

doxygen\_herezh

Généré par Doxygen 1.9.2



---

<b>1 Index des modules</b>	<b>1</b>
1.1 Modules	1
<b>2 Index hiérarchique</b>	<b>3</b>
2.1 Hiérarchie des classes	3
<b>3 Index des classes</b>	<b>21</b>
3.1 Liste des classes	21
<b>4 Index des fichiers</b>	<b>45</b>
4.1 Liste des fichiers	45
<b>5 Documentation des modules</b>	<b>59</b>
5.1 Les_algorithmes_de_resolutions_globales	59
5.1.1 Description détaillée	60
5.2 Les_lois_anisotropes	60
5.2.1 Description détaillée	61
5.3 Les_conteneurs_energies	61
5.3.1 Description détaillée	61
5.4 Les_lois_hyperelastiques	61
5.4.1 Description détaillée	62
5.5 Les_lois_hypoelastiques	62
5.5.1 Description détaillée	63
5.6 Les_lois_hysteresis	63
5.6.1 Description détaillée	63
5.7 Les_lois_hooke	63
5.7.1 Description détaillée	64
5.8 Les_lois_iso_non_lineaire	64
5.8.1 Description détaillée	64
5.9 Les_loiscombinees	65
5.9.1 Description détaillée	65
5.10 Les_lois_de_plasticite	65
5.10.1 Description détaillée	65
5.11 Les_lois_concernant_thermique	66
5.11.1 Description détaillée	66
5.12 Groupe_sur_les_contacts	66
5.12.1 Description détaillée	67
5.13 Les_Elements_de_geometrie	67
5.13.1 Description détaillée	67
5.14 Les_Elements_de_frontiere	68
5.14.1 Description détaillée	68
5.15 Groupe_des_deformations	68
5.15.1 Description détaillée	69
5.16 Groupe_des_metrique	69

---

5.16.1 Description détaillée	70
5.17 Groupe_concernant_les_points_integrations	70
5.17.1 Description détaillée	70
5.18 Group_types_enumeres	71
5.18.1 Description détaillée	77
5.18.2 Documentation des variables	78
5.18.2.1 map_Enum_ddl	78
5.18.2.2 remplir_map	78
5.19 Groupe_conteneurs_bloc	78
5.19.1 Description détaillée	78
5.20 Groupe_relatif_aux_entrees_sorties	79
5.20.1 Description détaillée	79
5.21 Les_classes_Ddl_en_tout_genre	79
5.21.1 Description détaillée	80
5.22 Les_classes_relatives_aux_conditions_limites	80
5.22.1 Description détaillée	80
5.23 Les_Maillages	80
5.23.1 Description détaillée	81
5.24 Les_classes_exportation_en_variables	81
5.24.1 Description détaillée	81
5.25 Les_parametres_generaux	82
5.25.1 Description détaillée	82
5.26 Les_classes_Reference	82
5.26.1 Description détaillée	83
5.26.2 Documentation des fonctions	83
5.26.2.1 operator<<()	83
5.27 Les_classes_Matrices	84
5.27.1 Description détaillée	84
5.28 Les_sorties_geomview	84
5.28.1 Description détaillée	85
5.29 Les_sorties_Gid	85
5.29.1 Description détaillée	85
5.30 Les_sorties_gmsh	85
5.30.1 Description détaillée	86
5.31 Les_sorties_au_format_maple	86
5.31.1 Description détaillée	86
5.32 Les_sorties_generiques	87
5.32.1 Description détaillée	87
5.33 Les_sorties_vrml	87
5.33.1 Description détaillée	88
5.34 Les_Tableaux_generiques	88
5.34.1 Description détaillée	88



---

5.35	Les_classes_cooronnee	89
5.35.1	Description détaillée	89
5.36	Les_classes_cooronnee1	89
5.36.1	Description détaillée	90
5.37	Les_classes_cooronnee2	90
5.37.1	Description détaillée	90
5.38	Les_classes_cooronnee3	91
5.38.1	Description détaillée	91
5.39	Les_classes_Base	91
5.39.1	Description détaillée	92
5.40	Les_classes_Base3D3	92
5.40.1	Description détaillée	92
5.41	Les_classes_tenseurs_virtuelles_ordre2	93
5.41.1	Description détaillée	93
5.42	Les_classes_Maillons_tenseurs	93
5.42.1	Description détaillée	94
5.43	Les_classes_tenseurs_dim1_ordre2	94
5.43.1	Description détaillée	95
5.44	Les_classes_tenseurs_dim2_ordre2	95
5.44.1	Description détaillée	96
5.45	Les_classes_tenseurs_dim3_ordre2	96
5.45.1	Description détaillée	97
5.46	Les_classes_tenseurs_virtuelles_ordre4	97
5.46.1	Description détaillée	98
5.47	Les_classes_Maillons_tenseurs_ordre4	98
5.47.1	Description détaillée	98
5.48	Les_classes_Vecteur	99
5.48.1	Description détaillée	99
5.49	Les_conteneurs_ultra_basiques	99
5.49.1	Description détaillée	100
5.50	Group_Ddl_enum_etendu	100
5.50.1	Description détaillée	100
5.51	Les_list_io	101
5.51.1	Description détaillée	101
5.52	Les_classes_PlusieursCoordonnees	101
5.52.1	Description détaillée	101
5.53	Les_classes_Ponderation	102
5.53.1	Description détaillée	102
5.54	Les_listes_de_petits_tableaux_de_reels	102
5.54.1	Description détaillée	103
5.55	Les_pointeurs_dans_listes_de_petits_tableaux	103
5.55.1	Description détaillée	104

5.56	Group_TypeQuelconque	105
5.56.1	Description détaillée	105
5.57	Group_TypeQuelconque_enum_etendu	105
5.57.1	Description détaillée	106
5.58	Les_grandeurs_particulieres	106
5.58.1	Description détaillée	107
5.59	Les_classes_algo	108
5.59.1	Description détaillée	108
5.60	Les_courbes_1D	108
5.60.1	Description détaillée	110
5.61	Les_fonctions_nD	110
5.61.1	Description détaillée	110
5.62	Classes_utilitaires_sur_vecteurs_et_matrices	111
5.62.1	Description détaillée	111
5.63	Classes_gestions_arret_Herezh	111
5.63.1	Description détaillée	112
5.64	def_classes_suites_reel	112
5.64.1	Description détaillée	112
5.65	Groupe_concernant_le_chargement	113
5.65.1	Description détaillée	113
5.66	Groupe_des_elements_finis	113
5.66.1	Description détaillée	115
5.66.2	Documentation des fonctions	115
5.66.2.1	Dim_sig_eps()	115
5.66.2.2	InitialisationUmatAbaqus()	115
<b>6</b>	<b>Documentation des classes</b>	<b>117</b>
6.1	Référence de la structure __CLPK_complex	117
6.2	Référence de la structure __CLPK_doublecomplex	117
6.3	Référence de la classe HyperD::A_i	117
6.4	Référence de la classe HyperD::A_iAvecVarDdl	118
6.5	Référence de la classe HyperD::A_iAvecVarEps	119
6.6	Référence de la classe VariablesExporter::A_un_E	120
6.7	Référence de la classe VariablesExporter::A_un_NE	121
6.8	Référence de la classe Algo_edp	122
6.8.1	Description détaillée	123
6.8.2	Documentation des fonctions membres	124
6.8.2.1	Pilotage_kutta()	124
6.8.2.2	Runge_Kutta_step23()	125
6.8.2.3	Runge_Kutta_step34()	125
6.8.2.4	Runge_Kutta_step45()	126
6.9	Référence de la classe Algo_Integ1D	126

---

6.9.1 Description détaillée	127
6.9.2 Documentation des fonctions membres	127
6.9.2.1 IntegGauss()	127
6.10 Référence de la classe Algo_zero	127
6.10.1 Description détaillée	129
6.10.2 Documentation des fonctions membres	130
6.10.2.1 Affiche()	130
6.11 Référence de la classe AlgoBonelli	130
6.11.1 Description détaillée	133
6.11.2 Documentation des fonctions membres	133
6.11.2.1 CalEquilibre()	134
6.11.2.2 Ecrit_Base_info_Parametre()	134
6.11.2.3 Execution()	134
6.11.2.4 FinCalcul()	135
6.11.2.5 Info_commande_parametres()	135
6.11.2.6 InitAlgorithme()	135
6.11.2.7 Lecture_Base_info_Parametre()	136
6.11.2.8 lecture_Parametres()	136
6.11.2.9 MiseAJourAlgo()	136
6.11.2.10 New_idem()	136
6.11.2.11 SchemaXML_Algori()	137
6.12 Référence de la classe AlgoInformations	137
6.12.1 Description détaillée	138
6.12.2 Documentation des fonctions membres	139
6.12.2.1 CalEquilibre()	139
6.12.2.2 Execution()	139
6.12.2.3 FinCalcul()	140
6.12.2.4 InitAlgorithme()	140
6.12.2.5 MiseAJourAlgo()	140
6.12.2.6 New_idem()	141
6.12.2.7 SchemaXML_Algori()	141
6.13 Référence de la classe Algori	141
6.13.1 Description détaillée	148
6.13.2 Documentation des fonctions membres	149
6.13.2.1 AmortissementCinetique()	149
6.13.2.2 CalEquilibre()	149
6.13.2.3 Info_commande_parametres()	149
6.13.2.4 lecture_Parametres()	149
6.13.2.5 SchemaXML_Algori()	150
6.14 Référence de la classe Algori_chung_lee	150
6.14.1 Description détaillée	152
6.14.2 Documentation des fonctions membres	152

---

6.14.2.1 CalEquilibre()	153
6.14.2.2 Ecrit_Base_info_Parametre()	153
6.14.2.3 Execution()	153
6.14.2.4 FinCalcul()	154
6.14.2.5 Info_commande_parametres()	154
6.14.2.6 InitAlgorithme()	154
6.14.2.7 Lecture_Base_info_Parametre()	155
6.14.2.8 lecture_Parametres()	155
6.14.2.9 MiseAJourAlgo()	155
6.14.2.10 New_idem()	155
6.14.2.11 SchemaXML_Algori()	156
6.15 Référence de la classe AlgoriCombine	156
6.15.1 Description détaillée	158
6.15.2 Documentation des fonctions membres	158
6.15.2.1 Arret_du_comptage_CPU()	158
6.15.2.2 AutreSortieTempsCPU()	158
6.15.2.3 CalEquilibre()	158
6.15.2.4 Execution()	159
6.15.2.5 FinCalcul()	159
6.15.2.6 InitAlgorithme()	159
6.15.2.7 Lecture()	160
6.15.2.8 MiseAJourAlgo()	160
6.15.2.9 New_idem()	160
6.15.2.10 SchemaXML_Algori()	160
6.15.2.11 Sortie_temps_cpu()	161
6.16 Référence de la classe AlgoriDynaExpli	161
6.16.1 Description détaillée	163
6.16.2 Documentation des fonctions membres	163
6.16.2.1 CalEquilibre()	164
6.16.2.2 Ecrit_Base_info_Parametre()	164
6.16.2.3 Execution()	164
6.16.2.4 FinCalcul()	165
6.16.2.5 Info_commande_parametres()	165
6.16.2.6 InitAlgorithme()	165
6.16.2.7 Lecture_Base_info_Parametre()	166
6.16.2.8 lecture_Parametres()	166
6.16.2.9 MiseAJourAlgo()	166
6.16.2.10 New_idem()	166
6.16.2.11 SchemaXML_Algori()	167
6.17 Référence de la classe AlgoriDynaExpli_zhai	167
6.17.1 Description détaillée	169
6.17.2 Documentation des fonctions membres	169

---

6.17.2.1 CalEquilibre()	170
6.17.2.2 Ecrit_Base_info_Parametre()	170
6.17.2.3 Execution()	170
6.17.2.4 FinCalcul()	171
6.17.2.5 Info_commande_parametres()	171
6.17.2.6 InitAlgorithme()	171
6.17.2.7 Lecture_Base_info_Parametre()	172
6.17.2.8 lecture_Parametres()	172
6.17.2.9 MiseAJourAlgo()	172
6.17.2.10 New_idem()	172
6.17.2.11 SchemaXML_Algori()	173
6.18 Référence de la classe AlgoriFlambLineaire	173
6.18.1 Description détaillée	174
6.18.2 Documentation des fonctions membres	175
6.18.2.1 CalEquilibre()	175
6.18.2.2 Execution()	175
6.18.2.3 FinCalcul()	176
6.18.2.4 InitAlgorithme()	176
6.18.2.5 MiseAJourAlgo()	176
6.18.2.6 New_idem()	177
6.18.2.7 SchemaXML_Algori()	177
6.19 Référence de la classe AlgoriNewmark	177
6.19.1 Description détaillée	178
6.19.2 Documentation des fonctions membres	179
6.19.2.1 CalEquilibre()	179
6.19.2.2 Execution()	179
6.19.2.3 FinCalcul()	180
6.19.2.4 InitAlgorithme()	180
6.19.2.5 MiseAJourAlgo()	180
6.19.2.6 New_idem()	181
6.19.2.7 SchemaXML_Algori()	181
6.20 Référence de la classe AlgoriNonDyna	181
6.20.1 Description détaillée	182
6.20.2 Documentation des fonctions membres	183
6.20.2.1 CalEquilibre()	183
6.20.2.2 Execution()	183
6.20.2.3 FinCalcul()	184
6.20.2.4 InitAlgorithme()	184
6.20.2.5 MiseAJourAlgo()	184
6.20.2.6 New_idem()	185
6.20.2.7 SchemaXML_Algori()	185
6.21 Référence de la classe AlgoriRelaxDyna	185

6.21.1 Description détaillée	188
6.21.2 Documentation des fonctions membres	188
6.21.2.1 CalEquilibre()	188
6.21.2.2 Ecrit_Base_info_Parametre()	189
6.21.2.3 Execution()	189
6.21.2.4 FinCalcul()	189
6.21.2.5 Info_commande_parametres()	190
6.21.2.6 InitAlgorithme()	190
6.21.2.7 Lecture_Base_info_Parametre()	190
6.21.2.8 lecture_Parametres()	190
6.21.2.9 MiseAJourAlgo()	191
6.21.2.10 New_idem()	191
6.21.2.11 Repercussion_algo_sur_cinematique()	191
6.21.2.12 SchemaXML_Algori()	191
6.22 Référence de la classe AlgoriRemontErreur	192
6.22.1 Description détaillée	193
6.22.2 Documentation des fonctions membres	193
6.22.2.1 Execution()	193
6.23 Référence de la classe AlgoriRungeKutta	193
6.23.1 Description détaillée	196
6.23.2 Documentation des fonctions membres	196
6.23.2.1 CalEquilibre()	196
6.23.2.2 Ecrit_Base_info_Parametre()	196
6.23.2.3 Execution()	197
6.23.2.4 FinCalcul()	197
6.23.2.5 Info_commande_parametres()	197
6.23.2.6 InitAlgorithme()	198
6.23.2.7 Lecture_Base_info_Parametre()	198
6.23.2.8 lecture_Parametres()	198
6.23.2.9 MiseAJourAlgo()	198
6.23.2.10 New_idem()	199
6.23.2.11 SchemaXML_Algori()	199
6.24 Référence de la classe AlgoristatExpli	199
6.24.1 Description détaillée	201
6.24.2 Documentation des fonctions membres	202
6.24.2.1 CalEquilibre()	202
6.24.2.2 Ecrit_Base_info_Parametre()	202
6.24.2.3 Execution()	202
6.24.2.4 FinCalcul()	203
6.24.2.5 Info_commande_parametres()	203
6.24.2.6 InitAlgorithme()	203
6.24.2.7 Lecture_Base_info_Parametre()	204

---

6.24.2.8 lecture_Parametres()	204
6.24.2.9 MiseAJourAlgo()	204
6.24.2.10 New_idem()	204
6.24.2.11 SchemaXML_Algori()	205
6.25 Référence de la classe AlgoriTchamwa	205
6.25.1 Description détaillée	208
6.25.2 Documentation des fonctions membres	208
6.25.2.1 CalEquilibre()	208
6.25.2.2 Ecrit_Base_info_Parametre()	208
6.25.2.3 Execution()	209
6.25.2.4 FinCalcul()	209
6.25.2.5 Info_commande_parametres()	209
6.25.2.6 InitAlgorithme()	210
6.25.2.7 Lecture_Base_info_Parametre()	210
6.25.2.8 lecture_Parametres()	210
6.25.2.9 MiseAJourAlgo()	210
6.25.2.10 New_idem()	211
6.25.2.11 SchemaXML_Algori()	211
6.26 Référence de la classe AlgoUmatAbaqus	211
6.26.1 Description détaillée	212
6.26.2 Documentation des fonctions membres	213
6.26.2.1 CalEquilibre()	213
6.26.2.2 Execution()	213
6.26.2.3 FinCalcul()	214
6.26.2.4 InitAlgorithme()	214
6.26.2.5 MiseAJourAlgo()	214
6.26.2.6 New_idem()	215
6.26.2.7 SchemaXML_Algori()	215
6.27 Référence de la classe AlgoUtils	215
6.27.1 Description détaillée	216
6.27.2 Documentation des fonctions membres	217
6.27.2.1 CalEquilibre()	217
6.27.2.2 Execution()	217
6.27.2.3 FinCalcul()	218
6.27.2.4 InitAlgorithme()	218
6.27.2.5 MiseAJourAlgo()	218
6.27.2.6 New_idem()	219
6.27.2.7 SchemaXML_Algori()	219
6.28 Référence de la classe Animation_geomview	219
6.28.1 Documentation des fonctions membres	221
6.28.1.1 ChoixOrdre()	221
6.28.1.2 Ecriture_parametres_OrdreVisu()	221

---

6.28.1.3 ExeOrdre()	221
6.28.1.4 Lecture_parametres_OrdreVisu()	222
6.29 Référence de la classe Animation_maple	222
6.29.1 Documentation des fonctions membres	223
6.29.1.1 ChoixOrdre()	223
6.29.1.2 Ecriture_parametres_OrdreVisu()	223
6.29.1.3 ExeOrdre()	224
6.29.1.4 Initialisation()	224
6.29.1.5 Lecture_parametres_OrdreVisu()	224
6.30 Référence de la classe Animation_vrml	225
6.30.1 Documentation des fonctions membres	225
6.30.1.1 ChoixOrdre()	226
6.30.1.2 Ecriture_parametres_OrdreVisu()	226
6.30.1.3 ExeOrdre()	226
6.30.1.4 Lecture_parametres_OrdreVisu()	226
6.31 Référence de la classe Assemblage	227
6.32 Référence de la classe Banniere	227
6.33 Référence de la classe Base3D3B	228
6.33.1 Description détaillée	229
6.34 Référence de la classe Base3D3H	229
6.34.1 Description détaillée	230
6.35 Référence de la classe BaseB	230
6.35.1 Description détaillée	231
6.35.2 Documentation des fonctions membres	231
6.35.2.1 ChangeBase_curviligne()	231
6.35.2.2 ChangeBase_theta_vers_Xi()	232
6.36 Référence de la classe BaseB_0_t_tdt	232
6.36.1 Description détaillée	233
6.37 Référence de la classe BaseH	233
6.37.1 Description détaillée	234
6.37.2 Documentation des fonctions membres	234
6.37.2.1 ChangeBase_curviligne()	235
6.37.2.2 ChangeBase_theta_vers_Xi()	235
6.38 Référence de la classe BaseH_0_t_tdt	235
6.38.1 Description détaillée	236
6.39 Référence de la classe Biel_axi	237
6.39.1 Documentation des fonctions membres	241
6.39.1.1 Active_ddl_Sigma()	241
6.39.1.2 Active_premier_ddl_Sigma()	241
6.39.1.3 ContraintesAbsolues()	241
6.39.1.4 Dim_sig_eps()	241
6.39.1.5 EpaisseurMoyenne()	242



---

6.39.1.6 Epaisseurs()	242
6.39.1.7 ErreurElement()	242
6.39.1.8 Inactive_ddl_Sigma()	242
6.39.1.9 LectureContraintes()	242
6.39.1.10 Les_types_particuliers_internes()	243
6.39.1.11 Long_arrete_mini_sur_c()	243
6.39.1.12 MatricesGeometrique_Et_Initiale()	243
6.39.1.13 new_frontiere_lin()	243
6.39.1.14 new_frontiere_surf()	243
6.39.1.15 Plus_ddl_Sigma()	244
6.39.1.16 Tableau_de_Sig1()	244
6.40 Référence de la classe Biel_axiQ	244
6.40.1 Documentation des fonctions membres	248
6.40.1.1 Active_ddl_Sigma()	248
6.40.1.2 Active_premier_ddl_Sigma()	248
6.40.1.3 ContraintesAbsolues()	248
6.40.1.4 Dim_sig_eps()	248
6.40.1.5 EpaisseurMoyenne()	249
6.40.1.6 Epaisseurs()	249
6.40.1.7 ErreurElement()	249
6.40.1.8 Inactive_ddl_Sigma()	249
6.40.1.9 LectureContraintes()	249
6.40.1.10 Les_types_particuliers_internes()	250
6.40.1.11 Long_arrete_mini_sur_c()	250
6.40.1.12 MatricesGeometrique_Et_Initiale()	250
6.40.1.13 new_frontiere_lin()	250
6.40.1.14 new_frontiere_surf()	250
6.40.1.15 Plus_ddl_Sigma()	251
6.40.1.16 Tableau_de_Sig1()	251
6.41 Référence de la classe Biellette	251
6.41.1 Documentation des fonctions membres	254
6.41.1.1 Active_ddl_Sigma()	255
6.41.1.2 Active_premier_ddl_Sigma()	255
6.41.1.3 ContraintesAbsolues()	255
6.41.1.4 Dim_sig_eps()	255
6.41.1.5 ErreurElement()	255
6.41.1.6 Inactive_ddl_Sigma()	256
6.41.1.7 LectureContraintes()	256
6.41.1.8 Les_types_particuliers_internes()	256
6.41.1.9 Long_arrete_mini_sur_c()	256
6.41.1.10 MatricesGeometrique_Et_Initiale()	256
6.41.1.11 new_frontiere_lin()	257

6.41.1.12 new_frontiere_surf()	257
6.41.1.13 Plus_ddl_Sigma()	257
6.41.1.14 Section()	257
6.41.1.15 SectionMoyenne()	257
6.41.1.16 Tableau_de_Sig1()	258
6.42 Référence de la classe BielletteC1	258
6.42.1 Documentation des fonctions membres	261
6.42.1.1 Active_ddl_Sigma()	261
6.42.1.2 Active_premier_ddl_Sigma()	262
6.42.1.3 ContraintesAbsolues()	262
6.42.1.4 Dim_sig_eps()	262
6.42.1.5 ErreurElement()	262
6.42.1.6 Inactive_ddl_Sigma()	262
6.42.1.7 LectureContraintes()	263
6.42.1.8 Long_arrete_mini_sur_c()	263
6.42.1.9 MatricesGeometrique_Et_Initiale()	263
6.42.1.10 new_frontiere_lin()	263
6.42.1.11 new_frontiere_surf()	263
6.42.1.12 Plus_ddl_Sigma()	264
6.42.1.13 Section()	264
6.42.1.14 SectionMoyenne()	264
6.42.1.15 Tableau_de_Sig1()	264
6.43 Référence de la classe BielletteQ	265
6.43.1 Documentation des fonctions membres	268
6.43.1.1 Active_ddl_Sigma()	269
6.43.1.2 Active_premier_ddl_Sigma()	269
6.43.1.3 ContraintesAbsolues()	269
6.43.1.4 Dim_sig_eps()	269
6.43.1.5 ErreurElement()	269
6.43.1.6 Inactive_ddl_Sigma()	270
6.43.1.7 LectureContraintes()	270
6.43.1.8 Les_types_particuliers_internes()	270
6.43.1.9 Long_arrete_mini_sur_c()	270
6.43.1.10 MatricesGeometrique_Et_Initiale()	270
6.43.1.11 new_frontiere_lin()	271
6.43.1.12 new_frontiere_surf()	271
6.43.1.13 Plus_ddl_Sigma()	271
6.43.1.14 Tableau_de_Sig1()	271
6.44 Référence de la classe BielletteThermi	272
6.44.1 Documentation des fonctions membres	275
6.44.1.1 Active_ddl_Flux()	275
6.44.1.2 Active_premier_ddl_Flux()	276

6.44.1.3 Dim_flux_gradT()	276
6.44.1.4 ErreurElement()	276
6.44.1.5 FluxAbsolues()	276
6.44.1.6 Inactive_ddl_Flux()	276
6.44.1.7 LectureFlux()	277
6.44.1.8 Les_types_particuliers_internes()	277
6.44.1.9 Long_arrete_mini_sur_c()	277
6.44.1.10 MatricesGeometrique_Et_Initiale()	277
6.44.1.11 new_frontiere_lin()	277
6.44.1.12 new_frontiere_surf()	278
6.44.1.13 Plus_ddl_Flux()	278
6.44.1.14 Section()	278
6.44.1.15 SectionMoyenne()	278
6.44.1.16 Tableau_de_Flux1()	278
6.45 Référence du modèle de la classe BlocDdlLim< Bloc_particulier >	279
6.45.1 Description détaillée	280
6.45.2 cas d'un bloc template avec conditions limites	280
6.46 Référence de la classe BlocGen	280
6.47 Référence de la classe BlocGen_3_0	281
6.48 Référence de la classe BlocGen_3_1	282
6.49 Référence de la classe BlocGen_3_2	283
6.50 Référence de la classe BlocGen_4_0	284
6.51 Référence de la classe BlocGen_5_0	285
6.52 Référence de la classe BlocGen_6_0	286
6.53 Référence de la classe BlocGeneEtVecMultType	286
6.54 Référence de la classe BlocScal	287
6.55 Référence de la classe BlocScal_ou_fctnD	288
6.56 Référence de la classe BlocScalType	288
6.57 Référence de la classe BlocVec	289
6.58 Référence de la classe BlocVecMultType	290
6.59 Référence de la classe BlocVecType	290
6.60 Référence de la classe Cercle	291
6.61 Référence de la classe Tenseur1BBBB::ChangementIndex	292
6.62 Référence de la classe Tenseur1BBHH::ChangementIndex	292
6.63 Référence de la classe Tenseur1HHBB::ChangementIndex	292
6.64 Référence de la classe Tenseur1HHHH::ChangementIndex	293
6.65 Référence de la classe Tenseur2BB::ChangementIndex	293
6.66 Référence de la classe Tenseur2BBBB::ChangementIndex	293
6.67 Référence de la classe Tenseur2BBHH::ChangementIndex	293
6.68 Référence de la classe Tenseur2BH::ChangementIndex	293
6.69 Référence de la classe Tenseur2HB::ChangementIndex	294
6.70 Référence de la classe Tenseur2HH::ChangementIndex	294

6.71	Référence de la classe <code>Tenseur2HHBB::ChangementIndex</code>	294
6.72	Référence de la classe <code>Tenseur2HHHH::ChangementIndex</code>	294
6.73	Référence de la classe <code>Tenseur3BB::ChangementIndex</code>	294
6.74	Référence de la classe <code>Tenseur3BBBB::ChangementIndex</code>	295
6.75	Référence de la classe <code>Tenseur3BBHH::ChangementIndex</code>	295
6.76	Référence de la classe <code>Tenseur3BH::ChangementIndex</code>	295
6.77	Référence de la classe <code>Tenseur3HB::ChangementIndex</code>	295
6.78	Référence de la classe <code>Tenseur3HH::ChangementIndex</code>	295
6.79	Référence de la classe <code>Tenseur3HHBB::ChangementIndex</code>	296
6.80	Référence de la classe <code>Tenseur3HHHH::ChangementIndex</code>	296
6.81	Référence de la classe <code>Tenseur_ns2BB::ChangementIndex</code>	296
6.82	Référence de la classe <code>Tenseur_ns2HH::ChangementIndex</code>	296
6.83	Référence de la classe <code>Tenseur_ns3BB::ChangementIndex</code>	296
6.84	Référence de la classe <code>Tenseur_ns3HH::ChangementIndex</code>	297
6.85	Référence de la classe <code>TenseurQ3_troisSym_BBBB::ChangementIndex</code>	297
6.86	Référence de la classe <code>TenseurQ3_troisSym_HHHH::ChangementIndex</code>	297
6.87	Référence de la classe <code>Choix_grandeurs_maple</code>	297
6.87.1	Documentation des fonctions membres	298
6.87.1.1	<code>ChoixOrdre()</code>	298
6.87.1.2	<code>Ecriture_parametres_OrdreVisu()</code>	298
6.87.1.3	<code>ExeOrdre()</code>	299
6.87.1.4	<code>Initialisation()</code>	299
6.87.1.5	<code>Lecture_parametres_OrdreVisu()</code>	299
6.88	Référence de la classe <code>ChoixDesMaillages_vrml</code>	300
6.88.1	Documentation des fonctions membres	300
6.88.1.1	<code>ChoixOrdre()</code>	300
6.88.1.2	<code>Ecriture_parametres_OrdreVisu()</code>	301
6.88.1.3	<code>ExeOrdre()</code>	301
6.88.1.4	<code>Lecture_parametres_OrdreVisu()</code>	301
6.89	Référence de la classe <code>ClassPourEnum_ddl</code>	301
6.89.1	Description détaillée	302
6.90	Référence de la classe <code>ClassPourEnumTypeGrandeur</code>	302
6.90.1	Description détaillée	302
6.91	Référence de la classe <code>ClassPourEnumTypeQuelconque</code>	303
6.91.1	Description détaillée	303
6.92	Référence de la classe <code>ClassPourEnuTypeCL</code>	303
6.92.1	Description détaillée	304
6.93	Référence de la classe <code>ClassPourEnuTypeQuelParticulier</code>	304
6.93.1	Description détaillée	304
6.94	Référence de la classe <code>CompCol_ILUPreconditioner_double</code>	304
6.94.1	Documentation des fonctions membres	305
6.94.1.1	<code>solve()</code>	305

6.94.1.2	<a href="#">trans_solve()</a>	305
6.95	<a href="#">Référence de la classe CompFrotAbstraite</a>	306
6.95.1	<a href="#">Documentation des fonctions membres</a>	307
6.95.1.1	<a href="#">Modif_comp_tangent_simplifie()</a>	307
6.95.1.2	<a href="#">Test_loi_simplife()</a>	307
6.96	<a href="#">Référence de la classe CompFrotCoulomb</a>	308
6.96.1	<a href="#">Documentation des fonctions membres</a>	309
6.96.1.1	<a href="#">Affiche()</a>	309
6.96.1.2	<a href="#">Calcul_DFrottement_tdt()</a>	309
6.96.1.3	<a href="#">Calcul_Frottement()</a>	309
6.96.1.4	<a href="#">Ecriture_base_info_loi()</a>	310
6.96.1.5	<a href="#">Info_commande_LoisDeComp()</a>	310
6.96.1.6	<a href="#">Lecture_base_info_loi()</a>	310
6.96.1.7	<a href="#">LectureDonneesParticulieres()</a>	310
6.96.1.8	<a href="#">Nouvelle_loi_identique()</a>	310
6.96.1.9	<a href="#">TestCompleet()</a>	310
6.97	<a href="#">Référence de la classe CompRow_ILUPreconditioner_double</a>	311
6.97.1	<a href="#">Documentation des fonctions membres</a>	311
6.97.1.1	<a href="#">solve()</a>	311
6.97.1.2	<a href="#">trans_solve()</a>	312
6.98	<a href="#">Référence de la classe CompThermoPhysiqueAbstraite</a>	312
6.98.1	<a href="#">Description détaillée</a>	313
6.98.2	<a href="#">Documentation des fonctions membres</a>	313
6.98.2.1	<a href="#">Modif_comp_tangent_simplifie()</a>	313
6.98.2.2	<a href="#">Test_loi_simplife()</a>	314
6.99	<a href="#">Référence de la classe Condilineaire</a>	314
6.100	<a href="#">Référence de la classe CondLim</a>	315
6.101	<a href="#">Référence de la classe Hexa::ConsHexa</a>	316
6.102	<a href="#">Référence de la classe Hexa_cm1pti::ConsHexa_cm1pti</a>	317
6.103	<a href="#">Référence de la classe Hexa_cm27pti::ConsHexa_cm27pti</a>	318
6.104	<a href="#">Référence de la classe Hexa_cm64pti::ConsHexa_cm64pti</a>	319
6.105	<a href="#">Référence de la classe HexaQ::ConsHexaQ</a>	320
6.106	<a href="#">Référence de la classe HexaQ_cm1pti::ConsHexaQ_cm1pti</a>	321
6.107	<a href="#">Référence de la classe HexaQ_cm27pti::ConsHexaQ_cm27pti</a>	322
6.108	<a href="#">Référence de la classe HexaQ_cm64pti::ConsHexaQ_cm64pti</a>	323
6.109	<a href="#">Référence de la classe HexaQComp::ConsHexaQComp</a>	324
6.110	<a href="#">Référence de la classe HexaQComp_cm1pti::ConsHexaQComp_cm1pti</a>	325
6.111	<a href="#">Référence de la classe HexaQComp_cm27pti::ConsHexaQComp_cm27pti</a>	326
6.112	<a href="#">Référence de la classe HexaQComp_cm64pti::ConsHexaQComp_cm64pti</a>	327
6.113	<a href="#">Référence de la classe PentaL::ConsPentaL</a>	328
6.114	<a href="#">Référence de la classe PentaL_cm1pti::ConsPentaL_cm1pti</a>	329
6.115	<a href="#">Référence de la classe PentaL_cm2pti::ConsPentaL_cm2pti</a>	330

6.116	Référence de la classe <code>PentaL_cm6pti::ConsPentaL_cm6pti</code>	331
6.117	Référence de la classe <code>PentaQ::ConsPentaQ</code>	332
6.118	Référence de la classe <code>PentaQ_cm12pti::ConsPentaQ_cm12pti</code>	333
6.119	Référence de la classe <code>PentaQ_cm18pti::ConsPentaQ_cm18pti</code>	334
6.120	Référence de la classe <code>PentaQ_cm3pti::ConsPentaQ_cm3pti</code>	335
6.121	Référence de la classe <code>PentaQ_cm9pti::ConsPentaQ_cm9pti</code>	336
6.122	Référence de la classe <code>PentaQComp::ConsPentaQComp</code>	337
6.123	Référence de la classe <code>PentaQComp_cm12pti::ConsPentaQComp_cm12pti</code>	338
6.124	Référence de la classe <code>PentaQComp_cm18pti::ConsPentaQComp_cm18pti</code>	339
6.125	Référence de la classe <code>PentaQComp_cm9pti::ConsPentaQComp_cm9pti</code>	340
6.126	Référence de la classe <code>PoutSfe3::ConsPoutSfe3</code>	341
6.127	Référence de la classe <code>Quad::ConsQuad</code>	342
6.128	Référence de la classe <code>Quad_cm1pti::ConsQuad_cm1pti</code>	343
6.129	Référence de la classe <code>QuadAxiCCom::ConsQuadAxiCCom</code>	344
6.130	Référence de la classe <code>QuadAxiCCom_cm9pti::ConsQuadAxiCCom_cm9pti</code>	345
6.131	Référence de la classe <code>QuadAxiL1::ConsQuadAxiL1</code>	346
6.132	Référence de la classe <code>QuadAxiL1_cm1pti::ConsQuadAxiL1_cm1pti</code>	347
6.133	Référence de la classe <code>QuadAxiQ::ConsQuadAxiQ</code>	348
6.134	Référence de la classe <code>QuadAxiQComp::ConsQuadAxiQComp</code>	349
6.135	Référence de la classe <code>QuadAxiQComp_cm4pti::ConsQuadAxiQComp_cm4pti</code>	350
6.136	Référence de la classe <code>QuadCCom::ConsQuadCCom</code>	351
6.137	Référence de la classe <code>QuadCCom_cm9pti::ConsQuadCCom_cm9pti</code>	352
6.138	Référence de la classe <code>QuadQ::ConsQuadQ</code>	353
6.139	Référence de la classe <code>QuadQCom::ConsQuadQCom</code>	354
6.140	Référence de la classe <code>QuadQCom_cm4pti::ConsQuadQCom_cm4pti</code>	355
6.141	Référence de la classe <code>Tetra::ConsTetra</code>	356
6.142	Référence de la classe <code>TetraQ::ConsTetraQ</code>	357
6.143	Référence de la classe <code>TetraQ_15pti::ConsTetraQ_15pti</code>	358
6.144	Référence de la classe <code>TetraQ_cm1pti::ConsTetraQ_cm1pti</code>	359
6.145	Référence de la classe <code>ConstMath</code>	359
6.146	Référence de la classe <code>ConstPhysico</code>	360
6.147	Référence de la classe <code>TriaAxiL1::ConsTriaAxiL1</code>	360
6.148	Référence de la classe <code>TriaAxiQ3::ConsTriaAxiQ3</code>	361
6.149	Référence de la classe <code>TriaAxiQ3_cm1pti::ConsTriaAxiQ3_cm1pti</code>	362
6.150	Référence de la classe <code>TriaAxiQ3_cmpti1003::ConsTriaAxiQ3_cmpti1003</code>	363
6.151	Référence de la classe <code>TriaCub::ConsTriaCub</code>	364
6.152	Référence de la classe <code>TriaCub_cm4pti::ConsTriaCub_cm4pti</code>	365
6.153	Référence de la classe <code>TriaMembL1::ConsTriaMembL1</code>	366
6.154	Référence de la classe <code>TriaMembQ3::ConsTriaMembQ3</code>	367
6.155	Référence de la classe <code>TriaMembQ3_cm1pti::ConsTriaMembQ3_cm1pti</code>	368
6.156	Référence de la classe <code>TriaQ3_cmpti1003::ConsTriaQ3_cmpti1003</code>	369
6.157	Référence de la classe <code>TriaQSfe1::ConsTriaQSfe1</code>	370

6.158	Référence de la classe <code>TriaQSfe3::ConsTriaQSfe3</code>	371
6.159	Référence de la classe <code>TriaSfe1::ConsTriaSfe1</code>	372
6.160	Référence de la classe <code>TriaSfe1_cm5pti::ConsTriaSfe1_cm5pti</code>	373
6.161	Référence de la classe <code>TriaSfe2::ConsTriaSfe2</code>	374
6.162	Référence de la classe <code>TriaSfe3::ConsTriaSfe3</code>	375
6.163	Référence de la classe <code>TriaSfe3_3D::ConsTriaSfe3_3D</code>	376
6.164	Référence de la classe <code>TriaSfe3_cm12pti::ConsTriaSfe3_cm12pti</code>	377
6.165	Référence de la classe <code>TriaSfe3_cm13pti::ConsTriaSfe3_cm13pti</code>	378
6.166	Référence de la classe <code>TriaSfe3_cm3pti::ConsTriaSfe3_cm3pti</code>	379
6.167	Référence de la classe <code>TriaSfe3_cm4pti::ConsTriaSfe3_cm4pti</code>	380
6.168	Référence de la classe <code>TriaSfe3_cm5pti::ConsTriaSfe3_cm5pti</code>	381
6.169	Référence de la classe <code>TriaSfe3_cm6pti::ConsTriaSfe3_cm6pti</code>	382
6.170	Référence de la classe <code>TriaSfe3_cm7pti::ConsTriaSfe3_cm7pti</code>	383
6.171	Référence de la classe <code>TriaSfe3C::ConsTriaSfe3C</code>	384
6.172	Référence de la classe <code>Biel_axi::ConstrucElementbiel</code>	385
6.173	Référence de la classe <code>Biel_axiQ::ConstrucElementbiel</code>	386
6.174	Référence de la classe <code>Biellette::ConstrucElementbiel</code>	387
6.175	Référence de la classe <code>BielletteC1::ConstrucElementbiel</code>	388
6.176	Référence de la classe <code>BielletteThermi::ConstrucElementbiel</code>	389
6.177	Référence de la classe <code>BielletteQ::ConstrucElementbielQ</code>	390
6.178	Référence de la classe <code>ElemPoint::ConstrucElemPoint</code>	391
6.179	Référence de la classe <code>ElemPoint_CP::ConstrucElemPoint_CP</code>	392
6.180	Référence de la classe <code>Isovaleurs_Gmsh::ConstructTabScalVecTensGmsh</code>	392
6.181	Référence de la classe <code>GeomSeg::Construire_Gauss_Lobatto</code>	393
6.182	Référence de la classe <code>ElemGeomC0::ConteneurExtrapolation</code>	393
6.183	Référence de la classe <code>Coordonnee</code>	393
6.183.1	Description détaillée	396
6.183.2	Documentation des fonctions membres	396
6.183.2.1	<code>Change_dim()</code>	396
6.183.2.2	<code>Dimension()</code>	396
6.183.2.3	<code>Libere()</code>	396
6.183.2.4	<code>Norme()</code>	396
6.183.2.5	<code>Vect()</code>	396
6.183.2.6	<code>Zero()</code>	397
6.184	Référence de la classe <code>Coordonnee1</code>	397
6.184.1	Description détaillée	399
6.184.2	Documentation des fonctions membres	399
6.184.2.1	<code>Change_dim()</code>	399
6.184.2.2	<code>Dimension()</code>	399
6.184.2.3	<code>Libere()</code>	399
6.184.2.4	<code>Norme()</code>	399
6.184.2.5	<code>Vect()</code>	399

---

6.184.2.6 Zero()	399
6.185 Référence de la classe Coordonnee1B	400
6.185.1 Description détaillée	402
6.185.2 Documentation des fonctions membres	402
6.185.2.1 Change_dim()	402
6.185.2.2 Dimension()	402
6.185.2.3 Libere()	402
6.185.2.4 Norme()	402
6.185.2.5 Vect()	402
6.185.2.6 Zero()	402
6.186 Référence de la classe Coordonnee1H	403
6.186.1 Description détaillée	405
6.186.2 Documentation des fonctions membres	405
6.186.2.1 Change_dim()	405
6.186.2.2 Dimension()	405
6.186.2.3 Libere()	405
6.186.2.4 Norme()	405
6.186.2.5 Vect()	405
6.186.2.6 Zero()	405
6.187 Référence de la classe Coordonnee2	406
6.187.1 Description détaillée	407
6.187.2 Documentation des fonctions membres	408
6.187.2.1 Change_dim()	408
6.187.2.2 Dimension()	408
6.187.2.3 Libere()	408
6.187.2.4 Norme()	408
6.187.2.5 Vect()	408
6.187.2.6 Zero()	408
6.188 Référence de la classe Coordonnee2B	408
6.188.1 Description détaillée	411
6.188.2 Documentation des fonctions membres	411
6.188.2.1 Change_dim()	411
6.188.2.2 Dimension()	411
6.188.2.3 Libere()	411
6.188.2.4 Norme()	411
6.188.2.5 Vect()	411
6.188.2.6 Zero()	411
6.189 Référence de la classe Coordonnee2H	412
6.189.1 Description détaillée	414
6.189.2 Documentation des fonctions membres	414
6.189.2.1 Change_dim()	414
6.189.2.2 Dimension()	414



---

6.189.2.3 Libere()	414
6.189.2.4 Norme()	414
6.189.2.5 Vect()	414
6.189.2.6 Zero()	414
6.190 Référence de la classe Coordonnee3	415
6.190.1 Description détaillée	416
6.190.2 Documentation des fonctions membres	417
6.190.2.1 Change_dim()	417
6.190.2.2 Dimension()	417
6.190.2.3 Libere()	417
6.190.2.4 Norme()	417
6.190.2.5 Vect()	417
6.190.2.6 Zero()	417
6.191 Référence de la classe Coordonnee3B	417
6.191.1 Description détaillée	420
6.191.2 Documentation des fonctions membres	420
6.191.2.1 Change_dim()	420
6.191.2.2 Dimension()	420
6.191.2.3 Libere()	420
6.191.2.4 Norme()	420
6.191.2.5 Vect()	420
6.191.2.6 Zero()	420
6.192 Référence de la classe Coordonnee3H	421
6.192.1 Description détaillée	423
6.192.2 Documentation des fonctions membres	423
6.192.2.1 Change_dim()	423
6.192.2.2 Dimension()	423
6.192.2.3 Libere()	423
6.192.2.4 Norme()	423
6.192.2.5 Vect()	423
6.192.2.6 Zero()	423
6.193 Référence de la classe CoordonneeB	424
6.193.1 Description détaillée	426
6.193.2 Documentation des fonctions membres	426
6.193.2.1 Change_dim()	426
6.193.2.2 Dimension()	426
6.193.2.3 Libere()	426
6.193.2.4 Norme()	427
6.193.2.5 Vect()	427
6.193.2.6 Zero()	427
6.194 Référence de la classe CoordonneeH	427
6.194.1 Description détaillée	429

6.194.2 Documentation des fonctions membres	430
6.194.2.1 Change_dim()	430
6.194.2.2 Dimension()	430
6.194.2.3 Libere()	430
6.194.2.4 Norme()	430
6.194.2.5 Vect()	430
6.194.2.6 Zero()	430
6.195 Référence de la classe Courbe1D	430
6.195.1 Description détaillée	433
6.195.2 Documentation des fonctions membres	433
6.195.2.1 Affiche()	433
6.195.2.2 Complet_courbe()	433
6.195.2.3 DependAutreCourbes()	434
6.195.2.4 Der_sec()	434
6.195.2.5 Derivee()	434
6.195.2.6 Ecriture_base_info()	434
6.195.2.7 Info_commande_Courbes1D()	434
6.195.2.8 LectDonnParticulieres_courbes()	435
6.195.2.9 Lecture_base_info()	435
6.195.2.10 Lien_entre_courbe()	435
6.195.2.11 ListDependanceCourbes()	435
6.195.2.12 SchemaXML_Courbes1D()	435
6.195.2.13 Valeur()	435
6.195.2.14 Valeur_Et_der12()	436
6.195.2.15 Valeur_Et_derivee()	436
6.195.2.16 Valeur_Et_derivee_stricte()	436
6.195.2.17 Valeur_stricte()	436
6.196 Référence de la classe Courbe_cos	436
6.196.1 Description détaillée	438
6.196.2 Documentation des fonctions membres	438
6.196.2.1 Affiche()	438
6.196.2.2 Complet_courbe()	438
6.196.2.3 Der_sec()	439
6.196.2.4 Derivee()	439
6.196.2.5 Ecriture_base_info()	439
6.196.2.6 Info_commande_Courbes1D()	439
6.196.2.7 LectDonnParticulieres_courbes()	439
6.196.2.8 Lecture_base_info()	439
6.196.2.9 SchemaXML_Courbes1D()	439
6.196.2.10 Valeur()	440
6.196.2.11 Valeur_Et_der12()	440
6.196.2.12 Valeur_Et_derivee()	440

---

6.196.2.13 Valeur_Et_derivee_stricte()	440
6.196.2.14 Valeur_stricte()	440
6.197 Référence de la classe Courbe_expo2_n	440
6.197.1 Description détaillée	442
6.197.2 Documentation des fonctions membres	442
6.197.2.1 Affiche()	442
6.197.2.2 Complet_courbe()	442
6.197.2.3 Der_sec()	443
6.197.2.4 Derivee()	443
6.197.2.5 Ecriture_base_info()	443
6.197.2.6 Info_commande_Courbes1D()	443
6.197.2.7 LectDonnParticulieres_courbes()	443
6.197.2.8 Lecture_base_info()	443
6.197.2.9 SchemaXML_Courbes1D()	443
6.197.2.10 Valeur()	444
6.197.2.11 Valeur_Et_der12()	444
6.197.2.12 Valeur_Et_derivee()	444
6.197.2.13 Valeur_Et_derivee_stricte()	444
6.197.2.14 Valeur_stricte()	444
6.198 Référence de la classe Courbe_expo_n	444
6.198.1 Description détaillée	446
6.198.2 Documentation des fonctions membres	446
6.198.2.1 Affiche()	446
6.198.2.2 Complet_courbe()	446
6.198.2.3 Der_sec()	447
6.198.2.4 Derivee()	447
6.198.2.5 Ecriture_base_info()	447
6.198.2.6 Info_commande_Courbes1D()	447
6.198.2.7 LectDonnParticulieres_courbes()	447
6.198.2.8 Lecture_base_info()	447
6.198.2.9 SchemaXML_Courbes1D()	447
6.198.2.10 Valeur()	448
6.198.2.11 Valeur_Et_der12()	448
6.198.2.12 Valeur_Et_derivee()	448
6.198.2.13 Valeur_Et_derivee_stricte()	448
6.198.2.14 Valeur_stricte()	448
6.199 Référence de la classe Courbe_expoaff	448
6.199.1 Description détaillée	450
6.199.2 Documentation des fonctions membres	450
6.199.2.1 Affiche()	450
6.199.2.2 Complet_courbe()	450
6.199.2.3 Der_sec()	451

6.199.2.4	Derivee()	451
6.199.2.5	Ecriture_base_info()	451
6.199.2.6	Info_commande_Courbes1D()	451
6.199.2.7	LectDonnParticulieres_courbes()	451
6.199.2.8	Lecture_base_info()	451
6.199.2.9	SchemaXML_Courbes1D()	451
6.199.2.10	Valeur()	452
6.199.2.11	Valeur_Et_der12()	452
6.199.2.12	Valeur_Et_derivee()	452
6.199.2.13	Valeur_Et_derivee_stricte()	452
6.199.2.14	Valeur_stricte()	452
6.200	Référence de la classe Courbe_expression_litterale_1D	452
6.200.1	Description détaillée	454
6.200.2	Documentation des fonctions membres	454
6.200.2.1	Affiche()	454
6.200.2.2	Complet_courbe()	454
6.200.2.3	Der_sec()	455
6.200.2.4	Derivee()	455
6.200.2.5	Ecriture_base_info()	455
6.200.2.6	Info_commande_Courbes1D()	455
6.200.2.7	LectDonnParticulieres_courbes()	455
6.200.2.8	Lecture_base_info()	455
6.200.2.9	SchemaXML_Courbes1D()	455
6.200.2.10	Valeur()	456
6.200.2.11	Valeur_Et_der12()	456
6.200.2.12	Valeur_Et_derivee()	456
6.200.2.13	Valeur_Et_derivee_stricte()	456
6.200.2.14	Valeur_stricte()	456
6.201	Référence de la classe Courbe_expression_litterale_avec_derivees_1D	456
6.201.1	Description détaillée	458
6.201.2	Documentation des fonctions membres	458
6.201.2.1	Affiche()	458
6.201.2.2	Complet_courbe()	459
6.201.2.3	Der_sec()	459
6.201.2.4	Derivee()	459
6.201.2.5	Ecriture_base_info()	459
6.201.2.6	Info_commande_Courbes1D()	459
6.201.2.7	LectDonnParticulieres_courbes()	459
6.201.2.8	Lecture_base_info()	459
6.201.2.9	SchemaXML_Courbes1D()	460
6.201.2.10	Valeur()	460
6.201.2.11	Valeur_Et_der12()	460

---

6.201.2.12 Valeur_Et_derivee()	460
6.201.2.13 Valeur_Et_derivee_stricte()	460
6.201.2.14 Valeur_stricte()	460
6.202 Référence de la classe Courbe_In_cosh	461
6.202.1 Description détaillée	462
6.202.2 Documentation des fonctions membres	462
6.202.2.1 Affiche()	463
6.202.2.2 Complet_courbe()	463
6.202.2.3 Der_sec()	463
6.202.2.4 Derivee()	463
6.202.2.5 Ecriture_base_info()	463
6.202.2.6 Info_commande_Courbes1D()	463
6.202.2.7 LectDonnParticulieres_courbes()	463
6.202.2.8 Lecture_base_info()	464
6.202.2.9 SchemaXML_Courbes1D()	464
6.202.2.10 Valeur()	464
6.202.2.11 Valeur_Et_der12()	464
6.202.2.12 Valeur_Et_derivee()	464
6.202.2.13 Valeur_Et_derivee_stricte()	464
6.202.2.14 Valeur_stricte()	464
6.203 Référence de la classe Courbe_relax_expo	465
6.203.1 Description détaillée	466
6.203.2 Documentation des fonctions membres	466
6.203.2.1 Affiche()	466
6.203.2.2 Complet_courbe()	467
6.203.2.3 Der_sec()	467
6.203.2.4 Derivee()	467
6.203.2.5 Ecriture_base_info()	467
6.203.2.6 Info_commande_Courbes1D()	467
6.203.2.7 LectDonnParticulieres_courbes()	467
6.203.2.8 Lecture_base_info()	467
6.203.2.9 SchemaXML_Courbes1D()	468
6.203.2.10 Valeur()	468
6.203.2.11 Valeur_Et_der12()	468
6.203.2.12 Valeur_Et_derivee()	468
6.203.2.13 Valeur_Et_derivee_stricte()	468
6.203.2.14 Valeur_stricte()	468
6.204 Référence de la classe Courbe_sin	468
6.204.1 Description détaillée	470
6.204.2 Documentation des fonctions membres	470
6.204.2.1 Affiche()	470
6.204.2.2 Complet_courbe()	470

6.204.2.3 Der_sec()	471
6.204.2.4 Derivee()	471
6.204.2.5 Ecriture_base_info()	471
6.204.2.6 Info_commande_Courbes1D()	471
6.204.2.7 LectDonnParticulieres_courbes()	471
6.204.2.8 Lecture_base_info()	471
6.204.2.9 SchemaXML_Courbes1D()	471
6.204.2.10 Valeur()	472
6.204.2.11 Valeur_Et_der12()	472
6.204.2.12 Valeur_Et_derivee()	472
6.204.2.13 Valeur_Et_derivee_stricte()	472
6.204.2.14 Valeur_stricte()	472
6.205 Référence de la classe Courbe_un_moins_cos	472
6.205.1 Description détaillée	474
6.205.2 Documentation des fonctions membres	474
6.205.2.1 Affiche()	474
6.205.2.2 Complet_courbe()	474
6.205.2.3 Der_sec()	475
6.205.2.4 Derivee()	475
6.205.2.5 Ecriture_base_info()	475
6.205.2.6 Info_commande_Courbes1D()	475
6.205.2.7 LectDonnParticulieres_courbes()	475
6.205.2.8 Lecture_base_info()	475
6.205.2.9 SchemaXML_Courbes1D()	475
6.205.2.10 Valeur()	476
6.205.2.11 Valeur_Et_der12()	476
6.205.2.12 Valeur_Et_derivee()	476
6.205.2.13 Valeur_Et_derivee_stricte()	476
6.205.2.14 Valeur_stricte()	476
6.206 Référence de la classe CourbePolyHermite1D	476
6.206.1 Description détaillée	478
6.206.2 Documentation des fonctions membres	478
6.206.2.1 Affiche()	478
6.206.2.2 Complet_courbe()	479
6.206.2.3 Der_sec()	479
6.206.2.4 Derivee()	479
6.206.2.5 Ecriture_base_info()	479
6.206.2.6 Info_commande_Courbes1D()	479
6.206.2.7 LectDonnParticulieres_courbes()	479
6.206.2.8 Lecture_base_info()	479
6.206.2.9 SchemaXML_Courbes1D()	480
6.206.2.10 Valeur()	480

---

6.206.2.11 Valeur_Et_der12()	480
6.206.2.12 Valeur_Et_derivee()	480
6.206.2.13 Valeur_Et_derivee_stricte()	480
6.206.2.14 Valeur_stricte()	480
6.207 Référence de la classe CourbePolyLineaire1D	480
6.207.1 Description détaillée	482
6.207.2 Documentation des fonctions membres	482
6.207.2.1 Affiche()	483
6.207.2.2 Complet_courbe()	483
6.207.2.3 Der_sec()	483
6.207.2.4 Derivee()	483
6.207.2.5 Ecriture_base_info()	483
6.207.2.6 Info_commande_Courbes1D()	483
6.207.2.7 LectDonnParticulieres_courbes()	483
6.207.2.8 Lecture_base_info()	484
6.207.2.9 SchemaXML_Courbes1D()	484
6.207.2.10 Valeur()	484
6.207.2.11 Valeur_Et_der12()	484
6.207.2.12 Valeur_Et_derivee()	484
6.207.2.13 Valeur_Et_derivee_stricte()	484
6.207.2.14 Valeur_stricte()	484
6.208 Référence de la classe CourbePolynomiale	485
6.208.1 Description détaillée	486
6.208.2 Documentation des fonctions membres	486
6.208.2.1 Affiche()	486
6.208.2.2 Complet_courbe()	487
6.208.2.3 Der_sec()	487
6.208.2.4 Derivee()	487
6.208.2.5 Ecriture_base_info()	487
6.208.2.6 Info_commande_Courbes1D()	487
6.208.2.7 LectDonnParticulieres_courbes()	487
6.208.2.8 Lecture_base_info()	487
6.208.2.9 SchemaXML_Courbes1D()	488
6.208.2.10 Valeur()	488
6.208.2.11 Valeur_Et_der12()	488
6.208.2.12 Valeur_Et_derivee()	488
6.208.2.13 Valeur_Et_derivee_stricte()	488
6.208.2.14 Valeur_stricte()	488
6.209 Référence de la classe Courbure_t_tdt	488
6.209.1 Description détaillée	490
6.210 Référence de la classe Cpl1D	490
6.210.1 Description détaillée	491

6.210.2	Documentation des fonctions membres	491
6.210.2.1	Info_commande_Courbes1D()	491
6.210.2.2	LectDonnParticulieres_courbes()	491
6.210.2.3	SchemaXML_Courbes1D()	491
6.211	Référence de la classe CristaliniteAbstraite	492
6.212	Référence de la classe Cylindre	493
6.213	Référence de la classe Ddl	494
6.213.1	Description détaillée	495
6.214	Référence de la classe VariablesExporter::Ddl_a_un_element	495
6.215	Référence de la classe VariablesExporter::Ddl_a_un_noeud	497
6.216	Référence de la classe Ddl_enum_etendu	498
6.216.1	Description détaillée	499
6.217	Référence de la classe Ddl_etendu	500
6.217.1	Description détaillée	501
6.218	Référence de la classe VariablesExporter::Ddl_etendu_a_un_noeud	501
6.219	Référence de la classe DdlLim::Ddlbloque	503
6.220	Référence de la classe DdlElement	503
6.220.1	Description détaillée	504
6.221	Référence de la classe DdlLim	504
6.221.1	Description détaillée	506
6.221.2	Documentation des fonctions membres	506
6.221.2.1	Lecture_champ()	506
6.222	Référence de la classe DdlNoeudElement	506
6.222.1	Description détaillée	507
6.223	Référence de la classe Def_Umat	507
6.223.1	Description détaillée	509
6.223.2	Documentation des fonctions membres	509
6.223.2.1	Nevez_deformation()	509
6.223.2.2	operator=()	509
6.224	Référence de la classe Deformation	509
6.224.1	Description détaillée	514
6.224.2	Documentation des fonctions membres	515
6.224.2.1	Cal_explicit_t()	515
6.224.2.2	Cal_explicit_tdt()	515
6.225	Référence de la classe DeformationP2D	516
6.225.1	Documentation des fonctions membres	518
6.225.1.1	ChangeNumIntegSH()	518
6.225.1.2	Nevez_deformation()	518
6.225.1.3	operator=() [1/2]	518
6.225.1.4	operator=() [2/2]	518
6.226	Référence de la classe DeformationPP	518
6.226.1	Description détaillée	521



6.226.2 Documentation des fonctions membres	522
6.226.2.1 BasePassage()	522
6.226.2.2 Cal_explicit_t() [1/2]	522
6.226.2.3 Cal_explicit_t() [2/2]	522
6.226.2.4 Cal_explicit_tdt() [1/2]	522
6.226.2.5 Cal_explicit_tdt() [2/2]	522
6.226.2.6 Cal_implicit()	523
6.226.2.7 DernierPtInteg()	523
6.226.2.8 Nevez_deformation()	523
6.226.2.9 NevezPtInteg()	523
6.226.2.10 operator=()	523
6.226.2.11 PremierPtInteg()	523
6.226.2.12 RemontExp_t() [1/2]	523
6.226.2.13 RemontExp_t() [2/2]	523
6.226.2.14 RemontExp_tdt() [1/2]	524
6.226.2.15 RemontExp_tdt() [2/2]	524
6.226.2.16 RemontImp() [1/2]	524
6.226.2.17 RemontImp() [2/2]	524
6.227 Référence de la classe DeformationSfe1	524
6.227.1 Description détaillée	527
6.227.2 Documentation des fonctions membres	527
6.227.2.1 Affiche()	527
6.227.2.2 AfficheDataSpecif()	527
6.227.2.3 BasePassage() [1/2]	527
6.227.2.4 BasePassage() [2/2]	528
6.227.2.5 Cal_explicit_t() [1/2]	528
6.227.2.6 Cal_explicit_t() [2/2]	528
6.227.2.7 Cal_explicit_tdt() [1/2]	528
6.227.2.8 Cal_explicit_tdt() [2/2]	528
6.227.2.9 Cal_implicit()	528
6.227.2.10 ChangeNumInteg()	529
6.227.2.11 DernierPtInteg()	529
6.227.2.12 Nevez_deformation()	529
6.227.2.13 NevezPtInteg()	529
6.227.2.14 New_et_Initialise()	529
6.227.2.15 operator=()	529
6.227.2.16 Position_0()	529
6.227.2.17 Position_t()	529
6.227.2.18 Position_tdt()	530
6.227.2.19 PremierPtInteg()	530
6.227.2.20 Remont0_t_tdt() [1/2]	530
6.227.2.21 Remont0_t_tdt() [2/2]	530

6.227.2.22 RemontExp_t() [1/2]	530
6.227.2.23 RemontExp_t() [2/2]	530
6.227.2.24 RemontExp_tdt() [1/2]	530
6.227.2.25 RemontExp_tdt() [2/2]	530
6.227.2.26 RemontImp() [1/2]	531
6.227.2.27 RemontImp() [2/2]	531
6.227.2.28 Retour_pti_precedant()	531
6.228 Référence de la classe Deformees_geomview	531
6.228.1 Documentation des fonctions membres	532
6.228.1.1 ExeOrdre()	532
6.229 Référence de la classe Deformees_Gid	533
6.229.1 Documentation des fonctions membres	534
6.229.1.1 ChoixOrdre()	534
6.229.1.2 Ecriture_parametres_OrdreVisu()	534
6.229.1.3 ExeOrdre()	534
6.229.1.4 Lecture_parametres_OrdreVisu()	535
6.230 Référence de la classe Deformees_Gmsh	535
6.230.1 Documentation des fonctions membres	536
6.230.1.1 ChoixOrdre()	536
6.230.1.2 Ecriture_parametres_OrdreVisu()	536
6.230.1.3 ExeOrdre()	537
6.230.1.4 Lecture_parametres_OrdreVisu()	537
6.231 Référence de la classe Deformees_maple	537
6.231.1 Documentation des fonctions membres	538
6.231.1.1 ChoixOrdre()	538
6.231.1.2 ExeOrdre()	538
6.232 Référence de la classe Deformees_vrml	539
6.232.1 Documentation des fonctions membres	541
6.232.1.1 ChoixOrdre()	541
6.232.1.2 Ecriture_parametres_OrdreVisu()	541
6.232.1.3 ExeOrdre()	541
6.232.1.4 Lecture_parametres_OrdreVisu()	541
6.233 Référence de la classe Deux_String	542
6.233.1 Description détaillée	542
6.234 Référence de la classe Deux_String_un_entier	542
6.234.1 Description détaillée	543
6.235 Référence de la classe DeuxCoordonnees	543
6.235.1 Description détaillée	543
6.236 Référence de la classe DeuxDoubles	543
6.236.1 Description détaillée	544
6.237 Référence de la classe Deformees_vrml::Deuxentiers	544
6.238 Référence de la classe DeuxEntiers	544

---

6.238.1 Description détaillée	545
6.239 Référence de la classe <code>OrdreVisu::Deuxentiers</code>	545
6.240 Référence de la classe <code>Deuxentiers_enu</code>	545
6.241 Référence de la classe <code>Algori::DeuxString</code>	545
6.242 Référence de la classe <code>DiagPreconditioner_double</code>	546
6.242.1 Documentation des fonctions membres	546
6.242.1.1 <code>solve()</code>	547
6.242.1.2 <code>trans_solve()</code>	547
6.243 Référence de la classe <code>DiversStockage</code>	547
6.243.1 Description détaillée	547
6.244 Référence de la classe <code>HexaMemb::DonnComHexa</code>	548
6.245 Référence de la classe <code>PentaMemb::DonnComPenta</code>	550
6.246 Référence de la classe <code>QuadAxiMemb::DonnComQuad</code>	552
6.247 Référence de la classe <code>QuadraMemb::DonnComQuad</code>	554
6.248 Référence de la classe <code>SfeMembT::DonnComSfe</code>	556
6.249 Référence de la classe <code>TetraMemb::DonnComTetra</code>	558
6.250 Référence de la classe <code>TriaAxiMemb::DonnComTria</code>	560
6.251 Référence de la classe <code>TriaMemb::DonnComTria</code>	562
6.252 Référence de la classe <code>Biel_axi::Donnee_specif</code>	563
6.253 Référence de la classe <code>Biel_axiQ::Donnee_specif</code>	564
6.254 Référence de la classe <code>Biellette::Donnee_specif</code>	565
6.255 Référence de la classe <code>BielletteC1::Donnee_specif</code>	565
6.256 Référence de la classe <code>BielletteQ::Donnee_specif</code>	566
6.257 Référence de la classe <code>BielletteThermi::Donnee_specif</code>	566
6.258 Référence de la classe <code>QuadAxiMemb::Donnee_specif</code>	567
6.259 Référence de la classe <code>QuadraMemb::Donnee_specif</code>	567
6.260 Référence de la classe <code>SfeMembT::Donnee_specif</code>	568
6.261 Référence de la classe <code>TriaAxiMemb::Donnee_specif</code>	568
6.262 Référence de la classe <code>TriaMemb::Donnee_specif</code>	568
6.263 Référence de la classe <code>ElemPoint::DonneeCommune</code>	570
6.264 Référence de la classe <code>Droite</code>	571
6.265 Référence de la classe <code>Dynamiq</code>	572
6.266 Référence de la classe <code>EiContact</code>	573
6.266.1 Documentation des fonctions membres	575
6.266.1.1 <code>ConditionLi()</code>	575
6.266.1.2 <code>Contact()</code>	576
6.266.1.3 <code>Valeur_fct_nD()</code>	576
6.267 Référence de la classe <code>ElemGeomC0</code>	576
6.268 Référence de la classe <code>ElemGeomC1</code>	578
6.269 Référence de la classe <code>ElemMeca</code>	580
6.269.1 Documentation des fonctions membres	586
6.269.1.1 <code>ErreurElement()</code>	586

6.269.1.2 Init_hourglass_comp()	587
6.269.1.3 Tableau_de_Sig1()	587
6.270 Référence de la classe ElemPoint	587
6.270.1 Documentation des fonctions membres	590
6.270.1.1 Active_ddl_Sigma()	590
6.270.1.2 Active_premier_ddl_Sigma()	590
6.270.1.3 ContraintesAbsolues()	590
6.270.1.4 Dim_sig_eps()	590
6.270.1.5 ErreurElement()	591
6.270.1.6 Inactive_ddl_Sigma()	591
6.270.1.7 LectureContraintes()	591
6.270.1.8 Long_arrete_mini_sur_c()	591
6.270.1.9 MatricesGeometrique_Et_Initiale()	591
6.270.1.10 new_frontiere_lin()	591
6.270.1.11 new_frontiere_surf()	591
6.270.1.12 Plus_ddl_Sigma()	591
6.270.1.13 Tableau_de_Sig1()	592
6.271 Référence de la classe ElemPoint_CP	592
6.272 Référence de la classe ElemThermi	593
6.272.1 Documentation des fonctions membres	598
6.272.1.1 ErreurElement()	598
6.272.1.2 Init_hourglass_comp()	598
6.272.1.3 Tableau_de_Flux1()	599
6.273 Référence de la classe ElFrontiere	599
6.273.1 Documentation des fonctions membres	601
6.273.1.1 BonCote_t()	601
6.274 Référence de la classe EnergieMeca	601
6.275 Référence de la classe EnergieThermi	602
6.276 Référence de la classe Entier_et_Double	602
6.276.1 Description détaillée	603
6.277 Référence de la classe Epai	603
6.277.1 Description détaillée	603
6.278 Référence de la classe Err_inconnue_ElemMeca	603
6.279 Référence de la classe ErrCalculFct_nD	604
6.279.1 Description détaillée	604
6.280 Référence de la classe ErrJacobienNegatif_ElemMeca	604
6.281 Référence de la classe ErrJacobienNegatif_ElemThermi	604
6.282 Référence de la classe ErrMathUtil2	604
6.282.1 Description détaillée	604
6.283 Référence de la classe ErrNonConvergence_loiDeComportement	604
6.283.1 Description détaillée	605
6.284 Référence de la classe ErrNonConvergence_Newton	605

6.284.1 Description détaillée	605
6.285 Référence de la classe UtilLecture::ErrNouvelEnreg	605
6.286 Référence de la classe UtilLecture::ErrNouvelEnregCVisu	605
6.287 Référence de la classe UtilLecture::ErrNouvelleDonnee	606
6.288 Référence de la classe UtilLecture::ErrNouvelleDonneeCVisu	606
6.289 Référence de la classe ErrResolve_system_lineaire	606
6.290 Référence de la classe ErrSortie	606
6.290.1 Description détaillée	606
6.291 Référence de la classe ErrSortieFinale	607
6.291.1 Description détaillée	607
6.292 Référence de la classe ErrVarJacobienMini_ElemMeca	607
6.293 Référence de la classe ErrVarJacobienMini_ElemThermi	607
6.294 Référence de la classe Expli	607
6.295 Référence de la classe Expli_t_tdt	608
6.296 Référence de la classe F1_plus_F2	609
6.296.1 Description détaillée	611
6.296.2 Documentation des fonctions membres	611
6.296.2.1 Affiche()	611
6.296.2.2 Complet_courbe()	611
6.296.2.3 DependAutreCourbes()	611
6.296.2.4 Der_sec()	612
6.296.2.5 Derivee()	612
6.296.2.6 Ecriture_base_info()	612
6.296.2.7 Info_commande_Courbes1D()	612
6.296.2.8 LectDonnParticulieres_courbes()	612
6.296.2.9 Lecture_base_info()	612
6.296.2.10 Lien_entre_courbe()	612
6.296.2.11 ListDependanceCourbes()	613
6.296.2.12 SchemaXML_Courbes1D()	613
6.296.2.13 Valeur()	613
6.296.2.14 Valeur_Et_der12()	613
6.296.2.15 Valeur_Et_derivee()	613
6.296.2.16 Valeur_Et_derivee_stricte()	613
6.296.2.17 Valeur_stricte()	613
6.297 Référence de la classe F1_rond_F2	614
6.297.1 Description détaillée	615
6.297.2 Documentation des fonctions membres	616
6.297.2.1 Affiche()	616
6.297.2.2 Complet_courbe()	616
6.297.2.3 DependAutreCourbes()	616
6.297.2.4 Der_sec()	616
6.297.2.5 Derivee()	616

6.297.2.6	Ecriture_base_info()	616
6.297.2.7	Info_commande_Courbes1D()	616
6.297.2.8	LectDonnParticulieres_courbes()	617
6.297.2.9	Lecture_base_info()	617
6.297.2.10	Lien_entre_courbe()	617
6.297.2.11	ListDependanceCourbes()	617
6.297.2.12	SchemaXML_Courbes1D()	617
6.297.2.13	Valeur()	617
6.297.2.14	Valeur_Et_der12()	617
6.297.2.15	Valeur_Et_derivee()	618
6.297.2.16	Valeur_Et_derivee_stricte()	618
6.297.2.17	Valeur_stricte()	618
6.298	Référence de la classe F_cycle_add	618
6.298.1	Description détaillée	620
6.298.2	Documentation des fonctions membres	620
6.298.2.1	Affiche()	620
6.298.2.2	Complet_courbe()	620
6.298.2.3	DependAutreCourbes()	620
6.298.2.4	Der_sec()	621
6.298.2.5	Derivee()	621
6.298.2.6	Ecriture_base_info()	621
6.298.2.7	Info_commande_Courbes1D()	621
6.298.2.8	LectDonnParticulieres_courbes()	621
6.298.2.9	Lecture_base_info()	621
6.298.2.10	Lien_entre_courbe()	621
6.298.2.11	ListDependanceCourbes()	622
6.298.2.12	SchemaXML_Courbes1D()	622
6.298.2.13	Valeur()	622
6.298.2.14	Valeur_Et_der12()	622
6.298.2.15	Valeur_Et_derivee()	622
6.298.2.16	Valeur_Et_derivee_stricte()	622
6.298.2.17	Valeur_stricte()	622
6.299	Référence de la classe F_cyclique	623
6.299.1	Description détaillée	624
6.299.2	Documentation des fonctions membres	625
6.299.2.1	Affiche()	625
6.299.2.2	Complet_courbe()	625
6.299.2.3	DependAutreCourbes()	625
6.299.2.4	Der_sec()	625
6.299.2.5	Derivee()	625
6.299.2.6	Ecriture_base_info()	625
6.299.2.7	Info_commande_Courbes1D()	626

6.299.2.8 LectDonnParticulieres_courbes()	626
6.299.2.9 Lecture_base_info()	626
6.299.2.10 Lien_entre_courbe()	626
6.299.2.11 ListDependanceCourbes()	626
6.299.2.12 SchemaXML_Courbes1D()	626
6.299.2.13 Valeur()	626
6.299.2.14 Valeur_Et_der12()	627
6.299.2.15 Valeur_Et_derivee()	627
6.299.2.16 Valeur_Et_derivee_stricte()	627
6.299.2.17 Valeur_stricte()	627
6.300 Référence de la classe F_nD_courbe1D	627
6.300.1 Description détaillée	629
6.300.2 Documentation des fonctions membres	629
6.300.2.1 Affiche() [1/2]	629
6.300.2.2 Affiche() [2/2]	629
6.300.2.3 Complet_Fonction() [1/2]	629
6.300.2.4 Complet_Fonction() [2/2]	629
6.300.2.5 DependAutreFoncCourbes() [1/2]	629
6.300.2.6 DependAutreFoncCourbes() [2/2]	630
6.300.2.7 Ecriture_base_info() [1/2]	630
6.300.2.8 Ecriture_base_info() [2/2]	630
6.300.2.9 Info_commande_Fonctions_nD() [1/2]	630
6.300.2.10 Info_commande_Fonctions_nD() [2/2]	630
6.300.2.11 LectDonnParticulieres_Fonction_nD() [1/2]	630
6.300.2.12 LectDonnParticulieres_Fonction_nD() [2/2]	630
6.300.2.13 Lecture_base_info() [1/2]	630
6.300.2.14 Lecture_base_info() [2/2]	631
6.300.2.15 Lien_entre_fonc_courbe() [1/2]	631
6.300.2.16 Lien_entre_fonc_courbe() [2/2]	631
6.300.2.17 ListDependanceCourbes() [1/2]	631
6.300.2.18 ListDependanceCourbes() [2/2]	631
6.300.2.19 Mise_a_jour_variables_globales() [1/2]	631
6.300.2.20 Mise_a_jour_variables_globales() [2/2]	631
6.300.2.21 NbComposante() [1/2]	631
6.300.2.22 NbComposante() [2/2]	631
6.300.2.23 NbVariable()	632
6.300.2.24 operator=() [1/2]	632
6.300.2.25 operator=() [2/2]	632
6.300.2.26 SchemaXML_Fonctions_nD() [1/2]	632
6.300.2.27 SchemaXML_Fonctions_nD() [2/2]	632
6.300.2.28 Valeur_FnD_interne() [1/2]	632
6.300.2.29 Valeur_FnD_interne() [2/2]	632

---

6.300.2.30 Valeur_pour_variables_globales()	632
6.300.2.31 Valeur_pour_variables_globales_interne()	633
6.301 Référence de la classe F_union_1D	633
6.301.1 Description détaillée	634
6.301.2 Documentation des fonctions membres	635
6.301.2.1 Affiche()	635
6.301.2.2 Complet_courbe()	635
6.301.2.3 DependAutreCourbes()	635
6.301.2.4 Der_sec()	635
6.301.2.5 Derivee()	635
6.301.2.6 Ecriture_base_info()	635
6.301.2.7 Info_commande_Courbes1D()	636
6.301.2.8 LectDonnParticulieres_courbes()	636
6.301.2.9 Lecture_base_info()	636
6.301.2.10 Lien_entre_courbe()	636
6.301.2.11 ListDependanceCourbes()	636
6.301.2.12 SchemaXML_Courbes1D()	636
6.301.2.13 Valeur()	636
6.301.2.14 Valeur_Et_der12()	637
6.301.2.15 Valeur_Et_derivee()	637
6.301.2.16 Valeur_Et_derivee_stricte()	637
6.301.2.17 Valeur_stricte()	637
6.302 Référence de la classe ElContact::Fct_nD_contact	638
6.303 Référence de la classe Fin_geomview	639
6.303.1 Documentation des fonctions membres	640
6.303.1.1 Ecriture_parametres_OrdreVisu()	640
6.303.1.2 ExeOrdre()	640
6.303.1.3 Lecture_parametres_OrdreVisu()	640
6.304 Référence de la classe Fin_Gid	641
6.304.1 Documentation des fonctions membres	642
6.304.1.1 Ecriture_parametres_OrdreVisu()	642
6.304.1.2 ExeOrdre()	642
6.304.1.3 Lecture_parametres_OrdreVisu()	642
6.305 Référence de la classe Fin_Gmsh	643
6.305.1 Documentation des fonctions membres	644
6.305.1.1 Ecriture_parametres_OrdreVisu()	644
6.305.1.2 ExeOrdre()	644
6.305.1.3 Lecture_parametres_OrdreVisu()	644
6.306 Référence de la classe Fin_maple	645
6.306.1 Documentation des fonctions membres	645
6.306.1.1 Ecriture_parametres_OrdreVisu()	645
6.306.1.2 ExeOrdre()	646



6.306.1.3 Lecture_parametres_OrdreVisu()	646
6.307 Référence de la classe Fin_vrml	646
6.307.1 Documentation des fonctions membres	647
6.307.1.1 Ecriture_parametres_OrdreVisu()	647
6.307.1.2 ExeOrdre()	647
6.307.1.3 Lecture_parametres_OrdreVisu()	648
6.308 Référence de la classe flambe_lin	648
6.309 Référence du modèle de la classe FMV_Vector< TYPE >	649
6.310 Référence de la classe Fonc_scalcombinees_nD	650
6.310.1 Description détaillée	652
6.310.2 Documentation des fonctions membres	652
6.310.2.1 Affiche() [1/2]	652
6.310.2.2 Affiche() [2/2]	652
6.310.2.3 Complet_Fonction() [1/2]	652
6.310.2.4 Complet_Fonction() [2/2]	652
6.310.2.5 DependAutreFoncCourbes() [1/2]	652
6.310.2.6 DependAutreFoncCourbes() [2/2]	652
6.310.2.7 Ecriture_base_info() [1/2]	653
6.310.2.8 Ecriture_base_info() [2/2]	653
6.310.2.9 Info_commande_Fonctions_nD() [1/2]	653
6.310.2.10 Info_commande_Fonctions_nD() [2/2]	653
6.310.2.11 LectDonnParticulieres_Fonction_nD() [1/2]	653
6.310.2.12 LectDonnParticulieres_Fonction_nD() [2/2]	653
6.310.2.13 Lecture_base_info() [1/2]	653
6.310.2.14 Lecture_base_info() [2/2]	653
6.310.2.15 Lien_entre_fonc_courbe() [1/2]	654
6.310.2.16 Lien_entre_fonc_courbe() [2/2]	654
6.310.2.17 ListDependanceCourbes() [1/2]	654
6.310.2.18 ListDependanceCourbes() [2/2]	654
6.310.2.19 ListDependanceFonctions() [1/2]	654
6.310.2.20 ListDependanceFonctions() [2/2]	654
6.310.2.21 Mise_a_jour_variables_globales() [1/2]	654
6.310.2.22 Mise_a_jour_variables_globales() [2/2]	654
6.310.2.23 NbComposante() [1/2]	655
6.310.2.24 NbComposante() [2/2]	655
6.310.2.25 operator=() [1/2]	655
6.310.2.26 operator=() [2/2]	655
6.310.2.27 SchemaXML_Fonctions_nD() [1/2]	655
6.310.2.28 SchemaXML_Fonctions_nD() [2/2]	655
6.310.2.29 Valeur_FnD_interne() [1/2]	655
6.310.2.30 Valeur_FnD_interne() [2/2]	655
6.310.2.31 Valeur_pour_variables_globales()	656

6.310.2.32 Valeur_pour_variables_globales_interne()	656
6.311 Référence de la classe Fonction_expression_litterale_nD	656
6.311.1 Description détaillée	657
6.311.2 Documentation des fonctions membres	658
6.311.2.1 Affiche() [1/2]	658
6.311.2.2 Affiche() [2/2]	658
6.311.2.3 Complet_Fonction() [1/2]	658
6.311.2.4 Complet_Fonction() [2/2]	658
6.311.2.5 Ecriture_base_info() [1/2]	658
6.311.2.6 Ecriture_base_info() [2/2]	658
6.311.2.7 Info_commande_Fonctions_nD() [1/2]	658
6.311.2.8 Info_commande_Fonctions_nD() [2/2]	659
6.311.2.9 LectDonnParticulieres_Fonction_nD() [1/2]	659
6.311.2.10 LectDonnParticulieres_Fonction_nD() [2/2]	659
6.311.2.11 Lecture_base_info() [1/2]	659
6.311.2.12 Lecture_base_info() [2/2]	659
6.311.2.13 Mise_a_jour_variables_globales() [1/2]	659
6.311.2.14 Mise_a_jour_variables_globales() [2/2]	659
6.311.2.15 NbComposante() [1/2]	659
6.311.2.16 NbComposante() [2/2]	660
6.311.2.17 NbVariable()	660
6.311.2.18 operator=() [1/2]	660
6.311.2.19 operator=() [2/2]	660
6.311.2.20 SchemaXML_Fonctions_nD() [1/2]	660
6.311.2.21 SchemaXML_Fonctions_nD() [2/2]	660
6.311.2.22 Valeur_FnD_interne() [1/2]	660
6.311.2.23 Valeur_FnD_interne() [2/2]	660
6.311.2.24 Valeur_pour_variables_globales()	661
6.311.2.25 Valeur_pour_variables_globales_interne()	661
6.312 Référence de la classe Fonction_externe_nD	661
6.312.1 Description détaillée	662
6.312.2 Documentation des fonctions membres	662
6.312.2.1 Affiche()	662
6.312.2.2 Complet_Fonction()	663
6.312.2.3 Ecriture_base_info()	663
6.312.2.4 Info_commande_Fonctions_nD()	663
6.312.2.5 LectDonnParticulieres_Fonction_nD()	663
6.312.2.6 Lecture_base_info()	663
6.312.2.7 Mise_a_jour_variables_globales()	663
6.312.2.8 NbComposante()	663
6.312.2.9 operator=()	663
6.312.2.10 SchemaXML_Fonctions_nD()	664

---

6.312.2.11 Valeur_FnD_interne()	664
6.312.2.12 Valeur_pour_variables_globales_interne()	664
6.313 Référence de la classe Fonction_nD	664
6.313.1 Description détaillée	669
6.314 Référence de la classe Force_hydroDyna	669
6.315 Référence de la classe Front	670
6.316 Référence de la classe Frontiere_initiale	671
6.316.1 Documentation des fonctions membres	672
6.316.1.1 ChoixOrdre()	672
6.316.1.2 ExeOrdre()	672
6.317 Référence de la classe Frontiere_initiale_geomview	673
6.317.1 Documentation des fonctions membres	674
6.317.1.1 Ecriture_parametres_OrdreVisu()	674
6.317.1.2 ExeOrdre()	674
6.317.1.3 Lecture_parametres_OrdreVisu()	675
6.318 Référence de la classe FrontPointF	675
6.318.1 Documentation des fonctions membres	676
6.318.1.1 Affiche()	676
6.318.1.2 AutreTangent()	677
6.318.1.3 BonCote_t()	677
6.318.1.4 BonCote_tdt()	677
6.318.1.5 DernierTangent()	677
6.318.1.6 Ecriture_base_info_ElFrontiere_pour_projection()	677
6.318.1.7 ElementGeometrique()	677
6.318.1.8 Frontiere()	677
6.318.1.9 InSurf()	677
6.318.1.10 Lecture_base_info_ElFrontiere_pour_projection()	678
6.318.1.11 Metrique()	678
6.318.1.12 NevezElemFront() [1/2]	678
6.318.1.13 NevezElemFront() [2/2]	678
6.318.1.14 operator=()	678
6.318.1.15 Phi()	678
6.318.1.16 Ref()	678
6.318.1.17 Tangent()	678
6.318.1.18 TangentRef()	679
6.318.1.19 TypeFrontiere()	679
6.319 Référence de la classe FrontQuadCC	679
6.319.1 Documentation des fonctions membres	680
6.319.1.1 Affiche()	680
6.319.1.2 AutreTangent()	680
6.319.1.3 BonCote_t()	680
6.319.1.4 BonCote_tdt()	680

6.319.1.5 DernierTangent()	681
6.319.1.6 Ecriture_base_info_EIFrontiere_pour_projection()	681
6.319.1.7 ElementGeometrique()	681
6.319.1.8 Frontiere()	681
6.319.1.9 InSurf()	681
6.319.1.10 Lecture_base_info_EIFrontiere_pour_projection()	681
6.319.1.11 Metrique()	681
6.319.1.12 NevezElemFront() [1/2]	681
6.319.1.13 NevezElemFront() [2/2]	681
6.319.1.14 operator=()	682
6.319.1.15 Phi()	682
6.319.1.16 Ref()	682
6.319.1.17 Tangent()	682
6.319.1.18 TangentRef()	682
6.319.1.19 TypeFrontiere()	682
6.320 Référence de la classe FrontQuadLine	683
6.320.1 Documentation des fonctions membres	684
6.320.1.1 Affiche()	684
6.320.1.2 AutreTangent()	684
6.320.1.3 BonCote_t()	684
6.320.1.4 BonCote_tdt()	684
6.320.1.5 DernierTangent()	684
6.320.1.6 Ecriture_base_info_EIFrontiere_pour_projection()	684
6.320.1.7 ElementGeometrique()	685
6.320.1.8 Frontiere()	685
6.320.1.9 InSurf()	685
6.320.1.10 Lecture_base_info_EIFrontiere_pour_projection()	685
6.320.1.11 Metrique()	685
6.320.1.12 NevezElemFront() [1/2]	685
6.320.1.13 NevezElemFront() [2/2]	685
6.320.1.14 operator=()	685
6.320.1.15 Phi()	685
6.320.1.16 Ref()	686
6.320.1.17 Tangent()	686
6.320.1.18 TangentRef()	686
6.320.1.19 TypeFrontiere()	686
6.321 Référence de la classe FrontQuadQC	686
6.321.1 Documentation des fonctions membres	687
6.321.1.1 Affiche()	687
6.321.1.2 AutreTangent()	688
6.321.1.3 BonCote_t()	688
6.321.1.4 BonCote_tdt()	688

6.321.1.5 DernierTangent()	688
6.321.1.6 Ecriture_base_info_EIFrontiere_pour_projection()	688
6.321.1.7 ElementGeometrique()	688
6.321.1.8 Frontiere()	688
6.321.1.9 InSurf()	688
6.321.1.10 Lecture_base_info_EIFrontiere_pour_projection()	689
6.321.1.11 Metrique()	689
6.321.1.12 NevezElemFront() [1/2]	689
6.321.1.13 NevezElemFront() [2/2]	689
6.321.1.14 operator=()	689
6.321.1.15 Phi()	689
6.321.1.16 Ref()	689
6.321.1.17 Tangent()	689
6.321.1.18 TangentRef()	690
6.321.1.19 TypeFrontiere()	690
6.322 Référence de la classe FrontQuadQuad	690
6.322.1 Documentation des fonctions membres	691
6.322.1.1 Affiche()	691
6.322.1.2 AutreTangent()	692
6.322.1.3 BonCote_t()	692
6.322.1.4 BonCote_tdt()	692
6.322.1.5 DernierTangent()	692
6.322.1.6 Ecriture_base_info_EIFrontiere_pour_projection()	692
6.322.1.7 ElementGeometrique()	692
6.322.1.8 Frontiere()	692
6.322.1.9 InSurf()	692
6.322.1.10 Lecture_base_info_EIFrontiere_pour_projection()	693
6.322.1.11 Metrique()	693
6.322.1.12 NevezElemFront() [1/2]	693
6.322.1.13 NevezElemFront() [2/2]	693
6.322.1.14 operator=()	693
6.322.1.15 Phi()	693
6.322.1.16 Ref()	693
6.322.1.17 Tangent()	693
6.322.1.18 TangentRef()	694
6.322.1.19 TypeFrontiere()	694
6.323 Référence de la classe FrontSegCub	694
6.323.1 Documentation des fonctions membres	695
6.323.1.1 Affiche()	695
6.323.1.2 AutreTangent()	696
6.323.1.3 BonCote_t()	696
6.323.1.4 BonCote_tdt()	696

6.323.1.5 DernierTangent()	696
6.323.1.6 Ecriture_base_info_EIFrontiere_pour_projection()	696
6.323.1.7 ElementGeometrique()	696
6.323.1.8 Frontiere()	696
6.323.1.9 InSurf()	696
6.323.1.10 Lecture_base_info_EIFrontiere_pour_projection()	697
6.323.1.11 LongueurApprox()	697
6.323.1.12 Metrique()	697
6.323.1.13 NevezElemFront() [1/2]	697
6.323.1.14 NevezElemFront() [2/2]	697
6.323.1.15 operator=()	697
6.323.1.16 Phi()	697
6.323.1.17 Ref()	697
6.323.1.18 Tangent()	697
6.323.1.19 TangentRef()	698
6.323.1.20 TypeFrontiere()	698
6.324 Référence de la classe FrontSegLine	698
6.324.1 Documentation des fonctions membres	699
6.324.1.1 Affiche()	699
6.324.1.2 AutreTangent()	700
6.324.1.3 BonCote_t()	700
6.324.1.4 BonCote_tdt()	700
6.324.1.5 DernierTangent()	700
6.324.1.6 Ecriture_base_info_EIFrontiere_pour_projection()	700
6.324.1.7 ElementGeometrique()	700
6.324.1.8 Frontiere()	700
6.324.1.9 InSurf()	700
6.324.1.10 Lecture_base_info_EIFrontiere_pour_projection()	701
6.324.1.11 LongueurApprox()	701
6.324.1.12 Metrique()	701
6.324.1.13 NevezElemFront() [1/2]	701
6.324.1.14 NevezElemFront() [2/2]	701
6.324.1.15 operator=()	701
6.324.1.16 Phi()	701
6.324.1.17 Ref()	701
6.324.1.18 Tangent()	701
6.324.1.19 TangentRef()	702
6.324.1.20 TypeFrontiere()	702
6.325 Référence de la classe FrontSegQuad	702
6.325.1 Documentation des fonctions membres	703
6.325.1.1 Affiche()	703
6.325.1.2 AutreTangent()	704

---

6.325.1.3 BonCote_t()	704
6.325.1.4 BonCote_tdt()	704
6.325.1.5 DernierTangent()	704
6.325.1.6 Ecriture_base_info_ElFrontiere_pour_projection()	704
6.325.1.7 ElementGeometrique()	704
6.325.1.8 Frontiere()	704
6.325.1.9 InSurf()	704
6.325.1.10 Lecture_base_info_ElFrontiere_pour_projection()	705
6.325.1.11 LongueurApprox()	705
6.325.1.12 Metrique()	705
6.325.1.13 NevezElemFront() [1/2]	705
6.325.1.14 NevezElemFront() [2/2]	705
6.325.1.15 operator=()	705
6.325.1.16 Phi()	705
6.325.1.17 Ref()	705
6.325.1.18 Tangent()	705
6.325.1.19 TangentRef()	706
6.325.1.20 TypeFrontiere()	706
6.326 Référence de la classe FrontTriaLine	706
6.326.1 Documentation des fonctions membres	707
6.326.1.1 Affiche()	707
6.326.1.2 AutreTangent()	708
6.326.1.3 BonCote_t()	708
6.326.1.4 BonCote_tdt()	708
6.326.1.5 DernierTangent()	708
6.326.1.6 Ecriture_base_info_ElFrontiere_pour_projection()	708
6.326.1.7 ElementGeometrique()	708
6.326.1.8 Frontiere()	708
6.326.1.9 InSurf()	708
6.326.1.10 Lecture_base_info_ElFrontiere_pour_projection()	709
6.326.1.11 Metrique()	709
6.326.1.12 NevezElemFront() [1/2]	709
6.326.1.13 NevezElemFront() [2/2]	709
6.326.1.14 operator=()	709
6.326.1.15 Phi()	709
6.326.1.16 Ref()	709
6.326.1.17 Tangent()	709
6.326.1.18 TangentRef()	710
6.326.1.19 TypeFrontiere()	710
6.327 Référence de la classe FrontTriaQuad	710
6.327.1 Documentation des fonctions membres	711
6.327.1.1 Affiche()	711

6.327.1.2 AutreTangent()	712
6.327.1.3 BonCote_t()	712
6.327.1.4 BonCote_tdt()	712
6.327.1.5 DernierTangent()	712
6.327.1.6 Ecriture_base_info_EIFrontiere_pour_projection()	712
6.327.1.7 ElementGeometrique()	712
6.327.1.8 Frontiere()	712
6.327.1.9 InSurf()	712
6.327.1.10 Lecture_base_info_EIFrontiere_pour_projection()	713
6.327.1.11 Metrique()	713
6.327.1.12 NevezElemFront() [1/2]	713
6.327.1.13 NevezElemFront() [2/2]	713
6.327.1.14 operator=()	713
6.327.1.15 Phi()	713
6.327.1.16 Ref()	713
6.327.1.17 Tangent()	713
6.327.1.18 TangentRef()	714
6.327.1.19 TypeFrontiere()	714
6.328 Référence de la classe LesCondLim::Gene_asso	714
6.329 Référence de la classe GeomHexaCom	714
6.329.1 Documentation des fonctions membres	715
6.329.1.1 Interieur()	715
6.329.1.2 Maxi_Coor_dans_directionGM()	715
6.330 Référence de la classe GeomHexaCubique	716
6.330.1 Documentation des fonctions membres	717
6.330.1.1 Dphi()	717
6.330.1.2 newElemGeomC0()	717
6.330.1.3 Phi()	717
6.331 Référence de la classe GeomHexalin	718
6.331.1 Documentation des fonctions membres	719
6.331.1.1 Dphi() [1/2]	719
6.331.1.2 Dphi() [2/2]	719
6.331.1.3 Interieur()	719
6.331.1.4 newElemGeomC0()	719
6.331.1.5 Phi() [1/2]	720
6.331.1.6 Phi() [2/2]	720
6.332 Référence de la classe GeomHexaQuad	720
6.332.1 Documentation des fonctions membres	721
6.332.1.1 Dphi()	721
6.332.1.2 newElemGeomC0()	722
6.332.1.3 Phi()	722
6.333 Référence de la classe GeomHexaQuadComp	722



---

6.333.1 Documentation des fonctions membres	723
6.333.1.1 Dphi()	723
6.333.1.2 newElemGeomC0()	724
6.333.1.3 Phi()	724
6.334 Référence de la classe GeomPentaCom	724
6.334.1 Documentation des fonctions membres	725
6.334.1.1 Interieur()	725
6.334.1.2 Maxi_Coor_dans_directionGM()	725
6.335 Référence de la classe GeomPentaL	726
6.335.1 Documentation des fonctions membres	727
6.335.1.1 Dphi()	727
6.335.1.2 newElemGeomC0()	727
6.335.1.3 Phi()	727
6.336 Référence de la classe GeomPentaQ	728
6.336.1 Documentation des fonctions membres	729
6.336.1.1 Dphi()	729
6.336.1.2 newElemGeomC0()	729
6.336.1.3 Phi()	729
6.337 Référence de la classe GeomPentaQComp	730
6.337.1 Documentation des fonctions membres	731
6.337.1.1 Dphi()	731
6.337.1.2 newElemGeomC0()	731
6.337.1.3 Phi()	731
6.338 Référence de la classe GeomPoint	731
6.338.1 Documentation des fonctions membres	732
6.338.1.1 Dphi()	732
6.338.1.2 Interieur()	732
6.338.1.3 Maxi_Coor_dans_directionGM()	733
6.338.1.4 newElemGeomC0()	733
6.338.1.5 Phi()	733
6.339 Référence de la classe GeomQuadrangle	733
6.339.1 Documentation des fonctions membres	734
6.339.1.1 Dphi()	734
6.339.1.2 Interieur()	734
6.339.1.3 Maxi_Coor_dans_directionGM()	735
6.339.1.4 newElemGeomC0()	735
6.339.1.5 Phi()	735
6.340 Référence de la classe GeomSeg	735
6.340.1 Documentation des fonctions membres	736
6.340.1.1 Dphi()	737
6.340.1.2 Interieur()	737
6.340.1.3 Maxi_Coor_dans_directionGM()	737

6.340.1.4 newElemGeomC0()	737
6.340.1.5 Phi()	737
6.341 Référence de la classe GeomTetraCom	738
6.341.1 Documentation des fonctions membres	739
6.341.1.1 Interieur()	739
6.341.1.2 Maxi_Coor_dans_directionGM()	739
6.342 Référence de la classe GeomTetraL	739
6.342.1 Documentation des fonctions membres	740
6.342.1.1 Dphi()	740
6.342.1.2 Interieur()	741
6.342.1.3 newElemGeomC0()	741
6.342.1.4 Phi()	741
6.343 Référence de la classe GeomTetraQ	741
6.343.1 Documentation des fonctions membres	742
6.343.1.1 Dphi()	743
6.343.1.2 Interieur()	743
6.343.1.3 newElemGeomC0()	743
6.343.1.4 Phi()	743
6.344 Référence de la classe GeomTriangle	743
6.344.1 Documentation des fonctions membres	744
6.344.1.1 Dphi()	744
6.344.1.2 Interieur()	745
6.344.1.3 Maxi_Coor_dans_directionGM()	745
6.344.1.4 newElemGeomC0()	745
6.344.1.5 Phi()	745
6.345 Référence de la classe gijHH_0_et_giH_0	745
6.346 Référence de la classe TypeQuelconque::Grandeur	746
6.346.1 Description détaillée	748
6.347 Référence de la classe Grandeur_BaseH	748
6.347.1 Description détaillée	749
6.347.2 Documentation des fonctions membres	750
6.347.2.1 Affectation_numerique()	750
6.347.2.2 Change_repere()	750
6.347.2.3 Ecriture_grandeur()	750
6.347.2.4 Grandeur_brut()	750
6.347.2.5 GrandeurNumOrdre()	750
6.347.2.6 InitParDefaut()	750
6.347.2.7 Lecture_grandeur()	750
6.347.2.8 NbMaxiNumeroOrdre()	750
6.347.2.9 New_idem_grandeur()	751
6.347.2.10 operator*=(())	751
6.347.2.11 operator/=(())	751

6.347.2.12 Type_enumGrandeurParticuliere()	751
6.347.2.13 Type_grandeurAssocie()	751
6.347.2.14 Type_structure_grandeurAssocie()	751
6.348 Référence de la classe Grandeur_coordonnee	751
6.348.1 Description détaillée	753
6.348.2 Documentation des fonctions membres	753
6.348.2.1 Affectation_numerique()	753
6.348.2.2 Change_repere()	753
6.348.2.3 Ecriture_grandeur()	753
6.348.2.4 Grandeur_brut()	753
6.348.2.5 GrandeurNumOrdre()	753
6.348.2.6 InitParDefaut()	753
6.348.2.7 Lecture_grandeur()	754
6.348.2.8 NbMaxiNumeroOrdre()	754
6.348.2.9 New_idem_grandeur()	754
6.348.2.10 operator*=(())	754
6.348.2.11 operator/=(())	754
6.348.2.12 Type_enumGrandeurParticuliere()	754
6.348.2.13 Type_grandeurAssocie()	754
6.348.2.14 Type_structure_grandeurAssocie()	754
6.349 Référence de la classe Grandeur_Ddl_etendu	754
6.349.1 Description détaillée	756
6.349.2 Documentation des fonctions membres	756
6.349.2.1 Affectation_numerique()	756
6.349.2.2 Change_repere()	756
6.349.2.3 Ecriture_grandeur()	756
6.349.2.4 Grandeur_brut()	756
6.349.2.5 GrandeurNumOrdre()	757
6.349.2.6 InitParDefaut()	757
6.349.2.7 Lecture_grandeur()	757
6.349.2.8 NbMaxiNumeroOrdre()	757
6.349.2.9 New_idem_grandeur()	757
6.349.2.10 Nom_ref()	757
6.349.2.11 operator*=(())	757
6.349.2.12 operator/=(())	757
6.349.2.13 Type_enumGrandeurParticuliere()	757
6.349.2.14 Type_grandeurAssocie()	758
6.349.2.15 Type_structure_grandeurAssocie()	758
6.350 Référence de la classe Grandeur_defaut	758
6.350.1 Description détaillée	759
6.350.1.1 grandeur par défaut: TypeQuelconque::Grandeur::Grandeur_defaut	759
6.350.2 Documentation des fonctions membres	759

6.350.2.1 Affectation_numerique()	759
6.350.2.2 Change_repere()	759
6.350.2.3 Ecriture_grandeur()	759
6.350.2.4 Grandeur_brut()	759
6.350.2.5 GrandeurNumOrdre()	759
6.350.2.6 InitParDefaut()	760
6.350.2.7 Lecture_grandeur()	760
6.350.2.8 NbMaxiNumeroOrdre()	760
6.350.2.9 New_idem_grandeur()	760
6.350.2.10 operator*=(())	760
6.350.2.11 operator/=(())	760
6.350.2.12 Type_enumGrandeurParticuliere()	760
6.350.2.13 Type_grandeurAssocie()	760
6.350.2.14 Type_structure_grandeurAssocie()	760
6.351 Référence de la classe Grandeur_Double_Nommer_indicer	761
6.351.1 Description détaillée	762
6.351.2 Documentation des fonctions membres	762
6.351.2.1 Affectation_numerique()	762
6.351.2.2 Change_repere()	762
6.351.2.3 Ecriture_grandeur()	762
6.351.2.4 Grandeur_brut()	762
6.351.2.5 GrandeurNumOrdre()	763
6.351.2.6 InitParDefaut()	763
6.351.2.7 Lecture_grandeur()	763
6.351.2.8 NbMaxiNumeroOrdre()	763
6.351.2.9 New_idem_grandeur()	763
6.351.2.10 Nom_ref()	763
6.351.2.11 operator*=(())	763
6.351.2.12 operator/=(())	763
6.351.2.13 Type_enumGrandeurParticuliere()	763
6.351.2.14 Type_grandeurAssocie()	764
6.351.2.15 Type_structure_grandeurAssocie()	764
6.352 Référence de la classe Grandeur_scalaire_double	764
6.352.1 Description détaillée	765
6.352.2 Documentation des fonctions membres	765
6.352.2.1 Affectation_numerique()	765
6.352.2.2 Change_repere()	765
6.352.2.3 Ecriture_grandeur()	765
6.352.2.4 Grandeur_brut()	766
6.352.2.5 GrandeurNumOrdre()	766
6.352.2.6 InitParDefaut()	766
6.352.2.7 Lecture_grandeur()	766

---

6.352.2.8 NbMaxiNumeroOrdre()	766
6.352.2.9 New_idem_grandeur()	766
6.352.2.10 operator*=(())	766
6.352.2.11 operator/=(())	766
6.352.2.12 Type_enumGrandeurParticuliere()	766
6.352.2.13 Type_grandeurAssocie()	767
6.352.2.14 Type_structure_grandeurAssocie()	767
6.353 Référence de la classe Grandeur_scalaire_entier	767
6.353.1 Description détaillée	768
6.353.2 Documentation des fonctions membres	768
6.353.2.1 Affectation_numerique()	768
6.353.2.2 Change_repere()	768
6.353.2.3 Ecriture_grandeur()	768
6.353.2.4 Grandeur_brut()	769
6.353.2.5 GrandeurNumOrdre()	769
6.353.2.6 InitParDefaut()	769
6.353.2.7 Lecture_grandeur()	769
6.353.2.8 NbMaxiNumeroOrdre()	769
6.353.2.9 New_idem_grandeur()	769
6.353.2.10 operator*=(())	769
6.353.2.11 operator/=(())	769
6.353.2.12 Type_enumGrandeurParticuliere()	769
6.353.2.13 Type_grandeurAssocie()	770
6.353.2.14 Type_structure_grandeurAssocie()	770
6.354 Référence de la classe Grandeur_TenseurBB	770
6.354.1 Description détaillée	771
6.354.2 Documentation des fonctions membres	771
6.354.2.1 Affectation_numerique()	771
6.354.2.2 Change_repere()	771
6.354.2.3 Ecriture_grandeur()	772
6.354.2.4 Grandeur_brut()	772
6.354.2.5 GrandeurNumOrdre()	772
6.354.2.6 InitParDefaut()	772
6.354.2.7 Lecture_grandeur()	772
6.354.2.8 NbMaxiNumeroOrdre()	772
6.354.2.9 New_idem_grandeur()	772
6.354.2.10 operator*=(())	772
6.354.2.11 operator/=(())	772
6.354.2.12 Type_enumGrandeurParticuliere()	773
6.354.2.13 Type_grandeurAssocie()	773
6.354.2.14 Type_structure_grandeurAssocie()	773
6.355 Référence de la classe Grandeur_TenseurBH	773

6.355.1 Description détaillée	775
6.355.2 Documentation des fonctions membres	775
6.355.2.1 Affectation_numerique()	775
6.355.2.2 Change_repere()	775
6.355.2.3 Ecriture_grandeur()	775
6.355.2.4 Grandeur_brut()	775
6.355.2.5 GrandeurNumOrdre()	775
6.355.2.6 InitParDefaut()	775
6.355.2.7 Lecture_grandeur()	775
6.355.2.8 NbMaxiNumeroOrdre()	776
6.355.2.9 New_idem_grandeur()	776
6.355.2.10 operator*=(())	776
6.355.2.11 operator/=(())	776
6.355.2.12 Type_enumGrandeurParticuliere()	776
6.355.2.13 Type_grandeurAssocie()	776
6.355.2.14 Type_structure_grandeurAssocie()	776
6.356 Référence de la classe Grandeur_TenseurHB	776
6.356.1 Description détaillée	778
6.356.2 Documentation des fonctions membres	778
6.356.2.1 Affectation_numerique()	778
6.356.2.2 Change_repere()	778
6.356.2.3 Ecriture_grandeur()	778
6.356.2.4 Grandeur_brut()	778
6.356.2.5 GrandeurNumOrdre()	779
6.356.2.6 InitParDefaut()	779
6.356.2.7 Lecture_grandeur()	779
6.356.2.8 NbMaxiNumeroOrdre()	779
6.356.2.9 New_idem_grandeur()	779
6.356.2.10 operator*=(())	779
6.356.2.11 operator/=(())	779
6.356.2.12 Type_enumGrandeurParticuliere()	779
6.356.2.13 Type_grandeurAssocie()	779
6.356.2.14 Type_structure_grandeurAssocie()	780
6.357 Référence de la classe Grandeur_TenseurHH	780
6.357.1 Description détaillée	781
6.357.2 Documentation des fonctions membres	781
6.357.2.1 Affectation_numerique()	781
6.357.2.2 Change_repere()	781
6.357.2.3 Ecriture_grandeur()	781
6.357.2.4 Grandeur_brut()	782
6.357.2.5 GrandeurNumOrdre()	782
6.357.2.6 InitParDefaut()	782

---

6.357.2.7	Lecture_grandeur()	782
6.357.2.8	NbMaxiNumeroOrdre()	782
6.357.2.9	New_idem_grandeur()	782
6.357.2.10	operator*=(())	782
6.357.2.11	operator/=(())	782
6.357.2.12	Type_enumGrandeurParticuliere()	782
6.357.2.13	Type_grandeurAssocie()	783
6.357.2.14	Type_structure_grandeurAssocie()	783
6.358	Référence de la classe Grandeur_Vecteur	783
6.358.1	Description détaillée	784
6.358.2	Documentation des fonctions membres	784
6.358.2.1	Affectation_numerique()	784
6.358.2.2	Change_repere()	784
6.358.2.3	Ecriture_grandeur()	784
6.358.2.4	Grandeur_brut()	785
6.358.2.5	GrandeurNumOrdre()	785
6.358.2.6	InitParDefaut()	785
6.358.2.7	Lecture_grandeur()	785
6.358.2.8	NbMaxiNumeroOrdre()	785
6.358.2.9	New_idem_grandeur()	785
6.358.2.10	operator*=(())	785
6.358.2.11	operator/=(())	785
6.358.2.12	Type_enumGrandeurParticuliere()	785
6.358.2.13	Type_grandeurAssocie()	786
6.358.2.14	Type_structure_grandeurAssocie()	786
6.359	Référence de la classe Grandeur_Vecteur_Nommer	786
6.359.1	Description détaillée	787
6.359.2	Documentation des fonctions membres	787
6.359.2.1	Affectation_numerique()	787
6.359.2.2	Change_repere()	787
6.359.2.3	Ecriture_grandeur()	788
6.359.2.4	Grandeur_brut()	788
6.359.2.5	GrandeurNumOrdre()	788
6.359.2.6	InitParDefaut()	788
6.359.2.7	Lecture_grandeur()	788
6.359.2.8	NbMaxiNumeroOrdre()	788
6.359.2.9	New_idem_grandeur()	788
6.359.2.10	Nom_ref()	788
6.359.2.11	operator*=(())	788
6.359.2.12	operator/=(())	789
6.359.2.13	Type_enumGrandeurParticuliere()	789
6.359.2.14	Type_grandeurAssocie()	789

---

6.359.2.15 Type_structure_grandeurAssocie()	789
6.360 Référence de la classe Hart_Smith3D	789
6.360.1 Documentation des fonctions membres	792
6.360.1.1 Affiche()	792
6.360.1.2 Calcul_dsigma_deps()	792
6.360.1.3 Calcul_DsigmaHH_tdt()	792
6.360.1.4 Calcul_SigmaHH()	793
6.360.1.5 CalculGrandeurTravail()	793
6.360.1.6 Ecriture_base_info_loi()	793
6.360.1.7 HsurH0()	794
6.360.1.8 Info_commande_LoisDeComp()	794
6.360.1.9 Lecture_base_info_loi()	794
6.360.1.10 LectureDonneesParticulieres()	794
6.360.1.11 Module_young_equivalent()	794
6.360.1.12 Nouvelle_loi_identique()	794
6.360.1.13 TestComplet()	794
6.361 Référence de la classe Hexa	795
6.361.1 Documentation des fonctions membres	796
6.361.1.1 new_frontiere_lin()	796
6.361.1.2 new_frontiere_surf()	796
6.362 Référence de la classe Hexa_cm1pti	797
6.362.1 Documentation des fonctions membres	798
6.362.1.1 new_frontiere_lin()	798
6.362.1.2 new_frontiere_surf()	798
6.363 Référence de la classe Hexa_cm27pti	799
6.363.1 Documentation des fonctions membres	800
6.363.1.1 new_frontiere_lin()	800
6.363.1.2 new_frontiere_surf()	800
6.364 Référence de la classe Hexa_cm64pti	801
6.364.1 Documentation des fonctions membres	802
6.364.1.1 new_frontiere_lin()	802
6.364.1.2 new_frontiere_surf()	802
6.365 Référence de la classe HexaLMemb	803
6.365.1 Documentation des fonctions membres	804
6.365.1.1 new_frontiere_lin()	804
6.365.1.2 new_frontiere_surf()	804
6.366 Référence de la classe HexaMemb	805
6.366.1 Documentation des fonctions membres	808
6.366.1.1 Active_ddl_Sigma()	808
6.366.1.2 Active_premier_ddl_Sigma()	808
6.366.1.3 ContraintesAbsolues()	809
6.366.1.4 Dim_sig_eps()	809



---

6.366.1.5	ErreurElement()	809
6.366.1.6	Inactive_ddl_Sigma()	809
6.366.1.7	LectureContraintes()	809
6.366.1.8	Long_arrete_mini_sur_c()	809
6.366.1.9	new_frontiere_lin()	809
6.366.1.10	new_frontiere_surf()	809
6.366.1.11	Plus_ddl_Sigma()	810
6.366.1.12	Tableau_de_Sig1()	810
6.367	Référence de la classe HexaQ	810
6.367.1	Documentation des fonctions membres	812
6.367.1.1	new_frontiere_lin()	812
6.367.1.2	new_frontiere_surf()	812
6.368	Référence de la classe HexaQ_cm1pti	812
6.368.1	Documentation des fonctions membres	814
6.368.1.1	new_frontiere_lin()	814
6.368.1.2	new_frontiere_surf()	814
6.369	Référence de la classe HexaQ_cm27pti	814
6.369.1	Documentation des fonctions membres	816
6.369.1.1	new_frontiere_lin()	816
6.369.1.2	new_frontiere_surf()	816
6.370	Référence de la classe HexaQ_cm64pti	816
6.370.1	Documentation des fonctions membres	818
6.370.1.1	new_frontiere_lin()	818
6.370.1.2	new_frontiere_surf()	818
6.371	Référence de la classe HexaQComp	818
6.371.1	Documentation des fonctions membres	820
6.371.1.1	new_frontiere_lin()	820
6.371.1.2	new_frontiere_surf()	820
6.372	Référence de la classe HexaQComp_cm1pti	820
6.372.1	Documentation des fonctions membres	822
6.372.1.1	new_frontiere_lin()	822
6.372.1.2	new_frontiere_surf()	822
6.373	Référence de la classe HexaQComp_cm27pti	822
6.373.1	Documentation des fonctions membres	824
6.373.1.1	new_frontiere_lin()	824
6.373.1.2	new_frontiere_surf()	824
6.374	Référence de la classe HexaQComp_cm64pti	824
6.374.1	Documentation des fonctions membres	826
6.374.1.1	new_frontiere_lin()	826
6.374.1.2	new_frontiere_surf()	826
6.375	Référence de la classe Hoffman1	826
6.375.1	Documentation des fonctions membres	827

6.375.1.1 Affiche()	827
6.375.1.2 Cristalinite() [1/2]	827
6.375.1.3 Cristalinite() [2/2]	828
6.375.1.4 Ecriture_don_base_info()	828
6.375.1.5 fct_KT()	828
6.375.1.6 Info_commande_LoisCrista()	828
6.375.1.7 Lecture_don_base_info()	828
6.375.1.8 LectureDonneesLoiCrista()	828
6.375.1.9 New_et_Initialise()	829
6.376 Référence de la classe Hoffman2	829
6.376.1 Documentation des fonctions membres	830
6.376.1.1 Affiche()	830
6.376.1.2 Cristalinite() [1/2]	830
6.376.1.3 Cristalinite() [2/2]	830
6.376.1.4 Ecriture_don_base_info()	830
6.376.1.5 fct_KT()	830
6.376.1.6 Info_commande_LoisCrista()	831
6.376.1.7 Lecture_don_base_info()	831
6.376.1.8 LectureDonneesLoiCrista()	831
6.376.1.9 New_et_Initialise()	831
6.377 Référence de la classe Hyper1	831
6.377.1 Documentation des fonctions membres	833
6.377.1.1 TestComplet()	833
6.378 Référence de la classe Hyper10	833
6.378.1 Documentation des fonctions membres	835
6.378.1.1 TestComplet() [1/2]	835
6.378.1.2 TestComplet() [2/2]	835
6.379 Référence de la classe Hyper3D	836
6.379.1 Documentation des fonctions membres	838
6.379.1.1 InvariantDe_V_bllb_leps()	838
6.379.1.2 Invariants()	838
6.379.1.3 Invariants_et_var()	838
6.379.1.4 Invariants_et_varEps()	839
6.379.1.5 PoGrenoble()	839
6.379.1.6 Potentiel()	839
6.379.1.7 Potentiel_et_var()	839
6.379.1.8 PotentielPhase()	839
6.379.1.9 PotentielPhase_et_var()	839
6.380 Référence de la classe Hyper3DN	840
6.380.1 Documentation des fonctions membres	842
6.380.1.1 HsurH0()	842
6.380.1.2 Invariants()	842

6.380.1.3 Invariants_et_var()	842
6.381 Référence de la classe Hyper_externe_W	843
6.381.1 Documentation des fonctions membres	845
6.381.1.1 Affiche()	846
6.381.1.2 AfficheDataSpecif()	846
6.381.1.3 Calcul_dsigma_deps()	846
6.381.1.4 Calcul_DsigmaHH_tdt()	846
6.381.1.5 Calcul_SigmaHH()	847
6.381.1.6 CalculGrandeurTravail()	847
6.381.1.7 Ecriture_base_info_loi()	847
6.381.1.8 Grandeur_particuliere()	848
6.381.1.9 HsurH0()	848
6.381.1.10 Info_commande_LoisDeComp()	848
6.381.1.11 Lecture_base_info_loi()	848
6.381.1.12 LectureDonneesParticulieres()	848
6.381.1.13 ListeGrandeurs_particulieres()	848
6.381.1.14 Module_compressibilite_equivalent()	848
6.381.1.15 Module_young_equivalent()	849
6.381.1.16 New_et_Initialise()	849
6.381.1.17 Nouvelle_loi_identique()	849
6.381.1.18 TestComplet()	849
6.382 Référence de la classe Hyper_W_gene_3D	849
6.382.1 Documentation des fonctions membres	852
6.382.1.1 Calcul_derivee_numerique()	852
6.382.1.2 Grandeur_particuliere()	852
6.382.1.3 ListeGrandeurs_particulieres()	852
6.382.1.4 New_et_Initialise()	852
6.383 Référence de la classe HyperD	852
6.383.1 Documentation des fonctions membres	855
6.383.1.1 Activation_stockage_grandeurs_quelconques()	855
6.383.1.2 AfficheDataSpecif()	855
6.383.1.3 Calcul_dsigma_deps()	855
6.383.1.4 Calcul_DsigmaHH_tdt()	856
6.383.1.5 Calcul_SigmaHH()	856
6.383.1.6 CalculGrandeurTravail()	857
6.383.1.7 Grandeur_particuliere()	857
6.383.1.8 Insertion_conteneur_dans_save_result()	857
6.383.1.9 Invariants_et_var()	857
6.383.1.10 ListeGrandeurs_particulieres()	857
6.383.1.11 New_et_Initialise()	858
6.384 Référence du modèle de la classe HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >	858

6.384.1	Documentation des fonctions membres	860
6.384.1.1	AfficheDataSpecif()	860
6.384.1.2	Calcul_DsigmaHH_tdt()	861
6.384.1.3	Calcul_SigmaHH()	861
6.384.1.4	CalculGrandeurTravail()	862
6.384.1.5	New_et_Initialise()	862
6.385	Référence de la classe Hypo_hooke1D	862
6.385.1	Documentation des fonctions membres	865
6.385.1.1	Affiche()	865
6.385.1.2	BsurB0()	865
6.385.1.3	Calcul_dsigma_deps()	865
6.385.1.4	Calcul_DsigmaHH_tdt()	865
6.385.1.5	Calcul_SigmaHH()	866
6.385.1.6	CalculGrandeurTravail()	866
6.385.1.7	Ecriture_base_info_loi()	867
6.385.1.8	Eps22BH()	867
6.385.1.9	Eps33BH()	867
6.385.1.10	Grandeur_particuliere()	867
6.385.1.11	HsurH0()	867
6.385.1.12	Info_commande_LoisDeComp()	867
6.385.1.13	Lecture_base_info_loi()	867
6.385.1.14	LectureDonneesParticulieres()	868
6.385.1.15	ListeGrandeurs_particulieres()	868
6.385.1.16	Module_compressibilite_equivalent()	868
6.385.1.17	Module_young_equivalent()	868
6.385.1.18	New_et_Initialise()	868
6.385.1.19	Nouvelle_loi_identique()	868
6.385.1.20	TestComplet()	868
6.386	Référence de la classe Hypo_hooke2D_C	869
6.386.1	Documentation des fonctions membres	871
6.386.1.1	Affiche()	872
6.386.1.2	Calcul_DsigmaHH_tdt()	872
6.386.1.3	Calcul_SigmaHH()	872
6.386.1.4	CalculGrandeurTravail()	873
6.386.1.5	Contraintes_planes_de_3D()	873
6.386.1.6	Ecriture_base_info_loi()	873
6.386.1.7	Eps33BH()	873
6.386.1.8	Grandeur_particuliere()	873
6.386.1.9	HsurH0()	873
6.386.1.10	Info_commande_LoisDeComp()	874
6.386.1.11	Lecture_base_info_loi()	874
6.386.1.12	LectureDonneesParticulieres()	874

6.386.1.13	Module_compressibilite_equivalent()	874
6.386.1.14	Module_young_equivalent()	874
6.386.1.15	New_et_Initialise()	874
6.386.1.16	Nouvelle_loi_identique()	874
6.386.1.17	TestComplet()	875
6.387	Référence de la classe Hypo_hooke3D	875
6.387.1	Documentation des fonctions membres	878
6.387.1.1	Affiche()	878
6.387.1.2	Calcul_dsigma_deps()	878
6.387.1.3	Calcul_DsigmaHH_tdt()	878
6.387.1.4	Calcul_SigmaHH()	879
6.387.1.5	CalculGrandeurTravail()	879
6.387.1.6	Ecriture_base_info_loi()	879
6.387.1.7	Grandeur_particuliere()	879
6.387.1.8	HsurH0()	880
6.387.1.9	Info_commande_LoisDeComp()	880
6.387.1.10	Lecture_base_info_loi()	880
6.387.1.11	LectureDonneesParticulieres()	880
6.387.1.12	ListeGrandeurs_particulieres()	880
6.387.1.13	Module_compressibilite_equivalent()	880
6.387.1.14	Module_young_equivalent()	881
6.387.1.15	New_et_Initialise()	881
6.387.1.16	Nouvelle_loi_identique()	881
6.387.1.17	TestComplet()	881
6.388	Référence de la classe Hypo_ortho3D_entrainee	881
6.388.1	Documentation des fonctions membres	884
6.388.1.1	Affiche()	884
6.388.1.2	Calcul_dsigma_deps()	884
6.388.1.3	Calcul_DsigmaHH_tdt()	884
6.388.1.4	Calcul_SigmaHH()	885
6.388.1.5	CalculGrandeurTravail()	885
6.388.1.6	Ecriture_base_info_loi()	886
6.388.1.7	Grandeur_particuliere()	886
6.388.1.8	HsurH0()	886
6.388.1.9	Info_commande_LoisDeComp()	886
6.388.1.10	Lecture_base_info_loi()	886
6.388.1.11	LectureDonneesParticulieres()	886
6.388.1.12	ListeGrandeurs_particulieres()	887
6.388.1.13	Module_compressibilite_equivalent()	887
6.388.1.14	Module_young_equivalent()	887
6.388.1.15	New_et_Initialise()	887
6.388.1.16	Nouvelle_loi_identique()	887

---

6.388.1.17 TestComplet()	887
6.389 Référence de la classe Hysteresis1D	888
6.389.1 Documentation des fonctions membres	891
6.389.1.1 Affiche()	891
6.389.1.2 Calcul_DsigmaHH_tdt()	891
6.389.1.3 Calcul_SigmaHH()	892
6.389.1.4 CalculGrandeurTravail()	892
6.389.1.5 Ecriture_base_info_loi()	892
6.389.1.6 Grandeur_particuliere()	893
6.389.1.7 HsurH0()	893
6.389.1.8 Info_commande_LoisDeComp()	893
6.389.1.9 Lecture_base_info_loi()	893
6.389.1.10 LectureDonneesParticulieres()	893
6.389.1.11 ListeGrandeurs_particulieres()	893
6.389.1.12 Module_young_equivalent()	894
6.389.1.13 New_et_Initialise()	894
6.389.1.14 Nouvelle_loi_identique()	894
6.389.1.15 TestComplet()	894
6.390 Référence de la classe Hysteresis3D	894
6.390.1 Documentation des fonctions membres	900
6.390.1.1 Affiche() [1/2]	900
6.390.1.2 Affiche() [2/2]	900
6.390.1.3 Calcul_dsigma_deps()	900
6.390.1.4 Calcul_DsigmaHH_tdt()	900
6.390.1.5 Calcul_SigmaHH()	901
6.390.1.6 CalculGrandeurTravail()	901
6.390.1.7 Ecriture_base_info_loi() [1/2]	902
6.390.1.8 Ecriture_base_info_loi() [2/2]	902
6.390.1.9 Grandeur_particuliere()	902
6.390.1.10 HsurH0()	902
6.390.1.11 Info_commande_LoisDeComp() [1/2]	902
6.390.1.12 Info_commande_LoisDeComp() [2/2]	902
6.390.1.13 Lecture_base_info_loi()	902
6.390.1.14 LectureDonneesParticulieres()	903
6.390.1.15 ListeGrandeurs_particulieres()	903
6.390.1.16 Module_young_equivalent()	903
6.390.1.17 New_et_Initialise() [1/2]	903
6.390.1.18 New_et_Initialise() [2/2]	903
6.390.1.19 Nouvelle_loi_identique() [1/2]	903
6.390.1.20 Nouvelle_loi_identique() [2/2]	903
6.390.1.21 TestComplet() [1/2]	903
6.390.1.22 TestComplet() [2/2]	903

---

6.391	Référence de la classe Hysteresis_bulk	904
6.391.1	Documentation des fonctions membres	907
6.391.1.1	Activation_stockage_grandeurs_quelconques()	908
6.391.1.2	Affiche()	908
6.391.1.3	Calcul_dsigma_deps()	908
6.391.1.4	Calcul_DsigmaHH_tdt()	908
6.391.1.5	Calcul_SigmaHH()	909
6.391.1.6	CalculGrandeurTravail()	909
6.391.1.7	Ecriture_base_info_loi()	909
6.391.1.8	Grandeur_particuliere()	910
6.391.1.9	HsurH0()	910
6.391.1.10	Info_commande_LoisDeComp()	910
6.391.1.11	Insertion_conteneur_dans_save_result()	910
6.391.1.12	Lecture_base_info_loi()	910
6.391.1.13	LectureDonneesParticulieres()	910
6.391.1.14	ListeGrandeurs_particulieres()	910
6.391.1.15	Module_young_equivalent()	911
6.391.1.16	New_et_Initialise()	911
6.391.1.17	Nouvelle_loi_identique()	911
6.391.1.18	TestCompleet()	911
6.392	Référence de la classe I_O_Condilinaire	911
6.393	Référence de la classe ICPreconditioner_double	913
6.393.1	Documentation des fonctions membres	914
6.393.1.1	solve()	914
6.393.1.2	trans_solve()	914
6.394	Référence de la classe ILUPreconditioner_double	914
6.394.1	Documentation des fonctions membres	915
6.394.1.1	solve()	915
6.394.1.2	trans_solve()	915
6.395	Référence de la classe Impli	916
6.396	Référence de la classe ImpliNonDynaCont	917
6.396.1	Description détaillée	919
6.396.2	Documentation des fonctions membres	919
6.396.2.1	CalEquilibre()	919
6.396.2.2	Execution()	919
6.396.2.3	FinCalcul()	919
6.396.2.4	InitAlgorithme()	920
6.396.2.5	MiseAJourAlgo()	920
6.396.2.6	New_idem()	920
6.396.2.7	SchemaXML_Algori()	920
6.397	Référence de la classe Increment_vrml	921
6.397.1	Documentation des fonctions membres	921

6.397.1.1 ChoixOrdre()	922
6.397.1.2 Ecriture_parametres_OrdreVisu()	922
6.397.1.3 ExeOrdre()	922
6.397.1.4 Lecture_parametres_OrdreVisu()	922
6.398 Référence de la classe Info0_t_tdt	923
6.399 Référence de la classe InfoExp_t	924
6.400 Référence de la classe InfoExp_tdt	924
6.401 Référence de la classe InfoImp	925
6.402 Référence de la classe Spectre::Initialisation_description_spectre	926
6.403 Référence de la classe Initialisation_tab_Daa	926
6.403.1 Description détaillée	926
6.404 Référence de la classe Initialisation_tab_De	926
6.404.1 Description détaillée	926
6.405 Référence de la classe ElemPoint::inNeNpti	926
6.406 Référence de la classe DdlElement::Int_initer	927
6.407 Référence de la classe HyperD::Invariant	927
6.408 Référence de la classe Hyper3D::Invariant0QepsCosphi	928
6.409 Référence de la classe Hyper3D::Invariant2Qeps	928
6.410 Référence de la classe Hyper3D::Invariant2QepsCosphi	928
6.411 Référence de la classe Hyper_W_gene_3D::Invariantpost3D	928
6.412 Référence de la classe HyperD::Invariantpost3D	929
6.413 Référence de la classe Hyper3D::InvariantQeps	929
6.414 Référence de la classe Hyper3D::InvariantQepsCosphi	929
6.415 Référence de la classe HyperD::InvariantVarDdl	930
6.416 Référence de la classe HyperD::InvariantVarEps	931
6.417 Référence de la classe Iso_elas_expo1D	932
6.417.1 Documentation des fonctions membres	934
6.417.1.1 Activation_stockage_grandeurs_quelconques()	935
6.417.1.2 Affiche()	935
6.417.1.3 Calcul_DsigmaHH_tdt()	935
6.417.1.4 Calcul_SigmaHH()	935
6.417.1.5 CalculGrandeurTravail()	936
6.417.1.6 Ecriture_base_info_loi()	936
6.417.1.7 Grandeur_particuliere()	936
6.417.1.8 HsurH0()	936
6.417.1.9 Info_commande_LoisDeComp()	936
6.417.1.10 Insertion_conteneur_dans_save_result()	937
6.417.1.11 Lecture_base_info_loi()	937
6.417.1.12 LectureDonneesParticulieres()	937
6.417.1.13 ListeGrandeurs_particulieres()	937
6.417.1.14 Module_compressibilite_equivalent()	937
6.417.1.15 Module_young_equivalent()	937



---

6.417.1.16 New_et_Initialise()	937
6.417.1.17 Nouvelle_loi_identique()	938
6.417.1.18 TestComplet()	938
6.418 Référence de la classe Iso_elas_expo3D	938
6.418.1 Documentation des fonctions membres	940
6.418.1.1 Affiche()	940
6.418.1.2 Calcul_dsigma_deps()	940
6.418.1.3 Calcul_DsigmaHH_tdt()	941
6.418.1.4 Calcul_SigmaHH()	941
6.418.1.5 CalculGrandeurTravail()	942
6.418.1.6 Ecriture_base_info_loi()	942
6.418.1.7 HsurH0()	942
6.418.1.8 Info_commande_LoisDeComp()	942
6.418.1.9 Lecture_base_info_loi()	942
6.418.1.10 LectureDonneesParticulieres()	943
6.418.1.11 Module_young_equivalent()	943
6.418.1.12 Nouvelle_loi_identique()	943
6.418.1.13 TestComplet()	943
6.419 Référence de la classe Iso_elas_SE1D	944
6.419.1 Documentation des fonctions membres	946
6.419.1.1 Affiche()	946
6.419.1.2 Calcul_DsigmaHH_tdt()	946
6.419.1.3 Calcul_SigmaHH()	947
6.419.1.4 CalculGrandeurTravail()	947
6.419.1.5 Ecriture_base_info_loi()	948
6.419.1.6 HsurH0()	948
6.419.1.7 Info_commande_LoisDeComp()	948
6.419.1.8 Lecture_base_info_loi()	948
6.419.1.9 LectureDonneesParticulieres()	948
6.419.1.10 Module_young_equivalent()	948
6.419.1.11 Nouvelle_loi_identique()	948
6.419.1.12 TestComplet()	949
6.420 Référence de la classe IsoHyper3DFavier3	949
6.420.1 Documentation des fonctions membres	951
6.420.1.1 Affiche()	951
6.420.1.2 Ecriture_base_info_loi()	951
6.420.1.3 HsurH0()	951
6.420.1.4 Info_commande_LoisDeComp()	952
6.420.1.5 Lecture_base_info_loi()	952
6.420.1.6 LectureDonneesParticulieres()	952
6.420.1.7 Module_young_equivalent()	952
6.420.1.8 Nouvelle_loi_identique()	952

---

6.420.1.9 PoGrenoble() [1/3]	952
6.420.1.10 PoGrenoble() [2/3]	952
6.420.1.11 PoGrenoble() [3/3]	953
6.420.1.12 PoGrenoble_et_V() [1/2]	953
6.420.1.13 PoGrenoble_et_V() [2/2]	953
6.420.1.14 PoGrenoble_et_var()	953
6.420.1.15 PoGrenoble_et_VV() [1/2]	953
6.420.1.16 PoGrenoble_et_VV() [2/2]	953
6.420.1.17 PoGrenoblePhase()	953
6.420.1.18 PoGrenoblePhase_et_var()	953
6.420.1.19 TestComplet()	954
6.421 Référence de la classe IsoHyper3DOrgeas1	954
6.421.1 Documentation des fonctions membres	957
6.421.1.1 Affiche()	957
6.421.1.2 Ecriture_base_info_loi()	957
6.421.1.3 HsurH0()	957
6.421.1.4 Info_commande_LoisDeComp()	957
6.421.1.5 Lecture_base_info_loi()	957
6.421.1.6 LectureDonneesParticulieres()	957
6.421.1.7 Module_young_equivalent()	957
6.421.1.8 Nouvelle_loi_identique()	958
6.421.1.9 PoGrenoble() [1/3]	958
6.421.1.10 PoGrenoble() [2/3]	958
6.421.1.11 PoGrenoble() [3/3]	958
6.421.1.12 PoGrenoble_et_V() [1/2]	958
6.421.1.13 PoGrenoble_et_V() [2/2]	958
6.421.1.14 PoGrenoble_et_var()	958
6.421.1.15 PoGrenoble_et_VV() [1/2]	959
6.421.1.16 PoGrenoble_et_VV() [2/2]	959
6.421.1.17 PoGrenoblePhase()	959
6.421.1.18 PoGrenoblePhase_et_var()	959
6.421.1.19 TestComplet()	959
6.422 Référence de la classe IsoHyper3DOrgeas2	959
6.422.1 Documentation des fonctions membres	962
6.422.1.1 Affiche()	962
6.422.1.2 Ecriture_base_info_loi()	962
6.422.1.3 HsurH0()	962
6.422.1.4 Info_commande_LoisDeComp()	962
6.422.1.5 Lecture_base_info_loi()	962
6.422.1.6 LectureDonneesParticulieres()	962
6.422.1.7 Module_compressibilite_equivalent()	962
6.422.1.8 Module_young_equivalent()	963

---

6.422.1.9 Nouvelle_loi_identique()	963
6.422.1.10 PoGrenoble() [1/3]	963
6.422.1.11 PoGrenoble() [2/3]	963
6.422.1.12 PoGrenoble() [3/3]	963
6.422.1.13 PoGrenoble_et_V() [1/2]	963
6.422.1.14 PoGrenoble_et_V() [2/2]	963
6.422.1.15 PoGrenoble_et_var()	964
6.422.1.16 PoGrenoble_et_VV() [1/2]	964
6.422.1.17 PoGrenoble_et_VV() [2/2]	964
6.422.1.18 PoGrenoblePhase()	964
6.422.1.19 PoGrenoblePhase_et_var()	964
6.422.1.20 TestComplet()	964
6.423 Référence de la classe IsoHyperBulk3	964
6.423.1 Documentation des fonctions membres	966
6.423.1.1 Affiche()	966
6.423.1.2 Ecriture_base_info_loi()	966
6.423.1.3 HsurH0()	966
6.423.1.4 Info_commande_LoisDeComp()	966
6.423.1.5 Lecture_base_info_loi()	967
6.423.1.6 LectureDonneesParticulieres()	967
6.423.1.7 Module_compressibilite_equivalent()	967
6.423.1.8 Module_young_equivalent()	967
6.423.1.9 Nouvelle_loi_identique()	967
6.423.1.10 PoGrenoble() [1/3]	967
6.423.1.11 PoGrenoble() [2/3]	967
6.423.1.12 PoGrenoble() [3/3]	968
6.423.1.13 PoGrenoble_et_V() [1/2]	968
6.423.1.14 PoGrenoble_et_V() [2/2]	968
6.423.1.15 PoGrenoble_et_var()	968
6.423.1.16 PoGrenoble_et_VV() [1/2]	968
6.423.1.17 PoGrenoble_et_VV() [2/2]	968
6.423.1.18 PoGrenoblePhase()	968
6.423.1.19 PoGrenoblePhase_et_var()	968
6.423.1.20 TestComplet()	969
6.424 Référence de la classe IsoHyperBulk_gene	969
6.424.1 Documentation des fonctions membres	971
6.424.1.1 Affiche()	971
6.424.1.2 Ecriture_base_info_loi()	971
6.424.1.3 HsurH0()	971
6.424.1.4 Info_commande_LoisDeComp()	971
6.424.1.5 Lecture_base_info_loi()	972
6.424.1.6 LectureDonneesParticulieres()	972

6.424.1.7	Module_compressibilite_equivalent()	972
6.424.1.8	Module_young_equivalent()	972
6.424.1.9	Nouvelle_loi_identique()	972
6.424.1.10	PoGrenoble() [1/3]	972
6.424.1.11	PoGrenoble() [2/3]	972
6.424.1.12	PoGrenoble() [3/3]	973
6.424.1.13	PoGrenoble_et_V() [1/2]	973
6.424.1.14	PoGrenoble_et_V() [2/2]	973
6.424.1.15	PoGrenoble_et_var()	973
6.424.1.16	PoGrenoble_et_VV() [1/2]	973
6.424.1.17	PoGrenoble_et_VV() [2/2]	973
6.424.1.18	PoGrenoblePhase()	973
6.424.1.19	PoGrenoblePhase_et_var()	973
6.424.1.20	TestCompleto()	974
6.425	Référence de la classe Isovaleurs_geomview	974
6.425.1	Documentation des fonctions membres	975
6.425.1.1	Sortie_dessin_legende()	975
6.426	Référence de la classe Isovaleurs_Gid	976
6.426.1	Documentation des fonctions membres	978
6.426.1.1	ChoixOrdre()	978
6.426.1.2	Ecriture_parametres_OrdreVisu()	978
6.426.1.3	ExeOrdre()	978
6.426.1.4	Initialisation()	979
6.426.1.5	Lecture_parametres_OrdreVisu()	979
6.427	Référence de la classe Isovaleurs_Gmsh	979
6.427.1	Documentation des fonctions membres	982
6.427.1.1	ChoixOrdre()	982
6.427.1.2	Ecriture_parametres_OrdreVisu()	982
6.427.1.3	ExeOrdre()	982
6.427.1.4	Initialisation()	983
6.427.1.5	Lecture_parametres_OrdreVisu()	983
6.428	Référence de la classe Isovaleurs_vrml	983
6.428.1	Documentation des fonctions membres	985
6.428.1.1	ChoixOrdre()	985
6.428.1.2	Ecriture_parametres_OrdreVisu()	985
6.428.1.3	ExeOrdre()	985
6.428.1.4	Initialisation()	985
6.428.1.5	Lecture_parametres_OrdreVisu()	986
6.429	Référence du modèle de la classe LaLIST_io< T >	986
6.429.1	Description détaillée	987
6.430	Référence de la classe LectBloc	987
6.431	Référence de la classe LectBlocmot	988

---

6.432	Référence de la classe LesChargeExtSurElement . . . . .	988
6.433	Référence de la classe LesCondLim . . . . .	988
6.434	Référence de la classe LesContacts . . . . .	990
6.434.1	Documentation des fonctions membres . . . . .	991
6.434.1.1	CalculReaction() . . . . .	991
6.435	Référence de la classe LesCourbes1D . . . . .	991
6.435.1	Description détaillée . . . . .	992
6.436	Référence de la classe LesFonctions_nD . . . . .	992
6.436.1	Description détaillée . . . . .	993
6.437	Référence de la classe LesLoisDeComp . . . . .	993
6.438	Référence de la classe LesMaillages . . . . .	994
6.438.1	Description détaillée . . . . .	997
6.439	Référence de la classe LesMaillonsBB . . . . .	997
6.439.1	Description détaillée . . . . .	997
6.439.2	Documentation des données membres . . . . .	997
6.439.2.1	maille . . . . .	998
6.440	Référence de la classe LesMaillonsBBBB . . . . .	998
6.440.1	Description détaillée . . . . .	998
6.440.2	Documentation des données membres . . . . .	998
6.440.2.1	maille . . . . .	998
6.441	Référence de la classe LesMaillonsBBHH . . . . .	998
6.441.1	Description détaillée . . . . .	999
6.441.2	Documentation des données membres . . . . .	999
6.441.2.1	maille . . . . .	999
6.442	Référence de la classe LesMaillonsBH . . . . .	999
6.442.1	Description détaillée . . . . .	999
6.442.2	Documentation des données membres . . . . .	999
6.442.2.1	maille . . . . .	999
6.443	Référence de la classe LesMaillonsBHBH . . . . .	1000
6.443.1	Description détaillée . . . . .	1000
6.443.2	Documentation des données membres . . . . .	1000
6.443.2.1	maille . . . . .	1000
6.444	Référence de la classe LesMaillonsBHHB . . . . .	1000
6.444.1	Description détaillée . . . . .	1000
6.444.2	Documentation des données membres . . . . .	1000
6.444.2.1	maille . . . . .	1001
6.445	Référence de la classe LesMaillonsHB . . . . .	1001
6.445.1	Description détaillée . . . . .	1001
6.445.2	Documentation des données membres . . . . .	1001
6.445.2.1	maille . . . . .	1001
6.446	Référence de la classe LesMaillonsHBBH . . . . .	1001
6.446.1	Description détaillée . . . . .	1002

6.446.2 Documentation des données membres	1002
6.446.2.1 maille	1002
6.447 Référence de la classe LesMaillonsHBHB	1002
6.447.1 Description détaillée	1002
6.447.2 Documentation des données membres	1002
6.447.2.1 maille	1002
6.448 Référence de la classe LesMaillonsHH	1003
6.448.1 Description détaillée	1003
6.448.2 Documentation des données membres	1003
6.448.2.1 maille	1003
6.449 Référence de la classe LesMaillonsHHBB	1003
6.449.1 Description détaillée	1003
6.449.2 Documentation des données membres	1003
6.449.2.1 maille	1004
6.450 Référence de la classe LesMaillonsHHHH	1004
6.450.1 Description détaillée	1004
6.450.2 Documentation des données membres	1004
6.450.2.1 maille	1004
6.451 Référence de la classe LesPtIntegMecaInterne	1004
6.452 Référence de la classe LesPtIntegThermiInterne	1005
6.453 Référence de la classe LesReferences	1006
6.453.1 Description détaillée	1007
6.454 Référence de la classe LesSuiteReel	1007
6.454.1 Description détaillée	1007
6.455 Référence de la classe LesValVecPropres	1007
6.456 Référence du modèle de la classe List_io< T >	1008
6.456.1 Description détaillée	1008
6.457 Référence de la classe Algori::ListDeuxString	1009
6.458 Référence de la classe LesLoisDeComp::Loi	1009
6.459 Référence de la classe Loi_comp_abstraite	1010
6.459.1 Documentation des fonctions membres	1014
6.459.1.1 Calcul_dsigma_deps()	1014
6.459.1.2 Calcul_DsigmaHH_tdt()	1014
6.459.1.3 Modif_comp_tangent_simplifie()	1015
6.459.1.4 Test_loi_simplife()	1015
6.460 Référence de la classe Loi_de_Tait	1016
6.460.1 Documentation des fonctions membres	1017
6.460.1.1 Affiche()	1017
6.460.1.2 AfficheDataSpecif()	1017
6.460.1.3 Cal_donnees_thermiques()	1017
6.460.1.4 Calcul_DfluxH_tdt()	1018
6.460.1.5 Ecriture_base_info_loi()	1018

6.460.1.6	Grandeur_particuliere()	1018
6.460.1.7	Info_commande_LoisDeComp()	1018
6.460.1.8	Lecture_base_info_loi()	1018
6.460.1.9	LectureDonneesParticulieres()	1018
6.460.1.10	ListeGrandeurs_particulieres()	1019
6.460.1.11	New_et_Initialise()	1019
6.460.1.12	Nouvelle_loi_identique()	1019
6.460.1.13	TestComplet()	1019
6.461	Référence de la classe Loi_iso_elas1D	1019
6.461.1	Documentation des fonctions membres	1021
6.461.1.1	Activation_stockage_grandeurs_quelconques()	1022
6.461.1.2	Affiche()	1022
6.461.1.3	Calcul_DsigmaHH_tdt()	1022
6.461.1.4	Calcul_SigmaHH()	1022
6.461.1.5	CalculGrandeurTravail()	1023
6.461.1.6	Ecriture_base_info_loi()	1023
6.461.1.7	Grandeur_particuliere()	1023
6.461.1.8	HsurH0()	1023
6.461.1.9	Info_commande_LoisDeComp()	1023
6.461.1.10	Insertion_conteneur_dans_save_result()	1024
6.461.1.11	Lecture_base_info_loi()	1024
6.461.1.12	LectureDonneesParticulieres()	1024
6.461.1.13	ListeGrandeurs_particulieres()	1024
6.461.1.14	Module_compressibilite_equivalent()	1024
6.461.1.15	Module_young_equivalent()	1024
6.461.1.16	New_et_Initialise()	1024
6.461.1.17	Nouvelle_loi_identique()	1025
6.461.1.18	TestComplet()	1025
6.462	Référence de la classe Loi_iso_elas2D_C	1025
6.462.1	Documentation des fonctions membres	1027
6.462.1.1	Activation_stockage_grandeurs_quelconques()	1028
6.462.1.2	Affiche()	1028
6.462.1.3	Calcul_DsigmaHH_tdt()	1028
6.462.1.4	Calcul_SigmaHH()	1028
6.462.1.5	CalculGrandeurTravail()	1029
6.462.1.6	Contraintes_planes_de_3D()	1029
6.462.1.7	Ecriture_base_info_loi()	1029
6.462.1.8	Eps33BH()	1029
6.462.1.9	Grandeur_particuliere()	1029
6.462.1.10	HsurH0()	1030
6.462.1.11	Info_commande_LoisDeComp()	1030
6.462.1.12	Insertion_conteneur_dans_save_result()	1030

6.462.1.13	Lecture_base_info_loi()	1030
6.462.1.14	LectureDonneesParticulieres()	1030
6.462.1.15	ListeGrandeurs_particulieres()	1030
6.462.1.16	Module_compressibilite_equivalent()	1030
6.462.1.17	Module_young_equivalent()	1031
6.462.1.18	New_et_Initialise()	1031
6.462.1.19	Nouvelle_loi_identique()	1031
6.462.1.20	TestComplet()	1031
6.463	Référence de la classe Loi_iso_elas2D_D	1031
6.463.1	Documentation des fonctions membres	1033
6.463.1.1	Activation_stockage_grandeurs_quelconques()	1034
6.463.1.2	Affiche()	1034
6.463.1.3	Calcul_DsigmaHH_tdt()	1034
6.463.1.4	Calcul_SigmaHH()	1034
6.463.1.5	CalculGrandeurTravail()	1035
6.463.1.6	Ecriture_base_info_loi()	1035
6.463.1.7	Grandeur_particuliere()	1035
6.463.1.8	HsurH0()	1035
6.463.1.9	Info_commande_LoisDeComp()	1035
6.463.1.10	Insertion_conteneur_dans_save_result()	1036
6.463.1.11	Lecture_base_info_loi()	1036
6.463.1.12	LectureDonneesParticulieres()	1036
6.463.1.13	ListeGrandeurs_particulieres()	1036
6.463.1.14	Module_compressibilite_equivalent()	1036
6.463.1.15	Module_young_equivalent()	1036
6.463.1.16	New_et_Initialise()	1036
6.463.1.17	Nouvelle_loi_identique()	1037
6.463.1.18	TestComplet()	1037
6.464	Référence de la classe Loi_iso_elas3D	1037
6.464.1	Documentation des fonctions membres	1040
6.464.1.1	Activation_stockage_grandeurs_quelconques()	1040
6.464.1.2	Affiche()	1040
6.464.1.3	Calcul_dsigma_deps()	1040
6.464.1.4	Calcul_DsigmaHH_tdt()	1040
6.464.1.5	Calcul_SigmaHH()	1041
6.464.1.6	CalculGrandeurTravail()	1041
6.464.1.7	Ecriture_base_info_loi()	1041
6.464.1.8	Grandeur_particuliere()	1042
6.464.1.9	HsurH0()	1042
6.464.1.10	Info_commande_LoisDeComp()	1042
6.464.1.11	Insertion_conteneur_dans_save_result()	1042
6.464.1.12	Lecture_base_info_loi()	1042



---

6.464.1.13 LectureDonneesParticulieres()	1042
6.464.1.14 ListeGrandeurs_particulieres()	1042
6.464.1.15 Module_compressibilite_equivalent()	1043
6.464.1.16 Module_young_equivalent()	1043
6.464.1.17 New_et_Initialise()	1043
6.464.1.18 Nouvelle_loi_identique()	1043
6.464.1.19 TestComplet()	1043
6.465 Référence de la classe Loi_iso_thermo	1044
6.465.1 Documentation des fonctions membres	1045
6.465.1.1 Affiche()	1045
6.465.1.2 AfficheDataSpecif()	1045
6.465.1.3 Cal_donnees_thermiques()	1045
6.465.1.4 Calcul_DfluxH_tdt()	1045
6.465.1.5 Ecriture_base_info_loi()	1046
6.465.1.6 Grandeur_particuliere()	1046
6.465.1.7 Info_commande_LoisDeComp()	1046
6.465.1.8 Lecture_base_info_loi()	1046
6.465.1.9 LectureDonneesParticulieres()	1046
6.465.1.10 ListeGrandeurs_particulieres()	1046
6.465.1.11 New_et_Initialise()	1047
6.465.1.12 Nouvelle_loi_identique()	1047
6.465.1.13 TestComplet()	1047
6.466 Référence de la classe Loi_ortho2D_C_entrainee	1047
6.466.1 Documentation des fonctions membres	1050
6.466.1.1 Affiche()	1050
6.466.1.2 Calcul_dsigma_deps()	1050
6.466.1.3 Calcul_DsigmaHH_tdt()	1050
6.466.1.4 Calcul_SigmaHH()	1051
6.466.1.5 CalculGrandeurTravail()	1051
6.466.1.6 Contraintes_planes_de_3D()	1052
6.466.1.7 Ecriture_base_info_loi()	1052
6.466.1.8 Eps33BH()	1052
6.466.1.9 Grandeur_particuliere()	1052
6.466.1.10 HsurH0()	1052
6.466.1.11 Info_commande_LoisDeComp()	1052
6.466.1.12 Lecture_base_info_loi()	1052
6.466.1.13 LectureDonneesParticulieres()	1053
6.466.1.14 ListeGrandeurs_particulieres()	1053
6.466.1.15 Module_compressibilite_equivalent()	1053
6.466.1.16 Module_young_equivalent()	1053
6.466.1.17 New_et_Initialise()	1053
6.466.1.18 Nouvelle_loi_identique()	1053

6.466.1.19 TestComplet()	1053
6.467 Référence de la classe Loi_ortho_elas3D	1054
6.467.1 Documentation des fonctions membres	1057
6.467.1.1 Affiche()	1057
6.467.1.2 Calcul_dsigma_deps()	1057
6.467.1.3 Calcul_DsigmaHH_tdt()	1057
6.467.1.4 Calcul_SigmaHH()	1058
6.467.1.5 CalculGrandeurTravail()	1058
6.467.1.6 Ecriture_base_info_loi()	1059
6.467.1.7 Grandeur_particuliere()	1059
6.467.1.8 HsurH0()	1059
6.467.1.9 Info_commande_LoisDeComp()	1059
6.467.1.10 Lecture_base_info_loi()	1059
6.467.1.11 LectureDonneesParticulieres()	1059
6.467.1.12 ListeGrandeurs_particulieres()	1060
6.467.1.13 Module_compressibilite_equivalent()	1060
6.467.1.14 Module_young_equivalent()	1060
6.467.1.15 New_et_Initialise()	1060
6.467.1.16 Nouvelle_loi_identique()	1060
6.467.1.17 TestComplet()	1060
6.468 Référence de la classe Loi_rien1D	1061
6.468.1 Documentation des fonctions membres	1063
6.468.1.1 Affiche()	1063
6.468.1.2 Calcul_DsigmaHH_tdt()	1063
6.468.1.3 Calcul_SigmaHH()	1064
6.468.1.4 CalculGrandeurTravail()	1064
6.468.1.5 Ecriture_base_info_loi()	1064
6.468.1.6 HsurH0()	1065
6.468.1.7 Info_commande_LoisDeComp()	1065
6.468.1.8 Lecture_base_info_loi()	1065
6.468.1.9 LectureDonneesParticulieres()	1065
6.468.1.10 Module_young_equivalent()	1065
6.468.1.11 Nouvelle_loi_identique()	1065
6.468.1.12 TestComplet()	1065
6.469 Référence de la classe Loi_rien2D	1066
6.469.1 Documentation des fonctions membres	1068
6.469.1.1 Affiche()	1068
6.469.1.2 Calcul_DsigmaHH_tdt()	1068
6.469.1.3 Calcul_SigmaHH()	1069
6.469.1.4 Ecriture_base_info_loi()	1069
6.469.1.5 Info_commande_LoisDeComp()	1069
6.469.1.6 Nouvelle_loi_identique()	1069

---

6.469.1.7 TestComplet()	1069
6.470 Référence de la classe Loi_rien2D_C	1070
6.470.1 Documentation des fonctions membres	1072
6.470.1.1 Affiche()	1072
6.470.1.2 Calcul_DsigmaHH_tdt()	1072
6.470.1.3 Calcul_SigmaHH()	1073
6.470.1.4 CalculGrandeurTravail()	1073
6.470.1.5 Ecriture_base_info_loi()	1073
6.470.1.6 HsurH0()	1074
6.470.1.7 Info_commande_LoisDeComp()	1074
6.470.1.8 Lecture_base_info_loi()	1074
6.470.1.9 LectureDonneesParticulieres()	1074
6.470.1.10 Module_young_equivalent()	1074
6.470.1.11 Nouvelle_loi_identique()	1074
6.470.1.12 TestComplet()	1074
6.471 Référence de la classe Loi_rien2D_D	1075
6.471.1 Documentation des fonctions membres	1077
6.471.1.1 Affiche()	1077
6.471.1.2 Calcul_DsigmaHH_tdt()	1077
6.471.1.3 Calcul_SigmaHH()	1078
6.471.1.4 CalculGrandeurTravail()	1078
6.471.1.5 Ecriture_base_info_loi()	1078
6.471.1.6 HsurH0()	1079
6.471.1.7 Info_commande_LoisDeComp()	1079
6.471.1.8 Lecture_base_info_loi()	1079
6.471.1.9 LectureDonneesParticulieres()	1079
6.471.1.10 Module_young_equivalent()	1079
6.471.1.11 Nouvelle_loi_identique()	1079
6.471.1.12 TestComplet()	1079
6.472 Référence de la classe Loi_rien3D	1080
6.472.1 Documentation des fonctions membres	1082
6.472.1.1 Affiche()	1082
6.472.1.2 Calcul_dsigma_deps()	1082
6.472.1.3 Calcul_DsigmaHH_tdt()	1083
6.472.1.4 Calcul_SigmaHH()	1083
6.472.1.5 CalculGrandeurTravail()	1084
6.472.1.6 Ecriture_base_info_loi()	1084
6.472.1.7 HsurH0()	1084
6.472.1.8 Info_commande_LoisDeComp()	1084
6.472.1.9 Lecture_base_info_loi()	1084
6.472.1.10 LectureDonneesParticulieres()	1084
6.472.1.11 Module_young_equivalent()	1085

6.472.1.12 Nouvelle_loi_identique()	1085
6.472.1.13 TestComplet()	1085
6.473 Référence de la classe Loi_Umat	1085
6.473.1 Documentation des fonctions membres	1089
6.473.1.1 Affiche()	1089
6.473.1.2 Calcul_dsigma_deps()	1089
6.473.1.3 Calcul_DsigmaHH_tdt()	1089
6.473.1.4 Calcul_SigmaHH()	1090
6.473.1.5 CalculGrandeurTravail()	1090
6.473.1.6 Ecriture_base_info_loi()	1090
6.473.1.7 HsurH0()	1091
6.473.1.8 Info_commande_LoisDeComp()	1091
6.473.1.9 Lecture_base_info_loi()	1091
6.473.1.10 LectureDonneesParticulieres()	1091
6.473.1.11 Module_young_equivalent()	1091
6.473.1.12 New_et_Initialise()	1091
6.473.1.13 Nouvelle_loi_identique()	1091
6.473.1.14 RepercuteChangeTemperature()	1091
6.473.1.15 TestComplet()	1092
6.474 Référence de la classe LoiAbstraiteGeneral	1093
6.475 Référence de la classe LoiAdditiveEnSigma	1094
6.475.1 Documentation des fonctions membres	1097
6.475.1.1 Activation_donnees()	1097
6.475.1.2 Affiche()	1097
6.475.1.3 Calcul_dsigma_deps()	1097
6.475.1.4 Calcul_DsigmaHH_tdt()	1098
6.475.1.5 Calcul_SigmaHH()	1098
6.475.1.6 CalculGrandeurTravail()	1099
6.475.1.7 Comportement_3D_CP_DP_1D()	1099
6.475.1.8 Ecriture_base_info_loi()	1099
6.475.1.9 Grandeur_particuliere()	1099
6.475.1.10 HsurH0()	1099
6.475.1.11 IndiquePtIntegMecalInterne()	1099
6.475.1.12 Info_commande_LoisDeComp()	1099
6.475.1.13 Lecture_base_info_loi()	1100
6.475.1.14 LectureDonneesParticulieres()	1100
6.475.1.15 ListeGrandeurs_particulieres()	1100
6.475.1.16 Module_compressibilite_equivalent()	1100
6.475.1.17 Module_young_equivalent()	1100
6.475.1.18 New_et_Initialise()	1100
6.475.1.19 Nouvelle_loi_identique()	1100
6.475.1.20 RepercuteChangeTemperature()	1101

---

6.475.1.21 TestComplet()	1101
6.476 Référence de la classe LoiContraintesPlanes	1101
6.476.1 Documentation des fonctions membres	1104
6.476.1.1 Activation_donnees()	1104
6.476.1.2 Affiche()	1104
6.476.1.3 Calcul_dsigma_deps()	1104
6.476.1.4 Calcul_DsigmaHH_tdt()	1104
6.476.1.5 Calcul_SigmaHH()	1105
6.476.1.6 CalculGrandeurTravail()	1105
6.476.1.7 Contraintes_planes_de_3D()	1105
6.476.1.8 Ecriture_base_info_loi()	1106
6.476.1.9 Eps33BH()	1106
6.476.1.10 Grandeur_particuliere()	1106
6.476.1.11 HsurH0()	1106
6.476.1.12 IndiquePtIntegMecalInterne()	1106
6.476.1.13 Info_commande_LoisDeComp()	1106
6.476.1.14 Lecture_base_info_loi()	1106
6.476.1.15 LectureDonneesParticulieres()	1107
6.476.1.16 ListeGrandeurs_particulieres()	1107
6.476.1.17 Module_compressibilite_equivalent()	1107
6.476.1.18 Module_young_equivalent()	1107
6.476.1.19 New_et_Initialise()	1107
6.476.1.20 Nouvelle_loi_identique()	1107
6.476.1.21 RepercuteChangeTemperature()	1107
6.476.1.22 TestComplet()	1107
6.477 Référence de la classe LoiContraintesPlanesDouble	1108
6.477.1 Documentation des fonctions membres	1111
6.477.1.1 Activation_donnees()	1111
6.477.1.2 Affiche()	1111
6.477.1.3 BsurB0()	1111
6.477.1.4 Calcul_dsigma_deps()	1111
6.477.1.5 Calcul_DsigmaHH_tdt()	1112
6.477.1.6 Calcul_SigmaHH()	1112
6.477.1.7 CalculGrandeurTravail()	1113
6.477.1.8 d_BsurB0()	1113
6.477.1.9 d_HsurH0()	1113
6.477.1.10 Ecriture_base_info_loi()	1113
6.477.1.11 Eps22BH()	1113
6.477.1.12 Eps33BH()	1113
6.477.1.13 Grandeur_particuliere()	1113
6.477.1.14 HsurH0()	1114
6.477.1.15 IndiquePtIntegMecalInterne()	1114

6.477.1.16	Info_commande_LoisDeComp()	1114
6.477.1.17	Lecture_base_info_loi()	1114
6.477.1.18	LectureDonneesParticulieres()	1114
6.477.1.19	ListeGrandeurs_particulieres()	1114
6.477.1.20	Module_compressibilite_equivalent()	1114
6.477.1.21	Module_young_equivalent()	1115
6.477.1.22	New_et_Initialise()	1115
6.477.1.23	Nouvelle_loi_identique()	1115
6.477.1.24	RepercuteChangeTemperature()	1115
6.477.1.25	TestComplet()	1115
6.478	Référence de la classe LoiCritere	1116
6.478.1	Documentation des fonctions membres	1121
6.478.1.1	Activation_donnees()	1121
6.478.1.2	Activation_stockage_grandeurs_quelconques()	1121
6.478.1.3	Affiche()	1121
6.478.1.4	Calcul_dsigma_deps()	1121
6.478.1.5	Calcul_DsigmaHH_tdt()	1121
6.478.1.6	Calcul_SigmaHH()	1122
6.478.1.7	CalculGrandeurTravail()	1122
6.478.1.8	Comportement_3D_CP_DP_1D()	1123
6.478.1.9	Ecriture_base_info_loi()	1123
6.478.1.10	Grandeur_particuliere()	1123
6.478.1.11	HsurH0()	1123
6.478.1.12	IndiquePtIntegMecalInterne()	1123
6.478.1.13	Info_commande_LoisDeComp()	1123
6.478.1.14	Insertion_conteneur_dans_save_result()	1123
6.478.1.15	Lecture_base_info_loi()	1124
6.478.1.16	LectureDonneesParticulieres()	1124
6.478.1.17	ListeGrandeurs_particulieres()	1124
6.478.1.18	Module_compressibilite_equivalent()	1124
6.478.1.19	Module_young_equivalent()	1124
6.478.1.20	New_et_Initialise()	1124
6.478.1.21	Nouvelle_loi_identique()	1124
6.478.1.22	RepercuteChangeTemperature()	1125
6.478.1.23	TestComplet()	1125
6.479	Référence de la classe LoiDeformationsPlanes	1125
6.479.1	Documentation des fonctions membres	1127
6.479.1.1	Activation_donnees()	1128
6.479.1.2	Affiche()	1128
6.479.1.3	Calcul_dsigma_deps()	1128
6.479.1.4	Calcul_DsigmaHH_tdt()	1128
6.479.1.5	Calcul_SigmaHH()	1129

6.479.1.6 CalculGrandeurTravail()	1129
6.479.1.7 Ecriture_base_info_loi()	1129
6.479.1.8 Grandeur_particuliere()	1130
6.479.1.9 HsurH0()	1130
6.479.1.10 IndiquePtIntegMecalInterne()	1130
6.479.1.11 Info_commande_LoisDeComp()	1130
6.479.1.12 Lecture_base_info_loi()	1130
6.479.1.13 LectureDonneesParticulieres()	1130
6.479.1.14 ListeGrandeurs_particulieres()	1130
6.479.1.15 Module_compressibilite_equivalent()	1131
6.479.1.16 Module_young_equivalent()	1131
6.479.1.17 New_et_Initialise()	1131
6.479.1.18 Nouvelle_loi_identique()	1131
6.479.1.19 RepercuteChangeTemperature()	1131
6.479.1.20 TestComplet()	1131
6.480 Référence de la classe LoiDesMelangesEnSigma	1132
6.480.1 Documentation des fonctions membres	1135
6.480.1.1 Activation_donnees()	1135
6.480.1.2 Affiche()	1135
6.480.1.3 Calcul_dsigma_deps()	1135
6.480.1.4 Calcul_DsigmaHH_tdt()	1135
6.480.1.5 Calcul_SigmaHH()	1136
6.480.1.6 CalculGrandeurTravail()	1136
6.480.1.7 Comportement_3D_CP_DP_1D()	1137
6.480.1.8 Ecriture_base_info_loi()	1137
6.480.1.9 Grandeur_particuliere()	1137
6.480.1.10 HsurH0()	1137
6.480.1.11 IndiquePtIntegMecalInterne()	1137
6.480.1.12 Info_commande_LoisDeComp()	1137
6.480.1.13 Lecture_base_info_loi()	1137
6.480.1.14 LectureDonneesParticulieres()	1137
6.480.1.15 ListeGrandeurs_particulieres()	1138
6.480.1.16 Module_compressibilite_equivalent()	1138
6.480.1.17 Module_young_equivalent()	1138
6.480.1.18 New_et_Initialise()	1138
6.480.1.19 Nouvelle_loi_identique()	1138
6.480.1.20 RepercuteChangeTemperature()	1138
6.480.1.21 TestComplet()	1138
6.481 Référence de la classe Maheo_hyper	1139
6.481.1 Documentation des fonctions membres	1141
6.481.1.1 Affiche()	1141
6.481.1.2 Calcul_dsigma_deps()	1142

---

6.481.1.3	Calcul_DsigmaHH_tdt()	1142
6.481.1.4	Calcul_SigmaHH()	1142
6.481.1.5	CalculGrandeurTravail()	1143
6.481.1.6	Ecriture_base_info_loi()	1143
6.481.1.7	HsurH0()	1143
6.481.1.8	Info_commande_LoisDeComp()	1143
6.481.1.9	Lecture_base_info_loi()	1144
6.481.1.10	LectureDonneesParticulieres()	1144
6.481.1.11	Module_young_equivalent()	1144
6.481.1.12	Nouvelle_loi_identique()	1144
6.481.1.13	TestCompleto()	1144
6.482	Référence de la classe Mail_initiale_geomview	1145
6.482.1	Documentation des fonctions membres	1145
6.482.1.1	ExeOrdre()	1146
6.483	Référence de la classe Mail_initiale_Gid	1146
6.483.1	Documentation des fonctions membres	1147
6.483.1.1	ChoixOrdre()	1147
6.483.1.2	Ecriture_parametres_OrdreVisu()	1147
6.483.1.3	ExeOrdre()	1148
6.483.1.4	Initialisation()	1148
6.483.1.5	Lecture_parametres_OrdreVisu()	1148
6.484	Référence de la classe Mail_initiale_Gmsh	1149
6.484.1	Documentation des fonctions membres	1150
6.484.1.1	ChoixOrdre()	1150
6.484.1.2	Ecriture_parametres_OrdreVisu()	1150
6.484.1.3	ExeOrdre()	1150
6.484.1.4	Initialisation()	1150
6.484.1.5	Lecture_parametres_OrdreVisu()	1151
6.485	Référence de la classe Mail_initiale_vrml	1151
6.485.1	Documentation des fonctions membres	1152
6.485.1.1	ChoixOrdre()	1152
6.485.1.2	Ecriture_parametres_OrdreVisu()	1152
6.485.1.3	ExeOrdre()	1152
6.485.1.4	Lecture_parametres_OrdreVisu()	1153
6.486	Référence de la classe Maillage	1153
6.486.1	Description détaillée	1158
6.486.2	Documentation des fonctions membres	1158
6.486.2.1	Calcul_indice()	1158
6.486.2.2	CreeElemFront()	1158
6.486.2.3	Lecture_info_1element()	1158
6.487	Référence du modèle de la classe Map_io< T >	1158
6.487.1	Description détaillée	1159



---

6.488 Référence de la classe Mat_abstraite	1160
6.489 Référence de la classe Mat_creuse_CompCol	1162
6.489.1 Documentation des fonctions membres	1163
6.489.1.1 Affiche()	1163
6.489.1.2 Affiche1()	1164
6.489.1.3 Affiche2()	1164
6.489.1.4 Colonne()	1164
6.489.1.5 ColonneSpe()	1164
6.489.1.6 Existe()	1164
6.489.1.7 Initialise()	1164
6.489.1.8 Libere()	1164
6.489.1.9 Ligne()	1165
6.489.1.10 Ligne_set()	1165
6.489.1.11 LigneSpe()	1165
6.489.1.12 MetValColonne()	1165
6.489.1.13 MetValLigne()	1165
6.489.1.14 Nb_colonne()	1165
6.489.1.15 Nb_ligne()	1165
6.489.1.16 NouvelElement()	1165
6.489.1.17 operator>() [1/2]	1165
6.489.1.18 operator>() [2/2]	1166
6.489.1.19 operator*=(())	1166
6.489.1.20 operator+=(())	1166
6.489.1.21 operator-=(())	1166
6.489.1.22 operator=()	1166
6.489.1.23 operator==(())	1166
6.489.1.24 Place()	1166
6.489.1.25 Preparation_resol()	1166
6.489.1.26 Prod_Ligne_vec()	1167
6.489.1.27 Prod_mat_vec() [1/2]	1167
6.489.1.28 Prod_mat_vec() [2/2]	1167
6.489.1.29 Prod_vec_col()	1167
6.489.1.30 Prod_vec_mat() [1/2]	1167
6.489.1.31 Prod_vec_mat() [2/2]	1167
6.489.1.32 RemplaceColonneSpe()	1167
6.489.1.33 RemplaceLigneSpe()	1167
6.489.1.34 Resol_syst() [1/2]	1168
6.489.1.35 Resol_syst() [2/2]	1168
6.489.1.36 Resol_systID() [1/2]	1168
6.489.1.37 Resol_systID() [2/2]	1168
6.489.1.38 Resol_systID_2()	1168
6.489.1.39 Simple_Resol_syst() [1/2]	1168

6.489.1.40 Simple_Resol_syst() [2/2]	1169
6.489.1.41 Simple_Resol_systID() [1/2]	1169
6.489.1.42 Simple_Resol_systID() [2/2]	1169
6.489.1.43 Simple_Resol_systID_2()	1169
6.489.1.44 Symetrie()	1169
6.489.1.45 trans_mult()	1169
6.489.1.46 Transfert_vers_mat()	1170
6.489.1.47 vectT_mat_vec()	1170
6.490 Référence de la classe Mat_pleine	1170
6.490.1 Documentation des fonctions membres	1172
6.490.1.1 Affiche()	1172
6.490.1.2 Colonne()	1172
6.490.1.3 ColonneSpe()	1172
6.490.1.4 Existe()	1173
6.490.1.5 Initialise()	1173
6.490.1.6 Libere()	1173
6.490.1.7 Ligne()	1173
6.490.1.8 Ligne_set()	1173
6.490.1.9 LigneSpe()	1173
6.490.1.10 MetValColonne()	1173
6.490.1.11 MetValLigne()	1173
6.490.1.12 Nb_colonne()	1174
6.490.1.13 Nb_ligne()	1174
6.490.1.14 NouvelElement()	1174
6.490.1.15 operator>() [1/2]	1174
6.490.1.16 operator>() [2/2]	1174
6.490.1.17 operator*=(())	1174
6.490.1.18 operator+=(())	1174
6.490.1.19 operator-=(())	1174
6.490.1.20 operator=()	1174
6.490.1.21 operator==(())	1175
6.490.1.22 Place()	1175
6.490.1.23 Preparation_resol()	1175
6.490.1.24 Prod_Ligne_vec()	1175
6.490.1.25 Prod_mat_vec() [1/2]	1175
6.490.1.26 Prod_mat_vec() [2/2]	1175
6.490.1.27 Prod_vec_col()	1175
6.490.1.28 Prod_vec_mat() [1/2]	1175
6.490.1.29 Prod_vec_mat() [2/2]	1176
6.490.1.30 RemplaceColonneSpe()	1176
6.490.1.31 RemplaceLigneSpe()	1176
6.490.1.32 Resol_syst() [1/2]	1176

6.490.1.33 Resol_syst() [2/2]	1176
6.490.1.34 Resol_systID() [1/2]	1176
6.490.1.35 Resol_systID() [2/2]	1176
6.490.1.36 Resol_systID_2()	1177
6.490.1.37 Simple_Resol_syst() [1/2]	1177
6.490.1.38 Simple_Resol_syst() [2/2]	1177
6.490.1.39 Simple_Resol_systID() [1/2]	1177
6.490.1.40 Simple_Resol_systID() [2/2]	1177
6.490.1.41 Simple_Resol_systID_2()	1177
6.490.1.42 Symetrie()	1178
6.490.1.43 Transfert_vers_mat()	1178
6.490.1.44 vectT_mat_vec()	1178
6.491 Référence de la classe MatBand	1178
6.491.1 Documentation des fonctions membres	1180
6.491.1.1 Affiche()	1180
6.491.1.2 Affiche1()	1180
6.491.1.3 Affiche2()	1181
6.491.1.4 Colonne()	1181
6.491.1.5 ColonneSpe()	1181
6.491.1.6 Existe()	1181
6.491.1.7 Initialise()	1181
6.491.1.8 Libere()	1181
6.491.1.9 Ligne()	1181
6.491.1.10 Ligne_set()	1182
6.491.1.11 LigneSpe()	1182
6.491.1.12 MetValColonne()	1182
6.491.1.13 MetValLigne()	1182
6.491.1.14 Nb_colonne()	1182
6.491.1.15 Nb_ligne()	1182
6.491.1.16 NouvelElement()	1182
6.491.1.17 operator>() [1/2]	1182
6.491.1.18 operator>() [2/2]	1183
6.491.1.19 operator*=(())	1183
6.491.1.20 operator+=(())	1183
6.491.1.21 operator-=(())	1183
6.491.1.22 operator=()	1183
6.491.1.23 operator==(())	1183
6.491.1.24 Place()	1183
6.491.1.25 Preparation_resol()	1183
6.491.1.26 Prod_Ligne_vec()	1183
6.491.1.27 Prod_mat_vec() [1/2]	1184
6.491.1.28 Prod_mat_vec() [2/2]	1184

6.491.1.29 Prod_vec_col()	1184
6.491.1.30 Prod_vec_mat() [1/2]	1184
6.491.1.31 Prod_vec_mat() [2/2]	1184
6.491.1.32 RemplaceColonneSpe()	1184
6.491.1.33 RemplaceLigneSpe()	1184
6.491.1.34 Resol_syst() [1/2]	1185
6.491.1.35 Resol_syst() [2/2]	1185
6.491.1.36 Resol_systID() [1/2]	1185
6.491.1.37 Resol_systID() [2/2]	1185
6.491.1.38 Resol_systID_2()	1185
6.491.1.39 Simple_Resol_syst() [1/2]	1185
6.491.1.40 Simple_Resol_syst() [2/2]	1186
6.491.1.41 Simple_Resol_systID() [1/2]	1186
6.491.1.42 Simple_Resol_systID() [2/2]	1186
6.491.1.43 Simple_Resol_systID_2()	1186
6.491.1.44 Symetrie()	1186
6.491.1.45 Transfert_vers_mat()	1186
6.491.1.46 vectT_mat_vec()	1186
6.492 Référence de la classe MatDiag	1187
6.492.1 Documentation des fonctions membres	1189
6.492.1.1 Affiche()	1189
6.492.1.2 Affiche1()	1189
6.492.1.3 Affiche2()	1189
6.492.1.4 Colonne()	1189
6.492.1.5 ColonneSpe()	1189
6.492.1.6 Existe()	1190
6.492.1.7 Initialise()	1190
6.492.1.8 Libere()	1190
6.492.1.9 Ligne()	1190
6.492.1.10 Ligne_set()	1190
6.492.1.11 LigneSpe()	1190
6.492.1.12 MetValColonne()	1190
6.492.1.13 MetValLigne()	1190
6.492.1.14 Nb_colonne()	1191
6.492.1.15 Nb_ligne()	1191
6.492.1.16 NouvelElement()	1191
6.492.1.17 operator>() [1/2]	1191
6.492.1.18 operator>() [2/2]	1191
6.492.1.19 operator*=(())	1191
6.492.1.20 operator+=(())	1191
6.492.1.21 operator-=(())	1191
6.492.1.22 operator=()	1191

6.492.1.23	operator==( )	1192
6.492.1.24	Place( )	1192
6.492.1.25	Preparation_resol( )	1192
6.492.1.26	Prod_Ligne_vec( )	1192
6.492.1.27	Prod_mat_vec( ) [1/2]	1192
6.492.1.28	Prod_mat_vec( ) [2/2]	1192
6.492.1.29	Prod_vec_col( )	1192
6.492.1.30	Prod_vec_mat( ) [1/2]	1192
6.492.1.31	Prod_vec_mat( ) [2/2]	1193
6.492.1.32	RemplaceColonneSpe( )	1193
6.492.1.33	RemplaceLigneSpe( )	1193
6.492.1.34	Resol_syst( ) [1/2]	1193
6.492.1.35	Resol_syst( ) [2/2]	1193
6.492.1.36	Resol_systID( ) [1/2]	1193
6.492.1.37	Resol_systID( ) [2/2]	1193
6.492.1.38	Resol_systID_2( )	1194
6.492.1.39	Simple_Resol_syst( ) [1/2]	1194
6.492.1.40	Simple_Resol_syst( ) [2/2]	1194
6.492.1.41	Simple_Resol_systID( ) [1/2]	1194
6.492.1.42	Simple_Resol_systID( ) [2/2]	1194
6.492.1.43	Simple_Resol_systID_2( )	1194
6.492.1.44	Symetrie( )	1195
6.492.1.45	Transfert_vers_mat( )	1195
6.492.1.46	vectT_mat_vec( )	1195
6.493	Référence de la classe ElemMeca::MatGeomInit	1195
6.494	Référence de la classe ElemThermi::MatGeomInit	1196
6.495	Référence de la classe MathUtil2	1196
6.495.1	Documentation des fonctions membres	1197
6.495.1.1	ChBase( )	1197
6.496	Référence de la classe MatLapack	1198
6.496.1	Documentation des fonctions membres	1200
6.496.1.1	Affiche( )	1200
6.496.1.2	Affiche1( )	1200
6.496.1.3	Affiche2( )	1201
6.496.1.4	Change_Choix_resolution( )	1201
6.496.1.5	Colonne( )	1201
6.496.1.6	ColonneSpe( )	1201
6.496.1.7	Existe( )	1201
6.496.1.8	Initialise( )	1201
6.496.1.9	Libere( )	1201
6.496.1.10	Ligne( )	1201
6.496.1.11	Ligne_set( )	1202

6.496.1.12 LigneSpe()	1202
6.496.1.13 MetValColonne()	1202
6.496.1.14 MetValLigne()	1202
6.496.1.15 Nb_colonne()	1202
6.496.1.16 Nb_ligne()	1202
6.496.1.17 NouvelElement()	1202
6.496.1.18 operator>() [1/2]	1202
6.496.1.19 operator>() [2/2]	1203
6.496.1.20 operator*=(())	1203
6.496.1.21 operator+=(())	1203
6.496.1.22 operator-=(())	1203
6.496.1.23 operator=()	1203
6.496.1.24 operator==(())	1203
6.496.1.25 Place()	1203
6.496.1.26 Preparation_resol()	1203
6.496.1.27 Prod_Ligne_vec()	1203
6.496.1.28 Prod_mat_vec() [1/2]	1204
6.496.1.29 Prod_mat_vec() [2/2]	1204
6.496.1.30 Prod_vec_col()	1204
6.496.1.31 Prod_vec_mat() [1/2]	1204
6.496.1.32 Prod_vec_mat() [2/2]	1204
6.496.1.33 RemplaceColonneSpe()	1204
6.496.1.34 RemplaceLigneSpe()	1204
6.496.1.35 Resol_syst() [1/2]	1205
6.496.1.36 Resol_syst() [2/2]	1205
6.496.1.37 Resol_systID() [1/2]	1205
6.496.1.38 Resol_systID() [2/2]	1205
6.496.1.39 Resol_systID_2()	1205
6.496.1.40 Simple_Resol_syst() [1/2]	1205
6.496.1.41 Simple_Resol_syst() [2/2]	1206
6.496.1.42 Simple_Resol_systID() [1/2]	1206
6.496.1.43 Simple_Resol_systID() [2/2]	1206
6.496.1.44 Simple_Resol_systID_2()	1206
6.496.1.45 Symetrie()	1206
6.496.1.46 Transfert_vers_mat()	1206
6.496.1.47 vectT_mat_vec()	1206
6.497 Référence de la structure Matrix_	1207
6.498 Référence de la classe Met_abstraite	1207
6.498.1 Documentation des fonctions membres	1213
6.498.1.1 Cal_explicit_t()	1213
6.498.1.2 Cal_explicit_tdt()	1213
6.498.1.3 Cal_implicit()	1213

---

6.498.1.4	Calcul_M0()	1213
6.498.1.5	operator=()	1213
6.499	Référence de la classe Met_biellette	1214
6.499.1	Documentation des fonctions membres	1216
6.499.1.1	Calcul_giB_0()	1216
6.499.1.2	Calcul_giB_t()	1216
6.499.1.3	Calcul_giB_tdt()	1216
6.499.1.4	Djacobien_t()	1216
6.499.1.5	Djacobien_tdt()	1216
6.499.1.6	operator=()	1217
6.500	Référence de la classe Met_BielletteC1	1217
6.500.1	Documentation des fonctions membres	1219
6.500.1.1	Calcul_giB_0()	1219
6.500.1.2	Calcul_giB_t()	1219
6.500.1.3	Calcul_giB_tdt()	1219
6.500.1.4	Djacobien_t()	1219
6.500.1.5	Djacobien_tdt()	1219
6.500.1.6	operator=()	1220
6.501	Référence de la classe Met_ElemPoint	1220
6.501.1	Documentation des fonctions membres	1222
6.501.1.1	Calcul_giB_0()	1222
6.501.1.2	Calcul_giB_t()	1222
6.501.1.3	Calcul_giB_tdt()	1222
6.501.1.4	Calcul_grandeurs_Umat()	1222
6.501.1.5	operator=()	1223
6.502	Référence de la classe Met_PiPoCo	1223
6.502.1	Documentation des fonctions membres	1226
6.502.1.1	Calcul_giB_0()	1226
6.502.1.2	Calcul_giB_t()	1226
6.502.1.3	Calcul_giB_tdt()	1226
6.502.1.4	Calcul_M0()	1227
6.502.1.5	Calcul_Mt()	1227
6.502.1.6	Calcul_Mtdt()	1227
6.502.1.7	D_giB_t()	1227
6.502.1.8	D_giB_tdt()	1227
6.502.1.9	operator=()	1227
6.502.1.10	PlusInitVariables()	1228
6.503	Référence de la classe Met_Pout2D	1228
6.503.1	Documentation des fonctions membres	1230
6.503.1.1	Calcul_giB_0()	1230
6.503.1.2	Calcul_giB_t()	1230
6.503.1.3	Calcul_giB_tdt()	1230

6.503.1.4 Calcul_N_0()	1231
6.503.1.5 Calcul_N_t()	1231
6.503.1.6 Calcul_N_tdt()	1231
6.503.1.7 courbure_0()	1231
6.503.1.8 courbure_t()	1231
6.503.1.9 courbure_tdt()	1231
6.503.1.10 D_giB_t()	1231
6.503.1.11 D_giB_tdt()	1231
6.503.1.12 Dcourbure_t()	1232
6.503.1.13 Dcourbure_tdt()	1232
6.503.1.14 operator=() [1/2]	1232
6.503.1.15 operator=() [2/2]	1232
6.504 Référence de la classe Met_Sfe1	1232
6.504.1 Documentation des fonctions membres	1237
6.504.1.1 Affiche()	1237
6.504.1.2 Cal_pourMatMass()	1238
6.504.1.3 Calcul_giB_0()	1238
6.504.1.4 Calcul_giB_t()	1238
6.504.1.5 Calcul_giB_tdt()	1238
6.504.1.6 Calcul_M0()	1238
6.504.1.7 Calcul_Mt()	1238
6.504.1.8 Calcul_Mtdt()	1239
6.504.1.9 D_giB_t()	1239
6.504.1.10 D_giB_tdt()	1239
6.504.1.11 operator=()	1239
6.504.1.12 PlusInitVariables()	1239
6.505 Référence de la classe Met_triaMemb	1240
6.506 Référence de la classe MetAxisymetrique2D	1242
6.506.1 Documentation des fonctions membres	1244
6.506.1.1 Calcul_d_Mtdt()	1244
6.506.1.2 Calcul_d_V_moyt() [1/2]	1244
6.506.1.3 Calcul_d_V_moyt() [2/2]	1245
6.506.1.4 Calcul_d_V_moytdt() [1/2]	1245
6.506.1.5 Calcul_d_V_moytdt() [2/2]	1245
6.506.1.6 Calcul_d_Vt() [1/2]	1245
6.506.1.7 Calcul_d_Vt() [2/2]	1245
6.506.1.8 Calcul_d_Vtdt() [1/2]	1245
6.506.1.9 Calcul_d_Vtdt() [2/2]	1245
6.506.1.10 Calcul_giB_0()	1246
6.506.1.11 Calcul_giB_t()	1246
6.506.1.12 Calcul_giB_tdt()	1246
6.506.1.13 Calcul_gradVBB_moyen_t()	1246



6.506.1.14 Calcul_gradVBB_moyen_tdt()	1246
6.506.1.15 Calcul_gradVBB_t()	1246
6.506.1.16 Calcul_gradVBB_tdt()	1247
6.506.1.17 D_giB_t()	1247
6.506.1.18 D_giB_tdt()	1247
6.506.1.19 DgradVBB_t()	1247
6.506.1.20 DgradVBB_tdt()	1247
6.506.1.21 DgradVmoyBB_t()	1247
6.506.1.22 DgradVmoyBB_tdt()	1247
6.506.1.23 operator=()	1247
6.507 Référence de la classe MetAxisymetrique3D	1248
6.507.1 Documentation des fonctions membres	1250
6.507.1.1 Calcul_giB_0()	1250
6.507.1.2 Calcul_giB_t()	1250
6.507.1.3 Calcul_giB_tdt()	1250
6.507.1.4 Calcul_gradVBB_moyen_t()	1251
6.507.1.5 Calcul_gradVBB_moyen_tdt()	1251
6.507.1.6 Calcul_gradVBB_t()	1251
6.507.1.7 Calcul_gradVBB_tdt()	1251
6.507.1.8 D_giB_t()	1251
6.507.1.9 D_giB_tdt()	1251
6.507.1.10 DgradVBB_t()	1251
6.507.1.11 DgradVBB_tdt()	1252
6.507.1.12 DgradVmoyBB_t()	1252
6.507.1.13 DgradVmoyBB_tdt()	1252
6.507.1.14 operator=()	1252
6.508 Référence de la classe MooneyRivlin1D	1252
6.508.1 Documentation des fonctions membres	1254
6.508.1.1 Affiche()	1254
6.508.1.2 Calcul_DsigmaHH_tdt()	1254
6.508.1.3 Calcul_SigmaHH()	1255
6.508.1.4 CalculGrandeurTravail()	1255
6.508.1.5 Ecriture_base_info_loi()	1256
6.508.1.6 HsurH0()	1256
6.508.1.7 Info_commande_LoisDeComp()	1256
6.508.1.8 Lecture_base_info_loi()	1256
6.508.1.9 LectureDonneesParticulieres()	1256
6.508.1.10 Module_young_equivalent()	1256
6.508.1.11 Nouvelle_loi_identique()	1256
6.508.1.12 TestCompleto()	1257
6.509 Référence de la classe MooneyRivlin3D	1257
6.509.1 Documentation des fonctions membres	1259

6.509.1.1 Affiche()	1260
6.509.1.2 Calcul_dsigma_deps()	1260
6.509.1.3 Calcul_DsigmaHH_tdt()	1260
6.509.1.4 Calcul_SigmaHH()	1261
6.509.1.5 CalculGrandeurTravail()	1261
6.509.1.6 Ecriture_base_info_loi()	1261
6.509.1.7 HsurH0()	1261
6.509.1.8 Info_commande_LoisDeComp()	1262
6.509.1.9 Lecture_base_info_loi()	1262
6.509.1.10 LectureDonneesParticulieres()	1262
6.509.1.11 Module_young_equivalent()	1262
6.509.1.12 Nouvelle_loi_identique()	1262
6.509.1.13 TestComplet()	1262
6.510 Référence de la classe MotCle	1262
6.510.1 Description détaillée	1263
6.511 Référence du modèle de la classe MV_ColMat< TYPE >	1263
6.512 Référence de la classe MV_VecIndex	1263
6.513 Référence de la classe MV_Vecteur	1264
6.513.1 Description détaillée	1265
6.514 Référence du modèle de la classe MV_Vector< TYPE >	1265
6.515 Référence de la structure MV_Vector_	1266
6.516 Référence de la classe MvtSolide	1266
6.516.1 Description détaillée	1266
6.517 Référence de la classe Nb_assemb	1267
6.517.1 Description détaillée	1267
6.518 Référence de la classe Maillage::NBelemEtArete	1267
6.519 Référence de la classe Maillage::NBelemEtFace	1268
6.520 Référence de la classe Maillage::NBelemEtptInteg	1268
6.521 Référence de la classe Maillage::NBelemFAEtptInteg	1268
6.522 Référence de la classe Noeud	1269
6.522.1 Description détaillée	1272
6.523 Référence de la classe Maillage::Noeud_degre	1273
6.524 Référence de la classe Biel_axi::NombresConstruire	1273
6.525 Référence de la classe Biel_axiQ::NombresConstruire	1274
6.526 Référence de la classe Biellette::NombresConstruire	1274
6.527 Référence de la classe BielletteC1::NombresConstruire	1274
6.528 Référence de la classe BielletteQ::NombresConstruire	1275
6.529 Référence de la classe BielletteThermi::NombresConstruire	1275
6.530 Référence de la classe ElemPoint::NombresConstruire	1275
6.531 Référence de la classe HexaMemb::NombresConstruire	1276
6.532 Référence de la classe PentaMemb::NombresConstruire	1277
6.533 Référence de la classe QuadAxiMemb::NombresConstruire	1278

---

6.534	Référence de la classe <code>QuadraMemb::NombresConstruire</code> . . . . .	1279
6.535	Référence de la classe <code>SfeMembT::NombresConstruire</code> . . . . .	1280
6.536	Référence de la classe <code>TetraMemb::NombresConstruire</code> . . . . .	1281
6.537	Référence de la classe <code>TriaAxiMemb::NombresConstruire</code> . . . . .	1282
6.538	Référence de la classe <code>TriaMemb::NombresConstruire</code> . . . . .	1283
6.539	Référence de la classe <code>Hexa::NombresConstruireHexa</code> . . . . .	1284
6.540	Référence de la classe <code>HexaLMemb::NombresConstruireHexa</code> . . . . .	1285
6.541	Référence de la classe <code>Hexa_cm1pti::NombresConstruireHexa_cm1pti</code> . . . . .	1286
6.542	Référence de la classe <code>Hexa_cm27pti::NombresConstruireHexa_cm27pti</code> . . . . .	1287
6.543	Référence de la classe <code>Hexa_cm64pti::NombresConstruireHexa_cm64pti</code> . . . . .	1288
6.544	Référence de la classe <code>HexaQ::NombresConstruireHexaQ</code> . . . . .	1289
6.545	Référence de la classe <code>HexaQ_cm1pti::NombresConstruireHexaQ_cm1pti</code> . . . . .	1290
6.546	Référence de la classe <code>HexaQ_cm27pti::NombresConstruireHexaQ_cm27pti</code> . . . . .	1291
6.547	Référence de la classe <code>HexaQ_cm64pti::NombresConstruireHexaQ_cm64pti</code> . . . . .	1292
6.548	Référence de la classe <code>HexaQComp::NombresConstruireHexaQComp</code> . . . . .	1293
6.549	Référence de la classe <code>HexaQComp_cm1pti::NombresConstruireHexaQComp_cm1pti</code> . . . . .	1294
6.550	Référence de la classe <code>HexaQComp_cm27pti::NombresConstruireHexaQComp_cm27pti</code> . . . . .	1295
6.551	Référence de la classe <code>HexaQComp_cm64pti::NombresConstruireHexaQComp_cm64pti</code> . . . . .	1296
6.552	Référence de la classe <code>PentaL::NombresConstruirePentaL</code> . . . . .	1297
6.553	Référence de la classe <code>PentaL_cm1pti::NombresConstruirePentaL_cm1pti</code> . . . . .	1298
6.554	Référence de la classe <code>PentaL_cm2pti::NombresConstruirePentaL_cm2pti</code> . . . . .	1299
6.555	Référence de la classe <code>PentaL_cm6pti::NombresConstruirePentaL_cm6pti</code> . . . . .	1300
6.556	Référence de la classe <code>PentaQ::NombresConstruirePentaQ</code> . . . . .	1301
6.557	Référence de la classe <code>PentaQ_cm12pti::NombresConstruirePentaQ_cm12pti</code> . . . . .	1302
6.558	Référence de la classe <code>PentaQ_cm18pti::NombresConstruirePentaQ_cm18pti</code> . . . . .	1303
6.559	Référence de la classe <code>PentaQ_cm3pti::NombresConstruirePentaQ_cm3pti</code> . . . . .	1304
6.560	Référence de la classe <code>PentaQ_cm9pti::NombresConstruirePentaQ_cm9pti</code> . . . . .	1305
6.561	Référence de la classe <code>PentaQComp::NombresConstruirePentaQComp</code> . . . . .	1306
6.562	Référence de la classe <code>PentaQComp_cm12pti::NombresConstruirePentaQComp_cm12pti</code> . . . . .	1307
6.563	Référence de la classe <code>PentaQComp_cm18pti::NombresConstruirePentaQComp_cm18pti</code> . . . . .	1308
6.564	Référence de la classe <code>PentaQComp_cm9pti::NombresConstruirePentaQComp_cm9pti</code> . . . . .	1309
6.565	Référence de la classe <code>PoutSfe3::NombresConstruirePoutSfe3</code> . . . . .	1310
6.566	Référence de la classe <code>Quad::NombresConstruireQuad</code> . . . . .	1311
6.567	Référence de la classe <code>Quad_cm1pti::NombresConstruireQuad_cm1pti</code> . . . . .	1312
6.568	Référence de la classe <code>QuadAxiCCom::NombresConstruireQuadAxiCCom</code> . . . . .	1313
6.569	Référence de la classe <code>QuadAxiCCom_cm9pti::NombresConstruireQuadAxiCCom_cm9pti</code> . . . . .	1314
6.570	Référence de la classe <code>QuadAxiL1::NombresConstruireQuadAxiL1</code> . . . . .	1315
6.571	Référence de la classe <code>QuadAxiL1_cm1pti::NombresConstruireQuadAxiL1_cm1pti</code> . . . . .	1316
6.572	Référence de la classe <code>QuadAxiQ::NombresConstruireQuadAxiQ</code> . . . . .	1317
6.573	Référence de la classe <code>QuadAxiQComp::NombresConstruireQuadAxiQComp</code> . . . . .	1318
6.574	Référence de la classe <code>QuadAxiQComp_cm4pti::NombresConstruireQuadAxiQComp_cm4pti</code> . . . . .	1319
6.575	Référence de la classe <code>QuadCCom::NombresConstruireQuadCCom</code> . . . . .	1320

6.576	Référence de la classe QuadCCom_cm9pti::NombresConstruireQuadCCom_cm9pti	1321
6.577	Référence de la classe QuadQ::NombresConstruireQuadQ	1322
6.578	Référence de la classe QuadQCom::NombresConstruireQuadQCom	1323
6.579	Référence de la classe QuadQCom_cm4pti::NombresConstruireQuadQCom_cm4pti	1324
6.580	Référence de la classe Tetra::NombresConstruireTetra	1325
6.581	Référence de la classe TetraQ::NombresConstruireTetraQ	1326
6.582	Référence de la classe TetraQ_15pti::NombresConstruireTetraQ_15pti	1327
6.583	Référence de la classe TetraQ_cm1pti::NombresConstruireTetraQ_cm1pti	1328
6.584	Référence de la classe TriaAxiL1::NombresConstruireTriaAxiL1	1329
6.585	Référence de la classe TriaAxiQ3::NombresConstruireTriaAxiQ3	1330
6.586	Référence de la classe TriaAxiQ3_cm1pti::NombresConstruireTriaAxiQ3_cm1pti	1331
6.587	Référence de la classe TriaAxiQ3_cmpti1003::NombresConstruireTriaAxiQ3_cmpti1003	1332
6.588	Référence de la classe TriaCub::NombresConstruireTriaCub	1333
6.589	Référence de la classe TriaCub_cm4pti::NombresConstruireTriaCub_cm4pti	1334
6.590	Référence de la classe TriaMembL1::NombresConstruireTriaMembL1	1335
6.591	Référence de la classe TriaMembQ3::NombresConstruireTriaMembQ3	1336
6.592	Référence de la classe TriaMembQ3_cm1pti::NombresConstruireTriaMembQ3_cm1pti	1337
6.593	Référence de la classe TriaQ3_cmpti1003::NombresConstruireTriaQ3_cmpti1003	1338
6.594	Référence de la classe TriaQSfe1::NombresConstruireTriaQSfe1	1339
6.595	Référence de la classe TriaQSfe3::NombresConstruireTriaQSfe3	1340
6.596	Référence de la classe TriaSfe1::NombresConstruireTriaSfe1	1341
6.597	Référence de la classe TriaSfe1_cm5pti::NombresConstruireTriaSfe1_cm5pti	1342
6.598	Référence de la classe TriaSfe2::NombresConstruireTriaSfe2	1343
6.599	Référence de la classe TriaSfe3::NombresConstruireTriaSfe3	1344
6.600	Référence de la classe TriaSfe3_3D::NombresConstruireTriaSfe3_3D	1345
6.601	Référence de la classe TriaSfe3_cm12pti::NombresConstruireTriaSfe3_cm12pti	1346
6.602	Référence de la classe TriaSfe3_cm13pti::NombresConstruireTriaSfe3_cm13pti	1347
6.603	Référence de la classe TriaSfe3_cm3pti::NombresConstruireTriaSfe3_cm3pti	1348
6.604	Référence de la classe TriaSfe3_cm4pti::NombresConstruireTriaSfe3_cm4pti	1349
6.605	Référence de la classe TriaSfe3_cm5pti::NombresConstruireTriaSfe3_cm5pti	1350
6.606	Référence de la classe TriaSfe3_cm6pti::NombresConstruireTriaSfe3_cm6pti	1351
6.607	Référence de la classe TriaSfe3_cm7pti::NombresConstruireTriaSfe3_cm7pti	1352
6.608	Référence de la classe TriaSfe3C::NombresConstruireTriaSfe3C	1353
6.609	Référence de la classe OrdreVisu	1354
6.610	Référence de la classe Isovaleurs_Gid::P_gauss	1356
6.611	Référence de la classe Isovaleurs_Gmsh::P_gauss	1357
6.612	Référence de la classe ParaAlgoControle	1358
6.613	Référence de la classe ParaGlob	1362
6.614	Référence de la classe PentaL	1364
6.614.1	Documentation des fonctions membres	1366
6.614.1.1	new_frontiere_lin_rec()	1366
6.614.1.2	new_frontiere_lin_tri()	1366

---

6.614.1.3 new_frontiere_surf_rec()	1366
6.614.1.4 new_frontiere_surf_tri()	1366
6.615 Référence de la classe PentaL_cm1pti	1367
6.615.1 Documentation des fonctions membres	1368
6.615.1.1 new_frontiere_lin_rec()	1368
6.615.1.2 new_frontiere_lin_tri()	1368
6.615.1.3 new_frontiere_surf_rec()	1368
6.615.1.4 new_frontiere_surf_tri()	1369
6.616 Référence de la classe PentaL_cm2pti	1369
6.616.1 Documentation des fonctions membres	1371
6.616.1.1 new_frontiere_lin_rec()	1371
6.616.1.2 new_frontiere_lin_tri()	1371
6.616.1.3 new_frontiere_surf_rec()	1371
6.616.1.4 new_frontiere_surf_tri()	1371
6.617 Référence de la classe PentaL_cm6pti	1372
6.617.1 Documentation des fonctions membres	1373
6.617.1.1 new_frontiere_lin_rec()	1373
6.617.1.2 new_frontiere_lin_tri()	1373
6.617.1.3 new_frontiere_surf_rec()	1373
6.617.1.4 new_frontiere_surf_tri()	1374
6.618 Référence de la classe PentaMemb	1374
6.618.1 Documentation des fonctions membres	1377
6.618.1.1 Active_ddl_Sigma()	1377
6.618.1.2 Active_premier_ddl_Sigma()	1378
6.618.1.3 ContraintesAbsolues()	1378
6.618.1.4 Dim_sig_eps()	1378
6.618.1.5 ErreurElement()	1378
6.618.1.6 Frontiere_surfacique()	1378
6.618.1.7 Inactive_ddl_Sigma()	1378
6.618.1.8 LectureContraintes()	1378
6.618.1.9 Long_arrete_mini_sur_c()	1378
6.618.1.10 new_frontiere_lin()	1379
6.618.1.11 new_frontiere_surf()	1379
6.618.1.12 Plus_ddl_Sigma()	1379
6.618.1.13 Tableau_de_Sig1()	1379
6.619 Référence de la classe PentaQ	1379
6.619.1 Documentation des fonctions membres	1381
6.619.1.1 new_frontiere_lin_rec()	1381
6.619.1.2 new_frontiere_lin_tri()	1381
6.619.1.3 new_frontiere_surf_rec()	1381
6.619.1.4 new_frontiere_surf_tri()	1381
6.620 Référence de la classe PentaQ_cm12pti	1382

6.620.1	Documentation des fonctions membres	1383
6.620.1.1	new_frontiere_lin_rec()	1383
6.620.1.2	new_frontiere_lin_tri()	1383
6.620.1.3	new_frontiere_surf_rec()	1383
6.620.1.4	new_frontiere_surf_tri()	1384
6.621	Référence de la classe PentaQ_cm18pti	1384
6.621.1	Documentation des fonctions membres	1386
6.621.1.1	new_frontiere_lin_rec()	1386
6.621.1.2	new_frontiere_lin_tri()	1386
6.621.1.3	new_frontiere_surf_rec()	1386
6.621.1.4	new_frontiere_surf_tri()	1386
6.622	Référence de la classe PentaQ_cm3pti	1387
6.622.1	Documentation des fonctions membres	1388
6.622.1.1	new_frontiere_lin_rec()	1388
6.622.1.2	new_frontiere_lin_tri()	1388
6.622.1.3	new_frontiere_surf_rec()	1388
6.622.1.4	new_frontiere_surf_tri()	1389
6.623	Référence de la classe PentaQ_cm9pti	1389
6.623.1	Documentation des fonctions membres	1391
6.623.1.1	new_frontiere_lin_rec()	1391
6.623.1.2	new_frontiere_lin_tri()	1391
6.623.1.3	new_frontiere_surf_rec()	1391
6.623.1.4	new_frontiere_surf_tri()	1391
6.624	Référence de la classe PentaQComp	1392
6.624.1	Documentation des fonctions membres	1393
6.624.1.1	new_frontiere_lin_rec()	1393
6.624.1.2	new_frontiere_lin_tri()	1393
6.624.1.3	new_frontiere_surf_rec()	1393
6.624.1.4	new_frontiere_surf_tri()	1394
6.625	Référence de la classe PentaQComp_cm12pti	1394
6.625.1	Documentation des fonctions membres	1396
6.625.1.1	new_frontiere_lin_rec()	1396
6.625.1.2	new_frontiere_lin_tri()	1396
6.625.1.3	new_frontiere_surf_rec()	1396
6.625.1.4	new_frontiere_surf_tri()	1396
6.626	Référence de la classe PentaQComp_cm18pti	1397
6.626.1	Documentation des fonctions membres	1398
6.626.1.1	new_frontiere_lin_rec()	1398
6.626.1.2	new_frontiere_lin_tri()	1398
6.626.1.3	new_frontiere_surf_rec()	1398
6.626.1.4	new_frontiere_surf_tri()	1399
6.627	Référence de la classe PentaQComp_cm9pti	1399

6.627.1	Documentation des fonctions membres	1401
6.627.1.1	new_frontiere_lin_rec()	1401
6.627.1.2	new_frontiere_lin_tri()	1401
6.627.1.3	new_frontiere_surf_rec()	1401
6.627.1.4	new_frontiere_surf_tri()	1401
6.628	Référence de la classe PiPoCo	1402
6.628.1	Documentation des fonctions membres	1404
6.628.1.1	MatricesGeometrique_Et_Initiale()	1404
6.629	Référence de la classe Plan	1405
6.630	Référence de la classe Hyper3D::PoGrenoble_V	1405
6.631	Référence de la classe Hyper3D::PoGrenoble_VV	1406
6.632	Référence de la classe Hyper3D::PoGrenobleAvecPhaseAvecVar	1406
6.633	Référence de la classe Hyper3D::PoGrenobleAvecPhaseSansVar	1406
6.634	Référence de la classe Hyper3D::PoGrenobleSansPhaseAvecVar	1406
6.635	Référence de la classe Hyper3D::PoGrenobleSansPhaseSansVar	1407
6.636	Référence de la classe UtilLecture::PointFich	1407
6.637	Référence de la classe Poly_hyper3D	1408
6.637.1	Documentation des fonctions membres	1411
6.637.1.1	Affiche()	1411
6.637.1.2	Calcul_dsigma_deps()	1411
6.637.1.3	Calcul_DsigmaHH_tdt()	1411
6.637.1.4	Calcul_SigmaHH()	1412
6.637.1.5	CalculGrandeurTravail()	1412
6.637.1.6	Ecriture_base_info_loi()	1412
6.637.1.7	HsurH0()	1413
6.637.1.8	Info_commande_LoisDeComp()	1413
6.637.1.9	Lecture_base_info_loi()	1413
6.637.1.10	LectureDonneesParticulieres()	1413
6.637.1.11	Module_young_equivalent()	1413
6.637.1.12	Nouvelle_loi_identique()	1413
6.637.1.13	TestComplet()	1413
6.638	Référence de la classe Poly_Lagrange	1414
6.638.1	Description détaillée	1415
6.638.2	Documentation des fonctions membres	1415
6.638.2.1	Affiche()	1415
6.638.2.2	Complet_courbe()	1416
6.638.2.3	Der_sec()	1416
6.638.2.4	Derivee()	1416
6.638.2.5	Ecriture_base_info()	1416
6.638.2.6	Info_commande_Courbes1D()	1416
6.638.2.7	LectDonnParticulieres_courbes()	1416
6.638.2.8	Lecture_base_info()	1416

6.638.2.9 SchemaXML_Courbes1D()	1417
6.638.2.10 Valeur()	1417
6.638.2.11 Valeur_Et_der12()	1417
6.638.2.12 Valeur_Et_derivee()	1417
6.638.2.13 Valeur_Et_derivee_stricte()	1417
6.638.2.14 Valeur_stricte()	1417
6.639 Référence de la classe Ponderation	1417
6.639.1 Description détaillée	1418
6.640 Référence de la classe Ponderation_Conconsultable	1418
6.640.1 Description détaillée	1419
6.641 Référence de la classe Ponderation_GGlobal	1419
6.641.1 Description détaillée	1419
6.642 Référence de la classe Ponderation_temps	1420
6.642.1 Description détaillée	1420
6.643 Référence de la classe Ponderation_TypeQuelconque	1420
6.643.1 Description détaillée	1421
6.644 Référence de la classe Posi_ddl_noeud	1421
6.644.1 Description détaillée	1422
6.645 Référence de la classe Maillage::PosiEtNoeud	1423
6.646 Référence de la classe UtilLecture::Position_BI	1423
6.647 Référence de la classe HyperD::PotenAvecPhaseAvecVar	1424
6.648 Référence de la classe HyperD::PotenAvecPhaseSansVar	1425
6.649 Référence de la classe HyperD::PotenSansPhaseAvecVar	1426
6.650 Référence de la classe HyperD::PotenSansPhaseSansVar	1427
6.651 Référence de la classe Met_abstraite::Pour_def_Almansi	1427
6.652 Référence de la classe Met_abstraite::Pour_def_log	1428
6.653 Référence de la classe PoutSfe3	1429
6.653.1 Documentation des fonctions membres	1431
6.653.1.1 KSI()	1431
6.653.1.2 new_frontiere_lin()	1431
6.653.1.3 new_frontiere_surf()	1431
6.654 Référence de la classe PoutSimple1	1432
6.654.1 Documentation des fonctions membres	1434
6.654.1.1 Active_ddl_Sigma()	1434
6.654.1.2 Active_premier_ddl_Sigma()	1434
6.654.1.3 ContraintesAbsolues()	1435
6.654.1.4 Dim_sig_eps()	1435
6.654.1.5 H()	1435
6.654.1.6 Inactive_ddl_Sigma()	1435
6.654.1.7 KSlepais()	1435
6.654.1.8 LectureContraintes()	1435
6.654.1.9 Long_arrete_mini_sur_c()	1435



---

6.654.1.10 MatricesGeometrique_Et_Initiale()	1435
6.654.1.11 Nb_pt_int_epai()	1435
6.654.1.12 Nb_pt_int_surf()	1436
6.654.1.13 new_frontiere_lin()	1436
6.654.1.14 new_frontiere_surf()	1436
6.654.1.15 Plus_ddl_Sigma()	1436
6.654.1.16 Section()	1436
6.654.1.17 SectionMoyenne()	1436
6.655 Référence de la classe PoutTimo	1437
6.655.1 Documentation des fonctions membres	1439
6.655.1.1 new_frontiere_lin()	1439
6.655.1.2 new_frontiere_surf()	1439
6.656 Référence de la classe Prandtl_Reuss	1440
6.656.1 Documentation des fonctions membres	1442
6.656.1.1 Affiche()	1442
6.656.1.2 Calcul_DsigmaHH_tdt()	1442
6.656.1.3 Calcul_SigmaHH()	1443
6.656.1.4 CalculGrandeurTravail()	1443
6.656.1.5 Ecriture_base_info_loi()	1444
6.656.1.6 HsurH0()	1444
6.656.1.7 Info_commande_LoisDeComp()	1444
6.656.1.8 Lecture_base_info_loi()	1444
6.656.1.9 LectureDonneesParticulieres()	1444
6.656.1.10 Module_young_equivalent()	1444
6.656.1.11 New_et_Initialise()	1445
6.656.1.12 Nouvelle_loi_identique()	1445
6.656.1.13 TestCompleto()	1445
6.657 Référence de la classe Prandtl_Reuss1D	1445
6.657.1 Documentation des fonctions membres	1447
6.657.1.1 Affiche()	1447
6.657.1.2 Calcul_DsigmaHH_tdt()	1447
6.657.1.3 Calcul_SigmaHH()	1448
6.657.1.4 CalculGrandeurTravail()	1448
6.657.1.5 Ecriture_base_info_loi()	1449
6.657.1.6 HsurH0()	1449
6.657.1.7 Info_commande_LoisDeComp()	1449
6.657.1.8 Lecture_base_info_loi()	1449
6.657.1.9 LectureDonneesParticulieres()	1449
6.657.1.10 Module_young_equivalent()	1449
6.657.1.11 New_et_Initialise()	1450
6.657.1.12 Nouvelle_loi_identique()	1450
6.657.1.13 TestCompleto()	1450

6.658	Référence de la classe Prandtl_Reuss2D_D	1450
6.658.1	Documentation des fonctions membres	1452
6.658.1.1	Affiche()	1452
6.658.1.2	Calcul_DsigmaHH_tdt()	1452
6.658.1.3	Calcul_SigmaHH()	1453
6.658.1.4	CalculGrandeurTravail()	1453
6.658.1.5	Ecriture_base_info_loi()	1454
6.658.1.6	HsurH0()	1454
6.658.1.7	Info_commande_LoisDeComp()	1454
6.658.1.8	Lecture_base_info_loi()	1454
6.658.1.9	LectureDonneesParticulieres()	1454
6.658.1.10	Module_young_equivalent()	1454
6.658.1.11	New_et_Initialise()	1455
6.658.1.12	Nouvelle_loi_identique()	1455
6.658.1.13	TestComplet()	1455
6.659	Référence de la classe Pre_cond_double	1455
6.660	Référence de la classe Pression_appliquee	1456
6.661	Référence de la classe Projection_anisotrope_3D	1457
6.661.1	Documentation des fonctions membres	1460
6.661.1.1	Activation_donnees()	1460
6.661.1.2	Affiche()	1460
6.661.1.3	Calcul_dsigma_deps()	1461
6.661.1.4	Calcul_DsigmaHH_tdt()	1461
6.661.1.5	Calcul_SigmaHH()	1461
6.661.1.6	CalculGrandeurTravail()	1462
6.661.1.7	Comportement_3D_CP_DP_1D()	1462
6.661.1.8	Ecriture_base_info_loi()	1462
6.661.1.9	Grandeur_particuliere()	1462
6.661.1.10	HsurH0()	1463
6.661.1.11	Info_commande_LoisDeComp()	1463
6.661.1.12	Lecture_base_info_loi()	1463
6.661.1.13	LectureDonneesParticulieres()	1463
6.661.1.14	ListeGrandeurs_particulieres()	1463
6.661.1.15	Module_compressibilite_equivalent()	1463
6.661.1.16	Module_young_equivalent()	1463
6.661.1.17	New_et_Initialise()	1464
6.661.1.18	Nouvelle_loi_identique()	1464
6.661.1.19	RepercuteChangeTemperature()	1464
6.661.1.20	TestComplet()	1464
6.662	Référence de la classe Projet	1464
6.662.1	Description détaillée	1464
6.663	Référence de la classe PtIntegMecalInterne	1465

6.664	Référence de la classe PtIntegThermiInterne	1467
6.665	Référence de la classe Quad	1468
6.665.1	Documentation des fonctions membres	1470
6.665.1.1	new_frontiere_lin()	1470
6.665.1.2	new_frontiere_surf()	1470
6.666	Référence de la classe Quad_cm1pti	1470
6.666.1	Documentation des fonctions membres	1472
6.666.1.1	new_frontiere_lin()	1472
6.666.1.2	new_frontiere_surf()	1472
6.667	Référence de la classe QuadAxiCCom	1472
6.667.1	Documentation des fonctions membres	1474
6.667.1.1	new_frontiere_lin()	1474
6.667.1.2	new_frontiere_surf()	1474
6.668	Référence de la classe QuadAxiCCom_cm9pti	1474
6.668.1	Documentation des fonctions membres	1476
6.668.1.1	new_frontiere_lin()	1476
6.668.1.2	new_frontiere_surf()	1476
6.669	Référence de la classe QuadAxiL1	1476
6.669.1	Documentation des fonctions membres	1478
6.669.1.1	new_frontiere_lin()	1478
6.669.1.2	new_frontiere_surf()	1478
6.670	Référence de la classe QuadAxiL1_cm1pti	1478
6.670.1	Documentation des fonctions membres	1480
6.670.1.1	new_frontiere_lin()	1480
6.670.1.2	new_frontiere_surf()	1480
6.671	Référence de la classe QuadAxiMemb	1480
6.671.1	Documentation des fonctions membres	1483
6.671.1.1	Active_ddl_Sigma()	1483
6.671.1.2	Active_premier_ddl_Sigma()	1483
6.671.1.3	ContraintesAbsolues()	1484
6.671.1.4	Dim_sig_eps()	1484
6.671.1.5	ErreurElement()	1484
6.671.1.6	Inactive_ddl_Sigma()	1484
6.671.1.7	LectureContraintes()	1484
6.671.1.8	Long_arrete_mini_sur_c()	1484
6.671.1.9	Plus_ddl_Sigma()	1484
6.671.1.10	Tableau_de_Sig1()	1484
6.672	Référence de la classe QuadAxiQ	1485
6.672.1	Documentation des fonctions membres	1486
6.672.1.1	new_frontiere_lin()	1486
6.672.1.2	new_frontiere_surf()	1486
6.673	Référence de la classe QuadAxiQComp	1487

6.673.1 Documentation des fonctions membres	1488
6.673.1.1 new_frontiere_lin()	1488
6.673.1.2 new_frontiere_surf()	1488
6.674 Référence de la classe QuadAxiQComp_cm4pti	1488
6.674.1 Documentation des fonctions membres	1490
6.674.1.1 new_frontiere_lin()	1490
6.674.1.2 new_frontiere_surf()	1490
6.675 Référence de la classe QuadCCom	1490
6.675.1 Documentation des fonctions membres	1492
6.675.1.1 new_frontiere_lin()	1492
6.675.1.2 new_frontiere_surf()	1492
6.676 Référence de la classe QuadCCom_cm9pti	1492
6.676.1 Documentation des fonctions membres	1494
6.676.1.1 new_frontiere_lin()	1494
6.676.1.2 new_frontiere_surf()	1494
6.677 Référence de la classe QuadQ	1494
6.677.1 Documentation des fonctions membres	1496
6.677.1.1 new_frontiere_lin()	1496
6.677.1.2 new_frontiere_surf()	1496
6.678 Référence de la classe QuadQCom	1496
6.678.1 Documentation des fonctions membres	1498
6.678.1.1 new_frontiere_lin()	1498
6.678.1.2 new_frontiere_surf()	1498
6.679 Référence de la classe QuadQCom_cm4pti	1498
6.679.1 Documentation des fonctions membres	1500
6.679.1.1 new_frontiere_lin()	1500
6.679.1.2 new_frontiere_surf()	1500
6.680 Référence de la classe QuadraMemb	1500
6.680.1 Documentation des fonctions membres	1503
6.680.1.1 Active_ddl_Sigma()	1504
6.680.1.2 Active_premier_ddl_Sigma()	1504
6.680.1.3 ContraintesAbsolues()	1504
6.680.1.4 Dim_sig_eps()	1504
6.680.1.5 EpaisseurMoyenne()	1504
6.680.1.6 Epaisseurs()	1504
6.680.1.7 ErreurElement()	1504
6.680.1.8 Inactive_ddl_Sigma()	1504
6.680.1.9 LectureContraintes()	1505
6.680.1.10 Les_types_particuliers_internes()	1505
6.680.1.11 Long_arrete_mini_sur_c()	1505
6.680.1.12 Plus_ddl_Sigma()	1505
6.680.1.13 Tableau_de_Sig1()	1505

---

6.681	Référence de la classe <code>Quartic</code>	1505
6.682	Référence de la classe <code>quatre_string_un_entier</code>	1506
6.682.1	Description détaillée	1507
6.683	Référence de la classe <code>VariablesExporter::Quelconque_a_un_noeud</code>	1507
6.684	Référence de la classe <code>LesContacts::ReactCont</code>	1509
6.685	Référence de la classe <code>LesCondLim::ReactStoc</code>	1510
6.686	Référence de la classe <code>Reel16Pointe</code>	1510
6.687	Référence de la classe <code>Reel1Pointe</code>	1511
6.688	Référence de la classe <code>Reel21Pointe</code>	1512
6.689	Référence de la classe <code>Reel2Pointe</code>	1513
6.690	Référence de la classe <code>Reel36Pointe</code>	1514
6.691	Référence de la classe <code>Reel3Pointe</code>	1515
6.692	Référence de la classe <code>Reel4Pointe</code>	1516
6.693	Référence de la classe <code>Reel5Pointe</code>	1517
6.694	Référence de la classe <code>Reel6Pointe</code>	1518
6.695	Référence de la classe <code>Reel7Pointe</code>	1519
6.696	Référence de la classe <code>Reel81Pointe</code>	1520
6.697	Référence de la classe <code>Reel8Pointe</code>	1521
6.698	Référence de la classe <code>Reel9Pointe</code>	1522
6.699	Référence de la classe <code>Reels1</code>	1522
6.700	Référence de la classe <code>Reels16</code>	1522
6.701	Référence de la classe <code>Reels2</code>	1523
6.702	Référence de la classe <code>Reels21</code>	1523
6.703	Référence de la classe <code>Reels3</code>	1523
6.704	Référence de la classe <code>Reels36</code>	1523
6.705	Référence de la classe <code>Reels4</code>	1523
6.706	Référence de la classe <code>Reels5</code>	1523
6.707	Référence de la classe <code>Reels6</code>	1524
6.708	Référence de la classe <code>Reels7</code>	1524
6.709	Référence de la classe <code>Reels8</code>	1524
6.710	Référence de la classe <code>Reels81</code>	1524
6.711	Référence de la classe <code>Reels9</code>	1524
6.712	Référence de la classe <code>ReelsPointe</code>	1524
6.712.1	Description détaillée	1525
6.713	Référence de la classe <code>Reference</code>	1526
6.713.1	Description détaillée	1527
6.714	Référence de la classe <code>ReferenceAF</code>	1527
6.714.1	Description détaillée	1529
6.714.2	Documentation des fonctions membres	1529
6.714.2.1	<code>Affiche()</code>	1529
6.714.2.2	<code>Affiche_dans_lis()</code>	1529
6.714.2.3	<code>Ecriture_base_info()</code>	1529

6.714.2.4	Info_commande_Ref()	1529
6.714.2.5	Lecture_base_info()	1530
6.714.2.6	LectureReference()	1530
6.714.2.7	Nevez_Ref_copie()	1530
6.714.2.8	operator=()	1530
6.714.2.9	Supprime_doublons_internes()	1530
6.715	Référence de la classe ReferenceNE	1530
6.715.1	Description détaillée	1532
6.715.2	Documentation des fonctions membres	1532
6.715.2.1	Affiche() [1/2]	1532
6.715.2.2	Affiche() [2/2]	1532
6.715.2.3	Affiche_dans_lis()	1532
6.715.2.4	Ecriture_base_info()	1532
6.715.2.5	Info_commande_Ref()	1532
6.715.2.6	Lecture_base_info()	1533
6.715.2.7	LectureReference()	1533
6.715.2.8	Nevez_Ref_copie()	1533
6.715.2.9	operator=() [1/2]	1533
6.715.2.10	operator=() [2/2]	1533
6.715.2.11	Supprime_doublons_internes()	1533
6.716	Référence de la classe ReferencePtiAF	1533
6.716.1	Description détaillée	1535
6.716.2	Documentation des fonctions membres	1535
6.716.2.1	Affiche()	1535
6.716.2.2	Affiche_dans_lis()	1535
6.716.2.3	Ecriture_base_info()	1535
6.716.2.4	Info_commande_Ref()	1536
6.716.2.5	Lecture_base_info()	1536
6.716.2.6	LectureReference()	1536
6.716.2.7	Nevez_Ref_copie()	1536
6.716.2.8	operator=()	1536
6.716.2.9	Supprime_doublons_internes()	1536
6.717	Référence de la classe LesLoisDeComp::RefLoi	1536
6.718	Référence de la classe Resultats	1537
6.719	Référence de la classe Rgb	1537
6.720	Référence de la classe CristaliniteAbstraite::SaveCrista	1538
6.721	Référence de la classe Hoffman1::SaveCrista_Hoffman1	1538
6.721.1	Documentation des fonctions membres	1539
6.721.1.1	Affiche() [1/2]	1539
6.721.1.2	Affiche() [2/2]	1539
6.721.1.3	Ecriture_base_info()	1539
6.721.1.4	Lecture_base_info()	1540

6.721.1.5 Nevez_SaveCrista()	1540
6.721.1.6 operator=()	1540
6.721.1.7 TdtversT()	1540
6.721.1.8 TversTdt()	1540
6.722 Référence de la classe Hoffman2::SaveCrista_Hoffman2	1540
6.722.1 Documentation des fonctions membres	1541
6.722.1.1 Affiche() [1/2]	1541
6.722.1.2 Affiche() [2/2]	1541
6.722.1.3 Ecriture_base_info()	1541
6.722.1.4 Lecture_base_info()	1542
6.722.1.5 Nevez_SaveCrista()	1542
6.722.1.6 operator=()	1542
6.722.1.7 TdtversT()	1542
6.722.1.8 TversTdt()	1542
6.723 Référence de la classe Deformation::SaveDefResul	1542
6.724 Référence de la classe DeformationSfe1::SaveDefResulSfe1	1544
6.724.1 Documentation des fonctions membres	1545
6.724.1.1 Affiche() [1/2]	1545
6.724.1.2 Affiche() [2/2]	1545
6.724.1.3 Ecriture_base_info()	1545
6.724.1.4 Lecture_base_info()	1545
6.724.1.5 MiseAJourGrandeurs_a_0()	1545
6.724.1.6 MiseAJourGrandeurs_a_tdt()	1546
6.724.1.7 operator=()	1546
6.724.1.8 TdtversT()	1546
6.724.1.9 TversTdt()	1546
6.725 Référence de la classe CompFrotAbstraite::SaveResul	1546
6.726 Référence de la classe CompThermoPhysiqueAbstraite::SaveResul	1547
6.727 Référence de la classe Loi_comp_abstraite::SaveResul	1548
6.728 Référence de la classe Loi_comp_abstraite::SaveResul_C	1549
6.729 Référence de la classe Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C	1549
6.729.1 Documentation des fonctions membres	1550
6.729.1.1 Affiche()	1550
6.729.1.2 ChBase_des_grandeurs()	1551
6.729.1.3 Complete_SaveResul()	1551
6.729.1.4 Deformation_plastique()	1551
6.729.1.5 Ecriture_base_info()	1551
6.729.1.6 Lecture_base_info()	1551
6.729.1.7 Nevez_SaveResul()	1551
6.729.1.8 operator=()	1551
6.729.1.9 TdtversT()	1551
6.729.1.10 TversTdt()	1552

6.730	Référence de la classe <code>Loi_de_Tait::SaveResul_Loi_de_Tait</code>	1552
6.730.1	Documentation des fonctions membres	1553
6.730.1.1	<code>ChBase_des_grandeurs()</code>	1553
6.730.1.2	<code>Ecriture_base_info()</code>	1553
6.730.1.3	<code>Lecture_base_info()</code>	1553
6.730.1.4	<code>Nevez_SaveResul()</code>	1553
6.730.1.5	<code>operator=()</code>	1553
6.730.1.6	<code>TdtversT()</code>	1553
6.730.1.7	<code>TversTdt()</code>	1554
6.731	Référence de la classe <code>Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C</code>	1554
6.731.1	Documentation des fonctions membres	1555
6.731.1.1	<code>Affiche()</code>	1555
6.731.1.2	<code>ChBase_des_grandeurs()</code>	1555
6.731.1.3	<code>Complete_SaveResul()</code>	1555
6.731.1.4	<code>Deformation_plastique()</code>	1555
6.731.1.5	<code>Ecriture_base_info()</code>	1555
6.731.1.6	<code>Lecture_base_info()</code>	1556
6.731.1.7	<code>Map_type_quelconque()</code>	1556
6.731.1.8	<code>Nevez_SaveResul()</code>	1556
6.731.1.9	<code>operator=()</code>	1556
6.731.1.10	<code>TdtversT()</code>	1556
6.731.1.11	<code>TversTdt()</code>	1556
6.732	Référence de la classe <code>Loi_iso_thermo::SaveResul_Loi_iso_thermo</code>	1557
6.732.1	Documentation des fonctions membres	1558
6.732.1.1	<code>ChBase_des_grandeurs()</code>	1558
6.732.1.2	<code>Ecriture_base_info()</code>	1558
6.732.1.3	<code>Lecture_base_info()</code>	1558
6.732.1.4	<code>Nevez_SaveResul()</code>	1558
6.732.1.5	<code>operator=()</code>	1558
6.732.1.6	<code>TdtversT()</code>	1558
6.732.1.7	<code>TversTdt()</code>	1558
6.733	Référence de la classe <code>Loi_Umat::SaveResul_Loi_Umat</code>	1559
6.733.1	Documentation des fonctions membres	1560
6.733.1.1	<code>Affiche()</code>	1560
6.733.1.2	<code>ChBase_des_grandeurs()</code>	1560
6.733.1.3	<code>Complete_SaveResul()</code>	1560
6.733.1.4	<code>Ecriture_base_info()</code>	1560
6.733.1.5	<code>Lecture_base_info()</code>	1560
6.733.1.6	<code>Nevez_SaveResul()</code>	1560
6.733.1.7	<code>operator=()</code>	1560
6.733.1.8	<code>TdtversT()</code>	1560
6.733.1.9	<code>TversTdt()</code>	1561



---

6.734 Référence de la classe <code>LoiAdditiveEnSigma::SaveResul_LoiAdditiveEnSigma</code> . . . . .	1561
6.734.1 Documentation des fonctions membres . . . . .	1562
6.734.1.1 <code>Affiche()</code> . . . . .	1562
6.734.1.2 <code>ChBase_des_grandeurs()</code> . . . . .	1562
6.734.1.3 <code>Complete_SaveResul()</code> . . . . .	1562
6.734.1.4 <code>Deformation_plastique()</code> . . . . .	1562
6.734.1.5 <code>Ecriture_base_info()</code> . . . . .	1562
6.734.1.6 <code>Lecture_base_info()</code> . . . . .	1563
6.734.1.7 <code>Nevez_SaveResul()</code> . . . . .	1563
6.734.1.8 <code>operator=()</code> . . . . .	1563
6.734.1.9 <code>TdtversT()</code> . . . . .	1563
6.734.1.10 <code>TversTdt()</code> . . . . .	1563
6.735 Référence de la classe <code>LoiContraintesPlanes::SaveResul_LoiContraintesPlanes</code> . . . . .	1564
6.735.1 Documentation des fonctions membres . . . . .	1565
6.735.1.1 <code>Affiche()</code> . . . . .	1565
6.735.1.2 <code>ChBase_des_grandeurs()</code> . . . . .	1565
6.735.1.3 <code>Complete_SaveResul()</code> . . . . .	1565
6.735.1.4 <code>Deformation_plastique()</code> . . . . .	1565
6.735.1.5 <code>Ecriture_base_info()</code> . . . . .	1566
6.735.1.6 <code>Lecture_base_info()</code> . . . . .	1566
6.735.1.7 <code>Nevez_SaveResul()</code> . . . . .	1566
6.735.1.8 <code>operator=()</code> . . . . .	1566
6.735.1.9 <code>TdtversT()</code> . . . . .	1566
6.735.1.10 <code>TversTdt()</code> . . . . .	1566
6.736 Référence de la classe <code>LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble</code> . . . . .	1567
6.736.1 Documentation des fonctions membres . . . . .	1568
6.736.1.1 <code>Affiche()</code> . . . . .	1568
6.736.1.2 <code>ChBase_des_grandeurs()</code> . . . . .	1568
6.736.1.3 <code>Complete_SaveResul()</code> . . . . .	1568
6.736.1.4 <code>Deformation_plastique()</code> . . . . .	1568
6.736.1.5 <code>Ecriture_base_info()</code> . . . . .	1568
6.736.1.6 <code>Lecture_base_info()</code> . . . . .	1569
6.736.1.7 <code>Nevez_SaveResul()</code> . . . . .	1569
6.736.1.8 <code>operator=()</code> . . . . .	1569
6.736.1.9 <code>TdtversT()</code> . . . . .	1569
6.736.1.10 <code>TversTdt()</code> . . . . .	1569
6.737 Référence de la classe <code>LoiCritere::SaveResul_LoiCritere</code> . . . . .	1569
6.737.1 Documentation des fonctions membres . . . . .	1571
6.737.1.1 <code>Affiche()</code> . . . . .	1571
6.737.1.2 <code>ChBase_des_grandeurs()</code> . . . . .	1571
6.737.1.3 <code>Complete_SaveResul()</code> . . . . .	1571
6.737.1.4 <code>Deformation_plastique()</code> . . . . .	1571

---

6.737.1.5	Ecriture_base_info()	1571
6.737.1.6	Lecture_base_info()	1571
6.737.1.7	Map_type_quelconque()	1571
6.737.1.8	Nevez_SaveResul()	1571
6.737.1.9	operator=()	1572
6.737.1.10	TdtversT()	1572
6.737.1.11	TversTdt()	1572
6.738	Référence de la classe LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes	1572
6.738.1	Documentation des fonctions membres	1573
6.738.1.1	Affiche()	1573
6.738.1.2	ChBase_des_grandeurs()	1573
6.738.1.3	Complete_SaveResul()	1574
6.738.1.4	Deformation_plastique()	1574
6.738.1.5	Ecriture_base_info()	1574
6.738.1.6	Lecture_base_info()	1574
6.738.1.7	Nevez_SaveResul()	1574
6.738.1.8	operator=()	1574
6.738.1.9	TdtversT()	1574
6.738.1.10	TversTdt()	1574
6.739	Référence de la classe LoiDesMelangesEnSigma::SaveResul_LoiDesMelangesEnSigma	1575
6.739.1	Documentation des fonctions membres	1576
6.739.1.1	Affiche()	1576
6.739.1.2	ChBase_des_grandeurs()	1576
6.739.1.3	Complete_SaveResul()	1576
6.739.1.4	Deformation_plastique()	1576
6.739.1.5	Ecriture_base_info()	1576
6.739.1.6	Lecture_base_info()	1577
6.739.1.7	Nevez_SaveResul()	1577
6.739.1.8	operator=()	1577
6.739.1.9	TdtversT()	1577
6.739.1.10	TversTdt()	1577
6.740	Référence de la classe Hyper_externer_W::SaveResulHyper_externer_W	1578
6.740.1	Documentation des fonctions membres	1579
6.740.1.1	Affiche()	1579
6.740.1.2	ChBase_des_grandeurs()	1579
6.740.1.3	Complete_SaveResul()	1579
6.740.1.4	Ecriture_base_info()	1579
6.740.1.5	Lecture_base_info()	1580
6.740.1.6	Nevez_SaveResul()	1580
6.740.1.7	operator=()	1580
6.740.1.8	TdtversT()	1580
6.740.1.9	TversTdt()	1580

6.741	Référence de la classe <code>Hyper_W_gene_3D::SaveResulHyper_W_gene_3D</code>	1581
6.741.1	Documentation des fonctions membres	1582
6.741.1.1	<code>Affiche()</code>	1582
6.741.1.2	<code>ChBase_des_grandeurs()</code>	1582
6.741.1.3	<code>Complete_SaveResul()</code>	1582
6.741.1.4	<code>Ecriture_base_info()</code>	1582
6.741.1.5	<code>Lecture_base_info()</code>	1582
6.741.1.6	<code>Nevez_SaveResul()</code>	1582
6.741.1.7	<code>operator=()</code>	1583
6.741.1.8	<code>TdtversT()</code>	1583
6.741.1.9	<code>TversTdt()</code>	1583
6.742	Référence de la classe <code>HyperD::SaveResulHyperD</code>	1583
6.742.1	Documentation des fonctions membres	1584
6.742.1.1	<code>Affiche()</code>	1584
6.742.1.2	<code>ChBase_des_grandeurs()</code>	1584
6.742.1.3	<code>Complete_SaveResul()</code>	1585
6.742.1.4	<code>Ecriture_base_info()</code>	1585
6.742.1.5	<code>Lecture_base_info()</code>	1585
6.742.1.6	<code>Map_type_quelconque()</code>	1585
6.742.1.7	<code>Nevez_SaveResul()</code>	1585
6.742.1.8	<code>operator=()</code>	1585
6.742.1.9	<code>TdtversT()</code>	1585
6.742.1.10	<code>TversTdt()</code>	1585
6.743	Référence de la classe <code>HyperDN&lt; TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB &gt;← ::SaveResulHyperDN</code>	1586
6.743.1	Documentation des fonctions membres	1587
6.743.1.1	<code>Affiche()</code>	1587
6.743.1.2	<code>ChBase_des_grandeurs()</code>	1587
6.743.1.3	<code>Complete_SaveResul()</code>	1587
6.743.1.4	<code>Ecriture_base_info()</code>	1587
6.743.1.5	<code>Lecture_base_info()</code>	1587
6.743.1.6	<code>Nevez_SaveResul()</code>	1588
6.743.1.7	<code>operator=()</code>	1588
6.744	Référence de la classe <code>Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee</code>	1588
6.744.1	Documentation des fonctions membres	1589
6.744.1.1	<code>Affiche()</code>	1589
6.744.1.2	<code>ChBase_des_grandeurs()</code>	1589
6.744.1.3	<code>Complete_SaveResul()</code>	1590
6.744.1.4	<code>Ecriture_base_info()</code>	1590
6.744.1.5	<code>Lecture_base_info()</code>	1590
6.744.1.6	<code>Nevez_SaveResul()</code>	1590
6.744.1.7	<code>operator=()</code>	1590

6.744.1.8 TdtversT()	1590
6.744.1.9 TversTdt()	1590
6.745 Référence de la classe Hysteresis1D::SaveResulHysteresis1D	1591
6.745.1 Documentation des fonctions membres	1592
6.745.1.1 Affiche()	1592
6.745.1.2 ChBase_des_grandeurs()	1592
6.745.1.3 Complete_SaveResul()	1592
6.745.1.4 Ecriture_base_info()	1592
6.745.1.5 Lecture_base_info()	1593
6.745.1.6 Nevez_SaveResul()	1593
6.745.1.7 operator=()	1593
6.745.1.8 TdtversT()	1593
6.745.1.9 TversTdt()	1593
6.746 Référence de la classe Hysteresis3D::SaveResulHysteresis3D	1593
6.746.1 Documentation des fonctions membres	1595
6.746.1.1 Affiche()	1595
6.746.1.2 ChBase_des_grandeurs()	1595
6.746.1.3 Complete_SaveResul()	1595
6.746.1.4 Ecriture_base_info() [1/2]	1595
6.746.1.5 Ecriture_base_info() [2/2]	1595
6.746.1.6 Lecture_base_info() [1/2]	1596
6.746.1.7 Lecture_base_info() [2/2]	1596
6.746.1.8 Nevez_SaveResul() [1/2]	1596
6.746.1.9 Nevez_SaveResul() [2/2]	1596
6.746.1.10 operator=()	1596
6.746.1.11 TdtversT() [1/2]	1596
6.746.1.12 TdtversT() [2/2]	1596
6.746.1.13 TversTdt() [1/2]	1596
6.746.1.14 TversTdt() [2/2]	1596
6.747 Référence de la classe Hysteresis_bulk::SaveResulHysteresis_bulk	1597
6.747.1 Documentation des fonctions membres	1598
6.747.1.1 Affiche()	1598
6.747.1.2 ChBase_des_grandeurs()	1598
6.747.1.3 Complete_SaveResul()	1598
6.747.1.4 Ecriture_base_info()	1598
6.747.1.5 Lecture_base_info()	1599
6.747.1.6 Map_type_quelconque()	1599
6.747.1.7 Nevez_SaveResul()	1599
6.747.1.8 operator=()	1599
6.747.1.9 TdtversT()	1599
6.747.1.10 TversTdt()	1599
6.748 Référence de la classe Iso_elas_expo1D::SaveResulIso_elas_expo1D	1600

---

6.748.1	Documentation des fonctions membres	1601
6.748.1.1	Affiche()	1601
6.748.1.2	ChBase_des_grandeurs()	1601
6.748.1.3	Complete_SaveResul()	1601
6.748.1.4	Ecriture_base_info()	1601
6.748.1.5	Lecture_base_info()	1601
6.748.1.6	Map_type_quelconque()	1601
6.748.1.7	Nevez_SaveResul()	1602
6.748.1.8	operator=()	1602
6.748.1.9	TdtversT()	1602
6.748.1.10	TversTdt()	1602
6.749	Référence de la classe Hypo_hooke1D::SaveResulLoi_Hypo1D	1602
6.749.1	Documentation des fonctions membres	1603
6.749.1.1	Affiche()	1603
6.749.1.2	ChBase_des_grandeurs()	1604
6.749.1.3	Complete_SaveResul()	1604
6.749.1.4	Ecriture_base_info()	1604
6.749.1.5	Lecture_base_info()	1604
6.749.1.6	Nevez_SaveResul()	1604
6.749.1.7	operator=()	1604
6.749.1.8	TdtversT()	1604
6.749.1.9	TversTdt()	1604
6.750	Référence de la classe Hypo_hooke3D::SaveResulLoi_Hypo3D	1605
6.750.1	Documentation des fonctions membres	1606
6.750.1.1	Affiche()	1606
6.750.1.2	ChBase_des_grandeurs()	1606
6.750.1.3	Complete_SaveResul()	1606
6.750.1.4	Ecriture_base_info()	1606
6.750.1.5	Lecture_base_info()	1606
6.750.1.6	Nevez_SaveResul()	1606
6.750.1.7	operator=()	1607
6.750.1.8	TdtversT()	1607
6.750.1.9	TversTdt()	1607
6.751	Référence de la classe Loi_iso_elas1D::SaveResulLoi_iso_elas1D	1607
6.751.1	Documentation des fonctions membres	1608
6.751.1.1	Affiche()	1608
6.751.1.2	ChBase_des_grandeurs()	1608
6.751.1.3	Complete_SaveResul()	1609
6.751.1.4	Ecriture_base_info()	1609
6.751.1.5	Lecture_base_info()	1609
6.751.1.6	Map_type_quelconque()	1609
6.751.1.7	Nevez_SaveResul()	1609

---

6.751.1.8 operator=()	1609
6.751.1.9 TdtversT()	1609
6.751.1.10 TversTdt()	1609
6.752 Référence de la classe <code>Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D</code>	1610
6.752.1 Documentation des fonctions membres	1611
6.752.1.1 Affiche()	1611
6.752.1.2 ChBase_des_grandeurs()	1611
6.752.1.3 Complete_SaveResul()	1611
6.752.1.4 Ecriture_base_info()	1611
6.752.1.5 Lecture_base_info()	1611
6.752.1.6 Map_type_quelconque()	1611
6.752.1.7 Nevez_SaveResul()	1612
6.752.1.8 operator=()	1612
6.752.1.9 TdtversT()	1612
6.752.1.10 TversTdt()	1612
6.753 Référence de la classe <code>Loi_iso_elas3D::SaveResulLoi_iso_elas3D</code>	1612
6.753.1 Documentation des fonctions membres	1613
6.753.1.1 Affiche()	1613
6.753.1.2 ChBase_des_grandeurs()	1613
6.753.1.3 Complete_SaveResul()	1614
6.753.1.4 Ecriture_base_info()	1614
6.753.1.5 Lecture_base_info()	1614
6.753.1.6 Map_type_quelconque()	1614
6.753.1.7 Nevez_SaveResul()	1614
6.753.1.8 operator=()	1614
6.753.1.9 TdtversT()	1614
6.753.1.10 TversTdt()	1614
6.754 Référence de la classe <code>Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee</code>	1615
6.754.1 Documentation des fonctions membres	1616
6.754.1.1 Affiche()	1616
6.754.1.2 ChBase_des_grandeurs()	1616
6.754.1.3 Complete_SaveResul()	1616
6.754.1.4 Ecriture_base_info()	1616
6.754.1.5 Lecture_base_info()	1616
6.754.1.6 Nevez_SaveResul()	1616
6.754.1.7 operator=()	1617
6.754.1.8 TdtversT()	1617
6.754.1.9 TversTdt()	1617
6.755 Référence de la classe <code>Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D</code>	1617
6.755.1 Documentation des fonctions membres	1618
6.755.1.1 Affiche()	1618
6.755.1.2 ChBase_des_grandeurs()	1618

---

6.755.1.3 Complete_SaveResul()	1619
6.755.1.4 Ecriture_base_info()	1619
6.755.1.5 Lecture_base_info()	1619
6.755.1.6 Nevez_SaveResul()	1619
6.755.1.7 operator=()	1619
6.755.1.8 TdtversT()	1619
6.755.1.9 TversTdt()	1619
6.756 Référence de la classe Prandtl_Reuss::SaveResulPrandtl_Reuss	1620
6.756.1 Documentation des fonctions membres	1621
6.756.1.1 Affiche()	1621
6.756.1.2 ChBase_des_grandeurs()	1621
6.756.1.3 Complete_SaveResul()	1621
6.756.1.4 Deformation_plastique()	1621
6.756.1.5 Ecriture_base_info()	1621
6.756.1.6 Lecture_base_info()	1621
6.756.1.7 Nevez_SaveResul()	1621
6.756.1.8 operator=()	1622
6.756.1.9 TdtversT()	1622
6.756.1.10 TversTdt()	1622
6.757 Référence de la classe Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D	1622
6.757.1 Documentation des fonctions membres	1623
6.757.1.1 Affiche()	1623
6.757.1.2 ChBase_des_grandeurs()	1624
6.757.1.3 Complete_SaveResul()	1624
6.757.1.4 Deformation_plastique()	1624
6.757.1.5 Ecriture_base_info()	1624
6.757.1.6 Lecture_base_info()	1624
6.757.1.7 Nevez_SaveResul()	1624
6.757.1.8 operator=()	1624
6.757.1.9 TdtversT()	1624
6.757.1.10 TversTdt()	1625
6.758 Référence de la classe Prandtl_Reuss2D_D::SaveResulPrandtl_Reuss2D_D	1625
6.758.1 Documentation des fonctions membres	1626
6.758.1.1 Affiche()	1626
6.758.1.2 ChBase_des_grandeurs()	1626
6.758.1.3 Complete_SaveResul()	1626
6.758.1.4 Deformation_plastique()	1626
6.758.1.5 Ecriture_base_info()	1626
6.758.1.6 Lecture_base_info()	1627
6.758.1.7 Nevez_SaveResul()	1627
6.758.1.8 operator=()	1627
6.758.1.9 TdtversT()	1627

6.758.1.10 TversTdt()	1627
6.759 Référence de la classe Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D	1627
6.759.1 Documentation des fonctions membres	1628
6.759.1.1 Affiche()	1628
6.759.1.2 ChBase_des_grandeurs()	1629
6.759.1.3 Complete_SaveResul()	1629
6.759.1.4 Ecriture_base_info()	1629
6.759.1.5 Lecture_base_info()	1629
6.759.1.6 Nevez_SaveResul()	1629
6.759.1.7 operator=()	1629
6.759.1.8 TdtversT()	1629
6.759.1.9 TversTdt()	1629
6.760 Référence de la classe Sect	1630
6.760.1 Description détaillée	1630
6.761 Référence de la classe Sfeg	1631
6.762 Référence de la classe SfeMembT	1632
6.762.1 Documentation des fonctions membres	1636
6.762.1.1 Active_ddl_Sigma()	1636
6.762.1.2 Active_premier_ddl_Sigma()	1636
6.762.1.3 CalculNormale_noeud()	1636
6.762.1.4 ContraintesAbsolues()	1636
6.762.1.5 Dim_sig_eps()	1636
6.762.1.6 EpaisseurMoyenne()	1636
6.762.1.7 Epaisseurs()	1636
6.762.1.8 ErreurElement()	1637
6.762.1.9 Inactive_ddl_Sigma()	1637
6.762.1.10 LectureContraintes()	1637
6.762.1.11 Les_types_particuliers_internes()	1637
6.762.1.12 Long_arrete_mini_sur_c()	1637
6.762.1.13 Plus_ddl_Sigma()	1637
6.762.1.14 Tableau_de_Sig1()	1637
6.763 Référence de la classe Front::Signature_Front	1637
6.764 Référence de la classe SixpodeCos3phi	1638
6.764.1 Description détaillée	1639
6.764.2 Documentation des fonctions membres	1639
6.764.2.1 Affiche()	1639
6.764.2.2 Complet_courbe()	1640
6.764.2.3 Der_sec()	1640
6.764.2.4 Derivee()	1640
6.764.2.5 Ecriture_base_info()	1640
6.764.2.6 Info_commande_Courbes1D()	1640
6.764.2.7 LectDonnParticulieres_courbes()	1640



---

6.764.2.8 Lecture_base_info()	1640
6.764.2.9 SchemaXML_Courbes1D()	1641
6.764.2.10 Valeur()	1641
6.764.2.11 Valeur_Et_der12()	1641
6.764.2.12 Valeur_Et_derivee()	1641
6.764.2.13 Valeur_Et_derivee_stricte()	1641
6.764.2.14 Valeur_stricte()	1641
6.765 Référence de la classe Spectre	1642
6.765.1 Documentation des données membres	1643
6.765.1.1 spectre	1643
6.766 Référence de la classe Sphere	1643
6.767 Référence de la classe ElemMeca::StabMembBiel	1644
6.768 Référence de la classe Deformation::Stmet	1645
6.769 Référence de la classe CompThermoPhysiqueAbstraite::StockParaInt	1646
6.770 Référence de la classe String_et_entier	1646
6.770.1 Description détaillée	1647
6.771 Référence de la classe Suite_arithmetique	1647
6.771.1 Description détaillée	1648
6.771.2 Documentation des fonctions membres	1648
6.771.2.1 Affiche()	1648
6.771.2.2 Def_suite()	1648
6.771.2.3 Somme_Suite()	1648
6.771.2.4 U_n()	1648
6.772 Référence de la classe Suite_equidistante	1648
6.772.1 Description détaillée	1649
6.772.2 Documentation des fonctions membres	1649
6.772.2.1 Affiche()	1649
6.772.2.2 Def_suite()	1649
6.772.2.3 Somme_Suite()	1649
6.772.2.4 U_n()	1650
6.773 Référence de la classe Suite_geometrique	1650
6.773.1 Description détaillée	1651
6.773.2 Documentation des fonctions membres	1651
6.773.2.1 Affiche()	1651
6.773.2.2 Def_suite()	1651
6.773.2.3 Somme_Suite()	1651
6.773.2.4 U_n()	1651
6.774 Référence de la classe SuiteReel	1651
6.774.1 Description détaillée	1652
6.775 Référence de la classe Tab2_Grandeur_TenseurHH	1652
6.775.1 Description détaillée	1653
6.775.2 Documentation des fonctions membres	1653

6.775.2.1 Affectation_numerique()	1654
6.775.2.2 Change_repere()	1654
6.775.2.3 Ecriture_grandeur()	1654
6.775.2.4 Grandeur_brut()	1654
6.775.2.5 GrandeurNumOrdre()	1654
6.775.2.6 InitParDefaut()	1654
6.775.2.7 Lecture_grandeur()	1654
6.775.2.8 NbMaxiNumeroOrdre()	1654
6.775.2.9 New_idem_grandeur()	1654
6.775.2.10 operator*=(())	1655
6.775.2.11 operator/=(())	1655
6.775.2.12 Type_enumGrandeurParticuliere()	1655
6.775.2.13 Type_grandeurAssocie()	1655
6.775.2.14 Type_structure_grandeurAssocie()	1655
6.776 Référence de l'union Tab_car_double_int	1655
6.777 Référence de l'union Tab_car_double_int_1	1655
6.777.1 Description détaillée	1656
6.778 Référence de l'union Tab_car_et_double	1656
6.779 Référence de la classe Tab_Grandeur_BaseH	1656
6.779.1 Description détaillée	1657
6.779.2 Documentation des fonctions membres	1658
6.779.2.1 Affectation_numerique()	1658
6.779.2.2 Change_repere()	1658
6.779.2.3 Ecriture_grandeur()	1658
6.779.2.4 Grandeur_brut()	1658
6.779.2.5 GrandeurNumOrdre()	1658
6.779.2.6 InitParDefaut()	1658
6.779.2.7 Lecture_grandeur()	1658
6.779.2.8 NbMaxiNumeroOrdre()	1658
6.779.2.9 New_idem_grandeur()	1659
6.779.2.10 operator*=(())	1659
6.779.2.11 operator/=(())	1659
6.779.2.12 Type_enumGrandeurParticuliere()	1659
6.779.2.13 Type_grandeurAssocie()	1659
6.779.2.14 Type_structure_grandeurAssocie()	1659
6.780 Référence de la classe Tab_Grandeur_Coordonnee	1659
6.780.1 Description détaillée	1661
6.780.2 Documentation des fonctions membres	1661
6.780.2.1 Affectation_numerique()	1661
6.780.2.2 Change_repere()	1661
6.780.2.3 Ecriture_grandeur()	1661
6.780.2.4 Grandeur_brut()	1661

---

6.780.2.5	GrandeurNumOrdre()	1661
6.780.2.6	InitParDefaut()	1661
6.780.2.7	Lecture_grandeur()	1662
6.780.2.8	NbMaxiNumeroOrdre()	1662
6.780.2.9	New_idem_grandeur()	1662
6.780.2.10	operator*=(())	1662
6.780.2.11	operator/=(())	1662
6.780.2.12	Type_enumGrandeurParticuliere()	1662
6.780.2.13	Type_grandeurAssocie()	1662
6.780.2.14	Type_structure_grandeurAssocie()	1662
6.781	Référence de la classe Tab_Grandeur_Ddl_etendu	1663
6.781.1	Description détaillée	1664
6.781.2	Documentation des fonctions membres	1664
6.781.2.1	Affectation_numerique()	1664
6.781.2.2	Change_repere()	1664
6.781.2.3	Ecriture_grandeur()	1664
6.781.2.4	Grandeur_brut()	1664
6.781.2.5	GrandeurNumOrdre()	1665
6.781.2.6	InitParDefaut()	1665
6.781.2.7	Lecture_grandeur()	1665
6.781.2.8	NbMaxiNumeroOrdre()	1665
6.781.2.9	New_idem_grandeur()	1665
6.781.2.10	operator*=(())	1665
6.781.2.11	operator/=(())	1665
6.781.2.12	Type_enumGrandeurParticuliere()	1665
6.781.2.13	Type_grandeurAssocie()	1665
6.781.2.14	Type_structure_grandeurAssocie()	1666
6.782	Référence de la classe Tab_Grandeur_scalaire_double	1666
6.782.1	Description détaillée	1667
6.782.2	Documentation des fonctions membres	1667
6.782.2.1	Affectation_numerique()	1667
6.782.2.2	Change_repere()	1667
6.782.2.3	Ecriture_grandeur()	1667
6.782.2.4	Grandeur_brut()	1668
6.782.2.5	GrandeurNumOrdre()	1668
6.782.2.6	InitParDefaut()	1668
6.782.2.7	Lecture_grandeur()	1668
6.782.2.8	NbMaxiNumeroOrdre()	1668
6.782.2.9	New_idem_grandeur()	1668
6.782.2.10	operator*=(())	1668
6.782.2.11	operator/=(())	1668
6.782.2.12	Type_enumGrandeurParticuliere()	1668

6.782.2.13	Type_grandeurAssocie()	1669
6.782.2.14	Type_structure_grandeurAssocie()	1669
6.783	Référence de la classe Tab_Grandeur_scalaire_entier	1669
6.783.1	Description détaillée	1670
6.783.2	Documentation des fonctions membres	1670
6.783.2.1	Affectation_numerique()	1670
6.783.2.2	Change_repere()	1670
6.783.2.3	Ecriture_grandeur()	1671
6.783.2.4	Grandeur_brut()	1671
6.783.2.5	GrandeurNumOrdre()	1671
6.783.2.6	InitParDefaut()	1671
6.783.2.7	Lecture_grandeur()	1671
6.783.2.8	NbMaxiNumeroOrdre()	1671
6.783.2.9	New_idem_grandeur()	1671
6.783.2.10	operator*=(())	1671
6.783.2.11	operator/=(())	1671
6.783.2.12	Type_enumGrandeurParticuliere()	1672
6.783.2.13	Type_grandeurAssocie()	1672
6.783.2.14	Type_structure_grandeurAssocie()	1672
6.784	Référence de la classe Tab_Grandeur_TenseurBB	1672
6.784.1	Description détaillée	1673
6.784.2	Documentation des fonctions membres	1674
6.784.2.1	Affectation_numerique()	1674
6.784.2.2	Change_repere()	1674
6.784.2.3	Ecriture_grandeur()	1674
6.784.2.4	Grandeur_brut()	1674
6.784.2.5	GrandeurNumOrdre()	1674
6.784.2.6	InitParDefaut()	1674
6.784.2.7	Lecture_grandeur()	1674
6.784.2.8	NbMaxiNumeroOrdre()	1674
6.784.2.9	New_idem_grandeur()	1675
6.784.2.10	operator*=(())	1675
6.784.2.11	operator/=(())	1675
6.784.2.12	Type_enumGrandeurParticuliere()	1675
6.784.2.13	Type_grandeurAssocie()	1675
6.784.2.14	Type_structure_grandeurAssocie()	1675
6.785	Référence de la classe Tab_Grandeur_TenseurBH	1675
6.785.1	Description détaillée	1677
6.785.2	Documentation des fonctions membres	1677
6.785.2.1	Affectation_numerique()	1677
6.785.2.2	Change_repere()	1677
6.785.2.3	Ecriture_grandeur()	1677

---

6.785.2.4	Grandeur_brut()	1677
6.785.2.5	GrandeurNumOrdre()	1677
6.785.2.6	InitParDefaut()	1677
6.785.2.7	Lecture_grandeur()	1678
6.785.2.8	NbMaxiNumeroOrdre()	1678
6.785.2.9	New_idem_grandeur()	1678
6.785.2.10	operator*=(())	1678
6.785.2.11	operator/=(())	1678
6.785.2.12	Type_enumGrandeurParticuliere()	1678
6.785.2.13	Type_grandeurAssocie()	1678
6.785.2.14	Type_structure_grandeurAssocie()	1678
6.786	Référence de la classe Tab_Grandeur_TenseurHB	1678
6.786.1	Description détaillée	1680
6.786.2	Documentation des fonctions membres	1680
6.786.2.1	Affectation_numerique()	1680
6.786.2.2	Change_repere()	1680
6.786.2.3	Ecriture_grandeur()	1680
6.786.2.4	Grandeur_brut()	1680
6.786.2.5	GrandeurNumOrdre()	1680
6.786.2.6	InitParDefaut()	1680
6.786.2.7	Lecture_grandeur()	1681
6.786.2.8	NbMaxiNumeroOrdre()	1681
6.786.2.9	New_idem_grandeur()	1681
6.786.2.10	operator*=(())	1681
6.786.2.11	operator/=(())	1681
6.786.2.12	Type_enumGrandeurParticuliere()	1681
6.786.2.13	Type_grandeurAssocie()	1681
6.786.2.14	Type_structure_grandeurAssocie()	1681
6.787	Référence de la classe Tab_Grandeur_TenseurHH	1681
6.787.1	Description détaillée	1683
6.787.2	Documentation des fonctions membres	1683
6.787.2.1	Affectation_numerique()	1683
6.787.2.2	Change_repere()	1683
6.787.2.3	Ecriture_grandeur()	1683
6.787.2.4	Grandeur_brut()	1683
6.787.2.5	GrandeurNumOrdre()	1683
6.787.2.6	InitParDefaut()	1683
6.787.2.7	Lecture_grandeur()	1684
6.787.2.8	NbMaxiNumeroOrdre()	1684
6.787.2.9	New_idem_grandeur()	1684
6.787.2.10	operator*=(())	1684
6.787.2.11	operator/=(())	1684

---

6.787.2.12 Type_enumGrandeurParticuliere()	1684
6.787.2.13 Type_grandeurAssocie()	1684
6.787.2.14 Type_structure_grandeurAssocie()	1684
6.788 Référence de la classe Tab_Grandeur_Vecteur	1684
6.788.1 Description détaillée	1686
6.788.2 Documentation des fonctions membres	1686
6.788.2.1 Affectation_numerique()	1686
6.788.2.2 Change_repere()	1686
6.788.2.3 Ecriture_grandeur()	1686
6.788.2.4 Grandeur_brut()	1686
6.788.2.5 GrandeurNumOrdre()	1686
6.788.2.6 InitParDefaut()	1686
6.788.2.7 Lecture_grandeur()	1687
6.788.2.8 NbMaxiNumeroOrdre()	1687
6.788.2.9 New_idem_grandeur()	1687
6.788.2.10 operator*=(())	1687
6.788.2.11 operator/=(())	1687
6.788.2.12 Type_enumGrandeurParticuliere()	1687
6.788.2.13 Type_grandeurAssocie()	1687
6.788.2.14 Type_structure_grandeurAssocie()	1687
6.789 Référence de la classe Tab_Grandeur_Vecteur_Nommer	1687
6.789.1 Description détaillée	1689
6.789.2 Documentation des fonctions membres	1689
6.789.2.1 Affectation_numerique()	1689
6.789.2.2 Change_repere()	1689
6.789.2.3 Ecriture_grandeur()	1689
6.789.2.4 Grandeur_brut()	1689
6.789.2.5 GrandeurNumOrdre()	1689
6.789.2.6 InitParDefaut()	1690
6.789.2.7 Lecture_grandeur()	1690
6.789.2.8 NbMaxiNumeroOrdre()	1690
6.789.2.9 New_idem_grandeur()	1690
6.789.2.10 operator*=(())	1690
6.789.2.11 operator/=(())	1690
6.789.2.12 Type_enumGrandeurParticuliere()	1690
6.789.2.13 Type_grandeurAssocie()	1690
6.789.2.14 Type_structure_grandeurAssocie()	1690
6.790 Référence de la classe Table33	1691
6.791 Référence du modèle de la classe Tableau< T >	1691
6.791.1 Documentation des fonctions membres	1692
6.791.1.1 Change_taille()	1692
6.792 Référence du modèle de la classe Tableau2< T >	1692

6.793	Référence du modèle de la classe <code>Tableau4&lt; T &gt;</code>	1693
6.794	Référence de la classe <code>Tableau_3D</code>	1694
6.795	Référence de la classe <code>Tableau_4D</code>	1695
6.796	Référence de la classe <code>Tableau_5D</code>	1695
6.797	Référence de la classe <code>Tableau_double</code>	1696
6.798	Référence de la classe <code>Tableau_Grandeur_quelconque</code>	1697
6.798.1	Description détaillée	1698
6.798.2	Documentation des fonctions membres	1698
6.798.2.1	<code>Affectation_numerique()</code>	1698
6.798.2.2	<code>Change_repere()</code>	1698
6.798.2.3	<code>Ecriture_grandeur()</code>	1699
6.798.2.4	<code>Grandeur_brut()</code>	1699
6.798.2.5	<code>GrandeurNumOrdre()</code>	1699
6.798.2.6	<code>InitParDefaut()</code>	1699
6.798.2.7	<code>Lecture_grandeur()</code>	1699
6.798.2.8	<code>NbMaxiNumeroOrdre()</code>	1699
6.798.2.9	<code>New_idem_grandeur()</code>	1699
6.798.2.10	<code>operator*=(())</code>	1699
6.798.2.11	<code>operator/=(())</code>	1699
6.798.2.12	<code>operator=()</code>	1700
6.798.2.13	<code>Type_enumGrandeurParticuliere()</code>	1700
6.798.2.14	<code>Type_grandeurAssocie()</code>	1700
6.798.2.15	<code>Type_structure_grandeurAssocie()</code>	1700
6.799	Référence du modèle de la classe <code>TabOper&lt; T &gt;</code>	1700
6.800	Référence de la classe <code>TangenteHyperbolique</code>	1701
6.800.1	Description détaillée	1703
6.800.2	Documentation des fonctions membres	1703
6.800.2.1	<code>Affiche()</code>	1703
6.800.2.2	<code>Complet_courbe()</code>	1703
6.800.2.3	<code>Der_sec()</code>	1703
6.800.2.4	<code>Derivee()</code>	1703
6.800.2.5	<code>Ecriture_base_info()</code>	1704
6.800.2.6	<code>Info_commande_Courbes1D()</code>	1704
6.800.2.7	<code>LectDonnParticulieres_courbes()</code>	1704
6.800.2.8	<code>Lecture_base_info()</code>	1704
6.800.2.9	<code>SchemaXML_Courbes1D()</code>	1704
6.800.2.10	<code>Valeur()</code>	1704
6.800.2.11	<code>Valeur_Et_der12()</code>	1704
6.800.2.12	<code>Valeur_Et_derivee()</code>	1705
6.800.2.13	<code>Valeur_Et_derivee_stricte()</code>	1705
6.800.2.14	<code>Valeur_stricte()</code>	1705
6.801	Référence de la classe <code>Temps_CPU_HZpp</code>	1705

6.801.1 Description détaillée	1706
6.802 Référence de la classe Temps_CPU_HZpp_3	1706
6.802.1 Description détaillée	1706
6.803 Référence de la classe Tenseur1BB	1707
6.803.1 Description détaillée	1709
6.803.2 Documentation des fonctions membres	1709
6.803.2.1 Affectation_trans_dimension()	1709
6.803.2.2 Coor()	1709
6.803.2.3 Det()	1709
6.803.2.4 Ecriture()	1709
6.803.2.5 Inita()	1710
6.803.2.6 Inverse()	1710
6.803.2.7 Lecture()	1710
6.803.2.8 MaxiComposante()	1710
6.803.2.9 Monte2Indices()	1710
6.803.2.10 MonteDernierIndice()	1710
6.803.2.11 MontePremierIndice()	1710
6.803.2.12 operator"!=(	1710
6.803.2.13 operator&&()	1711
6.803.2.14 operator>()()	1711
6.803.2.15 operator*() [1/4]	1711
6.803.2.16 operator*() [2/4]	1711
6.803.2.17 operator*() [3/4]	1711
6.803.2.18 operator*() [4/4]	1711
6.803.2.19 operator*=(	1711
6.803.2.20 operator+()	1712
6.803.2.21 operator+=(	1712
6.803.2.22 operator-() [1/2]	1712
6.803.2.23 operator-() [2/2]	1712
6.803.2.24 operator-=(	1712
6.803.2.25 operator/()	1712
6.803.2.26 operator/=()	1712
6.803.2.27 operator=()	1712
6.803.2.28 operator==(	1713
6.803.2.29 Transpose()	1713
6.804 Référence de la classe Tenseur1BBBB	1713
6.804.1 Documentation des fonctions membres	1715
6.804.1.1 Affectation_trans_dimension()	1715
6.804.1.2 Change()	1715
6.804.1.3 ChangePlus()	1715
6.804.1.4 Ecriture()	1716
6.804.1.5 Inita()	1716



6.804.1.6 Lecture()	1716
6.804.1.7 MaxiComposante()	1716
6.804.1.8 operator&&()	1716
6.804.1.9 operator()()	1716
6.804.1.10 operator*()	1716
6.804.1.11 operator*=(())	1716
6.804.1.12 operator+()	1717
6.804.1.13 operator+=()	1717
6.804.1.14 operator-() [1/2]	1717
6.804.1.15 operator-() [2/2]	1717
6.804.1.16 operator-=()	1717
6.804.1.17 operator/()	1717
6.804.1.18 operator/=()	1717
6.804.1.19 operator=()	1717
6.804.1.20 operator==(())	1717
6.804.1.21 Prod_gauche()	1718
6.804.1.22 Transpose1et2avec3et4()	1718
6.805 Référence de la classe Tenseur1BBHH	1718
6.805.1 Documentation des fonctions membres	1720
6.805.1.1 Affectation_trans_dimension()	1720
6.805.1.2 Change()	1720
6.805.1.3 ChangePlus()	1720
6.805.1.4 Ecriture()	1720
6.805.1.5 Inita()	1721
6.805.1.6 Lecture()	1721
6.805.1.7 MaxiComposante()	1721
6.805.1.8 operator&&()	1721
6.805.1.9 operator()()	1721
6.805.1.10 operator*()	1721
6.805.1.11 operator*=(())	1721
6.805.1.12 operator+()	1721
6.805.1.13 operator+=()	1722
6.805.1.14 operator-() [1/2]	1722
6.805.1.15 operator-() [2/2]	1722
6.805.1.16 operator-=()	1722
6.805.1.17 operator/()	1722
6.805.1.18 operator/=()	1722
6.805.1.19 operator=()	1722
6.805.1.20 operator==(())	1722
6.805.1.21 Prod_gauche()	1722
6.805.1.22 Transpose1et2avec3et4()	1723
6.806 Référence de la classe Tenseur1BH	1723

6.806.1 Description détaillée	1725
6.806.2 Documentation des fonctions membres	1725
6.806.2.1 Affectation_trans_dimension()	1725
6.806.2.2 Coor()	1726
6.806.2.3 Det()	1726
6.806.2.4 Ecriture()	1726
6.806.2.5 II()	1726
6.806.2.6 III()	1726
6.806.2.7 Initia()	1726
6.806.2.8 Inverse()	1726
6.806.2.9 Lecture()	1726
6.806.2.10 MaxiComposante()	1727
6.806.2.11 operator!=(())	1727
6.806.2.12 operator&&()	1727
6.806.2.13 operator>()()	1727
6.806.2.14 operator*() [1/4]	1727
6.806.2.15 operator*() [2/4]	1727
6.806.2.16 operator*() [3/4]	1727
6.806.2.17 operator*() [4/4]	1727
6.806.2.18 operator*==(())	1728
6.806.2.19 operator+()	1728
6.806.2.20 operator+==(())	1728
6.806.2.21 operator-() [1/2]	1728
6.806.2.22 operator-() [2/2]	1728
6.806.2.23 operator-==(())	1728
6.806.2.24 operator/()	1728
6.806.2.25 operator/==(())	1729
6.806.2.26 operator=()	1729
6.806.2.27 operator==(())	1729
6.806.2.28 PermuteHautBas()	1729
6.806.2.29 Trace()	1729
6.806.2.30 Transpose()	1729
6.806.2.31 ValPropre() [1/2]	1729
6.806.2.32 ValPropre() [2/2]	1730
6.806.2.33 VecteursPropres()	1730
6.807 Référence de la classe Tenseur1HB	1730
6.807.1 Description détaillée	1733
6.807.2 Documentation des fonctions membres	1733
6.807.2.1 Affectation_trans_dimension()	1733
6.807.2.2 Coor()	1733
6.807.2.3 Det()	1733
6.807.2.4 Ecriture()	1734

6.807.2.5 II()	1734
6.807.2.6 III()	1734
6.807.2.7 Inita()	1734
6.807.2.8 Inverse()	1734
6.807.2.9 Lecture()	1734
6.807.2.10 MaxiComposante()	1734
6.807.2.11 operator"!=(())	1734
6.807.2.12 operator&&()	1735
6.807.2.13 operator>()()	1735
6.807.2.14 operator*() [1/4]	1735
6.807.2.15 operator*() [2/4]	1735
6.807.2.16 operator*() [3/4]	1735
6.807.2.17 operator*() [4/4]	1735
6.807.2.18 operator*==(())	1735
6.807.2.19 operator+()	1735
6.807.2.20 operator+==(())	1736
6.807.2.21 operator-() [1/2]	1736
6.807.2.22 operator-() [2/2]	1736
6.807.2.23 operator-==(())	1736
6.807.2.24 operator/()	1736
6.807.2.25 operator/=()	1736
6.807.2.26 operator=()	1736
6.807.2.27 operator==(())	1737
6.807.2.28 PermuteHautBas()	1737
6.807.2.29 Trace()	1737
6.807.2.30 Transpose()	1737
6.807.2.31 ValPropre() [1/2]	1737
6.807.2.32 ValPropre() [2/2]	1737
6.807.2.33 VecteursPropres()	1738
6.808 Référence de la classe Tenseur1HH	1738
6.808.1 Description détaillée	1740
6.808.2 Documentation des fonctions membres	1740
6.808.2.1 Affectation_trans_dimension()	1741
6.808.2.2 Baisse2Indices()	1741
6.808.2.3 BaisseDernierIndice()	1741
6.808.2.4 BaissePremierIndice()	1741
6.808.2.5 Coor()	1741
6.808.2.6 Det()	1741
6.808.2.7 Ecriture()	1741
6.808.2.8 Inita()	1742
6.808.2.9 Inverse()	1742
6.808.2.10 Lecture()	1742

6.808.2.11 MaxiComposante()	1742
6.808.2.12 operator"!=()	1742
6.808.2.13 operator&&()	1742
6.808.2.14 operator>()()	1742
6.808.2.15 operator*() [1/4]	1742
6.808.2.16 operator*() [2/4]	1743
6.808.2.17 operator*() [3/4]	1743
6.808.2.18 operator*() [4/4]	1743
6.808.2.19 operator*==(())	1743
6.808.2.20 operator+()	1743
6.808.2.21 operator+==(())	1743
6.808.2.22 operator-() [1/2]	1743
6.808.2.23 operator-() [2/2]	1744
6.808.2.24 operator--()	1744
6.808.2.25 operator/()	1744
6.808.2.26 operator/=()	1744
6.808.2.27 operator=()	1744
6.808.2.28 operator==(())	1744
6.808.2.29 Transpose()	1744
6.809 Référence de la classe Tenseur1HHBB	1745
6.809.1 Documentation des fonctions membres	1746
6.809.1.1 Affectation_trans_dimension()	1746
6.809.1.2 Change()	1746
6.809.1.3 ChangePlus()	1747
6.809.1.4 Ecriture()	1747
6.809.1.5 Initia()	1747
6.809.1.6 Lecture()	1747
6.809.1.7 MaxiComposante()	1747
6.809.1.8 operator&&()	1747
6.809.1.9 operator>()()	1747
6.809.1.10 operator*()	1747
6.809.1.11 operator*==(())	1748
6.809.1.12 operator+()	1748
6.809.1.13 operator+==(())	1748
6.809.1.14 operator-() [1/2]	1748
6.809.1.15 operator-() [2/2]	1748
6.809.1.16 operator--()	1748
6.809.1.17 operator/()	1748
6.809.1.18 operator/=()	1748
6.809.1.19 operator=()	1749
6.809.1.20 operator==(())	1749
6.809.1.21 Prod_gauche()	1749

---

6.809.1.22 Transpose1et2avec3et4()	1749
6.810 Référence de la classe Tenseur1HHHH	1749
6.810.1 Documentation des fonctions membres	1751
6.810.1.1 Affectation_trans_dimension()	1751
6.810.1.2 Change()	1751
6.810.1.3 ChangePlus()	1751
6.810.1.4 Ecriture()	1752
6.810.1.5 Inita()	1752
6.810.1.6 Lecture()	1752
6.810.1.7 MaxiComposante()	1752
6.810.1.8 operator&&()	1752
6.810.1.9 operator()()	1752
6.810.1.10 operator*()	1752
6.810.1.11 operator*=(())	1752
6.810.1.12 operator+()	1753
6.810.1.13 operator+=(())	1753
6.810.1.14 operator-() [1/2]	1753
6.810.1.15 operator-() [2/2]	1753
6.810.1.16 operator-=(())	1753
6.810.1.17 operator/()	1753
6.810.1.18 operator/=(())	1753
6.810.1.19 operator=()	1753
6.810.1.20 operator==(())	1753
6.810.1.21 Prod_gauche()	1754
6.810.1.22 Transpose1et2avec3et4()	1754
6.811 Référence de la classe Tenseur2BB	1754
6.811.1 Description détaillée	1756
6.811.2 Documentation des fonctions membres	1756
6.811.2.1 Affectation_trans_dimension()	1756
6.811.2.2 Coor()	1757
6.811.2.3 Det()	1757
6.811.2.4 Ecriture()	1757
6.811.2.5 Inita()	1757
6.811.2.6 Inverse()	1757
6.811.2.7 Lecture()	1757
6.811.2.8 MaxiComposante()	1757
6.811.2.9 Monte2Indices()	1758
6.811.2.10 MonteDernierIndice()	1758
6.811.2.11 MontePremierIndice()	1758
6.811.2.12 operator"!=(())	1758
6.811.2.13 operator&&()	1758
6.811.2.14 operator()()	1758

6.811.2.15 operator*() [1/4]	1758
6.811.2.16 operator*() [2/4]	1758
6.811.2.17 operator*() [3/4]	1759
6.811.2.18 operator*() [4/4]	1759
6.811.2.19 operator*=( )	1759
6.811.2.20 operator+( )	1759
6.811.2.21 operator+=( )	1759
6.811.2.22 operator-( ) [1/2]	1759
6.811.2.23 operator-( ) [2/2]	1759
6.811.2.24 operator-=( )	1760
6.811.2.25 operator/( )	1760
6.811.2.26 operator/=( )	1760
6.811.2.27 operator=( )	1760
6.811.2.28 operator==( )	1760
6.811.2.29 Transpose( )	1760
6.812 Référence de la classe Tenseur2BBBB	1761
6.812.1 Documentation des fonctions membres	1762
6.812.1.1 Affectation_trans_dimension( )	1762
6.812.1.2 Change( )	1763
6.812.1.3 ChangePlus( )	1763
6.812.1.4 Ecriture( )	1763
6.812.1.5 Inita( )	1763
6.812.1.6 Lecture( )	1763
6.812.1.7 MaxiComposante( )	1763
6.812.1.8 operator&&( )	1763
6.812.1.9 operator()( )	1763
6.812.1.10 operator*( )	1764
6.812.1.11 operator*=( )	1764
6.812.1.12 operator+( )	1764
6.812.1.13 operator+=( )	1764
6.812.1.14 operator-( ) [1/2]	1764
6.812.1.15 operator-( ) [2/2]	1764
6.812.1.16 operator-=( )	1764
6.812.1.17 operator/( )	1764
6.812.1.18 operator/=( )	1765
6.812.1.19 operator=( )	1765
6.812.1.20 operator==( )	1765
6.812.1.21 Prod_gauche( )	1765
6.812.1.22 Transpose1et2avec3et4( )	1765
6.813 Référence de la classe Tenseur2BBHH	1765
6.813.1 Documentation des fonctions membres	1767
6.813.1.1 Affectation_trans_dimension( )	1767

6.813.1.2 Change()	1767
6.813.1.3 ChangePlus()	1767
6.813.1.4 Ecriture()	1768
6.813.1.5 Initia()	1768
6.813.1.6 Lecture()	1768
6.813.1.7 MaxiComposante()	1768
6.813.1.8 operator&&()	1768
6.813.1.9 operator>()()	1768
6.813.1.10 operator*()	1768
6.813.1.11 operator*=(())	1768
6.813.1.12 operator+()	1769
6.813.1.13 operator+=()	1769
6.813.1.14 operator-() [1/2]	1769
6.813.1.15 operator-() [2/2]	1769
6.813.1.16 operator-=()	1769
6.813.1.17 operator/()	1769
6.813.1.18 operator/=()	1769
6.813.1.19 operator=()	1769
6.813.1.20 operator==(())	1769
6.813.1.21 Prod_gauche()	1770
6.813.1.22 Transpose1et2avec3et4()	1770
6.814 Référence de la classe Tenseur2BH	1770
6.814.1 Description détaillée	1773
6.814.2 Documentation des fonctions membres	1773
6.814.2.1 Affectation_trans_dimension()	1773
6.814.2.2 Coor()	1773
6.814.2.3 Det()	1773
6.814.2.4 Ecriture()	1773
6.814.2.5 II()	1774
6.814.2.6 III()	1774
6.814.2.7 Initia()	1774
6.814.2.8 Inverse()	1774
6.814.2.9 Lecture()	1774
6.814.2.10 MaxiComposante()	1774
6.814.2.11 operator"!=(())	1774
6.814.2.12 operator&&()	1774
6.814.2.13 operator>()()	1775
6.814.2.14 operator*() [1/4]	1775
6.814.2.15 operator*() [2/4]	1775
6.814.2.16 operator*() [3/4]	1775
6.814.2.17 operator*() [4/4]	1775
6.814.2.18 operator*=(())	1775

6.814.2.19 operator+()	1775
6.814.2.20 operator+=()	1776
6.814.2.21 operator-() [1/2]	1776
6.814.2.22 operator-() [2/2]	1776
6.814.2.23 operator-=()	1776
6.814.2.24 operator/()	1776
6.814.2.25 operator/=()	1776
6.814.2.26 operator=()	1776
6.814.2.27 operator==(())	1776
6.814.2.28 PermuteHautBas()	1777
6.814.2.29 Trace()	1777
6.814.2.30 Transpose()	1777
6.814.2.31 ValPropre() [1/2]	1777
6.814.2.32 ValPropre() [2/2]	1777
6.814.2.33 VecteursPropres()	1778
6.815 Référence de la classe Tenseur2HB	1778
6.815.1 Description détaillée	1781
6.815.2 Documentation des fonctions membres	1781
6.815.2.1 Affectation_trans_dimension()	1781
6.815.2.2 Coor()	1781
6.815.2.3 Det()	1781
6.815.2.4 Ecriture()	1781
6.815.2.5 II()	1782
6.815.2.6 III()	1782
6.815.2.7 Inita()	1782
6.815.2.8 Inverse()	1782
6.815.2.9 Lecture()	1782
6.815.2.10 MaxiComposante()	1782
6.815.2.11 operator"!=(())	1782
6.815.2.12 operator&&()	1782
6.815.2.13 operator>()()	1783
6.815.2.14 operator*() [1/4]	1783
6.815.2.15 operator*() [2/4]	1783
6.815.2.16 operator*() [3/4]	1783
6.815.2.17 operator*() [4/4]	1783
6.815.2.18 operator*=(())	1783
6.815.2.19 operator+()	1783
6.815.2.20 operator+=()	1784
6.815.2.21 operator-() [1/2]	1784
6.815.2.22 operator-() [2/2]	1784
6.815.2.23 operator-=()	1784
6.815.2.24 operator/()	1784



6.815.2.25 operator/=( )	1784
6.815.2.26 operator=( )	1784
6.815.2.27 operator==( )	1784
6.815.2.28 PermuteHautBas()	1785
6.815.2.29 Trace()	1785
6.815.2.30 Transpose()	1785
6.815.2.31 ValPropre() [1/2]	1785
6.815.2.32 ValPropre() [2/2]	1785
6.815.2.33 VecteursPropres()	1786
6.816 Référence de la classe Tenseur2HH	1786
6.816.1 Description détaillée	1788
6.816.2 Documentation des fonctions membres	1789
6.816.2.1 Affectation_trans_dimension()	1789
6.816.2.2 Baisse2Indices()	1789
6.816.2.3 BaisseDernierIndice()	1789
6.816.2.4 BaissePremierIndice()	1789
6.816.2.5 Coor()	1789
6.816.2.6 Det()	1789
6.816.2.7 Ecriture()	1790
6.816.2.8 Inita()	1790
6.816.2.9 Inverse()	1790
6.816.2.10 Lecture()	1790
6.816.2.11 MaxiComposante()	1790
6.816.2.12 operator"!=( )	1790
6.816.2.13 operator&&( )	1790
6.816.2.14 operator()( )	1790
6.816.2.15 operator*( ) [1/4]	1791
6.816.2.16 operator*( ) [2/4]	1791
6.816.2.17 operator*( ) [3/4]	1791
6.816.2.18 operator*( ) [4/4]	1791
6.816.2.19 operator*=( )	1791
6.816.2.20 operator+( )	1791
6.816.2.21 operator+=( )	1791
6.816.2.22 operator-( ) [1/2]	1792
6.816.2.23 operator-( ) [2/2]	1792
6.816.2.24 operator-=( )	1792
6.816.2.25 operator/( )	1792
6.816.2.26 operator/=( )	1792
6.816.2.27 operator=( )	1792
6.816.2.28 operator==( )	1792
6.816.2.29 Transpose()	1792
6.817 Référence de la classe Tenseur2HHBB	1793

6.817.1 Documentation des fonctions membres	1794
6.817.1.1 Affectation_trans_dimension()	1794
6.817.1.2 Change()	1795
6.817.1.3 ChangePlus()	1795
6.817.1.4 Ecriture()	1795
6.817.1.5 Inita()	1795
6.817.1.6 Lecture()	1795
6.817.1.7 MaxiComposante()	1795
6.817.1.8 operator&&()	1795
6.817.1.9 operator()()	1795
6.817.1.10 operator*()	1796
6.817.1.11 operator*=(())	1796
6.817.1.12 operator+()	1796
6.817.1.13 operator+=()	1796
6.817.1.14 operator-() [1/2]	1796
6.817.1.15 operator-() [2/2]	1796
6.817.1.16 operator-=()	1796
6.817.1.17 operator/()	1796
6.817.1.18 operator/=()	1797
6.817.1.19 operator=()	1797
6.817.1.20 operator==(())	1797
6.817.1.21 Prod_gauche()	1797
6.817.1.22 Transpose1et2avec3et4()	1797
6.818 Référence de la classe Tenseur2HHHH	1797
6.818.1 Documentation des fonctions membres	1799
6.818.1.1 Affectation_trans_dimension()	1799
6.818.1.2 Change()	1799
6.818.1.3 ChangePlus()	1799
6.818.1.4 Ecriture()	1800
6.818.1.5 Inita()	1800
6.818.1.6 Lecture()	1800
6.818.1.7 MaxiComposante()	1800
6.818.1.8 operator&&()	1800
6.818.1.9 operator()()	1800
6.818.1.10 operator*()	1800
6.818.1.11 operator*=(())	1801
6.818.1.12 operator+()	1801
6.818.1.13 operator+=()	1801
6.818.1.14 operator-() [1/2]	1801
6.818.1.15 operator-() [2/2]	1801
6.818.1.16 operator-=()	1801
6.818.1.17 operator/()	1801

---

6.818.1.18 operator/=( )	1801
6.818.1.19 operator=( )	1801
6.818.1.20 operator==( )	1802
6.818.1.21 Prod_gauche( )	1802
6.818.1.22 Transpose1et2avec3et4( )	1802
6.819 Référence de la classe Tenseur3BB	1802
6.819.1 Description détaillée	1804
6.819.2 Documentation des fonctions membres	1805
6.819.2.1 Affectation_trans_dimension( )	1805
6.819.2.2 Coor( )	1805
6.819.2.3 Det( )	1805
6.819.2.4 Ecriture( )	1805
6.819.2.5 Init( )	1805
6.819.2.6 Inverse( )	1806
6.819.2.7 Lecture( )	1806
6.819.2.8 MaxiComposante( )	1806
6.819.2.9 Monte2Indices( )	1806
6.819.2.10 MonteDernierIndice( )	1806
6.819.2.11 MontePremierIndice( )	1806
6.819.2.12 operator"!=( )	1806
6.819.2.13 operator&&( )	1806
6.819.2.14 operator()( )	1807
6.819.2.15 operator*( ) [1/4]	1807
6.819.2.16 operator*( ) [2/4]	1807
6.819.2.17 operator*( ) [3/4]	1807
6.819.2.18 operator*( ) [4/4]	1807
6.819.2.19 operator*=( )	1807
6.819.2.20 operator+( )	1807
6.819.2.21 operator+=( )	1808
6.819.2.22 operator-( ) [1/2]	1808
6.819.2.23 operator-( ) [2/2]	1808
6.819.2.24 operator-=( )	1808
6.819.2.25 operator/( )	1808
6.819.2.26 operator/=( )	1808
6.819.2.27 operator=( )	1808
6.819.2.28 operator==( )	1808
6.819.2.29 Transpose( )	1809
6.820 Référence de la classe Tenseur3BBBB	1809
6.820.1 Documentation des fonctions membres	1811
6.820.1.1 Affectation_trans_dimension( )	1811
6.820.1.2 Change( )	1811
6.820.1.3 ChangePlus( )	1811

6.820.1.4	Ecriture()	1812
6.820.1.5	Inita()	1812
6.820.1.6	Lecture() [1/2]	1812
6.820.1.7	Lecture() [2/2]	1812
6.820.1.8	MaxiComposante() [1/2]	1812
6.820.1.9	MaxiComposante() [2/2]	1812
6.820.1.10	operator&&() [1/2]	1812
6.820.1.11	operator&&() [2/2]	1812
6.820.1.12	operator>() [1/2]	1813
6.820.1.13	operator>() [2/2]	1813
6.820.1.14	operator*() [1/2]	1813
6.820.1.15	operator*() [2/2]	1813
6.820.1.16	operator*=() [1/2]	1813
6.820.1.17	operator*=() [2/2]	1813
6.820.1.18	operator+() [1/2]	1813
6.820.1.19	operator+() [2/2]	1813
6.820.1.20	operator+=() [1/2]	1814
6.820.1.21	operator+=() [2/2]	1814
6.820.1.22	operator-() [1/4]	1814
6.820.1.23	operator-() [2/4]	1814
6.820.1.24	operator-() [3/4]	1814
6.820.1.25	operator-() [4/4]	1814
6.820.1.26	operator-=() [1/2]	1814
6.820.1.27	operator-=() [2/2]	1814
6.820.1.28	operator/() [1/2]	1815
6.820.1.29	operator/() [2/2]	1815
6.820.1.30	operator/=() [1/2]	1815
6.820.1.31	operator/=() [2/2]	1815
6.820.1.32	operator=() [1/2]	1815
6.820.1.33	operator=() [2/2]	1815
6.820.1.34	operator==() [1/2]	1815
6.820.1.35	operator==() [2/2]	1815
6.820.1.36	Prod_gauche()	1815
6.820.1.37	Transpose1et2avec3et4() [1/2]	1816
6.820.1.38	Transpose1et2avec3et4() [2/2]	1816
6.821	Référence de la classe Tenseur3BBHH	1816
6.821.1	Documentation des fonctions membres	1818
6.821.1.1	Affectation_trans_dimension()	1819
6.821.1.2	Change()	1819
6.821.1.3	ChangePlus()	1819
6.821.1.4	Ecriture()	1819
6.821.1.5	Inita()	1819

6.821.1.6 Lecture() [1/2]	1819
6.821.1.7 Lecture() [2/2]	1819
6.821.1.8 MaxiComposante() [1/2]	1820
6.821.1.9 MaxiComposante() [2/2]	1820
6.821.1.10 operator&&() [1/2]	1820
6.821.1.11 operator&&() [2/2]	1820
6.821.1.12 operator>() [1/2]	1820
6.821.1.13 operator>() [2/2]	1820
6.821.1.14 operator*() [1/2]	1820
6.821.1.15 operator*() [2/2]	1820
6.821.1.16 operator*=() [1/2]	1821
6.821.1.17 operator*=() [2/2]	1821
6.821.1.18 operator+() [1/2]	1821
6.821.1.19 operator+() [2/2]	1821
6.821.1.20 operator+=() [1/2]	1821
6.821.1.21 operator+=() [2/2]	1821
6.821.1.22 operator-() [1/4]	1821
6.821.1.23 operator-() [2/4]	1821
6.821.1.24 operator-() [3/4]	1821
6.821.1.25 operator-() [4/4]	1822
6.821.1.26 operator-=() [1/2]	1822
6.821.1.27 operator-=() [2/2]	1822
6.821.1.28 operator/() [1/2]	1822
6.821.1.29 operator/() [2/2]	1822
6.821.1.30 operator/=() [1/2]	1822
6.821.1.31 operator/=() [2/2]	1822
6.821.1.32 operator=() [1/2]	1822
6.821.1.33 operator=() [2/2]	1822
6.821.1.34 operator==() [1/2]	1823
6.821.1.35 operator==() [2/2]	1823
6.821.1.36 Prod_gauche()	1823
6.821.1.37 Transpose1et2avec3et4() [1/2]	1823
6.821.1.38 Transpose1et2avec3et4() [2/2]	1823
6.822 Référence de la classe Tenseur3BH	1823
6.822.1 Description détaillée	1826
6.822.2 Documentation des fonctions membres	1826
6.822.2.1 Affectation_trans_dimension()	1826
6.822.2.2 Coor()	1826
6.822.2.3 Det()	1826
6.822.2.4 Ecriture()	1826
6.822.2.5 II()	1827
6.822.2.6 III()	1827

6.822.2.7 Inita()	1827
6.822.2.8 Inverse()	1827
6.822.2.9 Lecture()	1827
6.822.2.10 MaxiComposante()	1827
6.822.2.11 operator"!=(())	1827
6.822.2.12 operator&&()	1827
6.822.2.13 operator>()()	1828
6.822.2.14 operator*() [1/4]	1828
6.822.2.15 operator*() [2/4]	1828
6.822.2.16 operator*() [3/4]	1828
6.822.2.17 operator*() [4/4]	1828
6.822.2.18 operator*==(())	1828
6.822.2.19 operator+()	1828
6.822.2.20 operator+==(())	1829
6.822.2.21 operator-() [1/2]	1829
6.822.2.22 operator-() [2/2]	1829
6.822.2.23 operator-==(())	1829
6.822.2.24 operator/()	1829
6.822.2.25 operator/=()	1829
6.822.2.26 operator=()	1829
6.822.2.27 operator==(())	1829
6.822.2.28 PermuteHautBas()	1830
6.822.2.29 Trace()	1830
6.822.2.30 Transpose()	1830
6.822.2.31 ValPropre() [1/2]	1830
6.822.2.32 ValPropre() [2/2]	1830
6.822.2.33 VecteursPropres()	1831
6.823 Référence de la classe Tenseur3HB	1831
6.823.1 Description détaillée	1834
6.823.2 Documentation des fonctions membres	1834
6.823.2.1 Affectation_trans_dimension()	1834
6.823.2.2 Coor()	1834
6.823.2.3 Det()	1834
6.823.2.4 Ecriture()	1834
6.823.2.5 II()	1835
6.823.2.6 III()	1835
6.823.2.7 Inita()	1835
6.823.2.8 Inverse()	1835
6.823.2.9 Lecture()	1835
6.823.2.10 MaxiComposante()	1835
6.823.2.11 operator"!=(())	1835
6.823.2.12 operator&&()	1835

6.823.2.13	operator>()	1836
6.823.2.14	operator*() [1/4]	1836
6.823.2.15	operator*() [2/4]	1836
6.823.2.16	operator*() [3/4]	1836
6.823.2.17	operator*() [4/4]	1836
6.823.2.18	operator*=( )	1836
6.823.2.19	operator+( )	1836
6.823.2.20	operator+=( )	1837
6.823.2.21	operator-( ) [1/2]	1837
6.823.2.22	operator-( ) [2/2]	1837
6.823.2.23	operator-=( )	1837
6.823.2.24	operator/( )	1837
6.823.2.25	operator/=( )	1837
6.823.2.26	operator=( )	1837
6.823.2.27	operator==( )	1837
6.823.2.28	PermuteHautBas()	1838
6.823.2.29	Trace()	1838
6.823.2.30	Transpose()	1838
6.823.2.31	ValPropre() [1/2]	1838
6.823.2.32	ValPropre() [2/2]	1838
6.823.2.33	VecteursPropres()	1839
6.824	Référence de la classe Tenseur3HH	1839
6.824.1	Description détaillée	1841
6.824.2	Documentation des fonctions membres	1842
6.824.2.1	Affectation_trans_dimension()	1842
6.824.2.2	Baisse2Indices()	1842
6.824.2.3	BaisseDernierIndice()	1842
6.824.2.4	BaissePremierIndice()	1842
6.824.2.5	Coor()	1842
6.824.2.6	Det()	1843
6.824.2.7	Ecriture()	1843
6.824.2.8	Inita()	1843
6.824.2.9	Inverse()	1843
6.824.2.10	Lecture()	1843
6.824.2.11	MaxiComposante()	1843
6.824.2.12	operator"!=( )	1843
6.824.2.13	operator&&( )	1843
6.824.2.14	operator>()()	1844
6.824.2.15	operator*() [1/4]	1844
6.824.2.16	operator*() [2/4]	1844
6.824.2.17	operator*() [3/4]	1844
6.824.2.18	operator*() [4/4]	1844

6.824.2.19 operator*=( )	1844
6.824.2.20 operator+( )	1844
6.824.2.21 operator+=( )	1845
6.824.2.22 operator-( ) [1/2]	1845
6.824.2.23 operator-( ) [2/2]	1845
6.824.2.24 operator==( )	1845
6.824.2.25 operator/( )	1845
6.824.2.26 operator/=( )	1845
6.824.2.27 operator=( )	1845
6.824.2.28 operator==( )	1845
6.824.2.29 Transpose( )	1846
6.825 Référence de la classe Tenseur3HHBB	1846
6.825.1 Documentation des fonctions membres	1848
6.825.1.1 Affectation_trans_dimension( )	1848
6.825.1.2 Change( )	1848
6.825.1.3 ChangePlus( )	1848
6.825.1.4 Ecriture( )	1849
6.825.1.5 Init( )	1849
6.825.1.6 Lecture( ) [1/2]	1849
6.825.1.7 Lecture( ) [2/2]	1849
6.825.1.8 MaxiComposante( ) [1/2]	1849
6.825.1.9 MaxiComposante( ) [2/2]	1849
6.825.1.10 operator&&( ) [1/2]	1849
6.825.1.11 operator&&( ) [2/2]	1849
6.825.1.12 operator()( ) [1/2]	1849
6.825.1.13 operator()( ) [2/2]	1850
6.825.1.14 operator*( ) [1/2]	1850
6.825.1.15 operator*( ) [2/2]	1850
6.825.1.16 operator*=( ) [1/2]	1850
6.825.1.17 operator*=( ) [2/2]	1850
6.825.1.18 operator+( ) [1/2]	1850
6.825.1.19 operator+( ) [2/2]	1850
6.825.1.20 operator+=( ) [1/2]	1851
6.825.1.21 operator+=( ) [2/2]	1851
6.825.1.22 operator-( ) [1/4]	1851
6.825.1.23 operator-( ) [2/4]	1851
6.825.1.24 operator-( ) [3/4]	1851
6.825.1.25 operator-( ) [4/4]	1851
6.825.1.26 operator==( ) [1/2]	1851
6.825.1.27 operator==( ) [2/2]	1851
6.825.1.28 operator/( ) [1/2]	1851
6.825.1.29 operator/( ) [2/2]	1852



6.825.1.30 operator/=( ) [1/2]	1852
6.825.1.31 operator/=( ) [2/2]	1852
6.825.1.32 operator=( ) [1/2]	1852
6.825.1.33 operator=( ) [2/2]	1852
6.825.1.34 operator==( ) [1/2]	1852
6.825.1.35 operator==( ) [2/2]	1852
6.825.1.36 Prod_gauche()	1852
6.825.1.37 Transpose1et2avec3et4() [1/2]	1852
6.825.1.38 Transpose1et2avec3et4() [2/2]	1853
6.826 Référence de la classe Tenseur3HHHH	1853
6.826.1 Documentation des fonctions membres	1855
6.826.1.1 Affectation_trans_dimension()	1855
6.826.1.2 Change()	1855
6.826.1.3 ChangePlus()	1856
6.826.1.4 Ecriture()	1856
6.826.1.5 Inita()	1856
6.826.1.6 Lecture() [1/2]	1856
6.826.1.7 Lecture() [2/2]	1856
6.826.1.8 MaxiComposante() [1/2]	1856
6.826.1.9 MaxiComposante() [2/2]	1856
6.826.1.10 operator&&() [1/2]	1856
6.826.1.11 operator&&() [2/2]	1857
6.826.1.12 operator>() [1/2]	1857
6.826.1.13 operator>() [2/2]	1857
6.826.1.14 operator*() [1/2]	1857
6.826.1.15 operator*() [2/2]	1857
6.826.1.16 operator*=( ) [1/2]	1857
6.826.1.17 operator*=( ) [2/2]	1857
6.826.1.18 operator+() [1/2]	1857
6.826.1.19 operator+() [2/2]	1858
6.826.1.20 operator+=( ) [1/2]	1858
6.826.1.21 operator+=( ) [2/2]	1858
6.826.1.22 operator-() [1/4]	1858
6.826.1.23 operator-() [2/4]	1858
6.826.1.24 operator-() [3/4]	1858
6.826.1.25 operator-() [4/4]	1858
6.826.1.26 operator-=( ) [1/2]	1858
6.826.1.27 operator-=( ) [2/2]	1859
6.826.1.28 operator/() [1/2]	1859
6.826.1.29 operator/() [2/2]	1859
6.826.1.30 operator/=( ) [1/2]	1859
6.826.1.31 operator/=( ) [2/2]	1859

6.826.1.32 operator=() [1/2]	1859
6.826.1.33 operator=() [2/2]	1859
6.826.1.34 operator==( ) [1/2]	1859
6.826.1.35 operator==( ) [2/2]	1859
6.826.1.36 Prod_gauche()	1860
6.826.1.37 Transpose1et2avec3et4() [1/2]	1860
6.826.1.38 Transpose1et2avec3et4() [2/2]	1860
6.827 Référence de la classe Tenseur_ns2BB	1860
6.827.1 Description détaillée	1862
6.827.2 Documentation des fonctions membres	1863
6.827.2.1 Affectation_trans_dimension()	1863
6.827.2.2 Coord()	1863
6.827.2.3 Det()	1863
6.827.2.4 Ecriture()	1863
6.827.2.5 Initia()	1863
6.827.2.6 Inverse()	1864
6.827.2.7 Lecture()	1864
6.827.2.8 MaxiComposante()	1864
6.827.2.9 Monte2Indices()	1864
6.827.2.10 MonteDernierIndice()	1864
6.827.2.11 MontePremierIndice()	1864
6.827.2.12 operator"!=( )	1864
6.827.2.13 operator&&( )	1864
6.827.2.14 operator()( )	1865
6.827.2.15 operator*( ) [1/4]	1865
6.827.2.16 operator*( ) [2/4]	1865
6.827.2.17 operator*( ) [3/4]	1865
6.827.2.18 operator*( ) [4/4]	1865
6.827.2.19 operator*=( )	1865
6.827.2.20 operator+( )	1865
6.827.2.21 operator+=( )	1866
6.827.2.22 operator-( ) [1/2]	1866
6.827.2.23 operator-( ) [2/2]	1866
6.827.2.24 operator-=( )	1866
6.827.2.25 operator/( )	1866
6.827.2.26 operator/=( )	1866
6.827.2.27 operator=( )	1866
6.827.2.28 operator==( )	1866
6.827.2.29 Transpose()	1867
6.828 Référence de la classe Tenseur_ns2HH	1867
6.828.1 Description détaillée	1869
6.828.2 Documentation des fonctions membres	1869

6.828.2.1 Affectation_trans_dimension()	1869
6.828.2.2 Baisse2Indices()	1870
6.828.2.3 BaisseDernierIndice()	1870
6.828.2.4 BaissePremierIndice()	1870
6.828.2.5 Coor()	1870
6.828.2.6 Det()	1870
6.828.2.7 Ecriture()	1870
6.828.2.8 Inita()	1870
6.828.2.9 Inverse()	1871
6.828.2.10 Lecture()	1871
6.828.2.11 MaxiComposante()	1871
6.828.2.12 operator"!=(())	1871
6.828.2.13 operator&&()	1871
6.828.2.14 operator>()()	1871
6.828.2.15 operator*() [1/4]	1871
6.828.2.16 operator*() [2/4]	1871
6.828.2.17 operator*() [3/4]	1872
6.828.2.18 operator*() [4/4]	1872
6.828.2.19 operator*==(())	1872
6.828.2.20 operator+()	1872
6.828.2.21 operator+==(())	1872
6.828.2.22 operator-() [1/2]	1872
6.828.2.23 operator-() [2/2]	1872
6.828.2.24 operator-==(())	1873
6.828.2.25 operator/()	1873
6.828.2.26 operator/=()	1873
6.828.2.27 operator=()	1873
6.828.2.28 operator==(())	1873
6.828.2.29 Transpose()	1873
6.829 Référence de la classe Tenseur_ns3BB	1873
6.829.1 Description détaillée	1876
6.829.2 Documentation des fonctions membres	1876
6.829.2.1 Affectation_trans_dimension()	1876
6.829.2.2 Coor()	1876
6.829.2.3 Det()	1876
6.829.2.4 Ecriture()	1877
6.829.2.5 Inita()	1877
6.829.2.6 Inverse()	1877
6.829.2.7 Lecture()	1877
6.829.2.8 MaxiComposante()	1877
6.829.2.9 Monte2Indices()	1877
6.829.2.10 MonteDernierIndice()	1877

6.829.2.11 MontePremierIndice()	1877
6.829.2.12 operator"!=( )	1878
6.829.2.13 operator&&( )	1878
6.829.2.14 operator()( )	1878
6.829.2.15 operator*( ) [1/4]	1878
6.829.2.16 operator*( ) [2/4]	1878
6.829.2.17 operator*( ) [3/4]	1878
6.829.2.18 operator*( ) [4/4]	1878
6.829.2.19 operator*=( )	1879
6.829.2.20 operator+( )	1879
6.829.2.21 operator+=( )	1879
6.829.2.22 operator-( ) [1/2]	1879
6.829.2.23 operator-( ) [2/2]	1879
6.829.2.24 operator--( )	1879
6.829.2.25 operator/( )	1879
6.829.2.26 operator/=( )	1879
6.829.2.27 operator=( )	1880
6.829.2.28 operator==( )	1880
6.829.2.29 Transpose( )	1880
6.830 Référence de la classe Tenseur_ns3HH	1880
6.830.1 Description détaillée	1882
6.830.2 Documentation des fonctions membres	1883
6.830.2.1 Affectation_trans_dimension( )	1883
6.830.2.2 Baisse2Indices( )	1883
6.830.2.3 BaisseDernierIndice( )	1883
6.830.2.4 BaissePremierIndice( )	1883
6.830.2.5 Coor( )	1883
6.830.2.6 Det( )	1884
6.830.2.7 Ecriture( )	1884
6.830.2.8 Inita( )	1884
6.830.2.9 Inverse( )	1884
6.830.2.10 Lecture( )	1884
6.830.2.11 MaxiComposante( )	1884
6.830.2.12 operator"!=( )	1884
6.830.2.13 operator&&( )	1884
6.830.2.14 operator()( )	1885
6.830.2.15 operator*( ) [1/4]	1885
6.830.2.16 operator*( ) [2/4]	1885
6.830.2.17 operator*( ) [3/4]	1885
6.830.2.18 operator*( ) [4/4]	1885
6.830.2.19 operator*=( )	1885
6.830.2.20 operator+( )	1885

6.830.2.21 operator+=()	1886
6.830.2.22 operator-() [1/2]	1886
6.830.2.23 operator-() [2/2]	1886
6.830.2.24 operator-=()	1886
6.830.2.25 operator/()	1886
6.830.2.26 operator/=()	1886
6.830.2.27 operator=()	1886
6.830.2.28 operator==(())	1886
6.830.2.29 Transpose()	1887
6.831 Référence de la classe TenseurBB	1887
6.831.1 Description détaillée	1889
6.831.2 Documentation des fonctions membres	1889
6.831.2.1 Affectation_trans_dimension()	1889
6.831.2.2 BaseAbsolue()	1889
6.831.2.3 Baselocale()	1890
6.831.2.4 ChBase()	1890
6.831.2.5 Coord()	1890
6.831.2.6 Det()	1890
6.831.2.7 Ecriture()	1890
6.831.2.8 Inita()	1890
6.831.2.9 Inverse()	1891
6.831.2.10 Lecture()	1891
6.831.2.11 MaxiComposante()	1891
6.831.2.12 Monte2Indices()	1891
6.831.2.13 MonteDernierIndice()	1891
6.831.2.14 MontePremierIndice()	1891
6.831.2.15 operator"!=(())	1891
6.831.2.16 operator&&()	1891
6.831.2.17 operator>()()	1892
6.831.2.18 operator*() [1/4]	1892
6.831.2.19 operator*() [2/4]	1892
6.831.2.20 operator*() [3/4]	1892
6.831.2.21 operator*() [4/4]	1892
6.831.2.22 operator*=(())	1892
6.831.2.23 operator+()	1892
6.831.2.24 operator+=()	1893
6.831.2.25 operator-() [1/2]	1893
6.831.2.26 operator-() [2/2]	1893
6.831.2.27 operator-=()	1893
6.831.2.28 operator/()	1893
6.831.2.29 operator/=()	1893
6.831.2.30 operator=()	1893

6.831.2.31 operator==( )	1893
6.831.2.32 Transpose( )	1894
6.831.2.33 Var_tenseur_dans_nouvelle_base( )	1894
6.832 Référence de la classe TenseurBBBB	1894
6.832.1 Description détaillée	1897
6.833 Référence de la classe TenseurBBHH	1897
6.833.1 Description détaillée	1899
6.834 Référence de la classe TenseurBH	1900
6.834.1 Description détaillée	1902
6.834.2 Documentation des fonctions membres	1902
6.834.2.1 Affectation_trans_dimension( )	1903
6.834.2.2 BaseAbsolue( )	1903
6.834.2.3 BaseLocale( )	1903
6.834.2.4 ChBase( )	1903
6.834.2.5 Coor( )	1904
6.834.2.6 Det( )	1904
6.834.2.7 Ecriture( )	1904
6.834.2.8 II( )	1904
6.834.2.9 III( )	1904
6.834.2.10 Inita( )	1904
6.834.2.11 Inverse( )	1904
6.834.2.12 Lecture( )	1905
6.834.2.13 MaxiComposante( )	1905
6.834.2.14 operator&&( )	1905
6.834.2.15 operator()( )	1905
6.834.2.16 operator*( ) [1/3]	1905
6.834.2.17 operator*( ) [2/3]	1905
6.834.2.18 operator*( ) [3/3]	1905
6.834.2.19 operator*=( )	1905
6.834.2.20 operator+( )	1906
6.834.2.21 operator+=( )	1906
6.834.2.22 operator-( ) [1/2]	1906
6.834.2.23 operator-( ) [2/2]	1906
6.834.2.24 operator-=( )	1906
6.834.2.25 operator/( )	1906
6.834.2.26 operator/=( )	1906
6.834.2.27 operator=( )	1907
6.834.2.28 operator==( )	1907
6.834.2.29 PermuteHautBas( )	1907
6.834.2.30 Trace( )	1907
6.834.2.31 Transpose( )	1907
6.834.2.32 ValPropre( ) [1/2]	1907

---

6.834.2.33 ValPropre() [2/2]	1907
6.834.2.34 Var_tenseur_dans_nouvelle_base()	1908
6.834.2.35 VecteursPropres()	1908
6.835 Référence de la classe TenseurBHBH	1909
6.835.1 Description détaillée	1910
6.836 Référence de la classe TenseurBHHB	1910
6.836.1 Description détaillée	1911
6.837 Référence de la classe TenseurHB	1911
6.837.1 Description détaillée	1914
6.837.2 Documentation des fonctions membres	1914
6.837.2.1 Affectation_trans_dimension()	1914
6.837.2.2 BaseAbsolue()	1914
6.837.2.3 ChBase()	1915
6.837.2.4 Coor()	1915
6.837.2.5 Det()	1915
6.837.2.6 Ecriture()	1915
6.837.2.7 II()	1915
6.837.2.8 III()	1915
6.837.2.9 Inita()	1916
6.837.2.10 Inverse()	1916
6.837.2.11 Lecture()	1916
6.837.2.12 MaxiComposante()	1916
6.837.2.13 operator&&()	1916
6.837.2.14 operator>()()	1916
6.837.2.15 operator*() [1/3]	1916
6.837.2.16 operator*() [2/3]	1916
6.837.2.17 operator*() [3/3]	1917
6.837.2.18 operator*=(())	1917
6.837.2.19 operator+()	1917
6.837.2.20 operator+=()	1917
6.837.2.21 operator-() [1/2]	1917
6.837.2.22 operator-() [2/2]	1917
6.837.2.23 operator-=(())	1917
6.837.2.24 operator/()	1918
6.837.2.25 operator/=()	1918
6.837.2.26 operator=()	1918
6.837.2.27 operator==(())	1918
6.837.2.28 PermuteHautBas()	1918
6.837.2.29 Trace()	1918
6.837.2.30 Transpose()	1918
6.837.2.31 ValPropre() [1/2]	1918
6.837.2.32 ValPropre() [2/2]	1919

6.837.2.33 Var_tenseur_dans_nouvelle_base()	1919
6.837.2.34 VecteursPropres()	1919
6.838 Référence de la classe TenseurHBBH	1920
6.838.1 Description détaillée	1921
6.839 Référence de la classe TenseurHBHB	1921
6.839.1 Description détaillée	1922
6.840 Référence de la classe TenseurHH	1922
6.840.1 Description détaillée	1924
6.840.2 Documentation des fonctions membres	1925
6.840.2.1 Affectation_trans_dimension()	1925
6.840.2.2 Baisse2Indices()	1925
6.840.2.3 BaisseDernierIndice()	1925
6.840.2.4 BaissePremierIndice()	1925
6.840.2.5 BaseAbsolue()	1925
6.840.2.6 Baselocale()	1926
6.840.2.7 ChBase()	1926
6.840.2.8 Coord()	1926
6.840.2.9 Det()	1926
6.840.2.10 Ecriture()	1927
6.840.2.11 Inita()	1927
6.840.2.12 Inverse()	1927
6.840.2.13 Lecture()	1927
6.840.2.14 MaxiComposante()	1927
6.840.2.15 operator"!=(())	1927
6.840.2.16 operator&&()	1927
6.840.2.17 operator>()()	1928
6.840.2.18 operator*() [1/4]	1928
6.840.2.19 operator*() [2/4]	1928
6.840.2.20 operator*() [3/4]	1928
6.840.2.21 operator*() [4/4]	1928
6.840.2.22 operator*=(())	1928
6.840.2.23 operator+()	1928
6.840.2.24 operator+=(())	1929
6.840.2.25 operator-() [1/2]	1929
6.840.2.26 operator-() [2/2]	1929
6.840.2.27 operator-=(())	1929
6.840.2.28 operator/()	1929
6.840.2.29 operator/=()	1929
6.840.2.30 operator=()	1929
6.840.2.31 operator==(())	1929
6.840.2.32 Transpose()	1930
6.840.2.33 Var_tenseur_dans_nouvelle_base()	1930



---

6.841	Référence de la classe TenseurHHBB	1930
6.841.1	Description détaillée	1932
6.842	Référence de la classe TenseurHHHH	1933
6.842.1	Description détaillée	1935
6.843	Référence de la classe TenseurQ1_troisSym_BBBB	1936
6.843.1	Documentation des fonctions membres	1937
6.843.1.1	Affectation_trans_dimension()	1937
6.843.1.2	Change()	1937
6.843.1.3	ChangePlus()	1937
6.843.1.4	Ecriture()	1938
6.843.1.5	Inita()	1938
6.843.1.6	Lecture()	1938
6.843.1.7	MaxiComposante()	1938
6.843.1.8	operator&&()	1938
6.843.1.9	operator()()	1938
6.843.1.10	operator*()	1938
6.843.1.11	operator*=(())	1939
6.843.1.12	operator+()	1939
6.843.1.13	operator+=(())	1939
6.843.1.14	operator-() [1/2]	1939
6.843.1.15	operator-() [2/2]	1939
6.843.1.16	operator-=(())	1939
6.843.1.17	operator/()	1939
6.843.1.18	operator/=(())	1939
6.843.1.19	operator=()	1939
6.843.1.20	operator==(())	1940
6.843.1.21	Prod_gauche()	1940
6.843.1.22	Transpose1et2avec3et4()	1940
6.844	Référence de la classe TenseurQ1_troisSym_HHHH	1940
6.844.1	Documentation des fonctions membres	1942
6.844.1.1	Affectation_trans_dimension()	1942
6.844.1.2	Change()	1942
6.844.1.3	ChangePlus()	1942
6.844.1.4	Ecriture()	1942
6.844.1.5	Inita()	1942
6.844.1.6	Lecture()	1943
6.844.1.7	MaxiComposante()	1943
6.844.1.8	operator&&()	1943
6.844.1.9	operator()()	1943
6.844.1.10	operator*()	1943
6.844.1.11	operator*=(())	1943
6.844.1.12	operator+()	1943

6.844.1.13 operator+=()	1943
6.844.1.14 operator-() [1/2]	1944
6.844.1.15 operator-() [2/2]	1944
6.844.1.16 operator-=()	1944
6.844.1.17 operator/()	1944
6.844.1.18 operator/=()	1944
6.844.1.19 operator=()	1944
6.844.1.20 operator==(())	1944
6.844.1.21 Prod_gauche()	1944
6.844.1.22 Transpose1et2avec3et4()	1944
6.845 Référence de la classe TenseurQ1geneBHBH	1945
6.845.1 Documentation des fonctions membres	1946
6.845.1.1 Affectation_trans_dimension()	1946
6.845.1.2 Change()	1946
6.845.1.3 ChangePlus()	1946
6.845.1.4 Ecriture()	1946
6.845.1.5 Inita()	1947
6.845.1.6 Lecture()	1947
6.845.1.7 MaxiComposante()	1947
6.845.1.8 operator&&()	1947
6.845.1.9 operator()()	1947
6.845.1.10 operator*()	1947
6.845.1.11 operator*=(())	1947
6.845.1.12 operator+()	1947
6.845.1.13 operator+=()	1948
6.845.1.14 operator-() [1/2]	1948
6.845.1.15 operator-() [2/2]	1948
6.845.1.16 operator-=()	1948
6.845.1.17 operator/()	1948
6.845.1.18 operator/=()	1948
6.845.1.19 operator=()	1948
6.845.1.20 operator==(())	1948
6.845.1.21 Prod_gauche()	1948
6.845.1.22 Transpose1et2avec3et4()	1949
6.846 Référence de la classe TenseurQ1geneBHHB	1949
6.846.1 Documentation des fonctions membres	1950
6.846.1.1 Affectation_trans_dimension()	1950
6.846.1.2 Change()	1950
6.846.1.3 ChangePlus()	1950
6.846.1.4 Ecriture()	1951
6.846.1.5 Inita()	1951
6.846.1.6 Lecture()	1951

6.846.1.7	MaxiComposante()	1951
6.846.1.8	operator&&()	1951
6.846.1.9	operator()()	1951
6.846.1.10	operator*()	1951
6.846.1.11	operator*=(())	1951
6.846.1.12	operator+()	1952
6.846.1.13	operator+=(())	1952
6.846.1.14	operator-() [1/2]	1952
6.846.1.15	operator-() [2/2]	1952
6.846.1.16	operator-=()	1952
6.846.1.17	operator/()	1952
6.846.1.18	operator/=()	1952
6.846.1.19	operator=()	1952
6.846.1.20	operator==(())	1952
6.846.1.21	Prod_gauche()	1953
6.846.1.22	Transpose1et2avec3et4()	1953
6.847	Référence de la classe TenseurQ1geneHBBH	1953
6.847.1	Documentation des fonctions membres	1954
6.847.1.1	Affectation_trans_dimension()	1954
6.847.1.2	Change()	1954
6.847.1.3	ChangePlus()	1955
6.847.1.4	Ecriture()	1955
6.847.1.5	Inita()	1955
6.847.1.6	Lecture()	1955
6.847.1.7	MaxiComposante()	1955
6.847.1.8	operator&&()	1955
6.847.1.9	operator()()	1955
6.847.1.10	operator*()	1955
6.847.1.11	operator*=(())	1956
6.847.1.12	operator+()	1956
6.847.1.13	operator+=(())	1956
6.847.1.14	operator-() [1/2]	1956
6.847.1.15	operator-() [2/2]	1956
6.847.1.16	operator-=()	1956
6.847.1.17	operator/()	1956
6.847.1.18	operator/=()	1956
6.847.1.19	operator=()	1957
6.847.1.20	operator==(())	1957
6.847.1.21	Prod_gauche()	1957
6.847.1.22	Transpose1et2avec3et4()	1957
6.848	Référence de la classe TenseurQ1geneHBHB	1957
6.848.1	Documentation des fonctions membres	1958

6.848.1.1 Affectation_trans_dimension()	1959
6.848.1.2 Change()	1959
6.848.1.3 ChangePlus()	1959
6.848.1.4 Ecriture()	1959
6.848.1.5 Inita()	1959
6.848.1.6 Lecture()	1959
6.848.1.7 MaxiComposante()	1959
6.848.1.8 operator&&()	1960
6.848.1.9 operator()()	1960
6.848.1.10 operator*()	1960
6.848.1.11 operator*=(())	1960
6.848.1.12 operator+()	1960
6.848.1.13 operator+=()	1960
6.848.1.14 operator-() [1/2]	1960
6.848.1.15 operator-() [2/2]	1960
6.848.1.16 operator-=(())	1961
6.848.1.17 operator/()	1961
6.848.1.18 operator/=()	1961
6.848.1.19 operator=()	1961
6.848.1.20 operator==(())	1961
6.848.1.21 Prod_gauche()	1961
6.848.1.22 Transpose1et2avec3et4()	1961
6.849 Référence de la classe TenseurQ2_troisSym_BBBB	1962
6.849.1 Documentation des fonctions membres	1963
6.849.1.1 Affectation_trans_dimension()	1963
6.849.1.2 Change()	1963
6.849.1.3 ChangePlus()	1964
6.849.1.4 Ecriture()	1964
6.849.1.5 Inita()	1964
6.849.1.6 Lecture()	1964
6.849.1.7 MaxiComposante()	1964
6.849.1.8 operator&&()	1964
6.849.1.9 operator()()	1964
6.849.1.10 operator*()	1964
6.849.1.11 operator*=(())	1965
6.849.1.12 operator+()	1965
6.849.1.13 operator+=()	1965
6.849.1.14 operator-() [1/2]	1965
6.849.1.15 operator-() [2/2]	1965
6.849.1.16 operator-=(())	1965
6.849.1.17 operator/()	1965
6.849.1.18 operator/=()	1965

---

6.849.1.19 operator=()	1966
6.849.1.20 operator==(())	1966
6.849.1.21 Prod_gauche()	1966
6.849.1.22 Transpose1et2avec3et4()	1966
6.850 Référence de la classe TenseurQ2_troisSym_HHHH	1966
6.850.1 Documentation des fonctions membres	1968
6.850.1.1 Affectation_trans_dimension()	1968
6.850.1.2 Change()	1968
6.850.1.3 ChangePlus()	1968
6.850.1.4 Ecriture()	1968
6.850.1.5 Inita()	1968
6.850.1.6 Lecture()	1969
6.850.1.7 MaxiComposante()	1969
6.850.1.8 operator&&()	1969
6.850.1.9 operator()(())	1969
6.850.1.10 operator*()	1969
6.850.1.11 operator*==(())	1969
6.850.1.12 operator+()	1969
6.850.1.13 operator+==(())	1969
6.850.1.14 operator-() [1/2]	1970
6.850.1.15 operator-() [2/2]	1970
6.850.1.16 operator-==(())	1970
6.850.1.17 operator/()	1970
6.850.1.18 operator/=()	1970
6.850.1.19 operator=()	1970
6.850.1.20 operator==(())	1970
6.850.1.21 Prod_gauche()	1970
6.850.1.22 Transpose1et2avec3et4()	1970
6.851 Référence de la classe TenseurQ2geneBBBB	1971
6.851.1 Documentation des fonctions membres	1972
6.851.1.1 Affectation_trans_dimension()	1972
6.851.1.2 Change()	1973
6.851.1.3 ChangePlus()	1973
6.851.1.4 Ecriture()	1973
6.851.1.5 Inita()	1973
6.851.1.6 Lecture()	1973
6.851.1.7 MaxiComposante()	1973
6.851.1.8 operator&&()	1973
6.851.1.9 operator()(())	1973
6.851.1.10 operator*()	1974
6.851.1.11 operator*==(())	1974
6.851.1.12 operator+()	1974

6.851.1.13 operator+=()	1974
6.851.1.14 operator-() [1/2]	1974
6.851.1.15 operator-() [2/2]	1974
6.851.1.16 operator-=()	1974
6.851.1.17 operator/()	1974
6.851.1.18 operator/=()	1975
6.851.1.19 operator=()	1975
6.851.1.20 operator==(())	1975
6.851.1.21 Prod_gauche()	1975
6.851.1.22 Transpose1et2avec3et4()	1975
6.852 Référence de la classe TenseurQ2geneBBHH	1975
6.852.1 Documentation des fonctions membres	1977
6.852.1.1 Affectation_trans_dimension()	1977
6.852.1.2 Change()	1977
6.852.1.3 ChangePlus()	1977
6.852.1.4 Ecriture()	1977
6.852.1.5 Inita()	1978
6.852.1.6 Lecture()	1978
6.852.1.7 MaxiComposante()	1978
6.852.1.8 operator&&()	1978
6.852.1.9 operator()()	1978
6.852.1.10 operator*()	1978
6.852.1.11 operator*=(())	1978
6.852.1.12 operator+()	1978
6.852.1.13 operator+=()	1979
6.852.1.14 operator-() [1/2]	1979
6.852.1.15 operator-() [2/2]	1979
6.852.1.16 operator-=()	1979
6.852.1.17 operator/()	1979
6.852.1.18 operator/=()	1979
6.852.1.19 operator=()	1979
6.852.1.20 operator==(())	1979
6.852.1.21 Prod_gauche()	1979
6.852.1.22 Transpose1et2avec3et4()	1980
6.853 Référence de la classe TenseurQ2geneHHBB	1980
6.853.1 Documentation des fonctions membres	1981
6.853.1.1 Affectation_trans_dimension()	1981
6.853.1.2 Change()	1982
6.853.1.3 ChangePlus()	1982
6.853.1.4 Ecriture()	1982
6.853.1.5 Inita()	1982
6.853.1.6 Lecture()	1982

6.853.1.7 MaxiComposante()	1982
6.853.1.8 operator&&()	1982
6.853.1.9 operator()()	1982
6.853.1.10 operator*()	1983
6.853.1.11 operator*=(())	1983
6.853.1.12 operator+()	1983
6.853.1.13 operator+=(())	1983
6.853.1.14 operator-() [1/2]	1983
6.853.1.15 operator-() [2/2]	1983
6.853.1.16 operator-=(())	1983
6.853.1.17 operator/()	1983
6.853.1.18 operator/=()	1984
6.853.1.19 operator=()	1984
6.853.1.20 operator==(())	1984
6.853.1.21 Prod_gauche()	1984
6.853.1.22 Transpose1et2avec3et4()	1984
6.854 Référence de la classe TenseurQ2geneHHHH	1984
6.854.1 Documentation des fonctions membres	1986
6.854.1.1 Affectation_trans_dimension()	1986
6.854.1.2 Change()	1986
6.854.1.3 ChangePlus()	1986
6.854.1.4 Ecriture()	1986
6.854.1.5 Inita()	1987
6.854.1.6 Lecture()	1987
6.854.1.7 MaxiComposante()	1987
6.854.1.8 operator&&()	1987
6.854.1.9 operator()()	1987
6.854.1.10 operator*()	1987
6.854.1.11 operator*=(())	1987
6.854.1.12 operator+()	1987
6.854.1.13 operator+=(())	1988
6.854.1.14 operator-() [1/2]	1988
6.854.1.15 operator-() [2/2]	1988
6.854.1.16 operator-=(())	1988
6.854.1.17 operator/()	1988
6.854.1.18 operator/=()	1988
6.854.1.19 operator=()	1988
6.854.1.20 operator==(())	1988
6.854.1.21 Prod_gauche()	1988
6.854.1.22 Transpose1et2avec3et4()	1989
6.855 Référence de la classe TenseurQ3_troisSym_BBBB	1989
6.855.1 Documentation des fonctions membres	1990

6.855.1.1 Affectation_trans_dimension()	1991
6.855.1.2 Change()	1991
6.855.1.3 ChangePlus()	1991
6.855.1.4 Ecriture()	1991
6.855.1.5 Inita()	1991
6.855.1.6 Lecture()	1991
6.855.1.7 MaxiComposante()	1991
6.855.1.8 operator&&()	1992
6.855.1.9 operator()()	1992
6.855.1.10 operator*()	1992
6.855.1.11 operator*=(())	1992
6.855.1.12 operator+()	1992
6.855.1.13 operator+=(())	1992
6.855.1.14 operator-() [1/2]	1992
6.855.1.15 operator-() [2/2]	1992
6.855.1.16 operator-=(())	1993
6.855.1.17 operator/()	1993
6.855.1.18 operator/=(())	1993
6.855.1.19 operator=()	1993
6.855.1.20 operator==(())	1993
6.855.1.21 Prod_gauche()	1993
6.855.1.22 Transpose1et2avec3et4()	1993
6.856 Référence de la classe TenseurQ3_troisSym_HHHH	1994
6.856.1 Documentation des fonctions membres	1995
6.856.1.1 Affectation_trans_dimension()	1995
6.856.1.2 Change()	1996
6.856.1.3 ChangePlus()	1996
6.856.1.4 Ecriture()	1996
6.856.1.5 Inita()	1996
6.856.1.6 Lecture()	1996
6.856.1.7 MaxiComposante()	1996
6.856.1.8 operator&&()	1996
6.856.1.9 operator()()	1996
6.856.1.10 operator*()	1997
6.856.1.11 operator*=(())	1997
6.856.1.12 operator+()	1997
6.856.1.13 operator+=(())	1997
6.856.1.14 operator-() [1/2]	1997
6.856.1.15 operator-() [2/2]	1997
6.856.1.16 operator-=(())	1997
6.856.1.17 operator/()	1997
6.856.1.18 operator/=(())	1998



6.856.1.19 operator=()	1998
6.856.1.20 operator==(())	1998
6.856.1.21 Prod_gauche()	1998
6.856.1.22 Transpose1et2avec3et4()	1998
6.857 Référence de la classe TenseurQ3geneBBBB	1998
6.857.1 Documentation des fonctions membres	2000
6.857.1.1 Affectation_trans_dimension()	2000
6.857.1.2 Change()	2000
6.857.1.3 ChangePlus()	2000
6.857.1.4 Ecriture()	2000
6.857.1.5 Inita()	2001
6.857.1.6 Lecture()	2001
6.857.1.7 MaxiComposante()	2001
6.857.1.8 operator&&()	2001
6.857.1.9 operator()(())	2001
6.857.1.10 operator*()	2001
6.857.1.11 operator*==(())	2001
6.857.1.12 operator+()	2001
6.857.1.13 operator+==(())	2002
6.857.1.14 operator-() [1/2]	2002
6.857.1.15 operator-() [2/2]	2002
6.857.1.16 operator-==(())	2002
6.857.1.17 operator/()	2002
6.857.1.18 operator/=()	2002
6.857.1.19 operator=()	2002
6.857.1.20 operator==(())	2002
6.857.1.21 Prod_gauche()	2002
6.857.1.22 Transpose1et2avec3et4()	2003
6.858 Référence de la classe TenseurQ3geneBBHH	2003
6.858.1 Documentation des fonctions membres	2004
6.858.1.1 Affectation_trans_dimension()	2004
6.858.1.2 Change()	2005
6.858.1.3 ChangePlus()	2005
6.858.1.4 Ecriture()	2005
6.858.1.5 Inita()	2005
6.858.1.6 Lecture()	2005
6.858.1.7 MaxiComposante()	2005
6.858.1.8 operator&&()	2005
6.858.1.9 operator()(())	2005
6.858.1.10 operator*()	2006
6.858.1.11 operator*==(())	2006
6.858.1.12 operator+()	2006

6.858.1.13 operator+=()	2006
6.858.1.14 operator-() [1/2]	2006
6.858.1.15 operator-() [2/2]	2006
6.858.1.16 operator-=()	2006
6.858.1.17 operator/()	2006
6.858.1.18 operator/=()	2007
6.858.1.19 operator=()	2007
6.858.1.20 operator==(())	2007
6.858.1.21 Prod_gauche()	2007
6.858.1.22 Transpose1et2avec3et4()	2007
6.859 Référence de la classe TenseurQ3geneHHBB	2007
6.859.1 Documentation des fonctions membres	2009
6.859.1.1 Affectation_trans_dimension()	2009
6.859.1.2 Change()	2009
6.859.1.3 ChangePlus()	2009
6.859.1.4 Ecriture()	2009
6.859.1.5 Inita()	2010
6.859.1.6 Lecture()	2010
6.859.1.7 MaxiComposante()	2010
6.859.1.8 operator&&()	2010
6.859.1.9 operator()()	2010
6.859.1.10 operator*()	2010
6.859.1.11 operator*=(())	2010
6.859.1.12 operator+()	2010
6.859.1.13 operator+=()	2011
6.859.1.14 operator-() [1/2]	2011
6.859.1.15 operator-() [2/2]	2011
6.859.1.16 operator-=()	2011
6.859.1.17 operator/()	2011
6.859.1.18 operator/=()	2011
6.859.1.19 operator=()	2011
6.859.1.20 operator==(())	2011
6.859.1.21 Prod_gauche()	2011
6.859.1.22 Transpose1et2avec3et4()	2012
6.860 Référence de la classe TenseurQ3geneHHHH	2012
6.860.1 Documentation des fonctions membres	2013
6.860.1.1 Affectation_trans_dimension()	2013
6.860.1.2 Change()	2014
6.860.1.3 ChangePlus()	2014
6.860.1.4 Ecriture()	2014
6.860.1.5 Inita()	2014
6.860.1.6 Lecture()	2014

6.860.1.7	MaxiComposante()	2014
6.860.1.8	operator&&()	2014
6.860.1.9	operator()()	2014
6.860.1.10	operator*()	2015
6.860.1.11	operator*=(())	2015
6.860.1.12	operator+()	2015
6.860.1.13	operator+=(())	2015
6.860.1.14	operator-() [1/2]	2015
6.860.1.15	operator-() [2/2]	2015
6.860.1.16	operator-=(())	2015
6.860.1.17	operator/()	2015
6.860.1.18	operator/=(())	2016
6.860.1.19	operator=()	2016
6.860.1.20	operator==(())	2016
6.860.1.21	Prod_gauche()	2016
6.860.1.22	Transpose1et2avec3et4()	2016
6.861	Référence de la classe Tetra	2016
6.861.1	Documentation des fonctions membres	2018
6.861.1.1	new_frontiere_lin()	2018
6.861.1.2	new_frontiere_surf()	2018
6.862	Référence de la classe TetraMemb	2018
6.862.1	Documentation des fonctions membres	2021
6.862.1.1	Active_ddl_Sigma()	2021
6.862.1.2	Active_premier_ddl_Sigma()	2021
6.862.1.3	ContraintesAbsolues()	2022
6.862.1.4	Dim_sig_eps()	2022
6.862.1.5	ErreurElement()	2022
6.862.1.6	Inactive_ddl_Sigma()	2022
6.862.1.7	LectureContraintes()	2022
6.862.1.8	Long_arrete_mini_sur_c()	2022
6.862.1.9	Plus_ddl_Sigma()	2022
6.862.1.10	Tableau_de_Sig1()	2022
6.863	Référence de la classe TetraQ	2023
6.863.1	Documentation des fonctions membres	2024
6.863.1.1	new_frontiere_lin()	2024
6.863.1.2	new_frontiere_surf()	2024
6.864	Référence de la classe TetraQ_15pti	2025
6.864.1	Documentation des fonctions membres	2026
6.864.1.1	new_frontiere_lin()	2026
6.864.1.2	new_frontiere_surf()	2026
6.865	Référence de la classe TetraQ_cm1pti	2027
6.865.1	Documentation des fonctions membres	2028

---

6.865.1.1 new_frontiere_lin()	2028
6.865.1.2 new_frontiere_surf()	2028
6.866 Référence de la classe ThermoDonnee	2028
6.867 Référence de la classe LesCondLim::TorseurReac	2029
6.868 Référence de la classe TreloarN	2030
6.868.1 Documentation des fonctions membres	2031
6.868.1.1 Affiche()	2031
6.868.1.2 Ecriture_base_info_loi()	2031
6.868.1.3 Info_commande LoisDeComp()	2031
6.868.1.4 Lecture_base_info_loi()	2031
6.868.1.5 LectureDonneesParticulieres()	2031
6.868.1.6 Nouvelle_loi_identique()	2032
6.868.1.7 Potentiel()	2032
6.868.1.8 Potentiel_et_var()	2032
6.868.1.9 TestComplet()	2032
6.869 Référence de la classe TriaAxiL1	2033
6.869.1 Documentation des fonctions membres	2034
6.869.1.1 new_frontiere_lin()	2034
6.869.1.2 new_frontiere_surf()	2034
6.870 Référence de la classe TriaAxiMemb	2034
6.870.1 Documentation des fonctions membres	2037
6.870.1.1 Active_ddl_Sigma()	2037
6.870.1.2 Active_premier_ddl_Sigma()	2037
6.870.1.3 ContraintesAbsolues()	2038
6.870.1.4 Dim_sig_eps()	2038
6.870.1.5 ErreurElement()	2038
6.870.1.6 Inactive_ddl_Sigma()	2038
6.870.1.7 LectureContraintes()	2038
6.870.1.8 Long_arrete_mini_sur_c()	2038
6.870.1.9 Plus_ddl_Sigma()	2038
6.870.1.10 Tableau_de_Sig1()	2038
6.871 Référence de la classe TriaAxiQ3	2039
6.871.1 Documentation des fonctions membres	2040
6.871.1.1 new_frontiere_lin()	2040
6.871.1.2 new_frontiere_surf()	2040
6.872 Référence de la classe TriaAxiQ3_cm1pti	2040
6.872.1 Documentation des fonctions membres	2042
6.872.1.1 new_frontiere_lin()	2042
6.872.1.2 new_frontiere_surf()	2042
6.873 Référence de la classe TriaAxiQ3_cmpti1003	2042
6.873.1 Documentation des fonctions membres	2044
6.873.1.1 new_frontiere_lin()	2044

---

6.873.1.2 new_frontiere_surf()	2044
6.874 Référence de la classe TriaCub	2044
6.874.1 Documentation des fonctions membres	2046
6.874.1.1 new_frontiere_lin()	2046
6.874.1.2 new_frontiere_surf()	2046
6.875 Référence de la classe TriaCub_cm4pti	2046
6.875.1 Documentation des fonctions membres	2048
6.875.1.1 new_frontiere_lin()	2048
6.875.1.2 new_frontiere_surf()	2048
6.876 Référence de la classe TriaMemb	2048
6.876.1 Documentation des fonctions membres	2051
6.876.1.1 Active_ddl_Sigma()	2052
6.876.1.2 Active_premier_ddl_Sigma()	2052
6.876.1.3 ContraintesAbsolues()	2052
6.876.1.4 Dim_sig_eps()	2052
6.876.1.5 EpaisseurMoyenne()	2052
6.876.1.6 Epaisseurs()	2052
6.876.1.7 ErreurElement()	2052
6.876.1.8 Inactive_ddl_Sigma()	2052
6.876.1.9 LectureContraintes()	2053
6.876.1.10 Les_types_particuliers_internes()	2053
6.876.1.11 Long_arrete_mini_sur_c()	2053
6.876.1.12 Plus_ddl_Sigma()	2053
6.876.1.13 Tableau_de_Sig1()	2053
6.877 Référence de la classe TriaMembL1	2053
6.877.1 Documentation des fonctions membres	2055
6.877.1.1 new_frontiere_lin()	2055
6.877.1.2 new_frontiere_surf()	2055
6.878 Référence de la classe TriaMembQ3	2055
6.878.1 Documentation des fonctions membres	2057
6.878.1.1 new_frontiere_lin()	2057
6.878.1.2 new_frontiere_surf()	2057
6.879 Référence de la classe TriaMembQ3_cm1pti	2057
6.879.1 Documentation des fonctions membres	2059
6.879.1.1 new_frontiere_lin()	2059
6.879.1.2 new_frontiere_surf()	2059
6.880 Référence de la classe TriaQ3_cmpti1003	2059
6.880.1 Documentation des fonctions membres	2061
6.880.1.1 new_frontiere_lin()	2061
6.880.1.2 new_frontiere_surf()	2061
6.881 Référence de la classe TriaQSfe1	2061
6.881.1 Documentation des fonctions membres	2063

6.881.1.1 KSI()	2063
6.881.1.2 new_frontiere_lin()	2063
6.881.1.3 new_frontiere_surf()	2063
6.882 Référence de la classe TriaQSfe3	2064
6.882.1 Documentation des fonctions membres	2066
6.882.1.1 KSI()	2066
6.882.1.2 new_frontiere_lin()	2066
6.882.1.3 new_frontiere_surf()	2066
6.883 Référence de la classe TriaSfe1	2067
6.883.1 Documentation des fonctions membres	2069
6.883.1.1 KSI()	2069
6.883.1.2 new_frontiere_lin()	2069
6.883.1.3 new_frontiere_surf()	2069
6.884 Référence de la classe TriaSfe1_cm5pti	2070
6.884.1 Documentation des fonctions membres	2072
6.884.1.1 KSI()	2072
6.884.1.2 new_frontiere_lin()	2072
6.884.1.3 new_frontiere_surf()	2072
6.885 Référence de la classe TriaSfe2	2073
6.885.1 Documentation des fonctions membres	2075
6.885.1.1 KSI()	2075
6.885.1.2 new_frontiere_lin()	2075
6.885.1.3 new_frontiere_surf()	2075
6.886 Référence de la classe TriaSfe3	2076
6.886.1 Documentation des fonctions membres	2078
6.886.1.1 KSI()	2078
6.886.1.2 new_frontiere_lin()	2078
6.886.1.3 new_frontiere_surf()	2078
6.887 Référence de la classe TriaSfe3_3D	2079
6.887.1 Documentation des fonctions membres	2081
6.887.1.1 KSI()	2081
6.887.1.2 new_frontiere_lin()	2081
6.887.1.3 new_frontiere_surf()	2081
6.888 Référence de la classe TriaSfe3_cm12pti	2082
6.888.1 Documentation des fonctions membres	2084
6.888.1.1 KSI()	2084
6.888.1.2 new_frontiere_lin()	2084
6.888.1.3 new_frontiere_surf()	2084
6.889 Référence de la classe TriaSfe3_cm13pti	2085
6.889.1 Documentation des fonctions membres	2087
6.889.1.1 KSI()	2087
6.889.1.2 new_frontiere_lin()	2087

---

6.889.1.3 new_frontiere_surf()	2087
6.890 Référence de la classe TriaSfe3_cm3pti	2088
6.890.1 Documentation des fonctions membres	2090
6.890.1.1 KSI()	2090
6.890.1.2 new_frontiere_lin()	2090
6.890.1.3 new_frontiere_surf()	2090
6.891 Référence de la classe TriaSfe3_cm4pti	2091
6.891.1 Documentation des fonctions membres	2093
6.891.1.1 KSI()	2093
6.891.1.2 new_frontiere_lin()	2093
6.891.1.3 new_frontiere_surf()	2093
6.892 Référence de la classe TriaSfe3_cm5pti	2094
6.892.1 Documentation des fonctions membres	2096
6.892.1.1 KSI()	2096
6.892.1.2 new_frontiere_lin()	2096
6.892.1.3 new_frontiere_surf()	2096
6.893 Référence de la classe TriaSfe3_cm6pti	2097
6.893.1 Documentation des fonctions membres	2099
6.893.1.1 KSI()	2099
6.893.1.2 new_frontiere_lin()	2099
6.893.1.3 new_frontiere_surf()	2099
6.894 Référence de la classe TriaSfe3_cm7pti	2100
6.894.1 Documentation des fonctions membres	2102
6.894.1.1 KSI()	2102
6.894.1.2 new_frontiere_lin()	2102
6.894.1.3 new_frontiere_surf()	2102
6.895 Référence de la classe TriaSfe3C	2103
6.895.1 Documentation des fonctions membres	2105
6.895.1.1 KSI()	2105
6.895.1.2 new_frontiere_lin()	2105
6.895.1.3 new_frontiere_surf()	2105
6.896 Référence de la classe TripodeCos3phi	2105
6.896.1 Description détaillée	2107
6.896.2 Documentation des fonctions membres	2107
6.896.2.1 Affiche()	2107
6.896.2.2 Complet_courbe()	2107
6.896.2.3 Der_sec()	2108
6.896.2.4 Derivee()	2108
6.896.2.5 Ecriture_base_info()	2108
6.896.2.6 Info_commande_Courbes1D()	2108
6.896.2.7 LectDonnParticulieres_courbes()	2108
6.896.2.8 Lecture_base_info()	2108

6.896.2.9 SchemaXML_Courbes1D()	2108
6.896.2.10 Valeur()	2109
6.896.2.11 Valeur_Et_der12()	2109
6.896.2.12 Valeur_Et_derivee()	2109
6.896.2.13 Valeur_Et_derivee_stricte()	2109
6.896.2.14 Valeur_stricte()	2109
6.897 Référence de la classe Trois_String	2109
6.897.1 Description détaillée	2110
6.898 Référence de la classe TroisEntiers	2110
6.898.1 Description détaillée	2111
6.899 Référence de la classe Type_Calcul	2111
6.900 Référence de la classe VariablesExporter::TypeEvoluee_a_un_element	2112
6.901 Référence de la classe VariablesExporter::TypeParticulier_a_un_element	2114
6.902 Référence de la classe VariablesExporter::TypeQuelc_arete_a_un_element	2116
6.903 Référence de la classe VariablesExporter::TypeQuelc_face_a_un_element	2118
6.904 Référence de la classe VariablesExporter::TypeQuelc_Une_composante_Grandeur_globale	2120
6.905 Référence de la classe TypeQuelconque	2121
6.905.1 Description détaillée	2122
6.906 Référence de la classe VariablesExporter::TypeQuelconque_a_Face_arete	2122
6.907 Référence de la classe VariablesExporter::TypeQuelconque_a_un_element	2124
6.908 Référence de la classe TypeQuelconque_enum_etendu	2125
6.908.1 Description détaillée	2126
6.909 Référence de la classe Umat_cont	2127
6.910 Référence de la classe UmatAbaqus	2128
6.911 Référence de la classe ElemPoint::UneFois	2130
6.912 Référence de la classe HexaMemb::UneFois	2132
6.913 Référence de la classe PentaMemb::UneFois	2134
6.914 Référence de la classe QuadAxiMemb::UneFois	2136
6.915 Référence de la classe QuadraMemb::UneFois	2138
6.916 Référence de la classe SfeMembT::UneFois	2140
6.917 Référence de la classe TetraMemb::UneFois	2142
6.918 Référence de la classe TriaAxiMemb::UneFois	2144
6.919 Référence de la classe TriaMemb::UneFois	2146
6.920 Référence de la classe Util	2147
6.920.1 Description détaillée	2149
6.921 Référence de la classe Deformation::UtilIndexDeformation	2149
6.922 Référence de la classe UtilLecture	2149
6.923 Référence de la classe UtilXML	2151
6.924 Référence de la classe Courbe1D::Valbool	2151
6.924.1 Description détaillée	2151
6.925 Référence de la classe Courbe1D::ValDer	2151
6.925.1 Description détaillée	2151



6.926	Référence de la classe Courbe1D::ValDer2	2151
6.926.1	Description détaillée	2152
6.927	Référence de la classe Courbe1D::ValDerbool	2152
6.927.1	Description détaillée	2152
6.928	Référence de la classe VariablesExporter	2152
6.929	Référence de la classe Vecteur	2153
6.929.1	Description détaillée	2157
6.930	Référence du modèle de la classe Vector_io< T >	2157
6.930.1	Description détaillée	2158
6.931	Référence de la classe VeurPropre	2158
6.932	Référence de la classe Visuali_geomview	2159
6.932.1	Documentation des fonctions membres	2159
6.932.1.1	Ecriture_parametres_OrdreVisu()	2159
6.932.1.2	ExeOrdre()	2160
6.932.1.3	Lecture_parametres_OrdreVisu()	2160
6.933	Référence de la classe Visuali_Gid	2160
6.933.1	Documentation des fonctions membres	2161
6.933.1.1	Ecriture_parametres_OrdreVisu()	2161
6.933.1.2	ExeOrdre()	2161
6.933.1.3	Lecture_parametres_OrdreVisu()	2162
6.934	Référence de la classe Visuali_Gmsh	2162
6.934.1	Documentation des fonctions membres	2163
6.934.1.1	Ecriture_parametres_OrdreVisu()	2163
6.934.1.2	ExeOrdre()	2163
6.934.1.3	Lecture_parametres_OrdreVisu()	2163
6.935	Référence de la classe Visuali_maple	2164
6.935.1	Documentation des fonctions membres	2164
6.935.1.1	Ecriture_parametres_OrdreVisu()	2164
6.935.1.2	ExeOrdre()	2165
6.935.1.3	Lecture_parametres_OrdreVisu()	2165
6.936	Référence de la classe Visuali_vrml	2165
6.936.1	Documentation des fonctions membres	2166
6.936.1.1	Ecriture_parametres_OrdreVisu()	2166
6.936.1.2	ExeOrdre()	2166
6.936.1.3	Lecture_parametres_OrdreVisu()	2167
6.937	Référence de la classe Visualisation	2167
6.938	Référence de la classe Visualisation_geomview	2167
6.939	Référence de la classe Visualisation_Gid	2168
6.940	Référence de la classe Visualisation_Gmsh	2168
6.941	Référence de la classe Visualisation_maple	2169

## 7 Documentation des fichiers

2171

7.1 Algori.h	2171
7.2 AlgoriCombine.h	2186
7.3 AlgoInformations.h	2189
7.4 AlgoriRemontErreur.h	2191
7.5 AlgoUmatAbaqus.h	2192
7.6 AlgoUtils.h	2195
7.7 Algori_chung_lee.h	2197
7.8 Algori_relax_dyna.h	2199
7.9 Algori_tchamwa.h	2204
7.10 AlgoriDynaExpli.h	2207
7.11 AlgoriDynaExpli_zhai.h	2210
7.12 AlgoRungeKutta.h	2212
7.13 AlgoriNewmark.h	2215
7.14 AlgoriMixte.h	2219
7.15 AlgoriFlambLineaire.h	2222
7.16 AlgoriNonDyna.h	2224
7.17 ImpliNonDynaCont.h	2227
7.18 AlgoBonelli.h	2229
7.19 Hypo_ortho3D_entrainee.h	2233
7.20 Loi_ortho2D_C_entrainee.h	2237
7.21 Loi_ortho3D_entrainee.h	2241
7.22 Projection_anisotrope_3D.h	2246
7.23 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Com↵ Loi_comp_abstraite.h	2251
7.23.1 Description détaillée	2251
7.24 ComLoi_comp_abstraite.h	2251
7.25 CompFrotAbstraite.h	2252
7.26 CompThermoPhysiqueAbstraite.h	2255
7.27 EnergieMeca.h	2259
7.28 EnergieThermi.h	2261
7.29 ExceptionsLoiComp.h	2263
7.30 CompFrotCoulomb.h	2264
7.31 Hart_Smith3D.h	2266
7.32 Hyper1.h	2269
7.33 Hyper10.h	2271
7.34 Hyper3D.h	2273
7.35 Hyper3D_save.h	2277
7.36 Hyper3DN.h	2278
7.37 Hyper_externe_W.h	2279
7.38 Hyper_w1.h	2283
7.39 Hyper_W_gene_3D.h	2285
7.40 HyperD.h	2289

---

7.41 HyperDN.h . . . . .	2295
7.42 IsoHyper3DFavier3.h . . . . .	2298
7.43 IsoHyper3DOrgeas1.h . . . . .	2300
7.44 IsoHyper3DOrgeas2.h . . . . .	2303
7.45 IsoHyperBulk3.h . . . . .	2306
7.46 IsoHyperBulk_gene.h . . . . .	2308
7.47 Maheo_hyper.h . . . . .	2310
7.48 MooneyRivlin1D.h . . . . .	2313
7.49 MooneyRivlin3D.h . . . . .	2315
7.50 Poly_hyper3D.h . . . . .	2318
7.51 TreloarN.h . . . . .	2321
7.52 Hypo_hooke1D.h . . . . .	2322
7.53 Hypo_hooke2D_C.h . . . . .	2327
7.54 Hypo_hooke3D.h . . . . .	2330
7.55 Copie_de_Hysteresis3D.h . . . . .	2334
7.56 Hysteresis1D.h . . . . .	2337
7.57 Hysteresis3D.h . . . . .	2342
7.58 Hysteresis_bulk.h . . . . .	2351
7.59 Loi_iso_elas1D.h . . . . .	2357
7.60 Loi_iso_elas2D_C.h . . . . .	2360
7.61 Loi_iso_elas2D_D.h . . . . .	2364
7.62 Loi_iso_elas3D.h . . . . .	2368
7.63 Iso_elas_expo1D.h . . . . .	2372
7.64 Iso_elas_expo3D.h . . . . .	2376
7.65 Iso_elas_SE1D.h . . . . .	2378
7.66 LesLoisDeComp.h . . . . .	2380
7.67 Loi_comp_abstraite.h . . . . .	2383
7.68 Loi_Umat.h . . . . .	2392
7.69 LoiAbstraiteGeneral.h . . . . .	2396
7.70 LoiAdditiveEnSigma.h . . . . .	2399
7.71 LoiContraintesPlanes.h . . . . .	2403
7.72 LoiContraintesPlanesDouble.h . . . . .	2409
7.73 LoiCritere.h . . . . .	2417
7.74 LoiDeformationsPlanes.h . . . . .	2425
7.75 LoiDesMelangesEnSigma.h . . . . .	2429
7.76 Loi_rien1D.h . . . . .	2434
7.77 Loi_rien2D.h . . . . .	2436
7.78 Loi_rien2D_C.h . . . . .	2437
7.79 Loi_rien2D_D.h . . . . .	2439
7.80 Loi_rien3D.h . . . . .	2441
7.81 Prandtl_Reuss.h . . . . .	2444
7.82 Prandtl_Reuss1D.h . . . . .	2447

7.83 Prandtl_Reuss2D_D.h	2450
7.84 Loi_de_Tait.h	2453
7.85 Loi_iso_thermo.h	2456
7.86 CristaliniteAbstraite.h	2459
7.87 Hoffman1.h	2462
7.88 Hoffman2.h	2464
7.89 ThermoDonnee.h	2467
7.90 Cercle.h	2468
7.91 Cylindre.h	2470
7.92 Droite.h	2471
7.93 ElContact.h	2473
7.94 LesContacts.h	2478
7.95 Plan.h	2484
7.96 Sphere.h	2485
7.97 ElemGeomC0.h	2487
7.98 ElemGeomC1.h	2491
7.99 GeomSeg.h	2492
7.100 GeomPoint.h	2494
7.101 GeomQuadrangle.h	2496
7.102 GeomTriangle.h	2498
7.103 GeomHexaCom.h	2500
7.104 GeomHexaCubique.h	2505
7.105 GeomHexalin.h	2509
7.106 GeomHexaQuad.h	2512
7.107 GeomHexaQuadComp.h	2515
7.108 GeomPentaCom.h	2518
7.109 GeomPentaL.h	2521
7.110 GeomPentaQ.h	2523
7.111 GeomPentaQComp.h	2525
7.112 GeomTetra.h	2527
7.113 GeomTetraCom.h	2529
7.114 GeomTetraL.h	2531
7.115 GeomTetraQ.h	2533
7.116 ElFrontiere.h	2535
7.117 Front.h	2538
7.118 FrontSegCub.h	2541
7.119 FrontSegLine.h	2543
7.120 FrontSegQuad.h	2545
7.121 FrontPointF.h	2547
7.122 FrontQuadCC.h	2550
7.123 FrontQuadLine.h	2552
7.124 FrontQuadQC.h	2554

---

7.125 FrontQuadQuad.h	2556
7.126 FrontTriaLine.h	2558
7.127 FrontTriaQuad.h	2560
7.128 Biel_axi.h	2562
7.129 Biel_axiQ.h	2571
7.130 Biellette.h	2581
7.131 BielletteC1.h	2589
7.132 BielletteQ.h	2597
7.133 DeformationP2D.h	2605
7.134 Met_biellette.h	2606
7.135 Met_BielletteC1.h	2608
7.136 Met_pout2D.h	2609
7.137 PoutSimple1.h	2611
7.138 PoutTimo.h	2616
7.139 Def_Umat.h	2620
7.140 Deformation.h	2621
7.141 DeformationPP.h	2630
7.142 Met_abstraite.h	2632
7.143 Met_abstraite_struc_donnees.h	2641
7.144 Met_PiPoCo.h	2653
7.145 MetAxisymetrique2D.h	2657
7.146 MetAxisymetrique3D.h	2659
7.147 PiPoCo.h	2661
7.148 ElemMeca.h	2664
7.149 ElemPoint.h	2677
7.150 ElemPoint_CP.h	2684
7.151 Met_ElemPoint.h	2686
7.152 UmatAbaqus.h	2688
7.153 ExceptionsElemMeca.h	2691
7.154 Hexa.h	2692
7.155 Hexa_cm1pti.h	2694
7.156 Hexa_cm27pti.h	2696
7.157 Hexa_cm64pti.h	2698
7.158 HexaLMemb.h	2700
7.159 HexaMemb.h	2702
7.160 HexaQ.h	2711
7.161 HexaQ_cm1pti.h	2713
7.162 HexaQ_cm27pti.h	2715
7.163 HexaQ_cm64pti.h	2717
7.164 HexaQComp.h	2719
7.165 HexaQComp_cm1pti.h	2721
7.166 HexaQComp_cm27pti.h	2723

7.167 HexaQComp_cm64pti.h	2726
7.168 LesChargeExtSurElement.h	2728
7.169 LesPtIntegMecaInterne.h	2730
7.170 PentaL.h	2732
7.171 PentaL_cm1pti.h	2734
7.172 PentaL_cm2pti.h	2736
7.173 PentaL_cm6pti.h	2738
7.174 PentaMemb.h	2741
7.175 PentaQ.h	2750
7.176 PentaQ_cm12pti.h	2752
7.177 PentaQ_cm18pti.h	2754
7.178 PentaQ_cm3pti.h	2756
7.179 PentaQ_cm9pti.h	2758
7.180 PentaQComp.h	2760
7.181 PentaQComp_cm12pti.h	2763
7.182 PentaQComp_cm18pti.h	2765
7.183 PentaQComp_cm9pti.h	2767
7.184 PtIntegMecaInterne.h	2769
7.185 QuadAxiCCom.h	2773
7.186 QuadAxiCCom_cm9pti.h	2775
7.187 QuadAxiL1.h	2777
7.188 QuadAxiL1_cm1pti.h	2779
7.189 QuadAxiMemb.h	2781
7.190 QuadAxiQ.h	2789
7.191 QuadAxiQComp.h	2791
7.192 QuadAxiQComp_cm4pti.h	2793
7.193 Quad.h	2795
7.194 Quad_cm1pti.h	2798
7.195 QuadCCom.h	2800
7.196 QuadCCom_cm9pti.h	2802
7.197 QuadQ.h	2804
7.198 QuadQCom.h	2806
7.199 QuadQCom_cm4pti.h	2808
7.200 QuadraMemb.h	2810
7.201 DeformationSfe1.h	2820
7.202 Met_Sfe1.h	2824
7.203 Met_Sfe1_struct_donnees.h	2833
7.204 PoutSfe3.h	2834
7.205 Sfeg.h	2836
7.206 SfeMembT.h	2837
7.207 TriaQSfe1.h	2848
7.208 TriaQSfe3.h	2850

---

7.209 TriaSfe1.h . . . . .	2853
7.210 TriaSfe1_cm5pti.h . . . . .	2855
7.211 TriaSfe2.h . . . . .	2857
7.212 TriaSfe3.h . . . . .	2860
7.213 TriaSfe3_3D.h . . . . .	2862
7.214 TriaSfe3_cm12pti.h . . . . .	2864
7.215 TriaSfe3_cm13pti.h . . . . .	2866
7.216 TriaSfe3_cm3pti.h . . . . .	2868
7.217 TriaSfe3_cm4pti.h . . . . .	2871
7.218 TriaSfe3_cm5pti.h . . . . .	2873
7.219 TriaSfe3_cm6pti.h . . . . .	2875
7.220 TriaSfe3_cm7pti.h . . . . .	2877
7.221 TriaSfe3C.h . . . . .	2880
7.222 Tetra.h . . . . .	2882
7.223 TetraMemb.h . . . . .	2884
7.224 TetraQ.h . . . . .	2892
7.225 TetraQ_cm15pti.h . . . . .	2894
7.226 TetraQ_cm1pti.h . . . . .	2897
7.227 TriaAxiL1.h . . . . .	2899
7.228 TriaAxiMemb.h . . . . .	2901
7.229 TriaAxiQ3.h . . . . .	2909
7.230 TriaAxiQ3_cm1pti.h . . . . .	2911
7.231 TriaAxiQ3_cmpti1003.h . . . . .	2913
7.232 Met_triaMemb.h . . . . .	2915
7.233 TriaCub.h . . . . .	2916
7.234 TriaCub_cm4pti.h . . . . .	2918
7.235 TriaMemb.h . . . . .	2920
7.236 TriaMembL1.h . . . . .	2930
7.237 TriaMembQ3.h . . . . .	2932
7.238 TriaMembQ3_cm1pti.h . . . . .	2934
7.239 TriaQ3_cmpti1003.h . . . . .	2936
7.240 BiелletteThermi.h . . . . .	2938
7.241 ElemThermi.h . . . . .	2946
7.242 ElemThermiGene.h . . . . .	2957
7.243 ExceptionsElemThermi.h . . . . .	2959
7.244 LesPtIntegThermiInterne.h . . . . .	2960
7.245 PtIntegThermiInterne.h . . . . .	2962
7.246 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ boolddl.h . . . . .	2964
7.246.1 Description détaillée . . . . .	2965
7.247 Enum_boolddl.h . . . . .	2965
7.248 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ calcul_masse.h . . . . .	2966

7.248.1 Description détaillée	2967
7.249 Enum_calcul_masse.h	2967
7.250 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ categorie_loi_comp.h	2968
7.250.1 Description détaillée	2969
7.251 Enum_categorie_loi_comp.h	2969
7.252 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ chargement.h	2970
7.252.1 Description détaillée	2971
7.253 Enum_chargement.h	2971
7.254 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ comp.h	2972
7.254.1 Description détaillée	2974
7.255 Enum_comp.h	2974
7.256 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ contrainte_mathematique.h	2975
7.256.1 Description détaillée	2977
7.257 Enum_contrainte_mathematique.h	2977
7.258 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ crista.h	2978
7.258.1 Description détaillée	2979
7.259 Enum_crista.h	2979
7.260 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ Critere_loi.h	2980
7.260.1 Description détaillée	2982
7.261 Enum_Critere_loi.h	2982
7.262 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ ddl.h	2983
7.262.1 Description détaillée	2985
7.263 Enum_ddl.h	2985
7.264 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ ddl_var_static.cc	2989
7.264.1 Description détaillée	2989
7.265 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ dure.h	2989
7.265.1 Description détaillée	2990
7.266 Enum_dure.h	2990
7.267 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ geom.h	2992
7.267.1 Description détaillée	2993
7.268 Enum_geom.h	2993
7.269 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ GrandeurGlobale.h	2994
7.269.1 Description détaillée	2995
7.270 Enum_GrandeurGlobale.h	2995



---

7.271	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ interpol.h	2996
7.271.1	Description détaillée	2997
7.272	Enum_interpol.h	2997
7.273	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ IO_XML.h	2998
7.273.1	Description détaillée	2999
7.274	Enum_IO_XML.h	2999
7.275	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ liaison_noeud.h	3000
7.275.1	Description détaillée	3001
7.276	Enum_liaison_noeud.h	3001
7.277	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ mat.h	3002
7.277.1	Description détaillée	3003
7.278	Enum_mat.h	3003
7.279	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ matrice.h	3004
7.279.1	Description détaillée	3005
7.280	Enum_matrice.h	3005
7.281	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ PiPoCo.h	3006
7.281.1	Description détaillée	3007
7.282	Enum_PiPoCo.h	3007
7.283	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ proj_aniso.h	3009
7.283.1	Description détaillée	3010
7.284	Enum_proj_aniso.h	3010
7.285	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ StabHourglass.h	3011
7.285.1	Description détaillée	3012
7.286	Enum_StabHourglass.h	3012
7.287	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ StabMembrane.h	3014
7.287.1	Description détaillée	3015
7.288	Enum_StabMembrane.h	3015
7.289	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ Suite.h	3016
7.289.1	Description détaillée	3017
7.290	Enum_Suite.h	3017
7.291	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ type_deformation.h	3018
7.291.1	Description détaillée	3019
7.292	Enum_type_deformation.h	3019

7.293	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ type_geom.h . . . . .	3021
7.293.1	Description détaillée . . . . .	3021
7.294	Enum_type_geom.h . . . . .	3022
7.295	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ type_pt_integ.h . . . . .	3022
7.295.1	Description détaillée . . . . .	3023
7.296	Enum_type_pt_integ.h . . . . .	3023
7.297	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ type_resolution_matri.h . . . . .	3024
7.297.1	Description détaillée . . . . .	3025
7.298	Enum_type_resolution_matri.h . . . . .	3026
7.299	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ type_stocke_deformation.h . . . . .	3027
7.299.1	Description détaillée . . . . .	3028
7.300	Enum_type_stocke_deformation.h . . . . .	3028
7.301	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ TypeQuelconque.h . . . . .	3030
7.301.1	Description détaillée . . . . .	3032
7.302	Enum_TypeQuelconque.h . . . . .	3032
7.303	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ variable_metrique.h . . . . .	3035
7.303.1	Description détaillée . . . . .	3036
7.304	Enum_variable_metrique.h . . . . .	3036
7.305	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ Courbe1D.h . . . . .	3037
7.305.1	Description détaillée . . . . .	3038
7.306	EnumCourbe1D.h . . . . .	3038
7.307	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ ElemTypeProblem.h . . . . .	3040
7.307.1	Description détaillée . . . . .	3040
7.308	EnumElemTypeProblem.h . . . . .	3041
7.309	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ Fonction_nD.h . . . . .	3042
7.309.1	Description détaillée . . . . .	3043
7.310	EnumFonction_nD.h . . . . .	3043
7.311	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ Langue.h . . . . .	3044
7.311.1	Description détaillée . . . . .	3045
7.312	EnumLangue.h . . . . .	3045
7.313	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_↔ TypeCalcul.h . . . . .	3046
7.313.1	Description détaillée . . . . .	3047
7.314	EnumTypeCalcul.h . . . . .	3048

7.315	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeGradient.h	3050
7.315.1	Description détaillée	3050
7.316	EnumTypeGradient.h	3051
7.317	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeGrandeur.h	3052
7.317.1	Description détaillée	3053
7.318	EnumTypeGrandeur.h	3053
7.319	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypePilotage.h	3054
7.319.1	Description détaillée	3055
7.320	EnumTypePilotage.h	3055
7.321	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeViteRotat.h	3056
7.321.1	Description détaillée	3057
7.322	EnumTypeViteRotat.h	3057
7.323	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeVitesseDefor.h	3058
7.323.1	Description détaillée	3059
7.324	EnumTypeVitesseDefor.h	3059
7.325	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeCL.h	3060
7.325.1	Description détaillée	3061
7.326	EnumTypeCL.h	3061
7.327	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeQuelParticulier.h	3063
7.327.1	Description détaillée	3064
7.328	EnumTypeQuelParticulier.h	3064
7.329	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/MotCle.h	3065
7.329.1	Description détaillée	3066
7.330	MotCle.h	3066
7.331	LesValVecPropres.h	3067
7.332	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/General/herezh.cc	3068
7.332.1	Description détaillée	3070
7.333	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/General/herezh.h	3070
7.333.1	Description détaillée	3071
7.334	herezh.h	3071
7.335	Projet.h	3072
7.336	Bloc.h	3075
7.337	LectBloc_T.h	3084
7.338	LectBlocMot.h	3086
7.339	UtilLecture.h	3088
7.340	UtilXML.h	3094
7.341	Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/BlocDdlLim.h	3096

7.341.1 Description détaillée	3096
7.342 BlocDdlLim.h	3096
7.343 Ddl.h	3099
7.344 Ddl_etendu.h	3102
7.345 DdlElement.h	3105
7.346 DdlLim.h	3107
7.347 DdlNoeudElement.h	3111
7.348 DiversStockage.h	3112
7.349 I_O_Condilinaire.h	3115
7.350 LesCondLim.h	3118
7.351 LesMaillages.h	3125
7.352 Maillage.h	3137
7.353 Noeud.h	3149
7.354 VariablesExporter.h	3159
7.355 Banniere.h	3168
7.356 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/Constante.h	3169
7.356.1 Description détaillée	3170
7.357 Constante.h	3170
7.358 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/← ConstantePrinc.h	3171
7.358.1 Description détaillée	3171
7.359 ConstantePrinc.h	3172
7.360 ConstMath.h	3172
7.361 ConstPhysico.h	3173
7.362 EnteteParaGlob.h	3174
7.363 ParaAlgoControle.h	3175
7.364 ParaGlob.h	3184
7.365 Type_Calcul.h	3188
7.366 TypeCalcul.h	3190
7.367 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/Lect_← reference.h	3190
7.367.1 Description détaillée	3191
7.368 Lect_reference.h	3191
7.369 LesReferences.h	3192
7.370 Reference.h	3195
7.371 ReferenceAF.h	3197
7.372 ReferenceFA.h	3200
7.373 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/← ReferenceNE.cc	3201
7.373.1 Description détaillée	3202
7.374 ReferenceNE.h	3202
7.375 ReferencePtiAF.h	3204
7.376 ExceptionsMatrices.h	3207

---

7.377 Mat_abstraite.h . . . . .	3208
7.378 Mat_pleine.h . . . . .	3213
7.379 MatBand.h . . . . .	3218
7.380 MatDiag.h . . . . .	3222
7.381 Mat_creuse_CompCol.h . . . . .	3226
7.382 clapack.h . . . . .	3230
7.383 MatLapack.h . . . . .	3303
7.384 vecdefs_GR.h . . . . .	3309
7.385 iotext_GR.h . . . . .	3310
7.386 mvblas_GR.h . . . . .	3310
7.387 mvmtpl_GR.h . . . . .	3313
7.388 mvvind_GR.h . . . . .	3317
7.389 mvvrf_GR.h . . . . .	3319
7.390 mvvtp_GR.h . . . . .	3319
7.391 diagpre_double_GR.h . . . . .	3326
7.392 icpre_double_GR.h . . . . .	3327
7.393 ilupre_double_GR.h . . . . .	3328
7.394 pre_cond_double.h . . . . .	3330
7.395 Assemblage.h . . . . .	3331
7.396 Condilinaire.h . . . . .	3333
7.397 CondLim.h . . . . .	3335
7.398 Frontiere_initiale.h . . . . .	3338
7.399 color.h . . . . .	3340
7.400 rgb.h . . . . .	3340
7.401 spectre.h . . . . .	3341
7.402 Animation_geomview.h . . . . .	3343
7.403 Deformees_geomview.h . . . . .	3345
7.404 Fin_geomview.h . . . . .	3346
7.405 Frontiere_initiale_geomview.h . . . . .	3347
7.406 Isovaleurs_geomview.h . . . . .	3348
7.407 Mail_initiale_geomview.h . . . . .	3350
7.408 Visuali_geomview.h . . . . .	3351
7.409 Visualisation_geomview.h . . . . .	3352
7.410 Deformees_Gid.h . . . . .	3354
7.411 Fin_Gid.h . . . . .	3356
7.412 Isovaleurs_Gid.h . . . . .	3357
7.413 Mail_initiale_Gid.h . . . . .	3362
7.414 Visuali_Gid.h . . . . .	3364
7.415 Visualisation_Gid.h . . . . .	3365
7.416 Deformees_Gmsh.h . . . . .	3367
7.417 Fin_Gmsh.h . . . . .	3369
7.418 Isovaleurs_Gmsh.h . . . . .	3371

7.419 Mail_initiale_Gmsh.h	3376
7.420 Visuali_Gmsh.h	3379
7.421 Visualisation_Gmsh.h	3380
7.422 Animation_maple.h	3383
7.423 Choix_grandeurs_maple.h	3385
7.424 Deformees_maple.h	3389
7.425 Fin_maple.h	3390
7.426 Visuali_maple.h	3392
7.427 Visualisation_maple.h	3393
7.428 OrdreVisu.h	3395
7.429 Resultats.h	3397
7.430 Visualisation.h	3399
7.431 Animation_vrml.h	3402
7.432 ChoixDesMaillages_vrml.h	3404
7.433 Deformees_vrml.h	3405
7.434 Fin_vrml.h	3407
7.435 Increment_vrml.h	3408
7.436 Isovaleurs_vrml.h	3409
7.437 Mail_initiale_vrml.h	3411
7.438 Visuali_vrml.h	3413
7.439 Tableau2_T.h	3414
7.440 Tableau4_T.h	3418
7.441 Tableau_3D.h	3422
7.442 Tableau_4D.h	3424
7.443 Tableau_5D.h	3426
7.444 Tableau_double.h	3429
7.445 Tableau_T.h	3431
7.446 TabOper_T.h	3441
7.447 utilDebug.h	3445
7.448 Coordonnee.h	3446
7.449 Coordonnee1.h	3454
7.450 Coordonnee2.h	3460
7.451 Coordonnee3.h	3466
7.452 Base.h	3471
7.453 Base3D3.h	3478
7.454 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/↔ Tenseur/DefValConstens.h	3480
7.454.1 Description détaillée	3481
7.454.2 Documentation des fonctions	3481
7.454.2.1 ConstantesTenseur()	3481
7.455 DefValConstens.h	3481
7.456 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/↔ Tenseur/EnteteTenseur.h	3482

---

7.456.1 Description détaillée	3482
7.457 EnteteTenseur.h	3483
7.458 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/↔ Tenseur/NevezTenseur.h	3484
7.458.1 Description détaillée	3485
7.459 NevezTenseur.h	3485
7.460 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/↔ Tenseur/NevezTenseurQ.h	3487
7.460.1 Description détaillée	3488
7.460.2 Documentation des fonctions	3488
7.460.2.1 NevezTenseurBBBB() [1/2]	3488
7.460.2.2 NevezTenseurBBBB() [2/2]	3488
7.460.2.3 NevezTenseurBBHH() [1/2]	3488
7.460.2.4 NevezTenseurBBHH() [2/2]	3489
7.460.2.5 NevezTenseurBHBH() [1/2]	3489
7.460.2.6 NevezTenseurBHBH() [2/2]	3489
7.460.2.7 NevezTenseurBHHB() [1/2]	3489
7.460.2.8 NevezTenseurBHHB() [2/2]	3489
7.460.2.9 NevezTenseurHBBH() [1/2]	3490
7.460.2.10 NevezTenseurHBBH() [2/2]	3490
7.460.2.11 NevezTenseurHBHB() [1/2]	3490
7.460.2.12 NevezTenseurHBHB() [2/2]	3490
7.460.2.13 NevezTenseurHHBB() [1/2]	3491
7.460.2.14 NevezTenseurHHBB() [2/2]	3491
7.460.2.15 NevezTenseurHHHH() [1/2]	3491
7.460.2.16 NevezTenseurHHHH() [2/2]	3491
7.461 NevezTenseurQ.h	3492
7.462 Tenseur.h	3495
7.463 Tenseur1.h	3507
7.464 Tenseur1_TroisSym.h	3515
7.465 Tenseur2.h	3519
7.466 Tenseur2_TroisSym.h	3529
7.467 Tenseur3.h	3533
7.468 Tenseur3_TroisSym.h	3544
7.469 TenseurO4.h	3548
7.470 TenseurQ-1.h	3553
7.471 TenseurQ-2.h	3560
7.472 TenseurQ-3.h	3567
7.473 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/↔ Tenseur/TenseurQ.h	3576
7.473.1 Description détaillée	3577
7.473.2 Documentation des fonctions	3577
7.473.2.1 LibereTenseurQ()	3577

7.473.3 la fonction libereTenseur libere tout l'espace de tous les types de tenseurs . . . . .	3577
7.474 TenseurQ.h . . . . .	3577
7.475 TenseurQ1gene.h . . . . .	3590
7.476 TenseurQ2gene.h . . . . .	3594
7.477 TenseurQ3.h.old.h . . . . .	3600
7.478 TenseurQ3gene.h . . . . .	3605
7.479 TypeConsTens.h . . . . .	3611
7.480 TypeConsTensPrinc.h . . . . .	3612
7.481 Vecteur.h . . . . .	3613
7.482 VeurPropre.h . . . . .	3617
7.483 Basiques.h . . . . .	3619
7.484 Ddl_enum_etendu-copie.h . . . . .	3628
7.485 Ddl_enum_etendu.h . . . . .	3630
7.486 Epai.h . . . . .	3633
7.487 List_io.h . . . . .	3634
7.488 Map_io.h . . . . .	3637
7.489 Nb_assemb.h . . . . .	3639
7.490 PlusieursCoordonnees.h . . . . .	3640
7.491 Ponderation.h . . . . .	3641
7.492 PtTabRel.h . . . . .	3646
7.493 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel← _Princ.h . . . . .	3653
7.493.1 Description détaillée . . . . .	3653
7.494 PtTabRel_Princ.h . . . . .	3653
7.495 Section.h . . . . .	3655
7.496 Temps_CPU_HZpp.h . . . . .	3656
7.497 Temps_CPU_HZpp_3.h . . . . .	3659
7.498 TypeQuelconque.h . . . . .	3660
7.499 TypeQuelconque_enum_etendu.h . . . . .	3664
7.500 TypeQuelconqueParticulier.h . . . . .	3667
7.501 Vector_io.h . . . . .	3694
7.502 LaLIST.H . . . . .	3695
7.503 Algo_edp.h . . . . .	3696
7.504 Algo_Integ1D.h . . . . .	3699
7.505 Algo_zero.h . . . . .	3700
7.506 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/CharUtil.h . . . . .	3704
7.506.1 Description détaillée . . . . .	3705
7.506.2 Documentation des fonctions . . . . .	3705
7.506.2.1 lect_return_defaut() . . . . .	3705
7.507 CharUtil.h . . . . .	3705
7.508 Courbe1D.h . . . . .	3707
7.509 Courbe_cos.h . . . . .	3710



---

7.510 Courbe_expo2_n.h . . . . .	3712
7.511 Courbe_expo_n.h . . . . .	3714
7.512 Courbe_expoaff.h . . . . .	3716
7.513 Courbe_expression_litterale_1D.h . . . . .	3718
7.514 Courbe_expression_litterale_avec_derivees_1D.h . . . . .	3720
7.515 Courbe_ln_cosh.h . . . . .	3722
7.516 Courbe_relax_expo.h . . . . .	3724
7.517 Courbe_sin.h . . . . .	3726
7.518 Courbe_un_moins_cos.h . . . . .	3728
7.519 CourbePolyHermite1D.h . . . . .	3730
7.520 CourbePolyLineaire1D.h . . . . .	3732
7.521 CourbePolyLineaire1D_simpli.h . . . . .	3734
7.522 CourbePolynomiale.h . . . . .	3736
7.523 F1_plus_F2.h . . . . .	3738
7.524 F1_rond_F2.h . . . . .	3740
7.525 F_cycle_add.h . . . . .	3742
7.526 F_cyclique.h . . . . .	3745
7.527 F_nD_courbe1D (copy: conflict on 2017-11-21).h . . . . .	3747
7.528 F_nD_courbe1D.h . . . . .	3749
7.529 F_union_1D.h . . . . .	3752
7.530 Fonc_scalcombinees_nD (copy: conflict on 2017-11-21).h . . . . .	3754
7.531 Fonc_scalcombinees_nD.h . . . . .	3756
7.532 Fonction_expression_litterale_nD (copy: conflict on 2017-11-21).h . . . . .	3760
7.533 Fonction_expression_litterale_nD.h . . . . .	3761
7.534 Fonction_externe_nD.h . . . . .	3763
7.535 Fonction_nD (conflict on 2019-05-08).h . . . . .	3766
7.536 Fonction_nD (copy: conflict on 2017-11-21).h . . . . .	3774
7.537 Fonction_nD.h . . . . .	3779
7.538 LesCourbes1D.h . . . . .	3788
7.539 LesFonctions_nD.h . . . . .	3791
7.540 Poly_Lagrange.h . . . . .	3793
7.541 SixpodeCos3phi.h . . . . .	3795
7.542 TangenteHyperbolique.h . . . . .	3797
7.543 TripodeCos3phi.h . . . . .	3799
7.544 Racine.h . . . . .	3801
7.545 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Handler_exception.h . . . . .	3805
7.545.1 Description détaillée . . . . .	3806
7.546 Handler_exception.h . . . . .	3806
7.547 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/MathUtil.h . . . . .	3807
7.547.1 Description détaillée . . . . .	3809
7.548 MathUtil.h . . . . .	3809
7.549 MathUtil2.h . . . . .	3811

7.550 MvtSolide.h . . . . .	3813
7.551 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Sortie.h . . . . .	3815
7.551.1 Description détaillée . . . . .	3815
7.552 Sortie.h . . . . .	3815
7.553 LesSuitesReel.h . . . . .	3817
7.554 Suite_arithmetique.h . . . . .	3818
7.555 Suite_equidistante.h . . . . .	3819
7.556 Suite_geometrique.h . . . . .	3820
7.557 SuiteReel.h . . . . .	3821
7.558 Util.h . . . . .	3823

# Chapitre 1

## Index des modules

### 1.1 Modules

Liste de tous les modules :

Les_algorithmes_de_resolutions_globales . . . . .	59
Les_lois_anisotropes . . . . .	60
Les_conteneurs_energies . . . . .	61
Les_lois_hyperelastiques . . . . .	61
Les_lois_hypoelastiques . . . . .	62
Les_lois_hysteresis . . . . .	63
Les_lois_hooke . . . . .	63
Les_lois_iso_non_lineaire . . . . .	64
Les_loiscombinees . . . . .	65
Les_lois_de_plasticite . . . . .	65
Les_lois_concernant_thermique . . . . .	66
Groupe_sur_les_contacts . . . . .	66
Les_Elements_de_geometrie . . . . .	67
Les_Elements_de_frontiere . . . . .	68
Groupe_des_deformations . . . . .	68
Groupe_des_metrrique . . . . .	69
Groupe_concernant_les_points_integraton . . . . .	70
Group_types_enumeres . . . . .	71
Goupe_conteneurs_bloc . . . . .	78
Goupe_relatif_aux_entrees_sorties . . . . .	79
Les_classes_Ddl_en_tout_genre . . . . .	79
Les_classes_relatives_aux_conditions_limites . . . . .	80
Les_Maillages . . . . .	80
Les_classes_exportation_en_variables . . . . .	81
Les_parametres_generaux . . . . .	82
Les_classes_Reference . . . . .	82
Les_classes_Matrices . . . . .	84
Les_sorties_geomview . . . . .	84
Les_sorties_Gid . . . . .	85
Les_sorties_gmsh . . . . .	85
Les_sorties_au_format_maple . . . . .	86
Les_sorties_generiques . . . . .	87
Les_sorties_vrml . . . . .	87
Les_Tableaux_generiques . . . . .	88
Les_classes_coordonnee . . . . .	89

Les_classes_coordonnee1 . . . . .	89
Les_classes_coordonnee2 . . . . .	90
Les_classes_coordonnee3 . . . . .	91
Les_classes_Base . . . . .	91
Les_classes_Base3D3 . . . . .	92
Les_classes_tenseurs_virtuelles_ordre2 . . . . .	93
Les_classes_Maillons_tenseurs . . . . .	93
Les_classes_tenseurs_dim1_ordre2 . . . . .	94
Les_classes_tenseurs_dim2_ordre2 . . . . .	95
Les_classes_tenseurs_dim3_ordre2 . . . . .	96
Les_classes_tenseurs_virtuelles_ordre4 . . . . .	97
Les_classes_Maillons_tenseurs_ordre4 . . . . .	98
Les_classes_Vecteur . . . . .	99
Les_conteneurs_ultra_basiques . . . . .	99
Group_Ddl_enum_etendu . . . . .	100
Les_list_io . . . . .	101
Les_classes_PlusieursCoordonnees . . . . .	101
Les_classes_Ponderation . . . . .	102
Les_listes_de_petits_tableaux_de_reels . . . . .	102
Les_pointeurs_dans_listes_de_petits_tableaux . . . . .	103
Group_TypeQuelconque . . . . .	105
Group_TypeQuelconque_enum_etendu . . . . .	105
Les_grandeurs_particulieres . . . . .	106
Les_classes_algo . . . . .	108
Les_courbes_1D . . . . .	108
Les_fonctions_nD . . . . .	110
Classes_utilitaires_sur_vecteurs_et_matrices . . . . .	111
Classes_gestions_arret_Herezh . . . . .	111
def_classes_suites_reel . . . . .	112
Groupe_concernant_le_chargement . . . . .	113
Groupe_des_elements_finis . . . . .	113

## Chapitre 2

# Index hiérarchique

### 2.1 Hiérarchie des classes

Cette liste d'héritage est classée approximativement par ordre alphabétique :

__CLPK_complex	117
__CLPK_doublecomplex	117
HyperD::A_i	117
HyperD::A_iAvecVarDdl	118
HyperD::A_iAvecVarEps	119
VariablesExporter::A_un_NE	121
VariablesExporter::A_un_E	120
VariablesExporter::Ddl_a_un_element	495
VariablesExporter::TypeQuelconque_a_un_element	2124
VariablesExporter::TypeEvoluee_a_un_element	2112
VariablesExporter::TypeParticulier_a_un_element	2114
VariablesExporter::TypeQuelconque_a_Face_arete	2122
VariablesExporter::TypeQuelc_arete_a_un_element	2116
VariablesExporter::TypeQuelc_face_a_un_element	2118
VariablesExporter::Ddl_etendu_a_un_noeud	501
VariablesExporter::Quelconque_a_un_noeud	1507
Algo_edp	122
Algo_Integ1D	126
Algo_zero	127
Algori	141
AlgoBonelli	130
AlgoInformations	137
AlgoUmatAbaqus	211
AlgoUtils	215
AlgoriCombine	156
AlgoriDynaExpli	161
AlgoriDynaExpli_zhai	167
AlgoriFlambLineaire	173
AlgoriNewmark	177
AlgoriNonDyna	181
AlgoriRelaxDyna	185
AlgoriRemontErreur	192
AlgoriRungeKutta	193
AlgoriTchamwa	205

Algori_chung_lee	150
AlgoristatExpli	199
ImpliNonDynaCont	917
Assemblage	227
Banniere	227
Base3D3B	228
Base3D3H	229
BaseB	230
BaseB_0_t_tdt	232
BaseH	233
BaseH_0_t_tdt	235
Bloc_particulier	
BlocDdlLim< Bloc_particulier >	279
BlocGen	280
BlocGen_3_0	281
BlocGen_3_1	282
BlocGen_3_2	283
BlocGen_4_0	284
BlocGen_5_0	285
BlocGen_6_0	286
BlocGeneEtVecMultType	286
BlocScal	287
BlocScal_ou_fctnD	288
BlocScalType	288
BlocVec	289
BlocVecMultType	290
BlocVecType	290
Cercle	291
Tenseur1BBBB::ChangementIndex	292
Tenseur1BBHH::ChangementIndex	292
Tenseur1HHBB::ChangementIndex	292
Tenseur1HHHH::ChangementIndex	293
Tenseur2BB::ChangementIndex	293
Tenseur2BBBB::ChangementIndex	293
Tenseur2BBHH::ChangementIndex	293
Tenseur2BH::ChangementIndex	293
Tenseur2HB::ChangementIndex	294
Tenseur2HH::ChangementIndex	294
Tenseur2HHBB::ChangementIndex	294
Tenseur2HHHH::ChangementIndex	294
Tenseur3BB::ChangementIndex	294
Tenseur3BBBB::ChangementIndex	295
Tenseur3BBHH::ChangementIndex	295
Tenseur3BH::ChangementIndex	295
Tenseur3HB::ChangementIndex	295
Tenseur3HH::ChangementIndex	295
Tenseur3HHBB::ChangementIndex	296
Tenseur3HHHH::ChangementIndex	296
Tenseur_ns2BB::ChangementIndex	296
Tenseur_ns2HH::ChangementIndex	296
Tenseur_ns3BB::ChangementIndex	296
Tenseur_ns3HH::ChangementIndex	297
TenseurQ3_troisSym_BBBB::ChangementIndex	297
TenseurQ3_troisSym_HHHH::ChangementIndex	297
ClassPourEnum_ddl	301
ClassPourEnumTypeGrandeur	302
ClassPourEnumTypeQuelconque	303
ClassPourEnuTypeCL	303

ClassPourEnuTypeQuelParticulier . . . . .	304
CompCol_Mat_double	
Mat_creuse_CompCol . . . . .	1162
Condilinaire . . . . .	314
I_O_Condilinaire . . . . .	911
CondLim . . . . .	315
ConstMath . . . . .	359
ConstPhysico . . . . .	360
ConstrucElement	
Biel_axi::ConstrucElementbiel . . . . .	385
Biel_axiQ::ConstrucElementbiel . . . . .	386
Biellette::ConstrucElementbiel . . . . .	387
BielletteC1::ConstrucElementbiel . . . . .	388
BielletteQ::ConstrucElementbielQ . . . . .	390
BielletteThermi::ConstrucElementbiel . . . . .	389
ElemPoint::ConstrucElemPoint . . . . .	391
ElemPoint_CP::ConstrucElemPoint_CP . . . . .	392
Hexa::ConsHexa . . . . .	316
HexaQ::ConsHexaQ . . . . .	320
HexaQComp::ConsHexaQComp . . . . .	324
HexaQComp_cm1pti::ConsHexaQComp_cm1pti . . . . .	325
HexaQComp_cm27pti::ConsHexaQComp_cm27pti . . . . .	326
HexaQComp_cm64pti::ConsHexaQComp_cm64pti . . . . .	327
HexaQ_cm1pti::ConsHexaQ_cm1pti . . . . .	321
HexaQ_cm27pti::ConsHexaQ_cm27pti . . . . .	322
HexaQ_cm64pti::ConsHexaQ_cm64pti . . . . .	323
Hexa_cm1pti::ConsHexa_cm1pti . . . . .	317
Hexa_cm27pti::ConsHexa_cm27pti . . . . .	318
Hexa_cm64pti::ConsHexa_cm64pti . . . . .	319
PentaL::ConsPentaL . . . . .	328
PentaL_cm1pti::ConsPentaL_cm1pti . . . . .	329
PentaL_cm2pti::ConsPentaL_cm2pti . . . . .	330
PentaL_cm6pti::ConsPentaL_cm6pti . . . . .	331
PentaQ::ConsPentaQ . . . . .	332
PentaQComp::ConsPentaQComp . . . . .	337
PentaQComp_cm12pti::ConsPentaQComp_cm12pti . . . . .	338
PentaQComp_cm18pti::ConsPentaQComp_cm18pti . . . . .	339
PentaQComp_cm9pti::ConsPentaQComp_cm9pti . . . . .	340
PentaQ_cm12pti::ConsPentaQ_cm12pti . . . . .	333
PentaQ_cm18pti::ConsPentaQ_cm18pti . . . . .	334
PentaQ_cm3pti::ConsPentaQ_cm3pti . . . . .	335
PentaQ_cm9pti::ConsPentaQ_cm9pti . . . . .	336
PoutSfe3::ConsPoutSfe3 . . . . .	341
Quad::ConsQuad . . . . .	342
QuadAxiCCom::ConsQuadAxiCCom . . . . .	344
QuadAxiCCom_cm9pti::ConsQuadAxiCCom_cm9pti . . . . .	345
QuadAxiL1::ConsQuadAxiL1 . . . . .	346
QuadAxiL1_cm1pti::ConsQuadAxiL1_cm1pti . . . . .	347
QuadAxiQ::ConsQuadAxiQ . . . . .	348
QuadAxiQComp::ConsQuadAxiQComp . . . . .	349
QuadAxiQComp_cm4pti::ConsQuadAxiQComp_cm4pti . . . . .	350
QuadCCom::ConsQuadCCom . . . . .	351
QuadCCom_cm9pti::ConsQuadCCom_cm9pti . . . . .	352
QuadQ::ConsQuadQ . . . . .	353
QuadQCom::ConsQuadQCom . . . . .	354
QuadQCom_cm4pti::ConsQuadQCom_cm4pti . . . . .	355
Quad_cm1pti::ConsQuad_cm1pti . . . . .	343
Tetra::ConsTetra . . . . .	356

TetraQ::ConsTetraQ . . . . .	357
TetraQ_15pti::ConsTetraQ_15pti . . . . .	358
TetraQ_cm1pti::ConsTetraQ_cm1pti . . . . .	359
TriaAxiL1::ConsTriaAxiL1 . . . . .	360
TriaAxiQ3::ConsTriaAxiQ3 . . . . .	361
TriaAxiQ3_cm1pti::ConsTriaAxiQ3_cm1pti . . . . .	362
TriaAxiQ3_cmpti1003::ConsTriaAxiQ3_cmpti1003 . . . . .	363
TriaCub::ConsTriaCub . . . . .	364
TriaCub_cm4pti::ConsTriaCub_cm4pti . . . . .	365
TriaMembL1::ConsTriaMembL1 . . . . .	366
TriaMembQ3::ConsTriaMembQ3 . . . . .	367
TriaMembQ3_cm1pti::ConsTriaMembQ3_cm1pti . . . . .	368
TriaQ3_cmpti1003::ConsTriaQ3_cmpti1003 . . . . .	369
TriaQSfe1::ConsTriaQSfe1 . . . . .	370
TriaQSfe3::ConsTriaQSfe3 . . . . .	371
TriaSfe1::ConsTriaSfe1 . . . . .	372
TriaSfe1_cm5pti::ConsTriaSfe1_cm5pti . . . . .	373
TriaSfe2::ConsTriaSfe2 . . . . .	374
TriaSfe3::ConsTriaSfe3 . . . . .	375
TriaSfe3C::ConsTriaSfe3C . . . . .	384
TriaSfe3_3D::ConsTriaSfe3_3D . . . . .	376
TriaSfe3_cm12pti::ConsTriaSfe3_cm12pti . . . . .	377
TriaSfe3_cm13pti::ConsTriaSfe3_cm13pti . . . . .	378
TriaSfe3_cm3pti::ConsTriaSfe3_cm3pti . . . . .	379
TriaSfe3_cm4pti::ConsTriaSfe3_cm4pti . . . . .	380
TriaSfe3_cm5pti::ConsTriaSfe3_cm5pti . . . . .	381
TriaSfe3_cm6pti::ConsTriaSfe3_cm6pti . . . . .	382
TriaSfe3_cm7pti::ConsTriaSfe3_cm7pti . . . . .	383
Isovaleurs_Gmsh::ConstructTabScalVecTensGmsh . . . . .	392
GeomSeg::Construire_Gauss_Lobatto . . . . .	393
ElemGeomC0::ConteneurExtrapolation . . . . .	393
Coordonnee . . . . .	393
Coordonnee1 . . . . .	397
Coordonnee2 . . . . .	406
Coordonnee3 . . . . .	415
CoordonneeB . . . . .	424
Coordonnee1B . . . . .	400
Coordonnee2B . . . . .	408
Coordonnee3B . . . . .	417
CoordonneeH . . . . .	427
Coordonnee1H . . . . .	403
Coordonnee2H . . . . .	412
Coordonnee3H . . . . .	421
Courbe1D . . . . .	430
CourbePolyHermite1D . . . . .	476
CourbePolyLineaire1D . . . . .	480
Cpl1D . . . . .	490
CourbePolynomiale . . . . .	485
Courbe_cos . . . . .	436
Courbe_expo2_n . . . . .	440
Courbe_expo_n . . . . .	444
Courbe_expoaff . . . . .	448
Courbe_expression_litterale_1D . . . . .	452
Courbe_expression_litterale_avec_derivees_1D . . . . .	456
Courbe_ln_cosh . . . . .	461
Courbe_relax_expo . . . . .	465
Courbe_sin . . . . .	468



Courbe_un_moins_cos . . . . .	472
F1_plus_F2 . . . . .	609
F1_rond_F2 . . . . .	614
F_cycle_add . . . . .	618
F_cyclique . . . . .	623
F_union_1D . . . . .	633
Poly_Lagrange . . . . .	1414
SixpodeCos3phi . . . . .	1638
TangenteHyperbolique . . . . .	1701
TripodeCos3phi . . . . .	2105
Courbure_t_tdt . . . . .	488
CristaliniteAbstraite . . . . .	492
Hoffman1 . . . . .	826
Hoffman2 . . . . .	829
Cylindre . . . . .	493
Ddl . . . . .	494
VariablesExporter::Ddl_a_un_noeud . . . . .	497
Ddl_enum_etendu . . . . .	498
Ddl_etendu . . . . .	500
DdlLim::Ddlbloque . . . . .	503
DdlElement . . . . .	503
DdlLim . . . . .	504
DdlNoeudElement . . . . .	506
Deformation . . . . .	509
Def_Umat . . . . .	507
DeformationPP . . . . .	518
DeformationP2D . . . . .	516
DeformationSfe1 . . . . .	524
Deux_String . . . . .	542
Deux_String_un_entier . . . . .	542
DeuxCoordonnees . . . . .	543
DeuxDoubles . . . . .	543
Deformees_vrml::Deuxentiers . . . . .	544
DeuxEntiers . . . . .	544
OrdreVisu::Deuxentiers . . . . .	545
Deuxentiers_enu . . . . .	545
Algori::DeuxString . . . . .	545
DiversStockage . . . . .	547
HexaMemb::DonnComHexa . . . . .	548
PentaMemb::DonnComPenta . . . . .	550
QuadAxiMemb::DonnComQuad . . . . .	552
QuadraMemb::DonnComQuad . . . . .	554
SfeMembT::DonnComSfe . . . . .	556
TetraMemb::DonnComTetra . . . . .	558
TriaAxiMemb::DonnComTria . . . . .	560
TriaMemb::DonnComTria . . . . .	562
Biel_axi::Donnee_specif . . . . .	563
Biel_axiQ::Donnee_specif . . . . .	564
Biellette::Donnee_specif . . . . .	565
BielletteC1::Donnee_specif . . . . .	565
BielletteQ::Donnee_specif . . . . .	566
BielletteThermi::Donnee_specif . . . . .	566
QuadAxiMemb::Donnee_specif . . . . .	567
QuadraMemb::Donnee_specif . . . . .	567
SfeMembT::Donnee_specif . . . . .	568
TriaAxiMemb::Donnee_specif . . . . .	568
TriaMemb::Donnee_specif . . . . .	568

ElemPoint::DonneeCommune . . . . .	570
Droite . . . . .	571
Dynamiq . . . . .	572
EIContact . . . . .	573
Element	
ElemMeca . . . . .	580
Biel_axi . . . . .	237
Biel_axiQ . . . . .	244
Biellette . . . . .	251
BielletteC1 . . . . .	258
BielletteQ . . . . .	265
ElemPoint . . . . .	587
ElemPoint_CP . . . . .	592
HexaMemb . . . . .	805
Hexa . . . . .	795
HexaLMemb . . . . .	803
HexaQ . . . . .	810
HexaQComp . . . . .	818
HexaQComp_cm1pti . . . . .	820
HexaQComp_cm27pti . . . . .	822
HexaQComp_cm64pti . . . . .	824
HexaQ_cm1pti . . . . .	812
HexaQ_cm27pti . . . . .	814
HexaQ_cm64pti . . . . .	816
Hexa_cm1pti . . . . .	797
Hexa_cm27pti . . . . .	799
Hexa_cm64pti . . . . .	801
PentaMemb . . . . .	1374
PentaL . . . . .	1364
PentaL_cm1pti . . . . .	1367
PentaL_cm2pti . . . . .	1369
PentaL_cm6pti . . . . .	1372
PentaQ . . . . .	1379
PentaQComp . . . . .	1392
PentaQComp_cm12pti . . . . .	1394
PentaQComp_cm18pti . . . . .	1397
PentaQComp_cm9pti . . . . .	1399
PentaQ_cm12pti . . . . .	1382
PentaQ_cm18pti . . . . .	1384
PentaQ_cm3pti . . . . .	1387
PentaQ_cm9pti . . . . .	1389
PiPoCo . . . . .	1402
PoutSimple1 . . . . .	1432
PoutTimo . . . . .	1437
QuadAxiMemb . . . . .	1480
QuadAxiCCom . . . . .	1472
QuadAxiCCom_cm9pti . . . . .	1474
QuadAxiL1 . . . . .	1476
QuadAxiL1_cm1pti . . . . .	1478
QuadAxiQ . . . . .	1485
QuadAxiQComp . . . . .	1487
QuadAxiQComp_cm4pti . . . . .	1488
QuadraMemb . . . . .	1500
Quad . . . . .	1468
QuadCCom . . . . .	1490
QuadCCom_cm9pti . . . . .	1492
QuadQ . . . . .	1494

QuadQCom	.1496
QuadQCom_cm4pti	.1498
Quad_cm1pti	.1470
SfeMembT	.1632
PoutSfe3	.1429
TriaQSfe1	.2061
TriaQSfe3	.2064
TriaSfe1	.2067
TriaSfe1_cm5pti	.2070
TriaSfe2	.2073
TriaSfe3	.2076
TriaSfe3C	.2103
TriaSfe3_3D	.2079
TriaSfe3_cm12pti	.2082
TriaSfe3_cm13pti	.2085
TriaSfe3_cm3pti	.2088
TriaSfe3_cm4pti	.2091
TriaSfe3_cm5pti	.2094
TriaSfe3_cm6pti	.2097
TriaSfe3_cm7pti	.2100
TetraMemb	.2018
Tetra	.2016
TetraQ	.2023
TetraQ_15pti	.2025
TetraQ_cm1pti	.2027
TriaAxiMemb	.2034
TriaAxiL1	.2033
TriaAxiQ3	.2039
TriaAxiQ3_cm1pti	.2040
TriaAxiQ3_cmpti1003	.2042
TriaMemb	.2048
TriaCub	.2044
TriaCub_cm4pti	.2046
TriaMembL1	.2053
TriaMembQ3	.2055
TriaMembQ3_cm1pti	.2057
TriaQ3_cmpti1003	.2059
ElemThermi	.593
BielletteThermi	.272
Element pour l'instat en essai pour une classe générale au dessus du solide et des fluides	
ElemMeca	.580
ElemGeom	
Sfeg	.1631
ElemGeomC0	.576
ElemGeomC1	.578
GeomHexaCom	.714
GeomHexaCubique	.716
GeomHexaQuad	.720
GeomHexaQuadComp	.722
GeomHexalin	.718
GeomHexalin	.718
GeomPentaCom	.724
GeomPentaL	.726
GeomPentaQ	.728
GeomPentaQComp	.730
GeomPoint	.731

GeomQuadrangle . . . . .	733
GeomSeg . . . . .	735
GeomTetraCom . . . . .	738
GeomTetraL . . . . .	739
GeomTetraQ . . . . .	741
GeomTriangle . . . . .	743
EIFrontiere . . . . .	599
FrontPointF . . . . .	675
FrontQuadCC . . . . .	679
FrontQuadLine . . . . .	683
FrontQuadQC . . . . .	686
FrontQuadQuad . . . . .	690
FrontSegCub . . . . .	694
FrontSegLine . . . . .	698
FrontSegQuad . . . . .	702
FrontTriaLine . . . . .	706
FrontTriaQuad . . . . .	710
EnergieMeca . . . . .	601
EnergieThermi . . . . .	602
Entier_et_Double . . . . .	602
Epai . . . . .	603
Err_inconnue_ElemMeca . . . . .	603
ErrCalculFct_nD . . . . .	604
ErrJacobienNegatif_ElemMeca . . . . .	604
ErrJacobienNegatif_ElemThermi . . . . .	604
ErrMathUtil2 . . . . .	604
ErrNonConvergence_loiDeComportement . . . . .	604
ErrNonConvergence_Newton . . . . .	605
UtilLecture::ErrNouvelEnreg . . . . .	605
UtilLecture::ErrNouvelEnregCVisu . . . . .	605
UtilLecture::ErrNouvelleDonnee . . . . .	606
UtilLecture::ErrNouvelleDonneeCVisu . . . . .	606
ErrResolve_system_lineaire . . . . .	606
ErrSortie . . . . .	606
ErrSortieFinale . . . . .	607
ErrVarJacobienMini_ElemMeca . . . . .	607
ErrVarJacobienMini_ElemThermi . . . . .	607
Expli . . . . .	607
Expli_t_tdt . . . . .	608
EIContact::Fct_nD_contact . . . . .	638
Fonction_nD . . . . .	664
F_nD_courbe1D . . . . .	627
F_nD_courbe1D . . . . .	627
Fonc_scalcombinees_nD . . . . .	650
Fonc_scalcombinees_nD . . . . .	650
Fonction_expression_litterale_nD . . . . .	656
Fonction_expression_litterale_nD . . . . .	656
Fonction_externes_nD . . . . .	661
Force_hydroDyna . . . . .	669
Front . . . . .	670
LesCondLim::Gene_asso . . . . .	714
gijHH_0_et_giH_0 . . . . .	745
TypeQuelconque::Grandeur . . . . .	746
Grandeur_BaseH . . . . .	748
Grandeur_Ddl_etendu . . . . .	754
Grandeur_Double_Nommer_indicer . . . . .	761
Grandeur_TenseurBB . . . . .	770

Grandeur_TenseurBH	773
Grandeur_TenseurHB	776
Grandeur_TenseurHH	780
Grandeur_Vecteur	783
Grandeur_Vecteur_Nommer	786
Grandeur_cooronnee	751
Grandeur_defaut	758
Grandeur_scalaire_double	764
Grandeur_scalaire_entier	767
Tab2_Grandeur_TenseurHH	1652
Tab_Grandeur_BaseH	1656
Tab_Grandeur_Coordonnee	1659
Tab_Grandeur_Ddl_etendu	1663
Tab_Grandeur_TenseurBB	1672
Tab_Grandeur_TenseurBH	1675
Tab_Grandeur_TenseurHB	1678
Tab_Grandeur_TenseurHH	1681
Tab_Grandeur_Vecteur	1684
Tab_Grandeur_Vecteur_Nommer	1687
Tab_Grandeur_scalaire_double	1666
Tab_Grandeur_scalaire_entier	1669
Tableau_Grandeur_quelconque	1697
HyperD< class Tenseur3HH, class Tenseur3BB, class Tenseur3BH, class Tenseur3HB >	852
Hyper3D	836
Hyper1	831
Hyper10	833
Hyper10	833
IsoHyper3DFavier3	949
IsoHyper3DOrgeas1	954
IsoHyper3DOrgeas2	959
IsoHyperBulk3	964
IsoHyperBulk_gene	969
Impli	916
flambe_lin	648
Info0_t_tdt	923
InfoExp_t	924
InfoExp_tdt	924
InfoImp	925
Spectre::Initialisation_description_spectre	926
Initialisation_tab_Daa	926
Initialisation_tab_De	926
ElemPoint::inNeNpti	926
DdlElement::Int_initer	927
HyperD::Invariant	927
HyperD::InvariantVarDdl	930
HyperD::InvariantVarEps	931
Hyper3D::Invariant0QepsCosphi	928
Hyper3D::Invariant2Qeps	928
Hyper3D::Invariant2QepsCosphi	928
Hyper_W_gene_3D::Invariantpost3D	928
HyperD::Invariantpost3D	929
Hyper3D::InvariantQeps	929
Hyper3D::InvariantQepsCosphi	929
LaLIST	
LaLIST_io< T >	986
List_io< T >	1008
LectBloc	987

LectBlocmot	988
LesChargeExtSurElement	988
LesCondLim	988
LesContacts	990
LesCourbes1D	991
LesFonctions_nD	992
LesLoisDeComp	993
LesMaillages	994
LesMaillonsBB	997
LesMaillonsBBBB	998
LesMaillonsBBHH	998
LesMaillonsBH	999
LesMaillonsBHBH	1000
LesMaillonsBHHB	1000
LesMaillonsHB	1001
LesMaillonsHBBH	1001
LesMaillonsHBHB	1002
LesMaillonsHH	1003
LesMaillonsHHBB	1003
LesMaillonsHHHH	1004
LesPtIntegMecalInterne	1004
LesPtIntegThermiInterne	1005
LesReferences	1006
LesSuiteReel	1007
LesValVecPropres	1007
Algori::ListDeuxString	1009
LesLoisDeComp::Loi	1009
LoiAbstraiteGeneral	1093
CompFrotAbstraite	306
CompFrotCoulomb	308
CompThermoPhysiqueAbstraite	312
Loi_de_Tait	1016
Loi_iso_thermo	1044
Loi_comp_abstraite	1010
HyperDN< class Tenseur3HH, class Tenseur3BB, class Tenseur3BH, class Tenseur3HB, class Tenseur_ns3HH, class Tenseur_ns3BB >	858
Hyper3DN	840
TreloarN	2030
HyperD	852
Hyper3D	836
HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >	858
Hyper_W_gene_3D	849
Hart_Smith3D	789
Hyper_externes_W	843
Maheo_hyper	1139
MooneyRivlin3D	1257
Poly_hyper3D	1408
Hypo_hooke1D	862
Hypo_hooke2D_C	869
Hypo_hooke3D	875
Hypo_ortho3D_entrainee	881
Hysteresis1D	888
Hysteresis3D	894
Hysteresis3D	894
Hysteresis_bulk	904
Iso_elas_SE1D	944
Iso_elas_expo1D	932

Iso_elas_expo3D	938
LoiAdditiveEnSigma	1094
LoiContraintesPlanes	1101
LoiContraintesPlanesDouble	1108
LoiCritere	1116
LoiDeformationsPlanes	1125
LoiDesMelangesEnSigma	1132
Loi_Umat	1085
Loi_iso_elas1D	1019
Loi_iso_elas2D_C	1025
Loi_iso_elas2D_D	1031
Loi_iso_elas3D	1037
Loi_ortho2D_C_entrainee	1047
Loi_ortho_elas3D	1054
Loi_rien1D	1061
Loi_rien2D	1066
Loi_rien2D_C	1070
Loi_rien2D_D	1075
Loi_rien3D	1080
MooneyRivlin1D	1252
Prandtl_Reuss	1440
Prandtl_Reuss1D	1445
Prandtl_Reuss2D_D	1450
Projection_anisotrope_3D	1457
Maillage	1153
map	
Map_io< T >	1158
Mat_abstraite	1160
MatBand	1178
MatDiag	1187
MatLapack	1198
Mat_creuse_CompCol	1162
Mat_pleine	1170
ElemMeca::MatGeomInit	1195
ElemThermi::MatGeomInit	1196
MathUtil2	1196
Matrix_	1207
Met_abstraite	1207
MetAxisymetrique2D	1242
MetAxisymetrique3D	1248
Met_BielletteC1	1217
Met_ElemPoint	1220
Met_PiPoCo	1223
Met_Pout2D	1228
Met_Sfe1	1232
Met_biellette	1214
Met_triaMemb	1240
MotCle	1262
MV_ColMat< TYPE >	1263
MV_VecIndex	1263
MV_Vector< TYPE >	1265
FMV_Vector< TYPE >	649
MV_Vector< double >	1265
MV_Vecteur	1264
MV_Vector_	1266
MvtSolide	1266
Nb_assemb	1267

Maillage::NBelemEtArete . . . . .	1267
Maillage::NBelemEtFace . . . . .	1268
Maillage::NBelemEtpInteg . . . . .	1268
Maillage::NBelemFAEtpInteg . . . . .	1268
Noeud . . . . .	1269
Maillage::Noeud_degre . . . . .	1273
Biel_axi::NombresConstruire . . . . .	1273
Biel_axiQ::NombresConstruire . . . . .	1274
Biellette::NombresConstruire . . . . .	1274
BielletteC1::NombresConstruire . . . . .	1274
BielletteQ::NombresConstruire . . . . .	1275
BielletteThermi::NombresConstruire . . . . .	1275
ElemPoint::NombresConstruire . . . . .	1275
HexaMemb::NombresConstruire . . . . .	1276
Hexa::NombresConstruireHexa . . . . .	1284
HexaLMemb::NombresConstruireHexa . . . . .	1285
HexaQ::NombresConstruireHexaQ . . . . .	1289
HexaQComp::NombresConstruireHexaQComp . . . . .	1293
HexaQComp_cm1pti::NombresConstruireHexaQComp_cm1pti . . . . .	1294
HexaQComp_cm27pti::NombresConstruireHexaQComp_cm27pti . . . . .	1295
HexaQComp_cm64pti::NombresConstruireHexaQComp_cm64pti . . . . .	1296
HexaQ_cm1pti::NombresConstruireHexaQ_cm1pti . . . . .	1290
HexaQ_cm27pti::NombresConstruireHexaQ_cm27pti . . . . .	1291
HexaQ_cm64pti::NombresConstruireHexaQ_cm64pti . . . . .	1292
Hexa_cm1pti::NombresConstruireHexa_cm1pti . . . . .	1286
Hexa_cm27pti::NombresConstruireHexa_cm27pti . . . . .	1287
Hexa_cm64pti::NombresConstruireHexa_cm64pti . . . . .	1288
PentaMemb::NombresConstruire . . . . .	1277
PentaL::NombresConstruirePentaL . . . . .	1297
PentaL_cm1pti::NombresConstruirePentaL_cm1pti . . . . .	1298
PentaL_cm2pti::NombresConstruirePentaL_cm2pti . . . . .	1299
PentaL_cm6pti::NombresConstruirePentaL_cm6pti . . . . .	1300
PentaQ::NombresConstruirePentaQ . . . . .	1301
PentaQComp::NombresConstruirePentaQComp . . . . .	1306
PentaQComp_cm12pti::NombresConstruirePentaQComp_cm12pti . . . . .	1307
PentaQComp_cm18pti::NombresConstruirePentaQComp_cm18pti . . . . .	1308
PentaQComp_cm9pti::NombresConstruirePentaQComp_cm9pti . . . . .	1309
PentaQ_cm12pti::NombresConstruirePentaQ_cm12pti . . . . .	1302
PentaQ_cm18pti::NombresConstruirePentaQ_cm18pti . . . . .	1303
PentaQ_cm3pti::NombresConstruirePentaQ_cm3pti . . . . .	1304
PentaQ_cm9pti::NombresConstruirePentaQ_cm9pti . . . . .	1305
QuadAxiMemb::NombresConstruire . . . . .	1278
QuadAxiCCom::NombresConstruireQuadAxiCCom . . . . .	1313
QuadAxiCCom_cm9pti::NombresConstruireQuadAxiCCom_cm9pti . . . . .	1314
QuadAxiL1::NombresConstruireQuadAxiL1 . . . . .	1315
QuadAxiL1_cm1pti::NombresConstruireQuadAxiL1_cm1pti . . . . .	1316
QuadAxiQ::NombresConstruireQuadAxiQ . . . . .	1317
QuadAxiQComp::NombresConstruireQuadAxiQComp . . . . .	1318
QuadAxiQComp_cm4pti::NombresConstruireQuadAxiQComp_cm4pti . . . . .	1319
QuadraMemb::NombresConstruire . . . . .	1279
Quad::NombresConstruireQuad . . . . .	1311
QuadCCom::NombresConstruireQuadCCom . . . . .	1320
QuadCCom_cm9pti::NombresConstruireQuadCCom_cm9pti . . . . .	1321
QuadQ::NombresConstruireQuadQ . . . . .	1322
QuadQCom::NombresConstruireQuadQCom . . . . .	1323
QuadQCom_cm4pti::NombresConstruireQuadQCom_cm4pti . . . . .	1324
Quad_cm1pti::NombresConstruireQuad_cm1pti . . . . .	1312



SfeMembT::NombresConstruire . . . . .	1280
PoutSfe3::NombresConstruirePoutSfe3 . . . . .	1310
TriaQSfe1::NombresConstruireTriaQSfe1 . . . . .	1339
TriaQSfe3::NombresConstruireTriaQSfe3 . . . . .	1340
TriaSfe1::NombresConstruireTriaSfe1 . . . . .	1341
TriaSfe1_cm5pti::NombresConstruireTriaSfe1_cm5pti . . . . .	1342
TriaSfe2::NombresConstruireTriaSfe2 . . . . .	1343
TriaSfe3::NombresConstruireTriaSfe3 . . . . .	1344
TriaSfe3C::NombresConstruireTriaSfe3C . . . . .	1353
TriaSfe3_3D::NombresConstruireTriaSfe3_3D . . . . .	1345
TriaSfe3_cm12pti::NombresConstruireTriaSfe3_cm12pti . . . . .	1346
TriaSfe3_cm13pti::NombresConstruireTriaSfe3_cm13pti . . . . .	1347
TriaSfe3_cm3pti::NombresConstruireTriaSfe3_cm3pti . . . . .	1348
TriaSfe3_cm4pti::NombresConstruireTriaSfe3_cm4pti . . . . .	1349
TriaSfe3_cm5pti::NombresConstruireTriaSfe3_cm5pti . . . . .	1350
TriaSfe3_cm6pti::NombresConstruireTriaSfe3_cm6pti . . . . .	1351
TriaSfe3_cm7pti::NombresConstruireTriaSfe3_cm7pti . . . . .	1352
TetraMemb::NombresConstruire . . . . .	1281
Tetra::NombresConstruireTetra . . . . .	1325
TetraQ::NombresConstruireTetraQ . . . . .	1326
TetraQ_15pti::NombresConstruireTetraQ_15pti . . . . .	1327
TetraQ_cm1pti::NombresConstruireTetraQ_cm1pti . . . . .	1328
TriaAxiMemb::NombresConstruire . . . . .	1282
TriaAxiL1::NombresConstruireTriaAxiL1 . . . . .	1329
TriaAxiQ3::NombresConstruireTriaAxiQ3 . . . . .	1330
TriaAxiQ3_cm1pti::NombresConstruireTriaAxiQ3_cm1pti . . . . .	1331
TriaAxiQ3_cmpti1003::NombresConstruireTriaAxiQ3_cmpti1003 . . . . .	1332
TriaMemb::NombresConstruire . . . . .	1283
TriaCub::NombresConstruireTriaCub . . . . .	1333
TriaCub_cm4pti::NombresConstruireTriaCub_cm4pti . . . . .	1334
TriaMembL1::NombresConstruireTriaMembL1 . . . . .	1335
TriaMembQ3::NombresConstruireTriaMembQ3 . . . . .	1336
TriaMembQ3_cm1pti::NombresConstruireTriaMembQ3_cm1pti . . . . .	1337
TriaQ3_cmpti1003::NombresConstruireTriaQ3_cmpti1003 . . . . .	1338
OrdreVisu . . . . .	1354
Animation_geomview . . . . .	219
Animation_maple . . . . .	222
Animation_vrml . . . . .	225
ChoixDesMaillages_vrml . . . . .	300
Choix_grandeurs_maple . . . . .	297
Deformees_Gid . . . . .	533
Deformees_Gmsh . . . . .	535
Deformees_vrml . . . . .	539
Deformees_geomview . . . . .	531
Deformees_maple . . . . .	537
Fin_maple . . . . .	645
Fin_vrml . . . . .	646
Fin_Gid . . . . .	641
Fin_Gmsh . . . . .	643
Fin_geomview . . . . .	639
Frontiere_initiale . . . . .	671
Frontiere_initiale_geomview . . . . .	673
Increment_vrml . . . . .	921
Isovaleurs_Gid . . . . .	976
Isovaleurs_Gmsh . . . . .	979
Isovaleurs_vrml . . . . .	983

Isovaleurs_geomview . . . . .	974
Mail_initiale_Gid . . . . .	1146
Mail_initiale_Gmsh . . . . .	1149
Mail_initiale_vrml . . . . .	1151
Mail_initiale_geomview . . . . .	1145
Visuali_Gid . . . . .	2160
Visuali_Gmsh . . . . .	2162
Visuali_geomview . . . . .	2159
Visuali_maple . . . . .	2164
Visuali_vrml . . . . .	2165
Isovaleurs_Gid::P_gauss . . . . .	1356
Isovaleurs_Gmsh::P_gauss . . . . .	1357
ParaAlgoControle . . . . .	1358
ParaGlob . . . . .	1362
Plan . . . . .	1405
Hyper3D::PoGrenoble_V . . . . .	1405
Hyper3D::PoGrenoble_VV . . . . .	1406
Hyper3D::PoGrenobleAvecPhaseAvecVar . . . . .	1406
Hyper3D::PoGrenobleAvecPhaseSansVar . . . . .	1406
Hyper3D::PoGrenobleSansPhaseAvecVar . . . . .	1406
Hyper3D::PoGrenobleSansPhaseSansVar . . . . .	1407
UtilLecture::PointFich . . . . .	1407
Ponderation . . . . .	1417
Ponderation_Consultable . . . . .	1418
Ponderation_GGlobal . . . . .	1419
Ponderation_temps . . . . .	1420
Ponderation_TypeQuelconque . . . . .	1420
Posi_ddl_noeud . . . . .	1421
Maillage::PosiEtNoeud . . . . .	1423
UtilLecture::Position_BI . . . . .	1423
HyperD::PotenAvecPhaseSansVar . . . . .	1425
HyperD::PotenAvecPhaseAvecVar . . . . .	1424
HyperD::PotenSansPhaseSansVar . . . . .	1427
HyperD::PotenSansPhaseAvecVar . . . . .	1426
Met_abstraite::Pour_def_Almansi . . . . .	1427
Met_abstraite::Pour_def_log . . . . .	1428
Pre_cond_double . . . . .	1455
CompCol_ILUPreconditioner_double . . . . .	304
CompRow_ILUPreconditioner_double . . . . .	311
DiagPreconditioner_double . . . . .	546
ICPreconditioner_double . . . . .	913
ILUPreconditioner_double . . . . .	914
Pression_appliquee . . . . .	1456
Projet . . . . .	1464
PtIntegMecalInterne . . . . .	1465
PtIntegThermilInterne . . . . .	1467
Quartic . . . . .	1505
quatre_string_un_entier . . . . .	1506
LesContacts::ReactCont . . . . .	1509
LesCondLim::ReactStoc . . . . .	1510
Reels1 . . . . .	1522
Reels16 . . . . .	1522
Reels2 . . . . .	1523
Reels21 . . . . .	1523
Reels3 . . . . .	1523
Reels36 . . . . .	1523
Reels4 . . . . .	1523

Reels5	1523
Reels6	1524
Reels7	1524
Reels8	1524
Reels81	1524
Reels9	1524
ReelsPointe	1524
Reel16Pointe	1510
Reel1Pointe	1511
Reel21Pointe	1512
Reel2Pointe	1513
Reel36Pointe	1514
Reel3Pointe	1515
Reel4Pointe	1516
Reel5Pointe	1517
Reel6Pointe	1518
Reel7Pointe	1519
Reel81Pointe	1520
Reel8Pointe	1521
Reel9Pointe	1522
Reference	1526
ReferenceAF	1527
ReferenceNE	1530
ReferenceNE	1530
ReferencePtiAF	1533
LesLoisDeComp::RefLoi	1536
Resultats	1537
Rgb	1537
CristaliniteAbstraite::SaveCrista	1538
Hoffman1::SaveCrista_Hoffman1	1538
Hoffman2::SaveCrista_Hoffman2	1540
Deformation::SaveDefResul	1542
DeformationSfe1::SaveDefResulSfe1	1544
CompFrotAbstraite::SaveResul	1546
CompThermoPhysiqueAbstraite::SaveResul	1547
Loi_de_Tait::SaveResul_Loi_de_Tait	1552
Loi_iso_thermo::SaveResul_Loi_iso_thermo	1557
Loi_comp_abstraite::SaveResul	1548
HyperD::SaveResulHyperD	1583
HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN	1586
Hyper_W_gene_3D::SaveResulHyper_W_gene_3D	1581
Hyper_externe_W::SaveResulHyper_externe_W	1578
Hypo_hooke1D::SaveResulLoi_Hypo1D	1602
Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C	1549
Hypo_hooke3D::SaveResulLoi_Hypo3D	1605
Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee	1588
Hysteresis1D::SaveResulHysteresis1D	1591
Hysteresis3D::SaveResulHysteresis3D	1593
Hysteresis3D::SaveResulHysteresis3D	1593
Hysteresis_bulk::SaveResulHysteresis_bulk	1597
Iso_elas_expo1D::SaveResulIso_elas_expo1D	1600
LoiAdditiveEnSigma::SaveResul_LoiAdditiveEnSigma	1561
LoiContraintesPlanes::SaveResul_LoiContraintesPlanes	1564
LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble	1567
LoiCritere::SaveResul_LoiCritere	1569
LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes	1572

LoiDesMelangesEnSigma::SaveResul_LoiDesMelangesEnSigma	1575
Loi_Umat::SaveResul_Loi_Umat	1559
Loi_iso_elas1D::SaveResulLoi_iso_elas1D	1607
Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C	1554
Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D	1610
Loi_iso_elas3D::SaveResulLoi_iso_elas3D	1612
Loi_ortho2D_C_entraine::SaveResulLoi_ortho2D_C_entraine	1615
Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D	1617
Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D	1622
Prandtl_Reuss2D_D::SaveResulPrandtl_Reuss2D_D	1625
Prandtl_Reuss::SaveResulPrandtl_Reuss	1620
Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D	1627
Loi_comp_abstraite::SaveResul_C	1549
Sect	1630
Front::Signature_Front	1637
Spectre	1642
Sphere	1643
ElemMeca::StabMembBiel	1644
Deformation::Stmet	1645
CompThermoPhysiqueAbstraite::StockParaInt	1646
String_et_entier	1646
SuiteReel	1651
Suite_arithmetique	1647
Suite_equidistante	1648
Suite_geometrique	1650
Tab_car_double_int	1655
Tab_car_double_int_1	1655
Tab_car_et_double	1656
Table33	1691
Tableau< T >	1691
TabOper< T >	1700
Tableau2< T >	1692
Tableau4< T >	1693
Tableau< double >	1691
Tableau_double	1696
Tableau_3D	1694
Tableau_4D	1695
Tableau_5D	1695
Temps_CPU_HZpp	1705
Temps_CPU_HZpp_3	1706
TenseurBB	1887
Tenseur1BB	1707
Tenseur2BB	1754
Tenseur3BB	1802
Tenseur_ns2BB	1860
Tenseur_ns3BB	1873
TenseurBBBB	1894
Tenseur1BBBB	1713
Tenseur2BBBB	1761
Tenseur3BBBB	1809
Tenseur3BBBB	1809
TenseurQ1_troisSym_BBBB	1936
TenseurQ2_troisSym_BBBB	1962
TenseurQ2geneBBBB	1971
TenseurQ3_troisSym_BBBB	1989
TenseurQ3geneBBBB	1998
TenseurBBHH	1897

Tenseur1BBHH . . . . .	1718
Tenseur2BBHH . . . . .	1765
Tenseur3BBHH . . . . .	1816
Tenseur3BBHH . . . . .	1816
TenseurQ2geneBBHH . . . . .	1975
TenseurQ3geneBBHH . . . . .	2003
TenseurBH . . . . .	1900
Tenseur1BH . . . . .	1723
Tenseur2BH . . . . .	1770
Tenseur3BH . . . . .	1823
TenseurBHBH . . . . .	1909
TenseurQ1geneBHBH . . . . .	1945
TenseurBHHB . . . . .	1910
TenseurQ1geneBHHB . . . . .	1949
TenseurHB . . . . .	1911
Tenseur1HB . . . . .	1730
Tenseur2HB . . . . .	1778
Tenseur3HB . . . . .	1831
TenseurHBBH . . . . .	1920
TenseurQ1geneHBBH . . . . .	1953
TenseurHBHB . . . . .	1921
TenseurQ1geneHBHB . . . . .	1957
TenseurHH . . . . .	1922
Tenseur1HH . . . . .	1738
Tenseur2HH . . . . .	1786
Tenseur3HH . . . . .	1839
Tenseur_ns2HH . . . . .	1867
Tenseur_ns3HH . . . . .	1880
TenseurHHBB . . . . .	1930
Tenseur1HHBB . . . . .	1745
Tenseur2HHBB . . . . .	1793
Tenseur3HHBB . . . . .	1846
Tenseur3HHBB . . . . .	1846
TenseurQ2geneHHBB . . . . .	1980
TenseurQ3geneHHBB . . . . .	2007
TenseurHHHH . . . . .	1933
Tenseur1HHHH . . . . .	1749
Tenseur2HHHH . . . . .	1797
Tenseur3HHHH . . . . .	1853
Tenseur3HHHH . . . . .	1853
TenseurQ1_troisSym_HHHH . . . . .	1940
TenseurQ2_troisSym_HHHH . . . . .	1966
TenseurQ2geneHHHH . . . . .	1984
TenseurQ3_troisSym_HHHH . . . . .	1994
TenseurQ3geneHHHH . . . . .	2012
ThermoDonnee . . . . .	2028
LesCondLim::TorseurReac . . . . .	2029
Trois_String . . . . .	2109
TroisEntiers . . . . .	2110
Type_Calcul . . . . .	2111
VariablesExporter::TypeQuelc_Une_composante_Grandeur_globale . . . . .	2120
TypeQuelconque . . . . .	2121
TypeQuelconque_enum_etendu . . . . .	2125
Umat_cont . . . . .	2127
UmatAbaqus . . . . .	2128
ElemPoint::UneFois . . . . .	2130

HexaMemb::UneFois . . . . .	2132
PentaMemb::UneFois . . . . .	2134
QuadAxiMemb::UneFois . . . . .	2136
QuadraMemb::UneFois . . . . .	2138
SfeMembT::UneFois . . . . .	2140
TetraMemb::UneFois . . . . .	2142
TriaAxiMemb::UneFois . . . . .	2144
TriaMemb::UneFois . . . . .	2146
Util . . . . .	2147
Deformation::UtilIndexDeformation . . . . .	2149
UtilLecture . . . . .	2149
UtilXML . . . . .	2151
Courbe1D::Valbool . . . . .	2151
Courbe1D::ValDer . . . . .	2151
Courbe1D::ValDer2 . . . . .	2151
Courbe1D::ValDerbool . . . . .	2152
VariablesExporter . . . . .	2152
Vecteur . . . . .	2153
MV_Vecteur . . . . .	1264
vector	
Vector_io< T > . . . . .	2157
VeurPropre . . . . .	2158
Visualisation . . . . .	2167
Visualisation_geomview . . . . .	2167
Visualisation_Gid . . . . .	2168
Visualisation_Gmsh . . . . .	2168
Visualisation_maple . . . . .	2169

# Chapitre 3

## Index des classes

### 3.1 Liste des classes

Liste des classes, structures, unions et interfaces avec une brève description :

<a href="#">__CLPK_complex</a>	117
<a href="#">__CLPK_doublecomplex</a>	117
<a href="#">HyperD::A_i</a>	117
<a href="#">HyperD::A_iAvecVarDdl</a>	118
<a href="#">HyperD::A_iAvecVarEps</a>	119
<a href="#">VariablesExporter::A_un_E</a>	120
<a href="#">VariablesExporter::A_un_NE</a>	121
<a href="#">Algo_edp</a>	
BUT: Algorithmes de base pour la résolution d'équations différentielles	122
<a href="#">Algo_Integ1D</a>	
Algorithme d'intégration 1D	126
<a href="#">Algo_zero</a>	
Algorithmes de base pour la recherche de zéro d'une fonction ou d'un ensemble de fonctions	127
<a href="#">AlgoBonelli</a>	
BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînés. Correspond à l'implantation de l'algorithme de Bonelli et Bursi. Le modèle est de type galerkin discontinu en temps P1-P1, pour la discrétisation. L'algorithme est de type prédiction suivi de 1 ou plusieurs cycles de correction. Il est explicite, dans le sens où on n'utilise pas de comportement tangent du matériau et que l'on contrôle exactement le nombre d'itération du processus	130
<a href="#">AlgoInformations</a>	
BUT: Calcul d'informations générales types, géométriques par exemple, ou de visualisation	137
<a href="#">Algori</a>	
BUT: Classe de base de Définition des différents algorithmes de résolution	141
<a href="#">Algori_chung_lee</a>	
BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînés. On ne prend pas en compte les phénomènes de contact. Ici il s'agit de l'algorithme proposé par Chung Lee	150
<a href="#">AlgoriCombine</a>	
BUT: Algorithme combiné, l'objectif est de mettre en œuvre plusieurs algorithmes existants, suivant une stratégie que l'utilisateur impose au travers de fonction nD particulières	156
<a href="#">AlgoriDynaExpli</a>	
BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînés. On ne prend pas en compte les phénomènes de contact	161

<a href="#">AlgoriDynaExpli_zhai</a>	
BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. On ne prend pas en compte les phénomènes de contact. La méthode implantée est celle de zhai . . . . .	167
<a href="#">AlgoriFlambLineaire</a>	
BUT: Algorithme de calcul pour le flambement linéaire, le calcul préliminaire est non dynamique, pour de la mecanique en coordonnees materielles entrainees . . . . .	173
<a href="#">AlgoriNewmark</a>	
BUT: Algorithme de calcul dynamique, pour de la mecanique du solide déformable en coordonnees materielles entrainees, en utilisant la discrétisation temporelle de newmark . . . . .	177
<a href="#">AlgoriNonDyna</a>	
BUT: Algorithme de calcul non dynamique, pour de la mecanique du solide déformable en coordonnees materielles entrainees . . . . .	181
<a href="#">AlgoriRelaxDyna</a>	
BUT: Algorithme de relaxation dynamique selon la méthode de Barnes, modifiée par Julien Trouflard puis généralisée . . . . .	185
<a href="#">AlgoriRemontErreur</a>	
BUT: Algorithme de calcul de la remontée des contraintes aux noeuds, puis de calcul d'erreur, après un calcul de mécanique . . . . .	192
<a href="#">AlgoriRungeKutta</a>	
BUT: Algorithme de calcul dynamique, méthode de Runge Kutta, pour de la mecanique du solide	193
<a href="#">AlgoristatExpli</a>	
BUT: Algorithme de calcul mixte: statique - pseudo-dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees . . . . .	199
<a href="#">AlgoriTchamwa</a>	
BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. On ne prend pas en compte les phénomènes de contact. Correspond à l'implantation de l'algo de wielgosz tchamwa . . . . .	205
<a href="#">AlgoUmatAbaqus</a>	
BUT: Algorithme de calcul permettant l'utilisation d'herezh++ comme Umat pour Abaqus . . . . .	211
<a href="#">AlgoUtils</a>	
BUT: Utilitaires divers. ex: transformation de maillage, quadratique incomplet en quadratique complet . . . . .	215
<a href="#">Animation_geomview</a> . . . . .	219
<a href="#">Animation_maple</a> . . . . .	222
<a href="#">Animation_vrml</a> . . . . .	225
<a href="#">Assemblage</a> . . . . .	227
<a href="#">Banniere</a> . . . . .	227
<a href="#">Base3D3B</a>	
Les base covariantes et absolus . . . . .	228
<a href="#">Base3D3H</a>	
Les base duales . . . . .	229
<a href="#">BaseB</a>	
Les base covariantes et absolus . . . . .	230
<a href="#">BaseB_0_t_tdt</a>	
Un groupe de 3 bases covariantes et absolus à 0, t et tdt . . . . .	232
<a href="#">BaseH</a>	
Les base duales . . . . .	233
<a href="#">BaseH_0_t_tdt</a>	
Un groupe de 3 bases contravariant et absolus à 0, t et tdt . . . . .	235
<a href="#">Biel_axi</a> . . . . .	237
<a href="#">Biel_axiQ</a> . . . . .	244
<a href="#">Biellette</a> . . . . .	251
<a href="#">BielletteC1</a> . . . . .	258
<a href="#">BielletteQ</a> . . . . .	265
<a href="#">BielletteThermi</a> . . . . .	272
<a href="#">BlocDdlLim&lt; Bloc_particulier &gt;</a> . . . . .	279
<a href="#">BlocGen</a> . . . . .	280



BlocGen_3_0	281
BlocGen_3_1	282
BlocGen_3_2	283
BlocGen_4_0	284
BlocGen_5_0	285
BlocGen_6_0	286
BlocGeneEtVecMultType	286
BlocScal	287
BlocScal_ou_fctnD	288
BlocScalType	288
BlocVec	289
BlocVecMultType	290
BlocVecType	290
Cercle	291
Tenseur1BBBB::ChangementIndex	292
Tenseur1BBHH::ChangementIndex	292
Tenseur1HHBB::ChangementIndex	292
Tenseur1HHHH::ChangementIndex	293
Tenseur2BB::ChangementIndex	293
Tenseur2BBBB::ChangementIndex	293
Tenseur2BBHH::ChangementIndex	293
Tenseur2BH::ChangementIndex	293
Tenseur2HB::ChangementIndex	294
Tenseur2HH::ChangementIndex	294
Tenseur2HHBB::ChangementIndex	294
Tenseur2HHHH::ChangementIndex	294
Tenseur3BB::ChangementIndex	294
Tenseur3BBBB::ChangementIndex	295
Tenseur3BBHH::ChangementIndex	295
Tenseur3BH::ChangementIndex	295
Tenseur3HB::ChangementIndex	295
Tenseur3HH::ChangementIndex	295
Tenseur3HHBB::ChangementIndex	296
Tenseur3HHHH::ChangementIndex	296
Tenseur_ns2BB::ChangementIndex	296
Tenseur_ns2HH::ChangementIndex	296
Tenseur_ns3BB::ChangementIndex	296
Tenseur_ns3HH::ChangementIndex	297
TenseurQ3_troisSym_BBBB::ChangementIndex	297
TenseurQ3_troisSym_HHHH::ChangementIndex	297
Choix_grandeurs_maple	297
ChoixDesMaillages_vrml	300
ClassPourEnum_ddl	
Classe utilitaire entre Enum_ddl et une map	301
ClassPourEnumTypeGrandeur	
Def de map qui fait la liaison entre les string et les énumérés	302
ClassPourEnumTypeQuelconque	
Def de la map qui fait la liaison entre les string et les énumérés	303
ClassPourEnumTypeCL	
Def de la map qui fait la liaison entre les string et les énumérés	303
ClassPourEnumTypeQuelParticulier	
Def de la map qui fait la liaison entre les string et les énumérés	304
CompCol_ILUPreconditioner_double	304
CompFrotAbstraite	306
CompFrotCoulomb	308
CompRow_ILUPreconditioner_double	311
CompThermoPhysiqueAbstraite	
Pour les comportements thermophysiques	312

Condilinaire	314
CondLim	315
Hexa::ConsHexa	316
Hexa_cm1pti::ConsHexa_cm1pti	317
Hexa_cm27pti::ConsHexa_cm27pti	318
Hexa_cm64pti::ConsHexa_cm64pti	319
HexaQ::ConsHexaQ	320
HexaQ_cm1pti::ConsHexaQ_cm1pti	321
HexaQ_cm27pti::ConsHexaQ_cm27pti	322
HexaQ_cm64pti::ConsHexaQ_cm64pti	323
HexaQComp::ConsHexaQComp	324
HexaQComp_cm1pti::ConsHexaQComp_cm1pti	325
HexaQComp_cm27pti::ConsHexaQComp_cm27pti	326
HexaQComp_cm64pti::ConsHexaQComp_cm64pti	327
PentaL::ConsPentaL	328
PentaL_cm1pti::ConsPentaL_cm1pti	329
PentaL_cm2pti::ConsPentaL_cm2pti	330
PentaL_cm6pti::ConsPentaL_cm6pti	331
PentaQ::ConsPentaQ	332
PentaQ_cm12pti::ConsPentaQ_cm12pti	333
PentaQ_cm18pti::ConsPentaQ_cm18pti	334
PentaQ_cm3pti::ConsPentaQ_cm3pti	335
PentaQ_cm9pti::ConsPentaQ_cm9pti	336
PentaQComp::ConsPentaQComp	337
PentaQComp_cm12pti::ConsPentaQComp_cm12pti	338
PentaQComp_cm18pti::ConsPentaQComp_cm18pti	339
PentaQComp_cm9pti::ConsPentaQComp_cm9pti	340
PoutSfe3::ConsPoutSfe3	341
Quad::ConsQuad	342
Quad_cm1pti::ConsQuad_cm1pti	343
QuadAxiCCom::ConsQuadAxiCCom	344
QuadAxiCCom_cm9pti::ConsQuadAxiCCom_cm9pti	345
QuadAxiL1::ConsQuadAxiL1	346
QuadAxiL1_cm1pti::ConsQuadAxiL1_cm1pti	347
QuadAxiQ::ConsQuadAxiQ	348
QuadAxiQComp::ConsQuadAxiQComp	349
QuadAxiQComp_cm4pti::ConsQuadAxiQComp_cm4pti	350
QuadCCom::ConsQuadCCom	351
QuadCCom_cm9pti::ConsQuadCCom_cm9pti	352
QuadQ::ConsQuadQ	353
QuadQCom::ConsQuadQCom	354
QuadQCom_cm4pti::ConsQuadQCom_cm4pti	355
Tetra::ConsTetra	356
TetraQ::ConsTetraQ	357
TetraQ_15pti::ConsTetraQ_15pti	358
TetraQ_cm1pti::ConsTetraQ_cm1pti	359
ConstMath	359
ConstPhysico	360
TriaAxiL1::ConsTriaAxiL1	360
TriaAxiQ3::ConsTriaAxiQ3	361
TriaAxiQ3_cm1pti::ConsTriaAxiQ3_cm1pti	362
TriaAxiQ3_cmpti1003::ConsTriaAxiQ3_cmpti1003	363
TriaCub::ConsTriaCub	364
TriaCub_cm4pti::ConsTriaCub_cm4pti	365
TriaMembL1::ConsTriaMembL1	366
TriaMembQ3::ConsTriaMembQ3	367
TriaMembQ3_cm1pti::ConsTriaMembQ3_cm1pti	368
TriaQ3_cmpti1003::ConsTriaQ3_cmpti1003	369

TriaQSfe1::ConsTriaQSfe1	370
TriaQSfe3::ConsTriaQSfe3	371
TriaSfe1::ConsTriaSfe1	372
TriaSfe1_cm5pti::ConsTriaSfe1_cm5pti	373
TriaSfe2::ConsTriaSfe2	374
TriaSfe3::ConsTriaSfe3	375
TriaSfe3_3D::ConsTriaSfe3_3D	376
TriaSfe3_cm12pti::ConsTriaSfe3_cm12pti	377
TriaSfe3_cm13pti::ConsTriaSfe3_cm13pti	378
TriaSfe3_cm3pti::ConsTriaSfe3_cm3pti	379
TriaSfe3_cm4pti::ConsTriaSfe3_cm4pti	380
TriaSfe3_cm5pti::ConsTriaSfe3_cm5pti	381
TriaSfe3_cm6pti::ConsTriaSfe3_cm6pti	382
TriaSfe3_cm7pti::ConsTriaSfe3_cm7pti	383
TriaSfe3C::ConsTriaSfe3C	384
Biel_axi::ConstrucElementbiel	385
Biel_axiQ::ConstrucElementbiel	386
Biellette::ConstrucElementbiel	387
BielletteC1::ConstrucElementbiel	388
BielletteThermi::ConstrucElementbiel	389
BielletteQ::ConstrucElementbielQ	390
ElemPoint::ConstrucElemPoint	391
ElemPoint_CP::ConstrucElemPoint_CP	392
Isovaleurs_Gmsh::ConstructTabScalVecTensGmsh	392
GeomSeg::Construire_Gauss_Lobatto	393
ElemGeomC0::ConteneurExtrapolation	393
Coordonnee	
Coordonnees simples sans variances	393
Coordonnee1	
Cas des coordonnées simples sans variance	397
Coordonnee1B	
Cas des coordonnées covariantes	400
Coordonnee1H	
Cas des coordonnées contravariantes	403
Coordonnee2	
Cas des coordonnées simples sans variance	406
Coordonnee2B	
Cas des coordonnées covariantes	408
Coordonnee2H	
Cas des coordonnées contravariantes	412
Coordonnee3	
Cas des coordonnées simples sans variance	415
Coordonnee3B	
Cas des coordonnées covariantes	417
Coordonnee3H	
Cas des coordonnées contravariantes	421
CoordonneeB	
Cas des coordonnées covariantes	424
CoordonneeH	
Cas des coordonnées contravariantes	427
Courbe1D	
Classe virtuelle permettant le calcul d'une fonction 1D ainsi qu'éventuellement un certain nombre d'information supplémentaires telles que dérivées	430
Courbe_cos	
Classe permettant le calcul d'une fonction 1D de type : $\text{ampli} \cdot \cos(\alpha \cdot x + \beta)$	436
Courbe_expo2_n	
Classe permettant le calcul d'une fonction 1D de type : $(\gamma + \alpha \cdot (x \cdot x)^\wedge n)$	440

<a href="#">Courbe_expo_n</a>	Classe permettant le calcul d'une fonction 1D de type : $f(x) = (\text{gamma} + \alpha * (x)^n)$ . . . . .	444
<a href="#">Courbe_expoaff</a>	Classe permettant le calcul d'une fonction 1D de type : $f(x) = \text{gamma} + \alpha * ( x ^n)$ . . . . .	448
<a href="#">Courbe_expression_litterale_1D</a>	Classe permettant le calcul d'une fonction 1D de type : une expression littérale . . . . .	452
<a href="#">Courbe_expression_litterale_avec_derivees_1D</a>	Classe permettant le calcul d'une fonction 1D et de sa dérivée de type : une expression littérale . . . . .	456
<a href="#">Courbe_In_cosh</a>	Classe permettant le calcul d'une fonction 1D de type : $f(x) = a * (\text{be} + \text{ga} * x)^n * \ln(\cosh(\text{de} * (\text{he} + \text{ee} * x))) + \text{ke} * (\text{ge} + \text{re} * x)$ . . . . .	461
<a href="#">Courbe_relax_expo</a>	Classe permettant le calcul d'une fonction 1D de type : $f(x) = (a - b) * \exp(-c * x) + b$ . . . . .	465
<a href="#">Courbe_sin</a>	Classe permettant le calcul d'une fonction 1D de type : $f(x) = \text{ampli} * \sin(\alpha * x + \text{beta})$ . . . . .	468
<a href="#">Courbe_un_moins_cos</a>	Classe permettant le calcul d'une fonction 1D de type : $f(x) = c * (1 - \cos((x - a) * \text{Pi} / (b - a)))$ . . . . .	472
<a href="#">CourbePolyHermite1D</a>	Classe permettant le calcul d'une fonction 1D polynomiale 1D par morceau, avec continuité de la dérivée via une interpolation de type hermite . . . . .	476
<a href="#">CourbePolyLineaire1D</a>	Classe permettant le calcul d'une fonction 1D de type : $f(x) = \text{poly-linéaire } c - a - d \text{ linéaire par morceaux limités par 2 points}$ . . . . .	480
<a href="#">CourbePolynomiale</a>	Classe permettant le calcul d'un polynôme 1D de type : $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots$ . . . . .	485
<a href="#">Courbure_t_tdt</a>	CLASSE CONTENEUR : cas des grandeurs relatives à la courbure, grandeurs à 0, t et tdt les données sont publiques . . . . .	488
<a href="#">Cpl1D</a>	Classe permettant le calcul d'une fonction 1D de type : poly-linéaire 1D simplifiée, qui hérite de la fonction poly-linéaire . . . . .	490
<a href="#">CristaliniteAbstraite</a>	. . . . .	492
<a href="#">Cylindre</a>	. . . . .	493
<a href="#">Ddl</a>	BUT: La classe <a href="#">Ddl</a> permet de déclarer des degrés de liberté. soit c'est une variable ou soit c'est une données qui peuvent être soit active soit hors-service ensuite soit il est fixe ou soit il est bloqué 1) libre, fixe, ou 2) hors service libre ou hors service fixe ==> une variable 3) lisible libre ou fixe ou 4) hors service libre ou fixe ==> une donnée 1) c'est pour les ddl qui sont des variables que le calcul va déterminer, 2) sont les variables qui sont hors service: on ne les considère pas 3) sont des variables que l'on peut utiliser mais que le calcul ne cherche pas à déterminer: elle sont en lecture seule en quelque sorte 4) se sont des données qui ne sont pas disponibles pour la consultation (par exemple hors temps) . . . . .	494
<a href="#">VariablesExporter::Ddl_a_un_element</a>	. . . . .	495
<a href="#">VariablesExporter::Ddl_a_un_noeud</a>	. . . . .	497
<a href="#">Ddl_enum_etendu</a>	La classe principale: <a href="#">Ddl_enum_etendu</a> . . . . .	498
<a href="#">Ddl_etendu</a>	BUT: class de stockage d'un ddl étendue, associé au type <a href="#">Ddl_enum_etendu</a> . En fait il s'agit du pendant des types <a href="#">Ddl</a> associé au type <code>enum_ddl</code> Noter que la grandeur stockée est un scalaire !!	500
<a href="#">VariablesExporter::Ddl_etendu_a_un_noeud</a>	. . . . .	501
<a href="#">DdlLim::Ddlbloque</a>	. . . . .	503
<a href="#">DdlElement</a>	BUT: Définition, gestion et stockage des ddl de l'élément . . . . .	503
<a href="#">DdlLim</a>	BUT: La classe <a href="#">DdlLim</a> permet de déclarer un ensemble de ddl coiffé par une référence utilisée pour les conditions limites . . . . .	504
<a href="#">DdlNoeudElement</a>	BUT: Déf des ddl liés à un noeud d'un élément (différent des ddl liés aux noeuds globaux) . . . . .	506

<a href="#">Def_Umat</a>	
	BUT: Calcul des differentes grandeurs liee a la deformation Dans le cas des grandeurs relatives aux procedures Umat. 507
<a href="#">Deformation</a>	
	BUT: Calcul des differentes grandeurs liee a la deformation La classe fonctionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stocke pas (ou très peu) . . . . . 509
<a href="#">DeformationP2D</a>	516
<a href="#">DeformationPP</a>	
	BUT: Calcul des differentes grandeurs liee a la deformation d'éléments poutres et plaques classiques. Par rapport à la classe <a href="#">Deformation</a> de base, ici on considère deux directions $\leftrightarrow$ : l'épaisseur, et l'axe ou le plan dans ces deux directions, il y a des fcts d'interpolation parti- culières. La classe fonctionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stock pas . . . . . 518
<a href="#">DeformationSfe1</a>	
	BUT: Calcul des differentes grandeurs liee a la deformation des elements sfe1 La classe fonc- tionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stock pas . . . . . 524
<a href="#">Deformees_geomview</a>	531
<a href="#">Deformees_Gid</a>	533
<a href="#">Deformees_Gmsh</a>	535
<a href="#">Deformees_maple</a>	537
<a href="#">Deformees_vrml</a>	539
<a href="#">Deux_String</a>	
	Cas de deux String . . . . . 542
<a href="#">Deux_String_un_entier</a>	
	Cas de deux String et un entier . . . . . 542
<a href="#">DeuxCoordonnees</a>	
	<a href="#">DeuxCoordonnees</a> : classe relative à 2 coordonnées . . . . . 543
<a href="#">DeuxDoubles</a>	
	Cas de 2 double . . . . . 543
<a href="#">Deformees_vrml::Deuxentiers</a>	544
<a href="#">DeuxEntiers</a>	
	Cas de 2 entiers . . . . . 544
<a href="#">OrdreVisu::Deuxentiers</a>	545
<a href="#">Deuxentiers_enu</a>	545
<a href="#">Algori::DeuxString</a>	545
<a href="#">DiagPreconditioner_double</a>	546
<a href="#">DiversStockage</a>	
	Classe servant a stocker des informations intermediaires . . . . . 547
<a href="#">HexaMemb::DonnComHexa</a>	548
<a href="#">PentaMemb::DonnComPenta</a>	550
<a href="#">QuadAxiMemb::DonnComQuad</a>	552
<a href="#">QuadraMemb::DonnComQuad</a>	554
<a href="#">SfeMembT::DonnComSfe</a>	556
<a href="#">TetraMemb::DonnComTetra</a>	558
<a href="#">TriaAxiMemb::DonnComTria</a>	560
<a href="#">TriaMemb::DonnComTria</a>	562
<a href="#">Biel_axi::Donnee_specif</a>	563
<a href="#">Biel_axiQ::Donnee_specif</a>	564
<a href="#">Biellette::Donnee_specif</a>	565
<a href="#">BielletteC1::Donnee_specif</a>	565
<a href="#">BielletteQ::Donnee_specif</a>	566
<a href="#">BielletteThermi::Donnee_specif</a>	566

QuadAxiMemb::Donnee_specif	567
QuadraMemb::Donnee_specif	567
SfeMembT::Donnee_specif	568
TriaAxiMemb::Donnee_specif	568
TriaMemb::Donnee_specif	568
ElemPoint::DonneeCommune	570
Droite	571
Dynamiq	572
EIContact	573
ElemGeomC0	576
ElemGeomC1	578
ElemMeca	580
ElemPoint	587
ElemPoint_CP	592
ElemThermi	593
EIFrontiere	599
EnergieMeca	601
EnergieThermi	602
Entier_et_Double	
Cas d' 1 entier et 1 double	602
Epai	
Conteneur très basique pour les épaisseurs	603
Err_inconnue_ElemMeca	603
ErrCalculFct_nD	
Gestion d'exception pour des erreurs d'appel de fonction nD	604
ErrJacobienNegatif_ElemMeca	604
ErrJacobienNegatif_ElemThermi	604
ErrMathUtil2	
Cas d'une erreur survenue à cause d'une non convergence	604
ErrNonConvergence_loiDeComportement	
Cas d'une erreur survenue à cause d'une non convergence pour la résolution d'une loi de comportement incrémentale	604
ErrNonConvergence_Newton	
Pour la gestion d'exception pour non convergence	605
UtilLecture::ErrNouvelEnreg	605
UtilLecture::ErrNouvelEnregCVisu	605
UtilLecture::ErrNouvelleDonnee	606
UtilLecture::ErrNouvelleDonneeCVisu	606
ErrResolve_system_lineaire	606
ErrSortie	
Gestion d'exception pour Sortie	606
ErrSortieFinale	
Gestion d'exception pour Sortie finale	607
ErrVarJacobienMini_ElemMeca	607
ErrVarJacobienMini_ElemThermi	607
Expli	607
Expli_t_tdt	608
F1_plus_F2	
Classe permettant le calcul d'une fonction 1D de type : $F(x) = (F1 + F2)(x) = F1(x) + F2(x)$	609
F1_rond_F2	
Classe permettant le calcul d'une fonction 1D composé : $F(x) = F1(F2(x)) = F1 \circ F2(x)$	614
F_cycle_add	
Classe permettant le calcul d'une fonction cyclique avec une amplification additive à chaque cycle	618
F_cyclique	
Classe permettant le calcul d'une fonction cyclique avec une amplification multiplicative à chaque cycle	623

<a href="#">F_nD_courbe1D</a>	
Classe permettant d'utiliser une fonction courbe à l'intérieur d'une fonction nD	627
<a href="#">F_union_1D</a>	
Classe permettant le calcul d'une fonction égale à l'union d'une suite de fonctions définies sur des segments contigus de manière à couvrir un segment global	633
<a href="#">ElContact::Fct_nD_contact</a>	638
<a href="#">Fin_geomview</a>	639
<a href="#">Fin_Gid</a>	641
<a href="#">Fin_Gmsh</a>	643
<a href="#">Fin_maple</a>	645
<a href="#">Fin_vrml</a>	646
<a href="#">flambe_lin</a>	648
<a href="#">FMV_Vector&lt; TYPE &gt;</a>	649
<a href="#">Fonc_scalcombinees_nD</a>	
Classe permettant le calcul d'une fonction scalaire nD correspondant à une combinaison d'autres fonctions scalaire nD	650
<a href="#">Fonction_expression_litterale_nD</a>	
Classe permettant le calcul d'une fonction nD de type : une expression littérale	656
<a href="#">Fonction_externe_nD</a>	
Classe permettant le calcul d'une fonction nD externe, définie par l'utilisateur, via 2 pipes nommés	661
<a href="#">Fonction_nD</a>	
Classe virtuelle d'interface permettant le calcul d'une fonction nD ainsi qu'éventuellement un certain nombre d'informations supplémentaires telles que dérivées	664
<a href="#">Force_hydroDyna</a>	669
<a href="#">Front</a>	670
<a href="#">Frontiere_initiale</a>	671
<a href="#">Frontiere_initiale_geomview</a>	673
<a href="#">FrontPointF</a>	675
<a href="#">FrontQuadCC</a>	679
<a href="#">FrontQuadLine</a>	683
<a href="#">FrontQuadQC</a>	686
<a href="#">FrontQuadQuad</a>	690
<a href="#">FrontSegCub</a>	694
<a href="#">FrontSegLine</a>	698
<a href="#">FrontSegQuad</a>	702
<a href="#">FrontTriaLine</a>	706
<a href="#">FrontTriaQuad</a>	710
<a href="#">LesCondLim::Gene_asso</a>	714
<a href="#">GeomHexaCom</a>	714
<a href="#">GeomHexaCubique</a>	716
<a href="#">GeomHexalin</a>	718
<a href="#">GeomHexaQuad</a>	720
<a href="#">GeomHexaQuadComp</a>	722
<a href="#">GeomPentaCom</a>	724
<a href="#">GeomPentaL</a>	726
<a href="#">GeomPentaQ</a>	728
<a href="#">GeomPentaQComp</a>	730
<a href="#">GeomPoint</a>	731
<a href="#">GeomQuadrangle</a>	733
<a href="#">GeomSeg</a>	735
<a href="#">GeomTetraCom</a>	738
<a href="#">GeomTetraL</a>	739
<a href="#">GeomTetraQ</a>	741
<a href="#">GeomTriangle</a>	743
<a href="#">gijHH_0_et_giH_0</a>	745

TypeQuelconque::Grandeur	Définition de la classe virtuelle qui serait déclinée en une grandeur particulière il s'agit ici de la classe conteneur . . . . .	746
Grandeur_BaseH	Grandeur BaseH: TypeQuelconque::Grandeur::Grandeur_BaseH . . . . .	748
Grandeur_cooronnee	Grandeur coordonnée: TypeQuelconque::Grandeur::Grandeur_cooronnee . . . . .	751
Grandeur_Ddl_etendu	Grandeur Ddl_etendu: TypeQuelconque::Grandeur::Grandeur_Ddl_etendu . . . . .	754
Grandeur_default	. . . . .	758
Grandeur_Double_Nommer_indicer	Grandeur scalaire double nommé + indice: TypeQuelconque::Grandeur::Grandeur_Double_↔ Nommer_indicer contient un nom d'identificateur et un indice qui est sensé resté fixe pendant les opérations la seule grandeur qui varie c'est le double: c'est donc équivalent à un scalaire double l'indice et le nom_ref, servent pour la gestion . . . . .	761
Grandeur_scalaire_double	Grandeur scalaire réel: TypeQuelconque::Grandeur::Grandeur_scalaire_double . . . . .	764
Grandeur_scalaire_entier	Grandeur scalaire réel: TypeQuelconque::Grandeur::Grandeur_scalaire_double . . . . .	767
Grandeur_TenseurBB	Grandeur TenseurBB: TypeQuelconque::Grandeur::Grandeur_TenseurBB  . . . . .	770
Grandeur_TenseurBH	Grandeur TenseurBH: TypeQuelconque::Grandeur::Grandeur_TenseurBH  . . . . .	773
Grandeur_TenseurHB	Grandeur TenseurHB: TypeQuelconque::Grandeur::Grandeur_TenseurHB  . . . . .	776
Grandeur_TenseurHH	Grandeur TenseurHH: TypeQuelconque::Grandeur::Grandeur_TenseurHH  . . . . .	780
Grandeur_Vecteur	Grandeur vecteur: TypeQuelconque::Grandeur::Grandeur_vecteur . . . . .	783
Grandeur_Vecteur_Nommer	Grandeur vecteur nommé : TypeQuelconque::Grandeur::Grandeur_Vecteur_Nommer contient un nom d'identificateur (utilisé par exemple pour un nom de fonc . . . . .	786
Hart_Smith3D	. . . . .	789
Hexa	. . . . .	795
Hexa_cm1pti	. . . . .	797
Hexa_cm27pti	. . . . .	799
Hexa_cm64pti	. . . . .	801
HexaLMemb	. . . . .	803
HexaMemb	. . . . .	805
HexaQ	. . . . .	810
HexaQ_cm1pti	. . . . .	812
HexaQ_cm27pti	. . . . .	814
HexaQ_cm64pti	. . . . .	816
HexaQComp	. . . . .	818
HexaQComp_cm1pti	. . . . .	820
HexaQComp_cm27pti	. . . . .	822
HexaQComp_cm64pti	. . . . .	824
Hoffman1	. . . . .	826
Hoffman2	. . . . .	829
Hyper1	. . . . .	831
Hyper10	. . . . .	833
Hyper3D	. . . . .	836
Hyper3DN	. . . . .	840
Hyper_externe_W	. . . . .	843
Hyper_W_gene_3D	. . . . .	849
HyperD	. . . . .	852
HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >	. . . . .	858
Hypo_hooke1D	. . . . .	862



Hypo_hooke2D_C	869
Hypo_hooke3D	875
Hypo_ortho3D_entrainee	881
Hysteresis1D	888
Hysteresis3D	894
Hysteresis_bulk	904
I_O_Condilinaire	911
ICPreconditioner_double	913
ILUPreconditioner_double	914
Impli	916
ImpliNonDynaCont	
BUT: Algorithme de calcul implicite non dynamique avec contact , pour de la mecanique en coordonnees materielles entrainees	917
Increment_vrml	921
Info0_t_tdt	923
InfoExp_t	924
InfoExp_tdt	924
InfoImp	925
Spectre::Initialisation_description_spectre	926
Initialisation_tab_Daa	
Initialisation_tab_Daa: definition d'une classe pour l'initialisation du tableau tab_Daa	926
Initialisation_tab_De	
Une classe secondaire: definition d'une classe pour l'initialisation du tableau tab_De et de la map qui relie les string et les Ddl_enum_etendu	926
ElemPoint::inNeNpti	926
DdlElement::Int_initer	927
HyperD::Invariant	927
Hyper3D::Invariant0QepsCosphi	928
Hyper3D::Invariant2Qeps	928
Hyper3D::Invariant2QepsCosphi	928
Hyper_W_gene_3D::Invariantpost3D	928
HyperD::Invariantpost3D	929
Hyper3D::InvariantQeps	929
Hyper3D::InvariantQepsCosphi	929
HyperD::InvariantVarDdl	930
HyperD::InvariantVarEps	931
Iso_elas_expo1D	932
Iso_elas_expo3D	938
Iso_elas_SE1D	944
IsoHyper3DFavier3	949
IsoHyper3DOrgeas1	954
IsoHyper3DOrgeas2	959
IsoHyperBulk3	964
IsoHyperBulk_gene	969
Isovaleurs_geomview	974
Isovaleurs_Gid	976
Isovaleurs_Gmsh	979
Isovaleurs_vrml	983
LaLIST_io< T >	
LaLIST_io classe template identique à List_io: existe pour des raisons historiques et en prevision de changements futurs eventuelles (?)	986
LectBloc	987
LectBlocmot	988
LesChargeExtSurElement	988
LesCondLim	988
LesContacts	990
LesCourbes1D	
Classe de gestion des differentes courbes 1D enregistrees	991

LesFonctions_nD	
Gestion des différentes fonctions multivariées enregistrées	992
LesLoisDeComp	993
LesMaillages	
LesMaillages: l'ensemble des maillages	994
LesMaillonsBB	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	997
LesMaillonsBBBB	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	998
LesMaillonsBBHH	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	998
LesMaillonsBH	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	999
LesMaillonsBHBH	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	1000
LesMaillonsBHHB	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	1000
LesMaillonsHB	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	1001
LesMaillonsHBBH	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	1001
LesMaillonsHBHB	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	1002
LesMaillonsHH	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	1003
LesMaillonsHHBB	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	1003
LesMaillonsHHHH	
Def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires	1004
LesPtIntegMecalInterne	1004
LesPtIntegThermilInterne	1005
LesReferences	
Gestion des listes de références	1006
LesSuiteReel	
Gestion des différentes Suites de réels enregistrées	1007
LesValVecPropres	1007
List_io< T >	
List_io classe template identique à list de stl, avec en plus une lecture et écriture	1008
Algori::ListDeuxString	1009
LesLoisDeComp::Loi	1009
Loi_comp_abstraite	1010
Loi_de_Tait	1016
Loi_iso_elas1D	1019
Loi_iso_elas2D_C	1025
Loi_iso_elas2D_D	1031
Loi_iso_elas3D	1037
Loi_iso_thermo	1044
Loi_ortho2D_C_entrainee	1047
Loi_ortho_elas3D	1054
Loi_rien1D	1061
Loi_rien2D	1066
Loi_rien2D_C	1070
Loi_rien2D_D	1075
Loi_rien3D	1080
Loi_Umat	1085
LoiAbstraiteGeneral	1093
LoiAdditiveEnSigma	1094
LoiContraintesPlanes	1101

LoiContraintesPlanesDouble	1108
LoiCritere	1116
LoiDeformationsPlanes	1125
LoiDesMelangesEnSigma	1132
Maheo_hyper	1139
Mail_initiale_geomview	1145
Mail_initiale_Gid	1146
Mail_initiale_Gmsh	1149
Mail_initiale_vrml	1151
Maillage	
Maillage: un maillage particulier	1153
Map_io< T >	
Map_io classe template de type map STL avec une lecture et écriture	1158
Mat_abstraite	1160
Mat_creuse_CompCol	1162
Mat_pleine	1170
MatBand	1178
MatDiag	1187
ElemMeca::MatGeomInit	1195
ElemThermi::MatGeomInit	1196
MathUtil2	1196
MatLapack	1198
Matrix_	1207
Met_abstraite	1207
Met_biellette	1214
Met_BielletteC1	1217
Met_ElemPoint	1220
Met_PiPoCo	1223
Met_Pout2D	1228
Met_Sfe1	1232
Met_triaMemb	1240
MetAxisymetrique2D	1242
MetAxisymetrique3D	1248
MooneyRivlin1D	1252
MooneyRivlin3D	1257
MotCle	
Ici l'énuméré est remplacé par une classe	1262
MV_ColMat< TYPE >	1263
MV_VecIndex	1263
MV_Vecteur	
Définition d'une classe de jonction entre les Vecteurs et les MV_Vecteurs	1264
MV_Vector< TYPE >	1265
MV_Vector_	1266
MvtSolide	1266
Nb_assemb	
Nb_assemb: description des numéros d'assemblage	1267
Maillage::NBelemEtArete	1267
Maillage::NBelemEtFace	1268
Maillage::NBelemEtptInteg	1268
Maillage::NBelemFAEtptInteg	1268
Noeud	
Noeud: un noeud	1269
Maillage::Noeud_degre	1273
Biel_axi::NombresConstruire	1273
Biel_axiQ::NombresConstruire	1274
Biellette::NombresConstruire	1274
BielletteC1::NombresConstruire	1274
BielletteQ::NombresConstruire	1275

BielletteThermi::NombresConstruire	1275
ElemPoint::NombresConstruire	1275
HexaMemb::NombresConstruire	1276
PentaMemb::NombresConstruire	1277
QuadAxiMemb::NombresConstruire	1278
QuadraMemb::NombresConstruire	1279
SfeMembT::NombresConstruire	1280
TetraMemb::NombresConstruire	1281
TriaAxiMemb::NombresConstruire	1282
TriaMemb::NombresConstruire	1283
Hexa::NombresConstruireHexa	1284
HexaLMemb::NombresConstruireHexa	1285
Hexa_cm1pti::NombresConstruireHexa_cm1pti	1286
Hexa_cm27pti::NombresConstruireHexa_cm27pti	1287
Hexa_cm64pti::NombresConstruireHexa_cm64pti	1288
HexaQ::NombresConstruireHexaQ	1289
HexaQ_cm1pti::NombresConstruireHexaQ_cm1pti	1290
HexaQ_cm27pti::NombresConstruireHexaQ_cm27pti	1291
HexaQ_cm64pti::NombresConstruireHexaQ_cm64pti	1292
HexaQComp::NombresConstruireHexaQComp	1293
HexaQComp_cm1pti::NombresConstruireHexaQComp_cm1pti	1294
HexaQComp_cm27pti::NombresConstruireHexaQComp_cm27pti	1295
HexaQComp_cm64pti::NombresConstruireHexaQComp_cm64pti	1296
PentaL::NombresConstruirePentaL	1297
PentaL_cm1pti::NombresConstruirePentaL_cm1pti	1298
PentaL_cm2pti::NombresConstruirePentaL_cm2pti	1299
PentaL_cm6pti::NombresConstruirePentaL_cm6pti	1300
PentaQ::NombresConstruirePentaQ	1301
PentaQ_cm12pti::NombresConstruirePentaQ_cm12pti	1302
PentaQ_cm18pti::NombresConstruirePentaQ_cm18pti	1303
PentaQ_cm3pti::NombresConstruirePentaQ_cm3pti	1304
PentaQ_cm9pti::NombresConstruirePentaQ_cm9pti	1305
PentaQComp::NombresConstruirePentaQComp	1306
PentaQComp_cm12pti::NombresConstruirePentaQComp_cm12pti	1307
PentaQComp_cm18pti::NombresConstruirePentaQComp_cm18pti	1308
PentaQComp_cm9pti::NombresConstruirePentaQComp_cm9pti	1309
PoutSfe3::NombresConstruirePoutSfe3	1310
Quad::NombresConstruireQuad	1311
Quad_cm1pti::NombresConstruireQuad_cm1pti	1312
QuadAxiCCom::NombresConstruireQuadAxiCCom	1313
QuadAxiCCom_cm9pti::NombresConstruireQuadAxiCCom_cm9pti	1314
QuadAxiL1::NombresConstruireQuadAxiL1	1315
QuadAxiL1_cm1pti::NombresConstruireQuadAxiL1_cm1pti	1316
QuadAxiQ::NombresConstruireQuadAxiQ	1317
QuadAxiQComp::NombresConstruireQuadAxiQComp	1318
QuadAxiQComp_cm4pti::NombresConstruireQuadAxiQComp_cm4pti	1319
QuadCCom::NombresConstruireQuadCCom	1320
QuadCCom_cm9pti::NombresConstruireQuadCCom_cm9pti	1321
QuadQ::NombresConstruireQuadQ	1322
QuadQCom::NombresConstruireQuadQCom	1323
QuadQCom_cm4pti::NombresConstruireQuadQCom_cm4pti	1324
Tetra::NombresConstruireTetra	1325
TetraQ::NombresConstruireTetraQ	1326
TetraQ_15pti::NombresConstruireTetraQ_15pti	1327
TetraQ_cm1pti::NombresConstruireTetraQ_cm1pti	1328
TriaAxiL1::NombresConstruireTriaAxiL1	1329
TriaAxiQ3::NombresConstruireTriaAxiQ3	1330
TriaAxiQ3_cm1pti::NombresConstruireTriaAxiQ3_cm1pti	1331

TriaAxiQ3_cmpti1003::NombresConstruireTriaAxiQ3_cmpti1003	1332
TriaCub::NombresConstruireTriaCub	1333
TriaCub_cm4pti::NombresConstruireTriaCub_cm4pti	1334
TriaMembL1::NombresConstruireTriaMembL1	1335
TriaMembQ3::NombresConstruireTriaMembQ3	1336
TriaMembQ3_cm1pti::NombresConstruireTriaMembQ3_cm1pti	1337
TriaQ3_cmpti1003::NombresConstruireTriaQ3_cmpti1003	1338
TriaQSfe1::NombresConstruireTriaQSfe1	1339
TriaQSfe3::NombresConstruireTriaQSfe3	1340
TriaSfe1::NombresConstruireTriaSfe1	1341
TriaSfe1_cm5pti::NombresConstruireTriaSfe1_cm5pti	1342
TriaSfe2::NombresConstruireTriaSfe2	1343
TriaSfe3::NombresConstruireTriaSfe3	1344
TriaSfe3_3D::NombresConstruireTriaSfe3_3D	1345
TriaSfe3_cm12pti::NombresConstruireTriaSfe3_cm12pti	1346
TriaSfe3_cm13pti::NombresConstruireTriaSfe3_cm13pti	1347
TriaSfe3_cm3pti::NombresConstruireTriaSfe3_cm3pti	1348
TriaSfe3_cm4pti::NombresConstruireTriaSfe3_cm4pti	1349
TriaSfe3_cm5pti::NombresConstruireTriaSfe3_cm5pti	1350
TriaSfe3_cm6pti::NombresConstruireTriaSfe3_cm6pti	1351
TriaSfe3_cm7pti::NombresConstruireTriaSfe3_cm7pti	1352
TriaSfe3C::NombresConstruireTriaSfe3C	1353
OrdreVisu	1354
Isovaleurs_Gid::P_gauss	1356
Isovaleurs_Gmsh::P_gauss	1357
ParaAlgoControle	1358
ParaGlob	1362
PentaL	1364
PentaL_cm1pti	1367
PentaL_cm2pti	1369
PentaL_cm6pti	1372
PentaMemb	1374
PentaQ	1379
PentaQ_cm12pti	1382
PentaQ_cm18pti	1384
PentaQ_cm3pti	1387
PentaQ_cm9pti	1389
PentaQComp	1392
PentaQComp_cm12pti	1394
PentaQComp_cm18pti	1397
PentaQComp_cm9pti	1399
PiPoCo	1402
Plan	1405
Hyper3D::PoGrenoble_V	1405
Hyper3D::PoGrenoble_VV	1406
Hyper3D::PoGrenobleAvecPhaseAvecVar	1406
Hyper3D::PoGrenobleAvecPhaseSansVar	1406
Hyper3D::PoGrenobleSansPhaseAvecVar	1406
Hyper3D::PoGrenobleSansPhaseSansVar	1407
UtilLecture::PointFich	1407
Poly_hyper3D	1408
Poly_Lagrange	
Classe permettant le calcul d'un polynôme 1D à l'aide d'une interpolation de Lagrange	1414
Ponderation	
Une seconde classe de travail qui permet d'utiliser une pondération qui dépend de n <i>Courbe1D</i> ,	
chacune fonction d'un ddl étendu	1417

<a href="#">Ponderation_Consultable</a>	
Une classe de travail, qui permet d'utiliser une pondération qui dépend de plusieurs courbe1D, chacune dépendant d'une grandeur consultable par un string . . . . .	1418
<a href="#">Ponderation_GGlobal</a>	
Une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble de grandeurs globales au travers d'une fonction nD . . . . .	1419
<a href="#">Ponderation_temps</a>	
Une classe de travail qui permet d'utiliser une pondération qui dépend du temps via une <a href="#">Courbe1D</a> . . . . .	1420
<a href="#">Ponderation_TypeQuelconque</a>	
Une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble de grandeurs quelconque au travers d'une fonction nD . . . . .	1420
<a href="#">Posi_ddl_noeud</a>	
<a href="#">Posi_ddl_noeud</a> : un conteneur qui associe numéro de noeud, de maillage, et enu ddl . . . . .	1421
<a href="#">Maillage::PosiEtNoeud</a> . . . . .	1423
<a href="#">UtilLecture::Position_BI</a> . . . . .	1423
<a href="#">HyperD::PotenAvecPhaseAvecVar</a> . . . . .	1424
<a href="#">HyperD::PotenAvecPhaseSansVar</a> . . . . .	1425
<a href="#">HyperD::PotenSansPhaseAvecVar</a> . . . . .	1426
<a href="#">HyperD::PotenSansPhaseSansVar</a> . . . . .	1427
<a href="#">Met_abstraite::Pour_def_Almansi</a> . . . . .	1427
<a href="#">Met_abstraite::Pour_def_log</a> . . . . .	1428
<a href="#">PoutSfe3</a> . . . . .	1429
<a href="#">PoutSimple1</a> . . . . .	1432
<a href="#">PoutTimo</a> . . . . .	1437
<a href="#">Prandtl_Reuss</a> . . . . .	1440
<a href="#">Prandtl_Reuss1D</a> . . . . .	1445
<a href="#">Prandtl_Reuss2D_D</a> . . . . .	1450
<a href="#">Pre_cond_double</a> . . . . .	1455
<a href="#">Pression_appliquee</a> . . . . .	1456
<a href="#">Projection_anisotrope_3D</a> . . . . .	1457
<a href="#">Projet</a>	
Definition des operations de haut niveau : Lecture, Calcul, Sortie des <a href="#">Resultats</a> . . . . .	1464
<a href="#">PtIntegMecalInterne</a> . . . . .	1465
<a href="#">PtIntegThermilInterne</a> . . . . .	1467
<a href="#">Quad</a> . . . . .	1468
<a href="#">Quad_cm1pti</a> . . . . .	1470
<a href="#">QuadAxiCCom</a> . . . . .	1472
<a href="#">QuadAxiCCom_cm9pti</a> . . . . .	1474
<a href="#">QuadAxiL1</a> . . . . .	1476
<a href="#">QuadAxiL1_cm1pti</a> . . . . .	1478
<a href="#">QuadAxiMemb</a> . . . . .	1480
<a href="#">QuadAxiQ</a> . . . . .	1485
<a href="#">QuadAxiQComp</a> . . . . .	1487
<a href="#">QuadAxiQComp_cm4pti</a> . . . . .	1488
<a href="#">QuadCCom</a> . . . . .	1490
<a href="#">QuadCCom_cm9pti</a> . . . . .	1492
<a href="#">QuadQ</a> . . . . .	1494
<a href="#">QuadQCom</a> . . . . .	1496
<a href="#">QuadQCom_cm4pti</a> . . . . .	1498
<a href="#">QuadraMemb</a> . . . . .	1500
<a href="#">Quartic</a> . . . . .	1505
<a href="#">quatre_string_un_entier</a>	
Cas de 4 string et un entier . . . . .	1506
<a href="#">VariablesExporter::Quelconque_a_un_noeud</a> . . . . .	1507
<a href="#">LesContacts::ReactCont</a> . . . . .	1509
<a href="#">LesCondLim::ReactStoc</a> . . . . .	1510
<a href="#">Reel16Pointe</a> . . . . .	1510

Reel1Pointe	1511
Reel21Pointe	1512
Reel2Pointe	1513
Reel36Pointe	1514
Reel3Pointe	1515
Reel4Pointe	1516
Reel5Pointe	1517
Reel6Pointe	1518
Reel7Pointe	1519
Reel81Pointe	1520
Reel8Pointe	1521
Reel9Pointe	1522
Reels1	1522
Reels16	1522
Reels2	1523
Reels21	1523
Reels3	1523
Reels36	1523
Reels4	1523
Reels5	1523
Reels6	1524
Reels7	1524
Reels8	1524
Reels81	1524
Reels9	1524
ReelsPointe	
ReelsPointe classe virtuelle de laquelle dérive les classe contenant un pointeur de réel	1524
Reference	
Class <a href="#">Reference</a> : classe général virtuel des références. Les classes dérivées permettent de définir précisément les différentes références : de noeuds, d'élément, d'arêtes, de faces etc..	
Une instance de la classe s'identifie a partir d'un nom	1526
ReferenceAF	
Def, stockage et manipulation des références pour les faces et les arêtes	1527
ReferenceNE	
Def, stockage et manipulation des références pour les noeuds et les éléments	1530
ReferencePtiAF	
Def, stockage et manipulation des références pour les pti de faces et d'arêtes	1533
LesLoisDeComp::RefLoi	1536
Resultats	1537
Rgb	1537
CristaliniteAbstraite::SaveCrista	1538
Hoffman1::SaveCrista_Hoffman1	1538
Hoffman2::SaveCrista_Hoffman2	1540
Deformation::SaveDefResul	1542
DeformationSfe1::SaveDefResulSfe1	1544
CompFrotAbstraite::SaveResul	1546
CompThermoPhysiqueAbstraite::SaveResul	1547
Loi_comp_abstraite::SaveResul	1548
Loi_comp_abstraite::SaveResul_C	1549
Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C	1549
Loi_de_Tait::SaveResul_Loi_de_Tait	1552
Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C	1554
Loi_iso_thermo::SaveResul_Loi_iso_thermo	1557
Loi_Umat::SaveResul_Loi_Umat	1559
LoiAdditiveEnSigma::SaveResul_LoiAdditiveEnSigma	1561
LoiContraintesPlanes::SaveResul_LoiContraintesPlanes	1564
LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble	1567
LoiCritere::SaveResul_LoiCritere	1569

LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes	1572
LoiDesMelangesEnSigma::SaveResul_LoiDesMelangesEnSigma	1575
Hyper_externe_W::SaveResulHyper_externe_W	1578
Hyper_W_gene_3D::SaveResulHyper_W_gene_3D	1581
HyperD::SaveResulHyperD	1583
HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN	1586
Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee	1588
Hysteresis1D::SaveResulHysteresis1D	1591
Hysteresis3D::SaveResulHysteresis3D	1593
Hysteresis_bulk::SaveResulHysteresis_bulk	1597
Iso_elas_expo1D::SaveResulIso_elas_expo1D	1600
Hypo_hooke1D::SaveResulLoi_Hypo1D	1602
Hypo_hooke3D::SaveResulLoi_Hypo3D	1605
Loi_iso_elas1D::SaveResulLoi_iso_elas1D	1607
Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D	1610
Loi_iso_elas3D::SaveResulLoi_iso_elas3D	1612
Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee	1615
Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D	1617
Prandtl_Reuss::SaveResulPrandtl_Reuss	1620
Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D	1622
Prandtl_Reuss2D_D::SaveResulPrandtl_Reuss2D_D	1625
Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D	1627
Sect	
Conteneur très basique pour les sections	1630
Sfeg	1631
SfeMembT	1632
Front::Signature_Front	1637
SixpodeCos3phi	
Classe permettant le calcul d'une fonction 1D de type : $f(x) = 1./(1.+gamma*cos(3*phi)^2)^n$	1638
Spectre	1642
Sphere	1643
ElemMeca::StabMembBiel	1644
Deformation::Stmet	1645
CompThermoPhysiqueAbstraite::StockParalnt	1646
String_et_entier	
Cas d'un string et un entier	1646
Suite_arithmetique	
Classe permettant des calculs relatifs à des suites arithmétique: $u_n=q U_{(n-1)}$	1647
Suite_equidistante	
Suite_equidistante: Classe permettant des calculs relatifs à des suites equidistante: $u_n=U_{(n-1)}$	1648
Suite_geometrique	
Suite_geometrique: cas d'une suite géométrique	1650
SuiteReel	
Classe d'interface (virtuelle pure) pour la création et l'utilisation de suite	1651
Tab2_Grandeur_TenseurHH	
Grandeur un tableau à 2 dim de TenseurHH: TypeQuelconque::Grandeur::Tab2_Grandeur_↔	
TenseurHH	1652
Tab_car_double_int	1655
Tab_car_double_int_1	
Définition d'une union qui lie les réels, les entiers et les caractères	1655
Tab_car_et_double	1656
Tab_Grandeur_BaseH	
Grandeur un tableau de Grandeur_BaseH: TypeQuelconque::Grandeur::Tab_Grandeur_BaseH	
tous les coordonnees doivent avoir la même dimension !! pour la routine GrandeurNumOrdre	1656
Tab_Grandeur_Coordonnee	
Grandeur un tableau de Coordonnee: TypeQuelconque::Grandeur::Tab_Grandeur_↔	
CoordonneeHH tous les coordonnees doivent avoir la même dimension !! pour la routine	
GrandeurNumOrdre	1659



<a href="#">Tab_Grandeur_Ddl_etendu</a>	Grandeur un tableau de <a href="#">Coordonnee</a> : TypeQuelconque::Grandeur::Tab_Grandeur_↔ CoordonneeHH tous les coordonnees doivent avoir la même dimension!! pour la routine GrandeurNumOrdre . . . . .	1663
<a href="#">Tab_Grandeur_scalaire_double</a>	Grandeur tableau de scalaires réel: TypeQuelconque::Grandeur::Tab_Grandeur_scalaire_↔ double . . . . .	1666
<a href="#">Tab_Grandeur_scalaire_entier</a>	Grandeur tableau de scalaires entière: TypeQuelconque::Grandeur::Tab_Grandeur_scalaire_↔ entière . . . . .	1669
<a href="#">Tab_Grandeur_TenseurBB</a>	Grandeur un tableau de <a href="#">TenseurBB</a> : TypeQuelconque::Grandeur::Tab_Grandeur_TenseurBB  tous les tenseurs doivent avoir la même dimension!! pour la routine GrandeurNumOrdre . . .	1672
<a href="#">Tab_Grandeur_TenseurBH</a>	Grandeur un tableau de <a href="#">TenseurBH</a> : TypeQuelconque::Grandeur::Tab_Grandeur_TenseurBH  tous les tenseurs doivent avoir la même dimension!! pour la routine GrandeurNumOrdre . . .	1675
<a href="#">Tab_Grandeur_TenseurHB</a>	Grandeur un tableau de <a href="#">TenseurHB</a> : TypeQuelconque::Grandeur::Tab_Grandeur_TenseurHB tous les tenseurs doivent avoir la même dimension!! pour la routine GrandeurNumOrdre . . .	1678
<a href="#">Tab_Grandeur_TenseurHH</a>	Grandeur un tableau de <a href="#">TenseurHH</a> : TypeQuelconque::Grandeur::Tab_Grandeur_TenseurHH  tous les tenseurs doivent avoir la même dimension!! pour la routine GrandeurNumOrdre . . .	1681
<a href="#">Tab_Grandeur_Vecteur</a>	Grandeur un tableau de <a href="#">Vecteur</a> : TypeQuelconque::Grandeur::Tab_Grandeur_Vecteur  tous les Vecteurs doivent avoir la même dimension!! pour la routine GrandeurNumOrdre ?? à voir . . .	1684
<a href="#">Tab_Grandeur_Vecteur_Nommer</a>	Grandeur un tableau de <a href="#">Grandeur_Vecteur_Nommer</a> : TypeQuelconque::Grandeur::Tab_↔ Grandeur_Grandeur_Vecteur_Nommer . . . . .	1687
<a href="#">Table33</a>	. . . . .	1691
<a href="#">Tableau&lt; T &gt;</a>	. . . . .	1691
<a href="#">Tableau2&lt; T &gt;</a>	. . . . .	1692
<a href="#">Tableau4&lt; T &gt;</a>	. . . . .	1693
<a href="#">Tableau_3D</a>	. . . . .	1694
<a href="#">Tableau_4D</a>	. . . . .	1695
<a href="#">Tableau_5D</a>	. . . . .	1695
<a href="#">Tableau_double</a>	. . . . .	1696
<a href="#">Tableau_Grandeur_quelconque</a>	Un tableau de grandeur quelconque 1697	
<a href="#">TabOper&lt; T &gt;</a>	. . . . .	1700
<a href="#">TangenteHyperbolique</a>	Classe permettant le calcul d'une fonction 1D de type : $f(x) = a+b*\tanh((x-c)/d)$ . . . . .	1701
<a href="#">Temps_CPU_HZpp</a>	<a href="#">Temps_CPU_HZpp</a> une classe dédiée à la gestion des temps d'exécution, il s'agit ici uniquement du temps user . . . . .	1705
<a href="#">Temps_CPU_HZpp_3</a>	<a href="#">Temps_CPU_HZpp_3</a> une classe dédiée à la gestion des temps d'exécution, ici on cumule 3 compteurs: user, système et réel . . . . .	1706
<a href="#">Tenseur1BB</a>	Definition des tenseur derivees de dimension1. cas des composantes deux fois covariantes . .	1707
<a href="#">Tenseur1BBBB</a>	. . . . .	1713
<a href="#">Tenseur1BBHH</a>	. . . . .	1718
<a href="#">Tenseur1BH</a>	Definition des tenseur derivees de dimension1. cas des composantes mixtes BH . . . . .	1723
<a href="#">Tenseur1HB</a>	Definition des tenseur derivees de dimension1. cas des composantes mixtes HB . . . . .	1730
<a href="#">Tenseur1HH</a>	Definition des tenseur derivees de dimension1. cas des composantes deux fois contravariantes	1738

Tenseur1HHBB	1745
Tenseur1HHHH	1749
Tenseur2BB	
Definition des tenseur derivees de dimension 2. cas des composantes deux fois covariantes symetriques	1754
Tenseur2BBBB	1761
Tenseur2BBHH	1765
Tenseur2BH	
Definition des tenseur derivees de dimension 2. cas des composantes mixtes BH	1770
Tenseur2HB	
Definition des tenseur derivees de dimension 2. cas des composantes mixtes HB	1778
Tenseur2HH	
Definition des tenseur derivees de dimension 2. cas des composantes deux fois contravariantes symetriques	1786
Tenseur2HHBB	1793
Tenseur2HHHH	1797
Tenseur3BB	
Definition des tenseur derivees de dimension 3. cas des composantes deux fois covariantes	1802
Tenseur3BBBB	1809
Tenseur3BBHH	1816
Tenseur3BH	
Definition des tenseur derivees de dimension 3. cas des composantes mixtes BH	1823
Tenseur3HB	
Definition des tenseur derivees de dimension 3. cas des composantes mixtes HB	1831
Tenseur3HH	
Definition des tenseur derivees de dimension 3. cas des composantes deux fois contravariantes	1839
Tenseur3HHBB	1846
Tenseur3HHHH	1853
Tenseur_ns2BB	
Definition des tenseur derivees de dimension 2. cas des composantes deux fois covariantes non symetriques pour les differencier la dimension = -2	1860
Tenseur_ns2HH	
Definition des tenseur derivees de dimension 2. cas des composantes deux fois contravariantes non symetriques pour les differencier la dimension = -2	1867
Tenseur_ns3BB	
Definition des tenseur derivees de dimension 3. cas des composantes deux fois covariantes non symetriques pour les differencier la dimension = -3	1873
Tenseur_ns3HH	
Definition des tenseur derivees de dimension 3. cas des composantes deux fois contravariantes non symetriques pour les differencier la dimension = -3	1880
TenseurBB	
TenseurBB: cas des composantes deux fois covariantes	1887
TenseurBBBB	
Cas des composantes 4 fois covariantes	1894
TenseurBBHH	
Cas des composantes mixte BBHH	1897
TenseurBH	
TenseurBH: cas des composantes mixtes BH	1900
TenseurBHBH	
Cas des composantes 2 fois mixtes BHBH	1909
TenseurBHHB	
Cas des composantes 2 fois mixtes BHHB	1910
TenseurHB	
TenseurHB: cas des composantes mixtes HB	1911
TenseurHBBH	
Cas des composantes 2 fois mixtes HBBH	1920
TenseurHBHB	
Cas des composantes 2 fois mixtes HBHB	1921

TenseurHH	
TenseurHH: cas des composantes deux fois contravariantes	1922
TenseurHHBB	
Cas des composantes mixte HHBB	1930
TenseurHHHH	
Cas des composantes 4 fois contravariantes	1933
TenseurQ1_troisSym_BBBB	1936
TenseurQ1_troisSym_HHHH	1940
TenseurQ1geneBHBH	1945
TenseurQ1geneBHHB	1949
TenseurQ1geneHBBH	1953
TenseurQ1geneHBHB	1957
TenseurQ2_troisSym_BBBB	1962
TenseurQ2_troisSym_HHHH	1966
TenseurQ2geneBBBB	1971
TenseurQ2geneBBHH	1975
TenseurQ2geneHHBB	1980
TenseurQ2geneHHHH	1984
TenseurQ3_troisSym_BBBB	1989
TenseurQ3_troisSym_HHHH	1994
TenseurQ3geneBBBB	1998
TenseurQ3geneBBHH	2003
TenseurQ3geneHHBB	2007
TenseurQ3geneHHHH	2012
Tetra	2016
TetraMemb	2018
TetraQ	2023
TetraQ_15pti	2025
TetraQ_cm1pti	2027
ThermoDonnee	2028
LesCondLim::TorseurReac	2029
TreloarN	2030
TriaAxiL1	2033
TriaAxiMemb	2034
TriaAxiQ3	2039
TriaAxiQ3_cm1pti	2040
TriaAxiQ3_cmpti1003	2042
TriaCub	2044
TriaCub_cm4pti	2046
TriaMemb	2048
TriaMembL1	2053
TriaMembQ3	2055
TriaMembQ3_cm1pti	2057
TriaQ3_cmpti1003	2059
TriaQSfe1	2061
TriaQSfe3	2064
TriaSfe1	2067
TriaSfe1_cm5pti	2070
TriaSfe2	2073
TriaSfe3	2076
TriaSfe3_3D	2079
TriaSfe3_cm12pti	2082
TriaSfe3_cm13pti	2085
TriaSfe3_cm3pti	2088
TriaSfe3_cm4pti	2091
TriaSfe3_cm5pti	2094
TriaSfe3_cm6pti	2097
TriaSfe3_cm7pti	2100

TriaSfe3C	2103
TripodeCos3phi	
Classe permettant le calcul d'une fonction 1D de type : $f(x) = 1./(1.+gamma*cos(3*phi))^n$	2105
Trois_String	
Cas de trois String	2109
TroisEntiers	
Cas de 3 entiers	2110
Type_Calcul	2111
VariablesExporter::TypeEvoluee_a_un_element	2112
VariablesExporter::TypeParticulier_a_un_element	2114
VariablesExporter::TypeQuelc_arete_a_un_element	2116
VariablesExporter::TypeQuelc_face_a_un_element	2118
VariablesExporter::TypeQuelc_Une_composante_Grandeur_globale	2120
TypeQuelconque	
TypeQuelconque : def de la classe générique globale qui sert a gérer la grandeur particulière qui est attachée via un pointeur	2121
VariablesExporter::TypeQuelconque_a_Face_arete	2122
VariablesExporter::TypeQuelconque_a_un_element	2124
TypeQuelconque_enum_etendu	
TypeQuelconque_enum_etendu: la classe qui gère les types quelconques étendus	2125
Umat_cont	2127
UmatAbaqus	2128
ElemPoint::UneFois	2130
HexaMemb::UneFois	2132
PentaMemb::UneFois	2134
QuadAxiMemb::UneFois	2136
QuadraMemb::UneFois	2138
SfeMembT::UneFois	2140
TetraMemb::UneFois	2142
TriaAxiMemb::UneFois	2144
TriaMemb::UneFois	2146
Util	
Util: divers utilitaires sur vecteurs, coordonnées, matrices ..	2147
Deformation::UtilIndexDeformation	2149
UtilLecture	2149
UtilXML	2151
Courbe1D::Valbool	
Conteneur public pour une valeur et un booléen pour le strictement inclus	2151
Courbe1D::ValDer	
Conteneur public pour une valeur et une dérivée	2151
Courbe1D::ValDer2	
Conteneur public pour une valeur, une dérivée première et une dérivée seconde	2151
Courbe1D::ValDerbool	
Conteneur public pour une valeur et une dérivée et un booléen pour le strictement inclus	2152
VariablesExporter	2152
Vecteur	
La classe <b>Vecteur</b> permet de déclarer des vecteurs d'une longueur prédefinie par une allocation dynamique de mémoire. Les composantes d'un vecteur de cette classe sont de type double. Correspond à un vecteur absolu ( par opposition avec des vecteurs avec des coordonnées covariantes ou contravariantes	2153
Vector_io< T >	
Création de conteneurs type vector stl avec surcharge de lecture écriture	2157
VeurPropre	2158
Visuali_geomview	2159
Visuali_Gid	2160
Visuali_Gmsh	2162
Visuali_maple	2164
Visuali_vrml	2165

---

Visualisation	2167
Visualisation_geomview	2167
Visualisation_Gid	2168
Visualisation_Gmsh	2168
Visualisation_maple	2169



# Chapitre 4

## Index des fichiers

### 4.1 Liste des fichiers

Liste de tous les fichiers documentés avec une brève description :

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoRef/ <a href="#">Algori.h</a> . . . . .	2171
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgorithmeCombiner/ <a href="#">AlgoriCombine.h</a> . . . . .	2186
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/ <a href="#">AlgoInformations.h</a> . . . . .	2189
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/ <a href="#">AlgoriRemontErreur.h</a> . . . . .	2191
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/ <a href="#">AlgoUmatAbaqus.h</a> . . . . .	2192
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/ <a href="#">AlgoUtils.h</a> . . . . .	2195
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/ <a href="#">Algori_chung_lee.h</a> 2197	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/ <a href="#">Algori_relax_dyna.h</a> 2199	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/ <a href="#">Algori_tchamwa.h</a> 2204	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/ <a href="#">AlgoriDynaExpli.h</a> 2207	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/ <a href="#">AlgoriDynaExpli_zhai.h</a> 2210	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/ <a href="#">AlgoRungeKutta.h</a> 2212	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaImplicite/ <a href="#">AlgoriNewmark.h</a> 2215	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoMixte/ <a href="#">AlgoriMixte.h</a> . . . . .	2219
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/ <a href="#">AlgoriFlambLineaire.h</a> 2222	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/ <a href="#">AlgoriNonDyna.h</a> 2224	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/ <a href="#">ImpliNonDynaCont.h</a> 2227	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinDiscontinu/DG_DynaExplicite/ <a href="#">AlgoBonelli.h</a> 2229	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/ <a href="#">ComLoi_comp_abstraite.h</a> Définition de fonctions templates utilisées par les classes dérivées de <a href="#">Loi_comp_abstraite</a> . . . . .	2251
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/ <a href="#">CompFrotAbstraite.h</a> . . . . .	2252
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/ <a href="#">CompThermoPhysiqueAbstraite.h</a>	2255
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/ <a href="#">ExceptionsLoiComp.h</a> . . . . .	2263
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/ <a href="#">LesLoisDeComp.h</a> . . . . .	2380

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Loi_comp_abstraite.h . . . . .	2383
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LoiAbstraiteGeneral.h . . . . .	2396
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Hypo_ortho3D_entraine.e.h 2233	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi_ortho2D_C_entraine.e.h 2237	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi_ortho3D_entraine.e.h 2241	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Projection_anisotrope_3D.h 2246	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Energies_meca/EnergieMeca.h .	2259
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Energies_thermique/EnergieThermi.h 2261	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Frottement/CompFrotCoulomb.h .	2264
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hart_Smith3D.h	2266
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper1.h . . . .	2269
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper10.h . . .	2271
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper3D.h . . .	2273
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper3D_save.h	2277
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper3DN.h . .	2278
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper_externe_W.h 2279	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper_w1.h . .	2283
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper_W_gene_3D.h 2285	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/HyperD.h . . . .	2289
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/HyperDN.h . . .	2295
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/IsoHyper3DFavier3.h 2298	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/IsoHyper3DOrgeas1.h 2300	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/IsoHyper3DOrgeas2.h 2303	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/IsoHyperBulk3.h	2306
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/IsoHyperBulk_gene.h 2308	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Maheo_hyper.h .	2310
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/MooneyRivlin1D.h 2313	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/MooneyRivlin3D.h 2315	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Poly_hyper3D.h	2318
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/TreloarN.h . . .	2321
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo_elastique/Hypo_hooke1D.h	2322
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo_elastique/Hypo_hooke2D_C.h 2327	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo_elastique/Hypo_hooke3D.h	2330
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Copie_de_Hysteresis3D.h 2334	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis1D.h . . . .	2337
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis3D.h . . . .	2342
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis_bulk.h . . .	2351
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_hooke/Loi_iso_elas1D.h	2357
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_hooke/Loi_iso_elas2D_C.h 2360	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_hooke/Loi_iso_elas2D_D.h 2364	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_hooke/Loi_iso_elas3D.h	2368



/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_nonlinear/Iso_elas_expo1D.h	
2372	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_nonlinear/Iso_elas_expo3D.h	
2376	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_nonlinear/Iso_elas_SE1D.h	
2378	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loi_Umat/Loi_Umat.h . . . . .	2392
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiAdditiveEnSigma.h	
2399	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiContraintesPlanes.h	
2403	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiContraintesPlanesDouble.h	
2409	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiCritere.h . . .	2417
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiDeformationsPlanes.h	
2425	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiDesMelangesEnSigma.h	
2429	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien1D.h . . .	2434
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien2D.h . . .	2436
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien2D_C.h . .	2437
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien2D_D.h . .	2439
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien3D.h . . .	2441
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl_Reuss.h . . . .	2444
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl_Reuss1D.h . . .	2447
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl_Reuss2D_D.h .	2450
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi_de_Tait.h . . . . .	2453
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi_iso_thermo.h . . .	2456
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/ThermoDonnee.h . . .	2467
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux_crista/CristaliniteAbstraite.h	
2459	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux_crista/Hoffman1.h	2462
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux_crista/Hoffman2.h	2464
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Cercle.h . . . . .	2468
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Cylindre.h . . . . .	2470
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Droite.h . . . . .	2471
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/EIContact.h . . . . .	2473
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/LesContacts.h . . . . .	2478
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Plan.h . . . . .	2484
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Sphere.h . . . . .	2485
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/ElemGeomC0.h	2487
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/ElemGeomC1.h	2491
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/Ligne/GeomSeg.h	
2492	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/Point/GeomPoint.h	
2494	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/surface/GeomQuadrangle.h	
2496	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/surface/GeomTriangle.h	
2498	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexaCom.h	
2500	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexaCubique.h	
2505	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexalin.h	
2509	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexaQuad.h	
2512	

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexaQuadComp.h	
2515	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomPentaCom.h	
2518	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomPentaL.h	
2521	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomPentaQ.h	
2523	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomPentaQComp.h	
2525	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomTetra.h	
2527	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomTetraCom.h	
2529	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomTetraL.h	
2531	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomTetraQ.h	
2533	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/ElFrontiere.h . . .	2535
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Front.h . . . . .	2538
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSegCub.h	
2541	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSegLine.h	
2543	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSegQuad.h	
2545	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Point/FrontPointF.h	2547
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuadCC.h	
2550	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuadLine.h	
2552	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuadQC.h	
2554	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuadQuad.h	
2556	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontTriaLine.h	
2558	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontTriaQuad.h	
2560	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca.h . . . . .	2664
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ExceptionsElemMeca.h . .	2691
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/LesChargeExtSurElement.h	2728
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/LesPtIntegMecalInterne.h	2730
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/PtIntegMecalInterne.h . . .	2769
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel_axi.h . . . . .	2562
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel_axiQ.h . . . . .	2571
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biellette.h . . . . .	2581
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/BielletteC1.h . . . . .	2589
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/BielletteQ.h . . . . .	2597
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/DeformationP2D.h	2605
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met_biellette.h . .	2606
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met_BielletteC1.h	2608
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met_pout2D.h . .	2609
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/PoutSimple1.h . .	2611
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/PoutTimo.h . . . . .	2616
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Def_Umat.h	
2620	

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Deformation.h	
2621	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/DeformationPP.h	
2630	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Met_abstraite.h	
2632	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Met_abstraite_struc_donnees.h	
2641	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Met_PiPoCo.h	
2653	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/MetAxisymetrique2D.h	
2657	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/MetAxisymetrique3D.h	
2659	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/PiPoCo.h	
2661	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.h	2677
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint_CP.h	2684
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/Met_ElemPoint.h	2686
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/UmatAbaqus.h	2688
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa.h	2692
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa_cm1pti.h	2694
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa_cm27pti.h	2696
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa_cm64pti.h	2698
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaLMemb.h	2700
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaMemb.h	2702
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ.h	2711
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ_cm1pti.h	2713
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ_cm27pti.h	2715
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ_cm64pti.h	2717
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp.h	2719
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp_cm1pti.h	2721
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp_cm27pti.h	2723
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp_cm64pti.h	2726
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL.h	2732
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL_cm1pti.h	2734
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL_cm2pti.h	2736
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL_cm6pti.h	2738
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaMemb.h	2741
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ.h	2750
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm12pti.h	2752
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm18pti.h	2754
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm3pti.h	2756
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm9pti.h	2758
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp.h	2760
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp_cm12pti.h	2763
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp_cm18pti.h	2765

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp_cm9pti.h	
	2767
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiCCom.h	
	2773
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiCCom_cm9pti.h	
	2775
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiL1.h	
	2777
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiL1_cm1pti.h	
	2779
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiMemb.h	
	2781
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiQ.h	
	2789
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiQComp.h	
	2791
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiQComp_cm4pti.h	
	2793
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad.h . . . . .	2795
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad_cm1pti.h	2798
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom.h . . . . .	2800
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom_cm9pti.h	
	2802
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQ.h . . . . .	2804
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom.h . . . . .	2806
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom_cm4pti.h	
	2808
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.h	2810
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/DeformationSfe1.h . . . . .	2820
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met_Sfe1.h . . . . .	2824
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met_Sfe1_struc_donnees.h	
	2833
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/PoutSfe3.h . . . . .	2834
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Sfeg.h . . . . .	2836
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.h . . . . .	2837
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe1.h . . . . .	2848
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe3.h . . . . .	2850
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1.h . . . . .	2853
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1_cm5pti.h . . . . .	2855
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe2.h . . . . .	2857
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3.h . . . . .	2860
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_3D.h . . . . .	2862
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm12pti.h . . . . .	2864
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm13pti.h . . . . .	2866
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm3pti.h . . . . .	2868
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm4pti.h . . . . .	2871
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm5pti.h . . . . .	2873
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm6pti.h . . . . .	2875
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm7pti.h . . . . .	2877
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3C.h . . . . .	2880
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/Tetra.h . . . . .	2882
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraMemb.h . . . . .	2884
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ.h . . . . .	2892
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ_cm15pti.h	2894
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ_cm1pti.h	2897
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria_axisymetrie/TriaAxiL1.h	
	2899

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria_axisymetrie/TriaAxiMemb.h	
2901	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria_axisymetrie/TriaAxiQ3.h	
2909	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria_axisymetrie/TriaAxiQ3_cm1pti.h	
2911	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria_axisymetrie/TriaAxiQ3_cmpti1003.h	
2913	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/Met_triMemb.h	2915
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub.h	2916
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub_cm4pti.h	2918
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.h	2920
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembL1.h	2930
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3.h	2932
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3_cm1pti.h	2934
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaQ3_cmpti1003.h	2936
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermi.h	2946
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermiGene.h	2957
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ExceptionsElemThermi.h	2959
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/LesPtIntegThermiInterne.h	2960
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/PtIntegThermiInterne.h	2962
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/Biellette/BielletteThermi.h	2938
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_boolddl.h	
Def de l'enuméré concernant les booléens ddl	2964
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_calcul_masse.h	
Def de l'enuméré concernant les types de calcul de masse	2966
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_categorie_loi_comp.h	
Def de l'enuméré concernant les booléens ddl	2968
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_chargement.h	
Def de l'enuméré concernant les types de chargement	2970
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_comp.h	
Def de l'enuméré permettant d'identifier chaque loi de comportement	2972
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_contrainte_mathematique.h	
Def de l'enuméré concernant les types de contraintes mathématiques	2975
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_crista.h	
Def Enumération des différents calcul de cristallinité	2978
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_Critere_loi.h	
Enumération des différentes méthodes permettant d'utiliser des critères sur les lois de comportement	2980
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_ddl.h	
Def de l'enuméré concernant les ddl	2983
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_ddl_var_static.cc	
Def de grandeurs statiques relatives aux Enum_ddl	2989
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_dure.h	
Défini une énumération en temps	2989
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_geom.h	
Définition de l'enuméré concernant les types de modélisations de géométrie	2992
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_GrandeurGlobale.h	
Définition de l'enuméré concernant les grandeurs globales	2994
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_interpol.h	
Définition de l'enuméré concernant les différents types d'interpolation	2996
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_IO_XML.h	
Enumération des différentes entree/sortie XML	2998
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_liaison_noeud.h	
Enuméré pour définir les différents types de liaison entre noeuds	3000



/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_mat.h	
Définition de l'énuméré concernant les types de matériau	3002
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_matrice.h	
Enumeration des différents type de matrice possible	3004
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_PiPoCo.h	
Définir un type énuméré pour la différenciation entre des éléments classiques et le cas poutre plaque ou coque	3006
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_proj_aniso.h	
Enumeration des différentes méthodes concernant les techniques de projection anisotrope	3009
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_StabHourglass.h	
Enumeration des différentes méthodes permettant de stabiliser les modes d'hourglass	3011
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_StabMembrane.h	
Enumeration des différentes méthodes permettant de stabiliser les membranes transversalement	3014
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_Suite.h	
Enumeration des différentes Suites existantes	3016
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_type_deformation.h	
Défini une énumération des différents types de déformations	3018
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_type_geom.h	
Def de l'énuméré concernant les types de géométrie	3021
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_type_pt_integ.h	
Def de l'énuméré concernant les types de point d'intégration	3022
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_type_resolution_matri.h	
Enumeration des différents type de résolution de systèmes matricielle possible, ainsi que du préconditionnement éventuel	3024
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_type_stocke_deformation.h	
Définir une énumération pour les type de stockage pour les données de déformations à chaque pt de gauss	3027
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_TypeQuelconque.h	
Définition de l'énuméré pour repérer un type quelconque	3030
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum_variable_metrrique.h	
Définition de l'énuméré concernant les grandeurs gérées par les classes métriques	3035
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumCourbe1D.h	
Enumeration des différentes courbes 1D existantes	3037
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumElemTypeProblem.h	
Def de l'énuméré concernant les types de problème	3040
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumFonction_nD.h	
Enumeration des différentes fonctions nD existantes	3042
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumLangue.h	
Def Enumeration des différents langages	3044
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeCalcul.h	
Def Enumeration concernant les différents types de calcul globaux	3046
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeGradient.h	
Def du gradient de vitesse pour différentes conditions	3050
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeGrandeur.h	
Def de l'énuméré concernant les types de structures de grandeurs	3052
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypePilotage.h	
Def de l'énuméré concernant les types de pilotage	3054
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeViteRotat.h	
Def de l'énuméré concernant les type de vitesse de rotation	3056
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeVitesseDeform.h	
Définition de l'énuméré concernant les types de vitesse de déformation	3058
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeCL.h	
Type énuméré pour définir les différentes sous-classes de conditions limites	3060
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeQuelParticulier.h	
Def de l'énuméré concernant les types quelconques particuliers	3063
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/MotCle.h	
Def Enumeration des différents mot cle dans le fichier d'entree	3065

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Flambage/LesValVecPropres.h	3067
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/General/herezh.cc	
Programme principal herezh++	3068
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/General/herezh.h	
Entête du programme herezh++	3070
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/General/Projet.h	3072
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h	3075
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/LectBloc_T.h	3084
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/LectBlocMot.h	3086
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.h	3088
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilXML.h	3094
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/BlocDdlLim.h	
Def classe et fonctions template pour la lecture de ddl lim	3096
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Ddl.h	3099
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Ddl_etendu.h	3102
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlElement.h	3105
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlLim.h	3107
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlNoeudElement.h	3111
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DiversStockage.h	3112
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/I_O_Condilinaire.h	3115
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesCondLim.h	3118
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesMaillages.h	3125
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.h	3137
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Noeud.h	3149
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h	3159
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/Banniere.h	3168
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/Constante.h	
Def de tableaux d'initialisation.	
3169	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ConstantePrinc.h	
Def de tableaux d'initialisation	3171
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ConstMath.h	3172
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ConstPhysico.h	3173
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/EnteteParaGlob.h	3174
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ParaAlgoControle.h	3175
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ParaGlob.h	3184
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/Type_Calcul.h	3188
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/TypeCalcul.h	3190
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/Lect_reference.h	
Lecture des references definies dans le fichier au format ".lis" de nom : nom_fichier	3190
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/LesReferences.h	3192
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/Reference.h	3195
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferenceAF.h	3197
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferenceFA.h	3200
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferenceNE.cc	
Def d'une variable static motCle	3201
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferenceNE.h	3202
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferencePtiAF.h	3204
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/ExceptionsMatrices.h	3207
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/Mat_abstraite.h	3208
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/Mat_pleine.h	3213
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/MatBand.h	3218
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/MatDiag.h	3222
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/matrices_creuses/Mat_creuse_CompCol.h	
3226	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/matrices_lapack/MatLapack.h	3303
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/matrices_lapack/Include_↵	
lapack/clapack.h	3230

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices_externes/definition/vecdefs_GR.h	
3309	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices_externes/MV++/iotext_GR.h	3310
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices_externes/MV++/mvblas_GR.h	3310
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices_externes/MV++/mvtmp_GR.h	3313
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices_externes/MV++/mvvind_GR.h	3317
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices_externes/MV++/mvvrf_GR.h	3319
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices_externes/MV++/mvvtp_GR.h	3319
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/diagpre_double_GR.h	
3326	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/icpre_double_GR.h	3327
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/ilupre_double_GR.h	3328
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/pre_cond_double.h	3330
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution_Condi/Assemblage.h	3331
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution_Condi/Condilinaire.h	3333
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution_Condi/CondLim.h	3335
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/OrdreVisu.h	3395
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Resultats.h	3397
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Visualisation.h	3399
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Commun_visu/Frontiere_initiale.h	3338
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext_visu/color.h	3340
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext_visu/rgb.h	3340
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext_visu/spectre.h	3341
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Animation_geomview.h	3343
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Deformees_geomview.h	3345
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Fin_geomview.h	3346
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Frontiere_initiale_geomview.h	
3347	
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Isovaleurs_geomview.h	3348
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Mail_initiale_geomview.h	3350
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Visuali_geomview.h	3351
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Visualisation_geomview.h	3352
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Deformees_Gid.h	3354
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Fin_Gid.h	3356
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Isovaleurs_Gid.h	3357
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Mail_initiale_Gid.h	3362
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Visuali_Gid.h	3364
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Visualisation_Gid.h	3365
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Deformees_Gmsh.h	3367
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Fin_Gmsh.h	3369
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Isovaleurs_Gmsh.h	3371
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Mail_initiale_Gmsh.h	3376
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Visuali_Gmsh.h	3379
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Visualisation_Gmsh.h	3380
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Animation_maple.h	3383
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Choix_grandeurs_maple.h	3385
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Deformees_maple.h	3389
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Fin_maple.h	3390
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Visuali_maple.h	3392
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Visualisation_maple.h	3393
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Animation_vrml.h	3402
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/ChoixDesMaillages_vrml.h	3404
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Deformees_vrml.h	3405
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Fin_vrml.h	3407
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Increment_vrml.h	3408
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Isovaleurs_vrml.h	3409
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Mail_initiale_vrml.h	3411
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Visuali_vrml.h	3413



/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau2_T.h	3414
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau4_T.h	3418
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_3D.h	3422
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_4D.h	3424
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_5D.h	3426
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_double.h	3429
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_T.h	3431
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/TabOper_T.h	3441
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/utilDebug.h	3445
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee.h	3446
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee1.h	3454
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee2.h	3460
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee3.h	3466
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Reperes_bases/Base.h	3471
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Reperes_bases/Base3D3.h	3478
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/DefValConstens.h	3480
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/EnteteTenseur.h	3482
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/NevezTenseur.h	3484
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/NevezTenseurQ.h	3487
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/Tenseur.h	3495
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/Tenseur1.h	3507
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/Tenseur1_TroisSym.h	3515
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/Tenseur2.h	3519
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/Tenseur2_TroisSym.h	3529
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/Tenseur3.h	3533
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/Tenseur3_TroisSym.h	3544
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurO4.h	3548
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ-1.h	3553
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ-2.h	3560
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ-3.h	3567
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ.h	3576
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ1gene.h	3590
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ2gene.h	3594
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ3.h.old.h	3600
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ3gene.h	3605
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TypeConstens.h	3611
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TypeConstensPrinc.h	3612
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Vecteurs/Vecteur.h	3613
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Vecteurs/VeurPropre.h	3617
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h	3619
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl_enum_etendu-copie.h	3628
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl_enum_etendu.h	3630
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Epai.h	3633
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/List_io.h	3634
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Map_io.h	3637
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Nb_assemb.h	3639
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PlusieursCoordonnees.h	3640
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.h	3641
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h	3646
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel_Princ.h	
Listes de petits tableaux de réels	3653
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Section.h	3655
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Temps_CPU_HZpp.h	3656
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Temps_CPU_HZpp_3.h	3659
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque.h	3660
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque_enum_etendu.h	3664
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h	3667
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Vector_io.h	3694

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/unix/LaLIST.H	3695
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo_edp.h	3696
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo_Integ1D.h	3699
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo_zero.h	3700
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/CharUtil.h	
Utilitaires divers concernant les caracteres	3704
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Handler_exception.h	
Gestion d'une liste d'Handlers d'exceptions système	3805
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/MathUtil.h	
Utilitaires divers de calculs élémentaires	3807
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/MathUtil2.h	3811
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Sortie.h	
Classes de gestion de l'arrêt d'Herezh	3815
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Util.h	3823
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe1D.h	3707
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_cos.h	3710
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_expo2_n.h	3712
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_expo_n.h	3714
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_expoaff.h	3716
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_expression_litterale_1D.h	3718
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_expression_litterale_avec_derivees_1D.h	3720
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_In_cosh.h	3722
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_relax_expo.h	3724
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_sin.h	3726
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe_un_moins_cos.h	3728
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyHermite1D.h	3730
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyLineaire1D.h	3732
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyLineaire1D_simpli.h	3734
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolynomiale.h	3736
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F1_plus_F2.h	3738
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F1_rond_F2.h	3740
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F_cycle_add.h	3742
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F_cyclique.h	3745
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F_nD_courbe1D (copy: conflict on 2017-11-21).h	3747
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F_nD_courbe1D.h	3749
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F_union_1D.h	3752
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonc_scalcombinees_nD (copy: conflict on 2017-11-21).h	3754
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonc_scalcombinees_nD.h	3756
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction_expression_litterale_nD (copy: conflict on 2017-11-21).h	3760
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction_expression_litterale_nD.h	3761
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction_externe_nD.h	3763
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction_nD (conflict on 2019-05-08).h	3766
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction_nD (copy: conflict on 2017-11-21).h	3774
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction_nD.h	3779
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/LesCourbes1D.h	3788
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/LesFonctions_nD.h	3791
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Poly_Lagrange.h	3793
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/SixpodeCos3phi.h	3795
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/TangenteHyperbolique.h	3797
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/TripodeCos3phi.h	3799
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/externe/Racine.h	3801
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/MvtSolide/MvtSolide.h	3813

---

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/LesSuitesReel.h . . . . .	3817
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite_arithmetique.h . . . . .	3818
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite_equidistante.h . . . . .	3819
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite_geometrique.h . . . . .	3820
/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/SuiteReel.h . . . . .	3821



# Chapitre 5

## Documentation des modules

### 5.1 Les\_algorithmes\_de\_resolutions\_globales

groupe des algorithmes de résolutions globales

#### Classes

- class [Algori](#)  
*BUT: Classe de base de Defintion des differents algorithmes de resolution.*
- class [AlgoriCombine](#)  
*BUT: Algorithme combiné, l'objectif est de mettre en oeuvre plusieurs algorithme existants, suivant une stratégie que l'utilisateur impose au travers de fonction nD particulières.*
- class [AlgoInformations](#)  
*BUT: Calcul d'informations générales types, géométriques par exemple, ou de visualisation.*
- class [AlgoriRemontErreur](#)  
*BUT: Algorithme de calcul de la remontée des contraintes aux noeuds, puis de calcul d'erreur, après un calcul de mécanique.*
- class [AlgoUmatAbaqus](#)  
*BUT: Algorithme de calcul permettant l'utilisation d'herezh++ comme Umat pour Abaqus.*
- class [AlgoUtils](#)  
*BUT: Utilitaires divers. ex: transformation de maillage, quadratique incomplet en quadratique complet.*
- class [Algori\\_chung\\_lee](#)  
*BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. On ne prend pas en compte les phénomènes de contact. Ici il s'agit de l'algorithme proposé par Chung Lee.*
- class [AlgoriRelaxDyna](#)  
*BUT: Algorithme de relaxation dynamique selon la méthode de Barnes, modifiée par Julien Troufflard puis généralisée.*
- class [AlgoriTchamwa](#)  
*BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. On ne prend pas en compte les phénomènes de contact. Correspond à l'implantation de l'algo de wielgosz tchamwa.*
- class [AlgoriDynaExpli](#)  
*BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. On ne prend pas en compte les phénomènes de contact.*
- class [AlgoriDynaExpli\\_zhai](#)  
*BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. On ne prend pas en compte les phénomènes de contact. La méthode implantée est celle de zhai.*
- class [AlgoriRungeKutta](#)

- class [AlgoriNewmark](#)  
*BUT: Algorithme de calcul dynamique, méthode de Runge Kutta, pour de la mecanique du solide.*
- class [AlgoriNewmark](#)  
*BUT: Algorithme de calcul dynamique, pour de la mecanique du solide déformable en coordonnees materielles entrainees, en utilisant la discrétisation temporelle de newmark .*
- class [AlgoriNewmark](#)  
*BUT: Algorithme de calcul mixte: statique - pseudo-dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees.*
- class [AlgoriFlambLineaire](#)  
*BUT: Algorithme de calcul pour le flambement linéaire, le calcul préliminaire est non dynamique, pour de la mecanique en coordonnees materielles entrainees.*
- class [AlgoriNonDyna](#)  
*BUT: Algorithme de calcul non dynamique, pour de la mecanique du solide déformable en coordonnees materielles entrainees.*
- class [ImpliNonDynaCont](#)  
*BUT: Algorithme de calcul implicit non dynamique avec contact , pour de la mecanique en coordonnees materielles entrainees.*
- class [AlgoBonelli](#)  
*BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. Correspond à l'implantation de l'algo de Bonelli et Bursi. Le modèle est de type galerkin discontinu en temps P1-P1, pour la discrétisation. L'algo est de type prédiction suivi de 1 ou plusieurs cycle de correction. Il est explicite, dans le sens où on n'utilise pas de comportement tangent du matériau et que l'on contrôle exactement le nombre d'itération du processus.*

### 5.1.1 Description détaillée

groupe des algorithmes de résolutions globales

BUT: groupe des algorithmes de résolutions globales

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

10/02/2004

## 5.2 Les\_lois\_anisotropes

Définition des lois anisotropes.

### Classes

- class [Hypo\\_ortho3D\\_entrainee](#)
- class [Loi\\_ortho2D\\_C\\_entrainee](#)
- class [Loi\\_ortho\\_elas3D](#)
- class [Projection\\_anisotrope\\_3D](#)

### 5.2.1 Description détaillée

Définition des lois anisotropes.

BUT: groupe des lois anisotropes

Auteur

Gérard Rio

Version

1.0

Date

11/06/2019

## 5.3 Les\_conteneurs\_energies

groupe des conteneurs pour les différentes énergies

### Classes

- class [EnergieMeca](#)
- class [EnergieThermi](#)
- class [ThermoDonnee](#)

### 5.3.1 Description détaillée

groupe des conteneurs pour les différentes énergies

BUT: groupe des conteneurs pour les différentes énergies

Auteur

Gérard Rio

Version

1.0

Date

14/03/2005

## 5.4 Les\_lois\_hyperelastiques

groupe des lois hyperelastiques

## Classes

- class [Hart\\_Smith3D](#)
- class [Hyper1](#)
- class [Hyper10](#)
- class [Hyper3D](#)
- class [Hyper3DN](#)
- class [Hyper\\_externe\\_W](#)
- class [Hyper\\_W\\_gene\\_3D](#)
- class [HyperD](#)
- class [HyperDN< TensHH, TensBB, TensBH, TensHB, Tens\\_nHH, Tens\\_nBB >](#)
- class [IsoHyper3DFavier3](#)
- class [IsoHyper3DOrgeas1](#)
- class [IsoHyper3DOrgeas2](#)
- class [IsoHyperBulk3](#)
- class [IsoHyperBulk\\_gene](#)
- class [Maheo\\_hyper](#)
- class [MooneyRivlin1D](#)
- class [MooneyRivlin3D](#)
- class [Poly\\_hyper3D](#)
- class [TreloarN](#)

### 5.4.1 Description détaillée

groupe des lois hyperélastiques

BUT: groupe des lois hyperélastiques

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

11/06/2019

## 5.5 Les\_lois\_hypoélastiques

Définition des lois hypoélastiques.

## Classes

- class [Hypo\\_hooke1D](#)
- class [Hypo\\_hooke2D\\_C](#)
- class [Hypo\\_hooke3D](#)



### 5.5.1 Description détaillée

Définition des lois hypoélastiques.

BUT: groupe des lois hypoélastiques

Auteur

Gérard Rio

Version

1.0

Date

28/06/2004

## 5.6 Les\_lois\_hysteresis

Définition des lois de type hystérésis.

### Classes

- class [Hysteresis1D](#)
- class [Hysteresis3D](#)
- class [Hysteresis\\_bulk](#)

### 5.6.1 Description détaillée

Définition des lois de type hystérésis.

BUT: groupe des lois de type hystérésis

Auteur

Gérard Rio

Version

1.0

Date

10/02/2004

## 5.7 Les\_lois\_hooke

Définition des lois de type Hooke.

## Classes

- class [Loi\\_iso\\_elas1D](#)
- class [Loi\\_iso\\_elas2D\\_C](#)
- class [Loi\\_iso\\_elas2D\\_D](#)
- class [Loi\\_iso\\_elas3D](#)

### 5.7.1 Description détaillée

Définition des lois de type Hooke.

BUT: groupe des lois de type Hooke

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

15/09/2020

## 5.8 Les\_lois\_iso\_non\_lineaire

Définition des lois de type iso non linéaire.

## Classes

- class [Iso\\_elas\\_expo1D](#)
- class [Iso\\_elas\\_expo3D](#)
- class [Iso\\_elas\\_SE1D](#)

### 5.8.1 Description détaillée

Définition des lois de type iso non linéaire.

BUT: groupe des lois de type iso non linéaire

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

15/9/2003

## 5.9 Les\_lois\_combinees

Définition des lois combinées.

### Classes

- class [LoiAdditiveEnSigma](#)
- class [LoiContraintesPlanes](#)
- class [LoiContraintesPlanesDouble](#)
- class [LoiCritere](#)
- class [LoiDeformationsPlanes](#)
- class [LoiDesMelangesEnSigma](#)

### 5.9.1 Description détaillée

Définition des lois combinées.

BUT: groupe des lois combinées

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

11/06/2003

## 5.10 Les\_lois\_de\_plasticite

Définition des lois de plasticité

### Classes

- class [Prandtl\\_Reuss](#)
- class [Prandtl\\_Reuss1D](#)
- class [Prandtl\\_Reuss2D\\_D](#)

### 5.10.1 Description détaillée

Définition des lois de plasticité

BUT: groupe des lois de plasticité

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

## 5.11 Les\_lois\_concernant\_thermique

Définition des lois concernant la thermique.

### Classes

- class [Loi\\_de\\_Tait](#)
- class [Loi\\_iso\\_thermo](#)
- class [CristaliniteAbstraite](#)
- class [Hoffman1](#)
- class [Hoffman2](#)

### 5.11.1 Description détaillée

Définition des lois concernant la thermique.

pour les calculs de cristalinité

BUT: groupe des lois concernant la thermique

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

14/04/2004

## 5.12 Groupe\_sur\_les\_contacts

groupe relatif aux contacts

### Classes

- class [Cercle](#)
- class [Cylindre](#)
- class [Droite](#)
- class [EIContact](#)
- class [LesContacts](#)
- class [Plan](#)
- class [Sphere](#)

### 5.12.1 Description détaillée

groupe relatif aux contacts

BUT: groupe relatif aux contacts

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

## 5.13 Les\_Elements\_de\_geometrie

groupe concernant les éléments de géométrie 1D, 2D, 3D

### Classes

- class [ElemGeomC0](#)
- class [ElemGeomC1](#)
- class [GeomSeg](#)
- class [GeomPoint](#)
- class [GeomQuadrangle](#)
- class [GeomTriangle](#)
- class [GeomHexaCom](#)
- class [GeomHexaCubique](#)
- class [GeomHexalin](#)
- class [GeomHexaQuad](#)
- class [GeomHexaQuadComp](#)
- class [GeomPentaCom](#)
- class [GeomPentaL](#)
- class [GeomPentaQ](#)
- class [GeomPentaQComp](#)
- class [GeomTetraCom](#)
- class [GeomTetraL](#)
- class [GeomTetraQ](#)

### 5.13.1 Description détaillée

groupe concernant les éléments de géométrie 1D, 2D, 3D

BUT: groupe concernant les éléments de géométrie 1D, 2D, 3D

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

## 5.14 Les\_Elements\_de\_frontiere

groupe concernant les éléments géométriques définissant les frontières 1D, 2D, 3D

### Classes

- class [EIFrontiere](#)
- class [Front](#)
- class [FrontSegCub](#)
- class [FrontSegLine](#)
- class [FrontSegQuad](#)
- class [FrontPointF](#)
- class [FrontQuadCC](#)
- class [FrontQuadLine](#)
- class [FrontQuadQC](#)
- class [FrontQuadQuad](#)
- class [FrontTriaLine](#)
- class [FrontTriaQuad](#)

### 5.14.1 Description détaillée

groupe concernant les éléments géométriques définissant les frontières 1D, 2D, 3D

BUT: groupe concernant les éléments géométriques définissant les frontières 1D, 2D, 3D

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.15 Groupe\_des\_deformations

groupe relatif aux calculs de déformation

### Classes

- class [DeformationP2D](#)
- class [Def\\_Umat](#)
  - BUT: Calcul des differentes grandeurs liee a la deformation*
  - Dans le cas des grandeurs relatives aux procédures Umat.*
- class [Deformation](#)
  - BUT: Calcul des differentes grandeurs liee a la deformation La classe fonctionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stocke pas (ou très peu).*
- class [DeformationPP](#)
  - BUT: Calcul des differentes grandeurs liee a la deformation d'éléments poutres et plaques classiques. Par rapport à la classe [Deformation](#) de base, ici on considère deux directions : l'épaisseur, et l'axe ou le plan dans ces deux directions, il y a des fcts d'interpolation parti- culières. La classe fonctionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stock pas.*
- class [DeformationSfe1](#)
  - BUT: Calcul des differentes grandeurs liee a la deformation des elements sfe1 La classe fonctionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stock pas.*

### 5.15.1 Description détaillée

groupe relatif aux calculs de déformation

BUT: Calcul des différentes grandeurs liées à la déformation. La classe fonctionne comme une boîte à outil. On y choisit ce dont on a besoin. Bien faire attention à l'ordre d'appel des différentes méthodes lorsque il faut suivre une chronologie. Cette classe calcule mais ne stocke pas (ou très peu).

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.16 Groupe\_des\_metrique

groupe relatif aux calculs de métriques

### Classes

- class [Met\\_biellette](#)
- class [Met\\_BielletteC1](#)
- class [Met\\_Pout2D](#)
- class [Met\\_abstraite](#)
- class [Expli](#)
- class [Umat\\_cont](#)
- class [Expli\\_t\\_tdt](#)
- class [Impli](#)
- class [gijHH\\_0\\_et\\_giH\\_0](#)
- class [Dynamiq](#)
- class [flambe\\_lin](#)
- class [Infolmp](#)
- class [InfoExp\\_t](#)
- class [InfoExp\\_tdt](#)
- class [Info0\\_t\\_tdt](#)
- class [Met\\_PiPoCo](#)
- class [MetAxisymetrique2D](#)
- class [MetAxisymetrique3D](#)
- class [Met\\_ElemPoint](#)
- class [Met\\_Sfe1](#)
- class [Courbure\\_t\\_tdt](#)

*CLASSE CONTENEUR : cas des grandeurs relatives à la courbure, grandeurs à 0, t et tdt les données sont publiques.*

### Définitions de type

- typedef [Umat\\_cont](#) [Info\\_et\\_metrique\\_0\\_t\\_tdt](#)

### 5.16.1 Description détaillée

groupe relatif aux calculs de métriques

BUT: [Met\\_abstraite](#) constitue une boîte à outil qui permet d'obtenir les diverses grandeurs liées à la métrique. Se rappeler qu'avant la récupération d'une grandeur il faut la calculer\* En phase de mise au point c'a-d avec l'existence de la variable MISE\_AU\_POINT il y a vérification des tailles etc, en phase normale il n'y a plus de vérif ! La métrique est conçue pour être commune à toute une classe d'éléments d'où une gestion de la mémoire particulière, vu la taille des différents tableaux en fonctionnement normal les méthodes ne testent pas les différentes tailles et si les tableaux sont correctement alloués ce qui impose de bien gérer l'allocation à l'aide du constructeur et des routines publiques de gestion, ceci à chaque changement de pb par exemple passage d'explicit à implicite

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.17 Groupe\_concernant\_les\_points\_intégration

groupe relatif aux points d'intégration

### Classes

- class [LesPtIntegMecalInterne](#)
- class [PtIntegMecalInterne](#)
- class [LesPtIntegThermilInterne](#)
- class [PtIntegThermilInterne](#)

### 5.17.1 Description détaillée

groupe relatif aux points d'intégration

BUT: groupe relatif aux points d'intégration

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

26/11/2006



## 5.18 Group\_types\_enumeres

Def de grandeurs énumérées: permet une meilleure lisibilité du code et éventuellement un gain de place.

### Classes

- class `ClassPourEnum_ddl`  
*classe utilitaire entre Enum\_ddl et une map*
- class `Deuxentiers_enu`
- class `ClassPourEnumTypeQuelconque`  
*def de la map qui fait la liaison entre les string et les énumérés*
- class `ClassPourEnumTypeGrandeur`  
*def de map qui fait la liaison entre les string et les énumérés*
- class `ClassPourEnuTypeCL`  
*def de la map qui fait la liaison entre les string et les énumérés*
- class `ClassPourEnuTypeQuelParticulier`  
*def de la map qui fait la liaison entre les string et les énumérés*
- class `MotCle`  
*ici l'énuméré est remplacé par une classe*

### Énumérations

- enum `Enum_boolddl` {  
LIBRE = 0 , FIXE , HSLIBRE , HSFIXE ,  
SOUSLIBRE , SURFIXE , LISIBLE\_LIBRE , LISIBLE\_FIXE ,  
LISIBLE\_SOUSLIBRE , LISIBLE\_SURFIXE , HS\_LISIBLE\_LIBRE , HS\_LISIBLE\_FIXE ,  
HS\_SOUSLIBRE , HS\_SURFIXE , HS\_LISIBLE\_SOUSLIBRE , HS\_LISIBLE\_SURFIXE }  
*Définition de l'énuméré concernant les booléens ddl.*
- enum `Enum_calcul_masse` { MASSE\_DIAG\_COEF\_EGAUX = 1 , MASSE\_DIAG\_COEF\_VAR , MASSE\_↔  
\_CONSISTANTE }  
*def de l'énuméré concernant les types de calcul de masse*
- enum `Enum_categorie_loi_comp` {  
CAT\_MECANIQUE =1 , CAT\_THERMO\_MECANIQUE , CAT\_THERMO\_PHYSIQUE , CAT\_FROTTEMENT  
,  
RIEN\_CATEGORIE\_LOI\_COMP }  
*Définition de l'énuméré concernant catégories de lois de comportement.*
- enum `Enum_chargement` {  
RIEN\_CHARGEMENT = 1 , UNIFORME , PRESSION , PONCTUELLE ,  
PRESSDIR , PHYDRO , LINEIQUE , VOLUMIQUE ,  
LINEIC\_SUIVEUSE , P\_HYDRODYNA , TORSEUR\_PONCT }  
*Définition de l'énuméré concernant les types de chargement.*
- enum `Enum_comp` {  
ISOELAS =1 , ISOELAS1D , ISOELAS2D\_D , ISOELAS2D\_C ,  
ISO\_ELAS\_ESPO1D , ISO\_ELAS\_SE1D , ISO\_ELAS\_ESPO3D , ORTHOELA3D ,  
ORTHOELA2D\_D , ORTHOELA2D\_C , HYPO\_ORTHO3D , HYPO\_ORTHO2D\_D ,  
HYPO\_ORTHO2D\_C , PROJECTION\_ANISOTROPE\_3D , VISCOELA , ISOHYPER ,  
ISOHYPER1 , ISOHYPER10 , ISOHYSTE , TRELOAR ,  
ISOHYPER3DFAVIER1 , ISOHYPER3DFAVIER2 , ISOHYPER3DFAVIER3 , ISOHYPER3DFAVIER4 ,  
ISOHYPER3DORGEAS1 , ISOHYPER3DORGEAS2 , ISOHYPERBULK3 , ISOHYPERBULK\_GENE ,  
PRANDTL\_REUSS , PRANDTL\_REUSS2D\_D , PRANDTL\_REUSS2D\_C , PRANDTL\_REUSS1D ,  
NEWTON1D , NEWTON2D\_C , NEWTON2D\_D , NEWTON3D ,  
HYPO\_ELAS3D , HYPO\_ELAS2D\_C , HYPO\_ELAS2D\_D , HYPO\_ELAS1D ,  
MAXWELL1D , MAXWELL2D\_C , MAXWELL2D\_D , MAXWELL3D ,  
LOI\_ADDITIVE\_EN\_SIGMA , LOI\_DES\_MELANGES\_EN\_SIGMA , LOI\_CRITERE , LOI\_CONTRAINTE↔  
\_PLANES ,

LOI\_CONTRAINTES\_PLANES\_DOUBLE , LOI\_DEFORMATIONS\_PLANES , HYSTERESIS\_1D ,  
HYSTERESIS\_3D ,  
HYSTERESIS\_BULK , LOI\_ISO\_THERMO , MOONEY\_RIVLIN\_1D , MOONEY\_RIVLIN\_3D ,  
POLY\_HYPER3D , HART\_SMITH3D , MAHEO\_HYPER , HYPER\_EXTERNE\_W ,  
LOI\_DE\_TAIT , LOI\_VIA\_UMAT , LOI\_VIA\_UMAT\_CP , LOI\_COULOMB ,  
LOI\_RIEN1D , LOI\_RIEN2D\_D , LOI\_RIEN2D\_C , LOI\_RIEN3D ,  
RIEN\_COMP }

*énuméré permettant d'identifier chaque loi de comportement*

— enum `Enum_comp_3D_CP_DP_1D` {  
COMP\_1D = 1 , COMP\_CONTRAINTES\_PLANES , COMP\_DEFORMATIONS\_PLANES , COMP\_3D ,  
RIEN\_COMP\_3D\_CP\_DP\_1D }

*énuméré permettant de savoir si une loi est : 1D, 2D en déformations planes ou contraintes planes, 3D générale*

— enum `Enum_contrainte_mathematique` {  
MULTIPLICATEUR\_DE\_LAGRANGE = 1 , PENALISATION , NEWTON\_LOCAL , PERTURBATION ,  
RIEN\_CONTRAINTE\_MATHEMATIQUE }

*Définition de l'énuméré concernant les types de contraintes mathématiques.*

— enum `Enum_crista` { HOFFMAN = 1 , HOFFMAN2 , CRISTA\_PAS\_DEFINI }

*Enumération des différents calcul de cristallinité*

— enum `Enum_Critere_Loi` {  
AUCUN\_CRITERE = 0 , PLISSEMENT\_MEMBRANE , PLISSEMENT\_BIEL , RUPTURE\_SIGMA\_PRINC ,  
RUPTURE\_EPS\_PRINC }

*Enumeration des différentes méthodes permettant d'utiliser des critères sur les lois de comportement.*

— enum `Enum_ddl` {  
X1 = 1 , X2 , X3 , EPAIS ,  
TEMP , UX , UY , UZ ,  
V1 , V2 , V3 , PR ,  
GAMMA1 , GAMMA2 , GAMMA3 , SIG11 ,  
SIG22 , SIG33 , SIG12 , SIG23 ,  
SIG13 , ERREUR , EPS11 , EPS22 ,  
EPS33 , EPS12 , EPS23 , EPS13 ,  
DEPS11 , DEPS22 , DEPS33 , DEPS12 ,  
DEPS23 , DEPS13 , PROP\_CRISTA , DELTA\_TEMP ,  
FLUXD1 , FLUXD2 , FLUXD3 , R\_TEMP ,  
GRADT1 , GRADT2 , GRADT3 , DGRADT1 ,  
DGRADT2 , DGRADT3 , R\_X1 , R\_X2 ,  
R\_X3 , R\_EPAIS , R\_V1 , R\_V2 ,  
R\_V3 , R\_GAMMA1 , R\_GAMMA2 , R\_GAMMA3 ,  
NU\_DDL }

— enum `Enum_dure` { TEMPS\_0 = 0 , TEMPS\_t , TEMPS\_tdt }

*Défini une énumération en temps.*

— enum `Enum_geom` {  
RIEN\_GEOM = 1 , TRIANGLE , QUADRANGLE , TETRAEDRE ,  
PENTAEDRE , HEXAEDRE , SEGMENT , TRIA\_AXI ,  
QUAD\_AXI , SEG\_AXI , POINT , POINT\_CP ,  
POUT , PS1 }

*Définition de l'énuméré concernant les types de modélisations de géométrie .*

— enum `Enum_GrandeurGlobale` {  
ENERGIE\_CINETIQUE = 0 , ENERGIE\_INTERNE , ENERGIE\_EXTERNE , ENERGIE\_BILAN ,  
QUANTITE\_MOUVEMENT , PUISSANCE\_ACCELERATION , PUISSANCE\_INTERNE , PUISSANCE\_↔  
EXTERNE ,  
PUISSANCE\_BILAN , ENERGIE\_ELASTIQUE , ENERGIE\_PLASTIQUE , ENERGIE\_VISQUEUSE ,  
ENERGIE\_HOURGLASS\_ , ENERGIE\_PENALISATION , ENERGIE\_FROT\_ELAST , ENERGIE\_FROT\_↔  
PLAST ,  
ENERGIE\_FROT\_VISQ , ENERGIE\_VISCO\_NUMERIQUE , ENERGIE\_BULK\_VISCOSITY , PUISSANCE\_↔  
\_BULK\_VISCOSITY ,  
VOLUME\_TOTAL\_MATIERE , ENERGIE\_STABILISATION\_MEMB\_BIEL , VOL\_TOTAL2D\_AVEC\_↔  
PLAN\_YZ , VOL\_TOTAL2D\_AVEC\_PLAN\_XZ ,  
VOL\_TOTAL2D\_AVEC\_PLAN\_XY , NORME\_CONVERGENCE , COMPTEUR\_ITERATION\_ALGO\_↔  
GLOBAL , MAXPUSSEXT ,

MAXPUISSINT , MAXREACTION , MAXRESIDUGLOBAL , MAXdeltaX ,  
 MAXvarDeltaX , MAXvarDdl , COMPTEUR\_INCREMENT\_CHARGE\_ALGO\_GLOBAL , AMOR\_CINET ↔  
 \_VISQUEUX ,  
 TEMPS\_COURANT , ALGO\_GLOBAL\_ACTUEL }

*Définition de l'énuméré concernant les grandeurs globales.*

— enum `Enum_interpol` {

RIEN\_INTERPOL = 1 , CONSTANT , LINEAIRE , QUADRATIQUE ,  
 QUADRACOMPL , CUBIQUE , CUBIQUE\_INCOMPL , LINQUAD ,  
 HERMITE , SFE1 , SFE2 , SFE3 ,  
 SFE3C , QSFE3 , QSFE1 , SFE1\_3D ,  
 SFE2\_3D , SFE3\_3D , SFE3C\_3D , SEG1 ,  
 BIE1 , BIE2 }

*Définition de l'énuméré concernant les différents types d'interpolation.*

— enum `Enum_IO_XML` {

XML\_TYPE\_GLOBAUX = 1 , XML\_IO\_POINT\_INFO , XML\_IO\_POINT\_BI , XML\_IO\_ELEMENT\_FINI ,  
 XML\_ACTION\_INTERACTIVE , XML\_STRUCTURE\_DONNEE }

*Enumération des différentes entree/sortie XML.*

— enum `Enum_liaison_noeud` { PAS\_LIER = 0 , LIER\_COMPLET }

*Enuméré pour définir les différents types de liaison entre noeuds.*

— enum `Enum_mat` { ACIER = 1 , BETON , COMPOSITE }

*Définition de l'énuméré concernant les types de matériau.*

— enum `Enum_matrice` {

RIEN\_MATRICE = 1 , CARREE , CARREE\_SYMETRIQUE , RECTANGLE ,  
 BANDE\_SYMETRIQUE , BANDE\_NON\_SYMETRIQUE , CARREE\_LAPACK , CARREE\_SYMETRIQUE ↔  
 \_LAPACK ,  
 RECTANGLE\_LAPACK , BANDE\_SYMETRIQUE\_LAPACK , BANDE\_NON\_SYMETRIQUE\_LAPACK ,  
 TRIDIAGONALE\_GENE\_LAPACK ,  
 TRIDIAGONALE\_DEF\_POSITIVE\_LAPACK , CREUSE\_NON\_COMPRESSEE , CREUSE\_COMPRESSEE ↔  
 \_COLONNE , CREUSE\_COMPRESSEE\_LIGNE ,  
 DIAGONALE }

*Enumeration des différents type de matrice possible.*

— enum `Enum_PiPoCo` { NON\_PoutrePlaqueCoque = 0 , POUTRE , PLAQUE , COQUE }

*Définir un type énuméré pour la différenciation entre des éléments classiques et le cas poutre plaque ou coque.*

— enum `Enum_proj_aniso` { AUCUNE\_PROJ\_ANISO = 0 , PROJ\_ORTHO }

*Enumeration des différentes méthodes concernant les techniques de projection anisotrope.*

— enum `Enum_StabHourglass` { STABHOURGLASS\_NON\_DEFINIE = 0 , STABHOURGLASS\_PAR ↔  
 COMPORTEMENT , STABHOURGLASS\_PAR\_COMPORTEMENT\_REDUIT }

*Enumeration des différentes méthodes permettant de stabiliser les modes d'hourglass.*

— enum `Enum_StabMembraneBiel` {

STABMEMBRANE\_BIEL\_NON\_DEFINIE = 0 , STABMEMBRANE\_BIEL\_PREMIER\_ITER , STABMEMBRANE ↔  
 \_BIEL\_PREMIER\_ITER\_INCR , STABMEMBRANE\_BIEL\_Gerschgorin\_PREMIER\_ITER ,  
 STABMEMBRANE\_BIEL\_Gerschgorin\_PREMIER\_ITER\_INCR , STABMEMBRANE\_BIEL\_PREMIER ↔  
 \_ITER\_NORMALE\_AU\_NOEUD , STABMEMBRANE\_BIEL\_PREMIER\_ITER\_INCR\_NORMALE\_AU ↔  
 NOEUD , STABMEMBRANE\_BIEL\_Gerschgorin\_PREMIER\_ITER\_NORMALE\_AU\_NOEUD ,  
 STABMEMBRANE\_BIEL\_Gerschgorin\_PREMIER\_ITER\_INCR\_NORMALE\_AU\_NOEUD , STAB\_M ↔  
 B\_ITER\_1\_VIA\_F\_EXT , STAB\_M\_B\_ITER\_INCR\_1\_VIA\_F\_EXT , STAB\_M\_B\_ITER\_1\_VIA\_F\_EXT ↔  
 NORMALE\_AU\_NOEUD ,  
 STAB\_M\_B\_ITER\_INCR\_1\_VIA\_F\_EXT\_NORMALE\_AU\_NOEUD }

*Enumeration des différentes méthodes permettant de stabiliser les membranes transversalement.*

— enum `Enum_Suite` { SUITE\_EQUIDISTANTE = 1 , SUITE\_ARITHMETIQUE , SUITE\_GEOMETRIQUE ,  
 SUITE\_NON\_DEFINIE }

*Enumeration des différentes Suites existantes.*

— enum `Enum_type_deformation` {

DEFORMATION\_STANDART = 1 , DEFORMATION\_POUTRE\_PLAQUE\_STANDART , DEFORMATION ↔  
 LOGARITHMIQUE , DEF\_CUMUL\_CORROTATIONNEL ,  
 DEF\_CUMUL\_ROTATION\_PROPRE , DEFORMATION\_CUMU\_LOGARITHMIQUE }

*Défini une énumération des différents types de déformations.*

— enum `Enum_type_geom` {

POINT\_G = 1 , LIGNE , SURFACE , VOLUME ,  
 RIEN\_TYPE\_GEOM }

- *Définition de l'énuméré concernant les types de géométrie.*  
enum `Enum_type_pt_integ` { `PTI_GAUSS = 1` , `PTI_GAUSS_LOBATTO` }
- *Définition de l'énuméré concernant les types de point d'intégration.*  
enum `Enum_type_resolution_matri` {  
`RIEN_TYPE_RESOLUTION_MATRI = 1` , `CHOLESKY` , `SYMETRISATION_PUIS_CHOLESKY` , `GAUSS` ,  
`GAUSS_EXPERT` , `BI_CONJUG` , `BI_CONJUG_STAB` , `CONJUG_GRAD` ,  
`CONJUG_GRAD_SQUARE` , `CHEBYSHEV` , `GENE_MINI_RESIDUAL` , `ITERATION_RICHARSON` ,  
`QUASI_MINI_RESIDUAL` , `DIRECT_DIAGONAL` , `CRAMER` , `LU_EQUILIBRE` }  
*Enumeration des differents type de résolution de systèmes matricielle possible.*
- enum `Enum_preconditionnement` { `RIEN_PRECONDITIONNEMENT = 1` , `DIAGONAL` , `ICP` , `ILU` }  
*Enumeration des differents type de preconditionnement éventuel.*
- enum `Enum_type_stocke_deformation` { `SAVEDEFRESUL_GENERAL = 0` , `SAVEDEFRESUL_SFE1` }  
*Définir une énumération pour les type de stockage pour les données de déformations à chaque pt de gauss.*
- enum `EnumTypeQuelconque` {  
`RIEN_TYPEQUELCONQUE = 1` , `SIGMA_BARRE_BH_T` , `CONTRAINTE_INDIVIDUELLE_A_CHAQUE_LOI_A_T` ,  
`CONTRAINTE_INDIVIDUELLE_A_CHAQUE_LOI_A_T_SANS_PROPORTION` ,  
`CONTRAINTE_COURANTE` , `DEFORMATION_COURANTE` , `VITESSE_DEFORMATION_COURANTE` ,  
`ALMANSI` ,  
`GREEN_LAGRANGE` , `LOGARITHMIQUE` , `DELTA_DEF` , `ALMANSI_TOTAL` ,  
`GREEN_LAGRANGE_TOTAL` , `LOGARITHMIQUE_TOTALE` , `DEF_PRINCIPALES` , `SIGMA_PRINCIPALES` ,  
`VIT_PRINCIPALES` , `DEF_DUALE_MISES` , `DEF_DUALE_MISES_MAXI` , `CONTRAINTE_MISES` ,  
`CONTRAINTE_MISES_T` , `CONTRAINTE_TRESCA` , `CONTRAINTE_TRESCA_T` , `ERREUR_Q` ,  
`DEF_PLASTIQUE_CUMULEE` , `ERREUR_SIG_RELATIVE` , `TEMPERATURE_LOI_THERMO_PHYSIQUE` ,  
`PRESSON_LOI_THERMO_PHYSIQUE` ,  
`TEMPERATURE_TRANSITION` , `VOLUME_SPECIFIQUE` , `FLUXD` , `GRADT` ,  
`DGRADT` , `DELTA_GRADT` , `COEFF_DILATATION_LINEAIRE` , `CONDUCTIVITE` ,  
`CAPACITE_CALORIFIQUE` , `MODULE_COMPRESSIBILITE` , `MODULE_CISAILLEMENT` , `COEFF_COMPRESSIBILITE` ,  
`MODULE_COMPRESSIBILITE_TOTAL` , `MODULE_CISAILLEMENT_TOTAL` , `E_YOUNG` , `NU_YOUNG` ,  
`MU_VISCO` , `MU_VISCO_SPHERIQUE` , `MODULE_TANGENT_1D` , `COMPRESSIBILITE_TANGENTE` ,  
`NB_INVERSION` , `HYPER_CENTRE_HYSTERESIS` , `SIGMA_REF` , `Q_SIG_HYST_OI_A_R` ,  
`Q_SIG_HYST_R_A_T` , `Q_DELTA_SIG_HYST` , `COS_ALPHA_HYSTERESIS` , `COS3PHI_SIG_HYSTERESIS` ,  
`COS3PHI_DELTA_SIG_HYSTERESIS` , `FCT_AIDE` , `NB_ITER_TOTAL_RESIDU` , `NB_INCRE_TOTAL_RESIDU` ,  
`NB_APPEL_FCT` , `NB_STEP` , `ERREUR_RK` , `PRESSON_HYST_REF` ,  
`PRESSON_HYST` , `PRESSON_HYST_REF_M1` , `PRESSON_HYST_T` , `UN_DDL_ENUM_ETENDUE` ,  
`ENERGIE_ELASTIQUE_INDIVIDUELLE_A_CHAQUE_LOI_A_T` , `ENERGIE_PLASTIQUE_INDIVIDUELLE_A_CHAQUE_LOI_A_T` ,  
`ENERGIE_VISQUEUSE_INDIVIDUELLE_A_CHAQUE_LOI_A_T` , `PROPORTION_LOI_MELANGE` ,  
`FONC_PONDERATION` , `POSITION_GEOMETRIQUE` , `POSITION_GEOMETRIQUE_t` , `POSITION_GEOMETRIQUE_t0` ,  
`CRISTALINITE` , `VOLUME_ELEMENT` , `VOLUME_PTI` , `EPAISSEUR_MOY_INITIALE` ,  
`EPAISSEUR_MOY_FINALE` , `SECTION_MOY_INITIALE` , `SECTION_MOY_FINALE` , `EPAISSEUR_INITIALE` ,  
`EPAISSEUR_FINALE` , `SECTION_INITIALE` , `SECTION_FINALE` , `VOL_ELEM_AVEC_PLAN_REF` ,  
`INTEG_SUR_VOLUME` , `INTEG_SUR_VOLUME_ET_TEMPS` , `STATISTIQUE` , `STATISTIQUE_ET_TEMPS` ,  
`ENERGIE_HOURLASS` , `PUISSANCE_BULK` , `ENERGIE_BULK` , `ENERGIE_STABMEMB_BIEL` ,  
`FORCE_STABMEMB_BIEL` , `TENSEUR_COURBURE` , `COURBURES_PRINCIPALES` , `DIRECTIONS_PRINC_COURBURE` ,  
`DIRECTIONS_PRINC_SIGMA` , `DIRECTIONS_PRINC_DEF` , `DIRECTIONS_PRINC_D` , `REPERE_LOCAL_ORTHO` ,  
`REPERE_LOCAL_H` , `REPERE_LOCAL_B` , `REPERE_D_ANISOTROPIE` , `EPS_TRANSPORTEE_ANISO` ,  
`SIGMA_DANS_ANISO` , `DELTA_EPS_TRANSPORTEE_ANISO` , `DELTA_SIGMA_DANS_ANISO` , `PARAMETRE_LOCAL_ORTHO` ,  
`SPHERIQUE_EPS` , `Q_EPS` , `COS3PHI_EPS` , `SPHERIQUE_SIG` ,

```

Q_SIG , COS3PHI_SIG , SPHERIQUE_DEPS , V_vol ,
Q_DEPS , COS3PHI_DEPS , POTENTIEL , FCT_POTENTIEL_ND ,
INVAR_B1 , INVAR_B2 , INVAR_B3 , INVAR_J1 ,
INVAR_J2 , INVAR_J3 , DEF_EQUIVALENTE , DEF_EPAISSEUR ,
D_EPAISSEUR , DEF_LARGEUR , D_LARGEUR , DEF_MECHANIQUE ,
DEF ASSO_LOI , DEF_P_DANS_V_A , SIG_EPAISSEUR , SIG_LARGEUR ,
FORCE_GENE_EXT , FORCE_GENE_INT , FORCE_GENE_TOT , RESIDU_GLOBAL ,
FORCE_GENE_EXT_t , FORCE_GENE_INT_t , VECT_PRESSION , PRESSION_SCALAIRE ,
VECT_FORCE_VOLUM , VECT_DIR_FIXE , VECT_SURF_SUIV , VECT_HYDRODYNA_Fn ,
VECT_HYDRODYNA_Ft , VECT_HYDRODYNA_T , VECT_LINE , VECT_LINE_SUIV ,
VECT_REAC , VECT_REAC_N , NN_11 , NN_22 ,
NN_33 , NN_12 , NN_13 , NN_23 ,
MM_11 , MM_22 , MM_33 , MM_12 ,
MM_13 , MM_23 , DIRECTION_PLI , DIRECTION_PLI_NORMEE ,
INDIC_CAL_PLIS , NN_SURF , NN_SURF_t , NN_SURF_t0 ,
NOEUD_PROJECTILE_EN_CONTACT , NOEUD_FACETTE_EN_CONTACT , GLISSEMENT_CONTACT ,
PENETRATION_CONTACT ,
GLISSEMENT_CONTACT_T , PENETRATION_CONTACT_T , FORCE_CONTACT , FORCE_CONTACT↵
_T ,
CONTACT_NB_PENET , CONTACT_NB_DECOL , CONTACT_CAS_SOLIDE , CONTACT_ENERG↵
PENAL ,
CONTACT_COLLANT , NUM_ZONE_CONTACT , CONTACT_ENERG_GLISSE_ELAS , CONTACT↵
ENERG_GLISSE_PLAS ,
CONTACT_ENERG_GLISSE_VISQ , CONTACT_PENALISATION_N , CONTACT_PENALISATION_T ,
NORMALE_CONTACT ,
TEMPS_CPU_USER , TEMPS_CPU_LOI_COMP , TEMPS_CPU_METRIQUE , GENERIQUE_UNE↵
GRANDEUR_GLOBALE ,
GENERIQUE_UNE_CONSTANTE_GLOB_INT_UTILISATEUR , GENERIQUE_UNE_CONSTANTE↵
GLOB_DOUBLE_UTILISATEUR , GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_0
, GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_T ,
GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_TDT , DEPLACEMENT , VITESSE ,
DELTA_XI ,
XI_ITER_0 , MASSE_RELAX_DYN , COMP_TORSEUR_REACTION , NUM_NOEUD ,
NUM_MAIL_NOEUD , NUM_ELEMENT , NUM_MAIL_ELEM , NUM_PTI ,
NUM_FACE , NUM_ARETE }

```

*Définition de l'enuméré pour repérer un type quelconque.*

```

— enum Enum_variable_métrique {
iM0 =1 , iMt , idMt , iMtdt ,
idMtdt , iV0 , iVt , idVt ,
iVtdt , idVtdt , igiB_0 , igiB_t ,
igiB_tdt , igiH_0 , igiH_t , igiH_tdt ,
igijBB_0 , igijBB_t , igijBB_tdt , igijHH_0 ,
igijHH_t , igijHH_tdt , id_giB_t , id_giB_tdt ,
id_giH_t , id_giH_tdt , id_gijBB_t , id_gijBB_tdt ,
id_gijHH_t , id_gijHH_tdt , id_jacobien_t , id_jacobien_tdt ,
id2_gijBB_tdt , igradVmoyBB_t , igradVmoyBB_tdt , igradVBB_t ,
igradVBB_tdt , id_gradVmoyBB_t , id_gradVmoyBB_tdt , id_gradVBB_t ,
id_gradVBB_tdt }

```

*Définition de l'enuméré concernant les grandeurs gérées par les classes métriques générales.*

```

— enum Enum_variable_metsfe {
iP0 =1 , iPt , idPt , iPtdt ,
idPtdt , iaiB_0 , iaiB_t , iaiB_tdt ,
iaiH_0 , iaiH_t , iaiH_tdt , iaijBB_0 ,
iaijBB_t , iaijBB_tdt , iaijHH_0 , iaijHH_t ,
iaijHH_tdt , id_aiB_t , id_aiB_tdt , id_aiH_t ,
id_aiH_tdt , id_aijBB_t , id_aijBB_tdt , id_aijHH_t ,
id_aijHH_tdt , id_ajacobien_t , id_ajacobien_tdt }

```

*Définition de l'enuméré concernant les grandeurs spécifiques gérées par les classes métriques plaques, coques et poutres.*

- enum `EnumCourbe1D` {  
**COURBEPOLYLINEAIRE\_1\_D = 1**, **COURBE\_EXPOAFF**, **COURBE\_UN\_MOINS\_COS**, **CPL1D**,  
**COURBEPOLYNOMIALE**, **F1 Rond\_F2**, **F1\_PLUS\_F2**, **F\_CYCLIQUE**,  
**F\_CYCLE\_ADD**, **F\_UNION\_1D**, **COURBE\_TRIPODECOS3PHI**, **COURBE\_SIXPODECOS3PHI**,  
**COURBE\_POLY\_LAGRANGE**, **COURBE\_EXPO\_N**, **COURBE\_EXPO2\_N**, **COURBE\_RELAX\_EXPO**,  
**COURBE\_COS**, **COURBE\_SIN**, **COURBE\_TANH**, **COURBEPOLYHERMITE\_1\_D**,  
**COURBE\_LN\_COSH**, **COURBE\_EXPRESSION\_LITTERALE\_1D**, **COURBE\_EXPRESSION\_↔**  
**LITTERALE\_AVEC\_DERIVEE\_1D**, **AUCUNE\_COURBE1D** }  
*Enumeration des différentes courbes 1D existantes.*
- enum `EnumElemTypeProblem` {  
**MECA\_SOLIDE\_DEFORMABLE = 1**, **MECA\_SOLIDE\_INDEFORMABLE**, **MECA\_FLUIDE**, **THERMIQUE**  
**ELECTROMAGNETIQUE**, **RIEN\_PROBLEM** }  
*Définition de l'enuméré concernant les types de problème.*
- enum `EnumFonction_nD` {  
**FONCTION\_EXPRESSION\_LITTERALE\_nD = 1**, **FONCTION\_COURBE1D**, **FONC\_SCAL\_COMBINEES↔**  
**\_ND**, **FONCTION\_EXTERNE\_ND**,  
**AUCUNE\_FONCTION\_nD** }  
*Enumeration des différentes fonctions nD existantes.*
- enum `EnumLangue` { **FRANCAIS = 1**, **ENGLISH** }  
*Énumération des différents langages.*
- enum `EnumTypeCalcul` {  
**DYNA\_IMP = 1**, **DYNA\_EXP**, **DYNA\_EXP\_TCHAMWA**, **DYNA\_EXP\_CHUNG\_LEE**,  
**DYNA\_EXP\_ZHAI**, **DYNA\_RUNGE\_KUTTA**, **NON\_DYNA**, **NON\_DYNA\_CONT**,  
**FLAMB\_LINEAIRE**, **INFORMATIONS**, **UTILITAIRES**, **DEF\_SCHEMA\_XML**,  
**RIEN\_TYPECALCUL**, **UMAT\_ABAQUS**, **DYNA\_EXP\_BONELLI**, **RELAX\_DYNA**,  
**STAT\_DYNA\_EXP**, **COMBINER** }  
*identificateur principal du type de Calcul*
- enum `EnumSousTypeCalcul` {  
**aucun\_soustypedecalcul = 1**, **avec\_remonte**, **avec\_remonte\_erreur**, **avec\_remonte\_erreur\_relocation**  
**avec\_remonte\_erreur\_raffinement\_relocation**, **remonte**, **remonte\_erreur**, **remonte\_erreur\_relocation**  
**remonte\_erreur\_raffinement\_relocation**, **frontieres**, **visualisation**, **LinVersQuad**,  
**QuadIncVersQuadComp**, **relocPtMilieuQuad**, **saveCommandesVisu**, **lectureCommandesVisu**,  
**commandeInteractive**, **saveMaillagesEnCours**, **extrusion2D3D**, **creation\_reference**,  
**prevision\_visu\_sigma**, **prevision\_visu\_epsilon**, **prevision\_visu\_erreur**, **modif\_orientation\_element**,  
**creationMaillageSFE**, **suppression\_noeud\_non\_references**, **renumerotation\_des\_noeuds**, **fusion↔**  
**\_de\_noeuds**,  
**fusion\_elements**, **fusion\_maillages**, **creer\_sous\_maillage**, **calcul\_geometrique** }  
*identificateur secondaire optionnel du type de Calcul renseigne par exemple sur le fait de calculer l'erreur, une relocation, un raffinement, etc ..*
- enum `Enum_type_gradient` { **GRADVITESSE\_V\_VTDT = 1**, **GRADVITESSE\_VCONST**, **GRADVITESSE\_↔**  
**VT\_VTDT** }  
*def du gradient de vitesse pour différentes conditions*
- enum `EnumTypeGrandeur` {  
**RIEN\_TYPEGRANDEUR = 0**, **SCALAIRE**, **VECTEUR**, **TENSEUR**,  
**TENSEUR\_NON\_SYM**, **SCALAIRE\_ENTIER**, **SCALAIRE\_DOUBLE**, **TENSEURBB**,  
**TENSEURHH**, **TENSEURBH**, **TENSEURHB**, **TENSEUR\_NON\_SYM\_BB**,  
**TENSEUR\_NON\_SYM\_HH**, **COORDONNEE**, **COORDONNEEB**, **COORDONNEEH**,  
**BASE\_H**, **BASE\_B**, **CHAINE\_CAR**, **GRANDEUR\_QUELCONQUE** }  
*un énuméré pour les types de base*
- enum `EnumType2Niveau` {  
**TYPE\_SIMPLE = 0**, **TABLEAU\_T**, **TABLEAU2\_T**, **LISTE\_T**,  
**LISTE\_IO\_T**, **MAP\_T**, **VECTOR\_T** }  
*un énuméré pour les types de structure*
- enum `EnumTypePilotage` { **PILOTAGE\_BASIQUE = 1**, **PILOT\_GRADIENT**, **AUCUN\_PILOTAGE** }  
*Définition de l'enuméré concernant les types de pilotage.*

- enum `EnumTypeViteRotat` { `R_CORROTATIONNEL = 1` , `R_REF_ROT_PROPRE` , `R_ROT_↵`  
`LOGARITHMIQUE` }  
*Définition de l'énuméré concernant les type de vitesse de rotation.*
- enum `Enum_TypeVitDef` { `D_MOY_EPS_SUR_DT = 1` , `D_AVEC_V_A_T_PLUS_DT` , `D_AVEC_DX_SUR_↵`  
`_DT_A_T_PLUS_DT_SUR2` , `D_AVEC_V_MOY_A_T_PLUS_DT_SUR2` }  
*Définition de l'énuméré concernant les types de vitesse de déformation.*
- enum `EnuTypeCL` { `TANGENTE_CL` , `RIEN_TYPE_CL` }  
*Type énuméré pour définir les différentes sous-classes de conditions limites.*
- enum `EnuTypeQuelParticulier` {  
`RIEN_TYPE_QUELCONQUE_PARTICULIER = 1` , `PARTICULIER_TENSEURHH` , `PARTICULIER_↵`  
`TENSEURBB` , `PARTICULIER_COORDONNEE` ,  
`PARTICULIER_TABLEAU_COORDONNEE` , `PARTICULIER_VECTEUR` , `PARTICULIER_TABLEAU_↵`  
`VECTEUR` , `PARTICULIER_BASE_H` ,  
`PARTICULIER_BASE_B` , `PARTICULIER_TABLEAU_BASE_H` , `PARTICULIER_TABLEAU_BASE_B` ,  
`PARTICULIER_TABLEAU_TENSEURHH` ,  
`PARTICULIER_TABLEAU2_TENSEURHH` , `PARTICULIER_TENSEURBH` , `PARTICULIER_SCALAIRE_↵`  
`_DOUBLE` , `PARTICULIER_TABLEAU_SCALAIRE_DOUBLE` ,  
`PARTICULIER_SCALAIRE_ENTIER` , `PARTICULIER_TABLEAU_SCALAIRE_ENTIER` , `PARTICULIER_↵`  
`_TABLEAU_QUELCONQUE` , `PARTICULIER_TABLEAU_TENSEURBH` ,  
`PARTICULIER_TABLEAU_TENSEURBB` , `PARTICULIER_TENSEURHB` , `PARTICULIER_TABLEAU_↵`  
`TENSEURHB` , `PARTICULIER_DDL_ETENDU` ,  
`PARTICULIER_TABLEAU_DDL_ETENDU` , `PARTICULIER_VECTEUR_NOMMER` , `PARTICULIER_↵`  
`TABLEAU_VECTEUR_NOMMER` , `PARTICULIER_SCALAIRE_DOUBLE_NOMMER_INDICER` }  
*Définition de l'énuméré concernant les types quelconques particuliers.*

## Variables

- const int `nombre_maxi_de_type_de_Enum_crista = 2`
- static map< string, Enum\_ddl, std::less< string > > `ClassPourEnum_ddl::map_Enum_ddl`  
*def de la map qui fait la liaison entre les string et les énumérés*
- static `ClassPourEnum_ddl ClassPourEnum_ddl::remplir_map`  
*def de la grandeur statique qui permet de remplir la map*

### 5.18.1 Description détaillée

Def de grandeurs énumérées: permet une meilleure lisibilité du code et éventuellement un gain de place.

donne directement le premier et le second indice en fonction de l'enum et du nbcomposante pour les tenseurs sigma ou pour le tenseur epsilon ou encore pour le tenseur Depsilon

classe utilitaire pour les enum\_ddl

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.18.2 Documentation des variables

### 5.18.2.1 map\_Enum\_ddl

```
map< string, Enum_ddl, std::less< string > > ClassPourEnum_ddl::map_Enum_ddl [static]
```

def de la map qui fait la liaison entre les string et les énumérés

tout d'abord on définit la map qui permet de relier les chaines et les types énumérés 1) def de la map et des tableaux

### 5.18.2.2 remplir\_map

```
ClassPourEnum_ddl ClassPourEnum_ddl::remplir_map [static]
```

def de la grandeur statique qui permet de remplir la map

2) def de la grandeur statique qui permet de remplir la map

## 5.19 Groupe\_conteneurs\_bloc

groupe concernant la définition de bloc de stockage divers associant des opérations d'I/O

### Classes

- class [BlocScal](#)
- class [BlocScal\\_ou\\_fctnD](#)
- class [BlocScalType](#)
- class [BlocGen](#)
- class [BlocGen\\_3\\_1](#)
- class [BlocGen\\_3\\_2](#)
- class [BlocGen\\_3\\_0](#)
- class [BlocGen\\_4\\_0](#)
- class [BlocGen\\_5\\_0](#)
- class [BlocGen\\_6\\_0](#)
- class [BlocVec](#)
- class [BlocVecType](#)
- class [BlocVecMultType](#)
- class [BlocGeneEtVecMultType](#)

### 5.19.1 Description détaillée

groupe concernant la définition de bloc de stockage divers associant des opérations d'I/O

BUT: groupe concernant la définition de bloc de stockage divers associant des opérations d'I/O

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97



## 5.20 Goupe\_relatif\_aux\_entrees\_sorties

groupe relatif aux méthodes de lectures et d'écritures sur fichiers, écrans

### Classes

- class [LectBloc](#)
- class [LectBlocmot](#)
- class [UtilLecture](#)
- class [UtilXML](#)

### 5.20.1 Description détaillée

groupe relatif aux méthodes de lectures et d'écritures sur fichiers, écrans

BUT: groupe relatif aux méthodes de lectures et d'écritures sur fichiers, écrans

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.21 Les\_classes\_Ddl\_en\_tout\_genre

Def de classes relatives aux ddl en tout genre.

### Classes

- class [BlocDdlLim](#) < [Bloc\\_particulier](#) >
- class [Ddl](#)

*BUT: La classe [Ddl](#) permet de declarer des degres de liberte. soit c'est une variable ou soit c'est une données qui peuvent être soit active soit hors-service ensuite soit il est fixe ou soit il est bloqué 1) libre, fixe, ou 2) hors service libre ou hors service fixe ==> une variable 3) lisible libre ou fixe ou 4) hors service libre ou fixe ==> une donnée 1) c'est pour les ddl qui sont des variables que le calcul va déterminer, 2) sont les variables qui sont hors service: on ne les considère pas 3) sont des variables que l'on peut utiliser mais que le calcul ne cherche pas à déterminer: elle sont en lecture seule en quelque sorte 4) se sont des données qui ne sont pas dispon pour la consultation (par exemple hors temps)*

- class [Ddl\\_etendu](#)

*BUT: class de stockage d'un ddl étendue, associé au type [Ddl\\_enum\\_etendu](#). En fait il s'agit du pendant des types [Ddl](#) associé au type [enum\\_ddl](#) Noter que la grandeur stockée est un scalaire!!*

- class [DdlElement](#)

*BUT: Definition, gestion et stockage des ddl de l'element.*

- class [DdlLim](#)

*BUT: La classe [DdlLim](#) permet de declarer un ensemble de ddl coiffe par une reference utilise pour les conditions limites.*

- class [DdlNoeudElement](#)

*BUT: Def des ddl lie a un noeud d'un element ( different des ddl lies aux noeuds globaux)*

### 5.21.1 Description détaillée

Def de classes relatives aux ddl en tout genre.

BUT: def de classes relatives aux ddl en tout genre [Ddl](#), [Ddl\\_etendu](#), [DdlElement](#), [DdlLim](#) etc.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.22 Les\_classes\_relatives\_aux\_conditions\_limites

Gestion des differentes conditions limites.

### Classes

- class [I\\_O\\_Condilinaire](#)
- class [LesCondLim](#)

### 5.22.1 Description détaillée

Gestion des differentes conditions limites.

BUT: Gestion des differentes conditions limites

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.23 Les\_Maillages

Définition du groupe de maillage.

## Classes

- class [LesMaillages](#)  
*LesMaillages: l'ensemble des maillages.*
- class [Maillage](#)  
*Maillage: un maillage particulier.*
- class [Posi\\_ddl\\_noeud](#)  
*Posi\_ddl\_noeud: un conteneur qui associe numéro de noeud, de maillage, et enu ddl.*
- class [Noeud](#)  
*Noeud: un noeud.*

### 5.23.1 Description détaillée

Définition du groupe de maillage.

BUT: définir le groupe de maillage \*

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.24 Les\_classes\_exportation\_en\_variables

Définition de classes relatives a l'exportation en variables.

## Classes

- class [VariablesExporter](#)

### 5.24.1 Description détaillée

Définition de classes relatives a l'exportation en variables.

BUT: def de classes relatives a l'exportation en variables globales de grandeurs définies dans les maillages. Cela concerne les constantes utilisateur et les variables calculées par Herezh.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.25 Les\_parametres\_generaux

groupe relatif à la gestion des paramètres généraux

### Classes

- class [Banniere](#)
- class [ConstMath](#)
- class [ConstPhysico](#)
- class [ParaAlgoControle](#)
- class [ParaGlob](#)

### 5.25.1 Description détaillée

groupe relatif à la gestion des paramètres généraux

BUT: groupe relatif à la gestion des paramètres généraux

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

## 5.26 Les\_classes\_Reference

Définir les différentes références de noeuds, d'élément, d'arêtes, de faces, de pti etc..

### Classes

- class [LesReferences](#)  
*Gestion des listes de références.*
- class [Reference](#)  
*class [Reference](#): classe général virtuel des références. Les classes dérivées permettent de définir précisément les différentes références : de noeuds, d'élément, d'arêtes, de faces etc.. Une instance de la classe s'identifie à partir d'un nom*
- class [ReferenceAF](#)  
*Def, stockage et manipulation des références pour les faces et les arêtes.*
- class [ReferenceNE](#)  
*Def, stockage et manipulation des références pour les noeuds et les éléments.*
- class [ReferencePtiAF](#)  
*Def, stockage et manipulation des références pour les pti de faces et d'arêtes.*

## Fonctions

- `Tableau< Reference > Lect_reference` (char \*nom\_fichier)  
*Lecture des references definies dans le fichier au format ".lis" de nom : nom\_fichier.*
- `ostream & operator<<` (ostream &sort, const map< string, Reference \*, std::less< string > > &)  
*Gestion des listes de references.*

## Variables

- static `MotCle Reference::motCle`  
----- variables statiques -----

### 5.26.1 Description détaillée

Definir les différentes références de noeuds, d'élément, d'aretes, de faces, de pti etc..

BUT: Definir les différentes références de noeuds, d'élément, d'aretes, de faces, de pti etc..

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

06/02/00

### 5.26.2 Documentation des fonctions

#### 5.26.2.1 operator<<()

```
ostream & operator<< (
    ostream & sort,
    const map< string, Reference *, std::less< string > > & )
```

Gestion des listes de references.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.27 Les\_classes\_Matrices

les différentes classes de matrices.

### Classes

- class [ErrResolve\\_system\\_lineaire](#)
- class [Mat\\_abstraite](#)
- class [Mat\\_pleine](#)
- class [MatBand](#)
- class [MatDiag](#)
- class [Mat\\_creuse\\_CompCol](#)
- class [MatLapack](#)
- class [Pre\\_cond\\_double](#)
- class [Assemblage](#)
- class [Condilinaire](#)
- class [CondLim](#)

### 5.27.1 Description détaillée

les différentes classes de matrices.

BUT: les différentes classes de matrices.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.28 Les\_sorties\_geomview

groupe concernant le posttraitement avec geomview

### Classes

- class [Animation\\_geomview](#)
- class [Deformees\\_geomview](#)
- class [Fin\\_geomview](#)
- class [Frontiere\\_initiale\\_geomview](#)
- class [Isovaleurs\\_geomview](#)
- class [Mail\\_initiale\\_geomview](#)
- class [Visuali\\_geomview](#)
- class [Visualisation\\_geomview](#)

### 5.28.1 Description détaillée

groupe concernant le posttraitement avec geomview

BUT: groupe concernant le posttraitement avec geomview

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

03/02/2003

## 5.29 Les\_sorties\_Gid

groupe concernant le posttraitement avec Gid

### Classes

- class [Deformees\\_Gid](#)
- class [Fin\\_Gid](#)
- class [Isovaleurs\\_Gid](#)
- class [Mail\\_initiale\\_Gid](#)
- class [Visuali\\_Gid](#)
- class [Visualisation\\_Gid](#)

### 5.29.1 Description détaillée

groupe concernant le posttraitement avec Gid

BUT: groupe concernant le posttraitement avec Gid

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

24/09/2004

## 5.30 Les\_sorties\_gmsh

groupe concernant le posttraitement avec gmsh

## Classes

- class [Deformees\\_Gmsh](#)
- class [Fin\\_Gmsh](#)
- class [Isovaleurs\\_Gmsh](#)
- class [Mail\\_initiale\\_Gmsh](#)
- class [Visuali\\_Gmsh](#)
- class [Visualisation\\_Gmsh](#)

### 5.30.1 Description détaillée

groupe concernant le postraitement avec gmsh

BUT: groupe concernant le postraitement avec gmsh

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

07/01/2008

## 5.31 Les\_sorties\_au\_format\_maple

groupe concernant le postraitement au format maple: format ascii auto-documenté, type tableau bidimensionnel

## Classes

- class [Animation\\_maple](#)
- class [Choix\\_grandeurs\\_maple](#)
- class [Deformees\\_maple](#)
- class [Fin\\_maple](#)
- class [Visuali\\_maple](#)
- class [Visualisation\\_maple](#)

### 5.31.1 Description détaillée

groupe concernant le postraitement au format maple: format ascii auto-documenté, type tableau bidimensionnel

BUT: groupe concernant le postraitement au format maple: format ascii auto-documenté, type tableau bidimensionnel

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97



## 5.32 Les\_sorties\_generiques

groupe concernant le postraitement de manière générique

### Classes

- class [Frontiere\\_initiale](#)
- class [OrdreVisu](#)
- class [Resultats](#)
- class [Visualisation](#)

### 5.32.1 Description détaillée

groupe concernant le postraitement de manière générique

BUT: groupe concernant le postraitement de manière générique (classe virtuelle et sorties avec un format spécifique d'Herezh )

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.33 Les\_sorties\_vrml

groupe concernant le postraitement au format vrml

### Classes

- class [Animation\\_vrml](#)
- class [ChoixDesMaillages\\_vrml](#)
- class [Deformees\\_vrml](#)
- class [Fin\\_vrml](#)
- class [Increment\\_vrml](#)
- class [Isovaleurs\\_vrml](#)
- class [Mail\\_initiale\\_vrml](#)
- class [Visuali\\_vrml](#)

### 5.33.1 Description détaillée

groupe concernant le postraitement au format vrml

BUT: groupe concernant le postraitement au format vrml

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

03/02/2003

## 5.34 Les\_Tableaux\_generiques

groupe des tableaux génériques: typiquement des templates, mais pas seulement

### Classes

- class [Tableau2< T >](#)
- class [Tableau4< T >](#)
- class [Tableau\\_3D](#)
- class [Tableau\\_4D](#)
- class [Tableau\\_5D](#)
- class [Tableau\\_double](#)
- class [Tableau< T >](#)
- class [TabOper< T >](#)

### 5.34.1 Description détaillée

groupe des tableaux génériques: typiquement des templates, mais pas seulement

BUT: groupe des tableaux génériques: typiquement des templates, mais pas seulement

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

## 5.35 Les\_classes\_coordonnee

Définition des classes de type [Coordonnee](#), en coordonnées sans variance (ex: absolues) ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes.

### Classes

- class [Coordonnee](#)  
*Coordonnees simples sans variances.*
- class [CoordonneeH](#)  
*cas des coordonnées contravariantes*
- class [CoordonneeB](#)  
*cas des coordonnées covariantes*

### 5.35.1 Description détaillée

Définition des classes de type [Coordonnee](#), en coordonnées sans variance (ex: absolues) ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes.

BUT: Les classes de type [Coordonnee](#) servent à la localisation dans l'espace des objets tels que les points ou les noeuds etc.

Une instance de cette classe est caractérisée par le nombre de coordonnées et par la valeur de celles-ci.

Les valeurs des coordonnées sont de type double et sont stockées dans un tableau dont la taille depend de la dimension du problème.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.36 Les\_classes\_coordonnee1

Définition des classes de type [Coordonnee1](#), en coordonnées sans variance (ex: absolues) ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont une spécialisation 1D des classes générales [Coordonnee](#).

### Classes

- class [Coordonnee1](#)  
*cas des coordonnées simples sans variance*
- class [Coordonnee1H](#)  
*cas des coordonnées contravariantes*
- class [Coordonnee1B](#)  
*cas des coordonnées covariantes*

### 5.36.1 Description détaillée

Définition des classes de type [Coordonnee1](#), en coordonnées sans variance (ex: absolues) ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont une spécialisation 1D des classes générales [Coordonnee](#).

BUT: Les classes [Coordonnee1](#) servent a la localisation dans l'espace 1D des objets tels que les noeuds ou les points. Ces classes dérivent des Classes génériques [Coordonnee](#).

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

## 5.37 Les\_classes\_cooronnee2

Définition des classes de type [Coordonnee2](#), en coordonnées sans variance (ex: absolues) ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont une spécialisation 2D des classes générales [Coordonnee](#).

### Classes

- class [Coordonnee2](#)  
*cas des coordonnées simples sans variance*
- class [Coordonnee2H](#)  
*cas des coordonnées contravariantes*
- class [Coordonnee2B](#)  
*cas des coordonnées covariantes*

### 5.37.1 Description détaillée

Définition des classes de type [Coordonnee2](#), en coordonnées sans variance (ex: absolues) ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont une spécialisation 2D des classes générales [Coordonnee](#).

BUT: Les classes [Coordonnee1](#) servent a la localisation dans l'espace 2D des objets tels que les noeuds ou les points. Ces classes dérivent des Classes génériques [Coordonnee](#).

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

## 5.38 Les\_classes\_coordonnee3

Définition des classes de type [Coordonnee3](#), en coordonnées sans variance (ex: absolues) ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont une spécialisation 3D des classes générales [Coordonnee](#).

### Classes

- class [Coordonnee3](#)  
*cas des coordonnées simples sans variance*
- class [Coordonnee3H](#)  
*cas des coordonnées contravariantes*
- class [Coordonnee3B](#)  
*cas des coordonnées covariantes*

### 5.38.1 Description détaillée

Définition des classes de type [Coordonnee3](#), en coordonnées sans variance (ex: absolues) ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont une spécialisation 3D des classes générales [Coordonnee](#).

BUT: Les classes [Coordonnee1](#) servent a la localisation dans l'espace 3D des objets tels que les noeuds ou les points. Ces classes dérivent des Classes génériques [Coordonnee](#).

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.39 Les\_classes\_Base

Définition des bases naturelles ou absolues, ou des bases duales.

### Classes

- class [BaseB](#)  
*Les base covariantes et absolus.*
- class [BaseH](#)  
*Les base duales.*
- class [BaseB\\_0\\_t\\_tdt](#)  
*un groupe de 3 bases covariantes et absolus à 0, t et tdt*
- class [BaseH\\_0\\_t\\_tdt](#)  
*un groupe de 3 bases contravariant et absolus à 0, t et tdt*

### 5.39.1 Description détaillée

Définition des bases naturelles ou absolues, ou des bases duales.

BUT: Les classes Base servent à définir des bases en 1D 2D 3D qui permettent d'exprimer des coordonnées, en absolue ou en locale. Ces bases correspondent à des bases naturelles ou des bases duales.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

## 5.40 Les\_classes\_Base3D3

Définition des bases naturelles ou absolues ou des bases duales, de 3 vecteurs de dim 3.

### Classes

- class [Base3D3B](#)  
*Les base covariantes et absolus.*
- class [Base3D3H](#)  
*Les base duales.*

### 5.40.1 Description détaillée

Définition des bases naturelles ou absolues ou des bases duales, de 3 vecteurs de dim 3.

BUT: Les classes Base3D3 servent à définir des bases de 3 vecteurs de dim 3 qui permettent d'exprimer des coordonnées, en absolue ou en locale. Ces bases correspondent à des bases naturelles ou des bases duales.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

## 5.41 Les\_classes\_tenseurs\_virtuelles\_ordre2

Définition des classes virtuelles pures de type Tenseur d'ordre 2, en coordonnées avec différentes variances.

### Classes

- class `TenseurHH`  
*TenseurHH: cas des composantes deux fois contravariantes.*
- class `TenseurBB`  
*TenseurBB: cas des composantes deux fois covariantes.*
- class `TenseurBH`  
*TenseurBH: cas des composantes mixtes BH.*
- class `TenseurHB`  
*TenseurHB: cas des composantes mixtes HB.*

### 5.41.1 Description détaillée

Définition des classes virtuelles pures de type Tenseur d'ordre 2, en coordonnées avec différentes variances.

///

BUT: Définir les tenseurs d'ordre 2 de différentes composantes. Les classes sont virtuelles pures. Elles se déclinent en fonction de la dimension du problème. L'objectif principal est de surcharger les différentes opérations classiques.

concernant le produit contracté un fois, en particulier pour les tenseurs mixtes il y a contraction du 2<sup>ème</sup> indice du premier tenseur avec le premier indice du second tenseur :  $A_{ij} * B_{jk} = C_{ik} \leftrightarrow A * B = C$  le tenseur inverse par rapport au produit contracté est défini de la manière suivante  $Inverse(A) * A = Id$ , ainsi l'inverse d'un tenseur BH est un BH idem pour les HB mais l'inverse d'un BB est un HH, et l'inverse d'un HH est un BB

NB: lorsque les tenseurs mixtes sont issues de tenseurs HH ou BB symétrique, l'ordre de contraction des indices n'a pas d'importance sur le résultat !!

le produit contracté de deux tenseurs symétriques quelconques ne donne pas un tenseur symétrique !!, donc par exemple la contraction d'un tenseur HB avec HH n'est pas forcément symétrique. Le résultat est symétrique SEULEMENT lorsque ces opérations sont effectuées avec le tenseur métrique.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.42 Les\_classes\_Maillons\_tenseurs

Définition des classes qui stockent les pointeurs sur les tenseurs intermédiaire.

## Classes

- class [LesMaillonsHH](#)  
*def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires*
- class [LesMaillonsBB](#)  
*def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires*
- class [LesMaillonsBH](#)  
*def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires*
- class [LesMaillonsHB](#)  
*def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires*

### 5.42.1 Description détaillée

Définition des classes qui stockent les pointeurs sur les tenseurs intermédiaire.

BUT: On gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsXX" qui stocke les pointeurs sur les tenseurs intermédiaire

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.43 Les\_classes\_tenseurs\_dim1\_ordre2

Définition des classes de dimension 1 de type Tenseur d'ordre 2, en coordonnées avec différentes variances.

## Classes

- class [Tenseur1HH](#)  
*Definition des tenseur derivees de dimension1. cas des composantes deux fois contravariantes.*
- class [Tenseur1BB](#)  
*Definition des tenseur derivees de dimension1. cas des composantes deux fois covariantes.*
- class [Tenseur1BH](#)  
*Definition des tenseur derivees de dimension1. cas des composantes mixtes BH.*
- class [Tenseur1HB](#)  
*Definition des tenseur derivees de dimension1. cas des composantes mixtes HB.*



### 5.43.1 Description détaillée

Définition des classes de dimension 1 de type Tenseur d'ordre 2, en coordonnées avec différentes variances.

BUT: Définir les tenseurs d'ordre 2 de différentes composantes, spécifiquement à la dimension 1. L'objectif principal est de surcharger les différentes opérations classiques.

concernant le produit contracté un fois, en particulier pour les tenseurs mixtes il y a contraction du 2<sup>ème</sup> indice du premier tenseur avec le premier indice du second tenseur :  $A_{ij} * B_{jk} = C_{ik} \leftrightarrow A * B = C$  le tenseur inverse par rapport au produit contracté est défini de la manière suivante  $Inverse(A) * A = Id$ , ainsi l'inverse d'un tenseur BH est un BH idem pour les HB mais l'inverse d'un BB est un HH, et l'inverse d'un HH est un BB

NB: lorsque les tenseurs mixtes sont issues de tenseurs HH ou BB symétrique, l'ordre de contraction des indices n'a pas d'importance sur le résultat !!

le produit contracté de deux tenseurs symétriques quelconques ne donne pas un tenseur symétrique !!, donc par exemple la contraction d'un tenseur HB avec HH n'est pas forcément symétrique. Le résultat est symétrique SEULEMENT lorsque ces opérations sont effectuées avec le tenseur métrique.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.44 Les\_classes\_tenseurs\_dim2\_ordre2

Définition des classes de dimension 2 de type Tenseur d'ordre 2, en coordonnées avec différentes variances.

### Classes

- class [Tenseur2HH](#)  
*Definition des tenseur derivees de dimension 2. cas des composantes deux fois contravariantes symetriques.*
- class [Tenseur\\_ns2HH](#)  
*Definition des tenseur derivees de dimension 2. cas des composantes deux fois contravariantes non symetriques pour les differencier la dimension = -2.*
- class [Tenseur2BB](#)  
*Definition des tenseur derivees de dimension 2. cas des composantes deux fois covariantes symetriques.*
- class [Tenseur\\_ns2BB](#)  
*Definition des tenseur derivees de dimension 2. cas des composantes deux fois covariantes non symetriques pour les differencier la dimension = -2.*
- class [Tenseur2BH](#)  
*Definition des tenseur derivees de dimension 2. cas des composantes mixtes BH.*
- class [Tenseur2HB](#)  
*Definition des tenseur derivees de dimension 2. cas des composantes mixtes HB.*

### 5.44.1 Description détaillée

Définition des classes de dimension 2 de type Tenseur d'ordre 2, en coordonnées avec différentes variances.

BUT: Définir les tenseurs d'ordre 2 de différentes composantes, spécifiquement à la dimension 2. L'objectif principal est de surcharger les différentes opérations classiques.

concernant le produit contracté un fois, en particulier pour les tenseurs mixtes il y a contraction du 2<sup>ème</sup> indice du premier tenseur avec le premier indice du second tenseur :  $A_{ij} * B_{jk} = C_{ik} \leftrightarrow A * B = C$  le tenseur inverse par rapport au produit contracté est défini de la manière suivante  $Inverse(A) * A = Id$ , ainsi l'inverse d'un tenseur BH est un BH idem pour les HB mais l'inverse d'un BB est un HH, et l'inverse d'un HH est un BB

NB: lorsque les tenseurs mixtes sont issues de tenseurs HH ou BB symétrique, l'ordre de contraction des indices n'a pas d'importance sur le résultat !!

le produit contracté de deux tenseurs symétriques quelconques ne donne pas un tenseur symétrique !!, donc par exemple la contraction d'un tenseur HB avec HH n'est pas forcément symétrique. Le résultat est symétrique SEULEMENT lorsque ces opérations sont effectuées avec le tenseur métrique.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.45 Les\_classes\_tenseurs\_dim3\_ordre2

Définition des classes de dimension 3 de type Tenseur d'ordre 2, en coordonnées avec différentes variances.

### Classes

- class [Tenseur3HH](#)  
*Definition des tenseur derivees de dimension 3. cas des composantes deux fois contravariantes.*
- class [Tenseur\\_ns3HH](#)  
*Definition des tenseur derivees de dimension 3. cas des composantes deux fois contravariantes non symetriques pour les differencier la dimension = -3.*
- class [Tenseur3BB](#)  
*Definition des tenseur derivees de dimension 3. cas des composantes deux fois covariantes.*
- class [Tenseur\\_ns3BB](#)  
*Definition des tenseur derivees de dimension 3. cas des composantes deux fois covariantes non symetriques pour les differencier la dimension = -3.*
- class [Tenseur3BH](#)  
*Definition des tenseur derivees de dimension 3. cas des composantes mixtes BH.*
- class [Tenseur3HB](#)  
*Definition des tenseur derivees de dimension 3. cas des composantes mixtes HB.*

### 5.45.1 Description détaillée

Définition des classes de dimension 3 de type Tenseur d'ordre 2, en coordonnées avec différentes variances.

BUT: Définir les tenseurs d'ordre 2 de différentes composantes, spécifiquement à la dimension 3. L'objectif principal est de surcharger les différentes opérations classiques.

concernant le produit contracté un fois, en particulier pour les tenseurs mixtes il y a contraction du 2<sup>ème</sup> indice du premier tenseur avec le premier indice du second tenseur :  $A_{ij} * B_{jk} = C_{ik} \leftrightarrow A * B = C$  le tenseur inverse par rapport au produit contracté est défini de la manière suivante  $Inverse(A) * A = Id$ , ainsi l'inverse d'un tenseur BH est un BH idem pour les HB mais l'inverse d'un BB est un HH, et l'inverse d'un HH est un BB

NB: lorsque les tenseurs mixtes sont issues de tenseurs HH ou BB symétrique, l'ordre de contraction des indices n'a pas d'importance sur le résultat !!

le produit contracté de deux tenseurs symétriques quelconques ne donne pas un tenseur symétrique !!, donc par exemple la contraction d'un tenseur HB avec HH n'est pas forcément symétrique. Le résultat est symétrique SEULEMENT lorsque ces opérations sont effectuées avec le tenseur métrique.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.46 Les\_classes\_tenseurs\_virtuelles\_ordre4

Définition des classes virtuelles pures de type Tenseur d'ordre 4, en coordonnées avec différentes variances.

### Classes

- class [TenseurHHHH](#)  
*cas des composantes 4 fois contravariantes*
- class [TenseurBBBB](#)  
*cas des composantes 4 fois covariantes*
- class [TenseurHHBB](#)  
*cas des composantes mixte HHBB*
- class [TenseurBBHH](#)  
*cas des composantes mixte BBHH*
- class [TenseurHBHB](#)  
*cas des composantes 2 fois mixtes HBHB*
- class [TenseurBHBH](#)  
*cas des composantes 2 fois mixtes BHBH*
- class [TenseurHBBH](#)  
*cas des composantes 2 fois mixtes HBBH*
- class [TenseurBHBB](#)  
*cas des composantes 2 fois mixtes BHBB*

### 5.46.1 Description détaillée

Définition des classes virtuelles pures de type Tenseur d'ordre 4, en coordonnées avec différentes variances.

BUT: Définir les tenseurs d'ordre 4 de différentes composantes. Les classes sont virtuelles pures. Elles se déclinent en fonction de la dimension du problème. L'objectif principal est de surcharger les différentes opérations classiques.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

2/5/2002

## 5.47 Les\_classes\_Maillons\_tenseurs\_ordre4

Définition des classes qui stockent les pointeurs sur les tenseurs intermédiaire d'ordre 4.

### Classes

- class [LesMaillonsHHHH](#)  
*def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires*
- class [LesMaillonsBBBB](#)  
*def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires*
- class [LesMaillonsHHBB](#)  
*def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires*
- class [LesMaillonsBBBH](#)  
*def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires*
- class [LesMaillonsHBHB](#)  
*def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires*
- class [LesMaillonsBHBH](#)  
*def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires*
- class [LesMaillonsHBBH](#)  
*def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires*
- class [LesMaillonsBHBB](#)  
*def d'un maillon de liste chaînée pour mémoriser les différents tenseurs intermédiaires*

### 5.47.1 Description détaillée

Définition des classes qui stockent les pointeurs sur les tenseurs intermédiaire d'ordre 4.

BUT: On gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsXXXX" qui stocke les pointeurs sur les tenseurs intermédiaire d'ordre 4

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

2/5/2002

## 5.48 Les\_classes\_Vecteur

Définition des classes de type "vecteur".

### Classes

- class [Vecteur](#)  
*La classe [Vecteur](#) permet de déclarer des vecteurs d'une longueur predefinie par une allocation dynamique de memoire. Les composantes d'un vecteur de cette classe sont de type double. Correspond à un vecteur absolu ( par opposition avec des vecteurs avec des coordonnees covariantes ou contravariantes.*
- class [MV\\_Vecteur](#)  
*définition d'une classe de jonction entre les Vecteurs et les MV\_Vecteurs*

### 5.48.1 Description détaillée

Définition des classes de type "vecteur".

Les classes de type "vecteur" permettent de déclarer des vecteurs d'une longueur predefinie par une allocation dynamique de memoire. Les composantes d'un vecteur sont de type double. correspond au vecteur absolu ( par opposition avec des vecteurs avec des coordonnees covariantes ou contravariantes

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.49 Les\_conteneurs\_ultra\_basiques

Listes des conteneurs ultra basiques.

### Classes

- class [DeuxEntiers](#)  
*cas de 2 entiers*
- class [DeuxDoubles](#)  
*cas de 2 double*
- class [Entier\\_et\\_Double](#)  
*cas d' 1 entier et 1 double*
- class [Deux\\_String](#)  
*cas de deux String*
- class [String\\_et\\_entier](#)  
*cas d'un string et un entier*
- class [Trois\\_String](#)  
*cas de trois String*
- class [quatre\\_string\\_un\\_entier](#)  
*cas de 4 string et un entier*
- class [TroisEntiers](#)  
*cas de 3 entiers*
- class [Deux\\_String\\_un\\_entier](#)  
*cas de deux String et un entier*

### 5.49.1 Description détaillée

Listes des conteneurs ultra basiques.

BUT: conteneurs ultra basiques, permettant entre autres \* la différenciation des types. \*

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

14/03/2003

## 5.50 Group\_Ddl\_enum\_etendu

classe qui permet de définir des identificateurs de grandeurs secondaires associées à des Enum\_ddl ddl de base.

### Classes

— class [Initialisation\\_tab\\_De](#)

*une classe secondaire: définition d'une classe pour l'initialisation du tableau tab\_De et de la map qui relie les string et les [Ddl\\_enum\\_etendu](#)*

— class [Ddl\\_enum\\_etendu](#)

*la classe principale: [Ddl\\_enum\\_etendu](#)*

### 5.50.1 Description détaillée

classe qui permet de définir des identificateurs de grandeurs secondaires associées à des Enum\_ddl ddl de base.

BUT: une classe qui permet de définir des identificateurs de grandeurs secondaires associées à des Enum\_ddl ddl de base. Ces grandeurs ne sont pas vraiment des ddl, mais ils s'en déduisent. Ainsi pour ne pas alourdir le type Enum ddl on crée cette classe associée. Elle est prévue en particulier pour gérer les sorties de résultats. les éléments sont ordonnés selon la règle suivante : si c'est un Enum\_ddl pur -> ordre de l'Enum si c'est un élément de tab\_De -> ordre dans tab\_De plus maxi des Enum\_ddl lorsqu'il s'agit d'un enum\_ddl tout simple, dans ce cas Non\_vide == true

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

## 5.51 Les\_list\_io

création de conteneurs type listes stl avec surcharge de lecture écriture

### Classes

- class [List\\_io< T >](#)  
*List\_io* classe template identique à *list* de stl, avec en plus une lecture et écriture.
- class [LaLIST\\_io< T >](#)  
*LaLIST\_io* classe template identique à *List\_io*: existe pour des raisons historiques et en prévision de changements futurs éventuelles (?)

### 5.51.1 Description détaillée

création de conteneurs type listes stl avec surcharge de lecture écriture

BUT: création de conteneurs type listes stl, qui comportent en plus une surcharge de lecture écriture. Idem pour LaList.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.52 Les\_classes\_PlusieursCoordonnees

classes relatives à des nombres fixes de coordonnées.

### Classes

- class [DeuxCoordonnees](#)  
*DeuxCoordonnees*: classe relative à 2 coordonnées.

### 5.52.1 Description détaillée

classes relatives à des nombres fixes de coordonnées.

///

BUT: classes relatives à des nombres fixes de coordonnées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

## 5.53 Les\_classes\_Ponderation

Def de classes conteneurs pour manipuler différentes pondérations.

### Classes

- class [Ponderation\\_Consultable](#)  
*une classe de travail, qui permet d'utiliser une pondération qui dépend de plusieurs courbe1D, chacune dépendant d'une grandeur consultable par un string*
- class [Ponderation\\_GGlobal](#)  
*une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble de grandeurs globales au travers d'une fonction nD*
- class [Ponderation\\_TypeQuelconque](#)  
*une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble de grandeurs quelconque au travers d'une fonction nD*
- class [Ponderation](#)  
*une seconde classe de travail qui permet d'utiliser une pondération qui dépend de n [Courbe1D](#), chacune fonction d'un ddl étendu*
- class [Ponderation\\_temps](#)  
*une classe de travail qui permet d'utiliser une pondération qui dépend du temps via une [Courbe1D](#)*

### 5.53.1 Description détaillée

Def de classes conteneurs pour manipuler différentes pondérations.

BUT: Def de classes conteneurs pour manipuler différentes pondérations

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

04/07/2016

## 5.54 Les\_listes\_de\_petits\_tableaux\_de\_reels

Listes de petits tableaux de réels.

### Classes

- class [Reels1](#)
- class [Reels2](#)
- class [Reels3](#)
- class [Reels4](#)
- class [Reels5](#)
- class [Reels6](#)
- class [Reels7](#)
- class [Reels8](#)
- class [Reels9](#)
- class [Reels16](#)
- class [Reels21](#)
- class [Reels36](#)
- class [Reels81](#)



### 5.54.1 Description détaillée

Listes de petits tableaux de réels.

cas des tableaux de 81 réels

cas des tableaux de 36 réels

cas des tableaux de 21 réels

cas des tableaux de 16 réels

cas des tableaux de 9 réels

cas des tableaux de 8 réels

cas des tableaux de sept réels

cas des tableaux de 6 réels

cas des tableaux de 5 réels

cas des tableaux de quatre réels

cas des tableaux de trois réels

cas des tableaux de 2 réels

cas des tableaux de 1 réel

BUT: Listes de petits tableaux de réels la définition exacte des listes globales est faite dans le fichier PtTabReI\_↔  
Prin.h qui n'est inclus qu'une seule fois dans le fichier du programme principal

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.55 Les pointeurs dans listes de petits tableaux

def de pointeur qui donne la position d'un élément dans une liste

## Classes

- class [ReelsPointe](#)  
*ReelsPointe* classe virtuelle de laquelle dérive les classe contenant un pointeur de réel.
- class [Reel1Pointe](#)
- class [Reel2Pointe](#)
- class [Reel3Pointe](#)
- class [Reel4Pointe](#)
- class [Reel5Pointe](#)
- class [Reel6Pointe](#)
- class [Reel7Pointe](#)
- class [Reel8Pointe](#)
- class [Reel9Pointe](#)
- class [Reel16Pointe](#)
- class [Reel21Pointe](#)
- class [Reel36Pointe](#)
- class [Reel81Pointe](#)

### 5.55.1 Description détaillée

def de pointeur qui donne la position d'un élément dans une liste

cas des tableaux de 81 réels

cas des tableaux de 36 réels

cas des tableaux de 21 réels

cas des tableaux de 16 réels

cas des tableaux de 9 réels

cas des tableaux de 8 réels

cas des tableaux de sept réels

cas des tableaux de 6 réels

cas des tableaux de 5 réels

cas des tableaux de quatre réels

cas des tableaux de trois réels

cas des tableaux de 2 réels

BUT: def de pointeur qui donne la position d'un élément dans une liste

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.56 Group\_TypeQuelconque

classe qui permet de définir des identificateurs de grandeurs quelconque.

### Classes

— class [TypeQuelconque](#)

*TypeQuelconque* : def de la classe générique globale qui sert a gérer la grandeur particulière qui est attachée via un pointeur.

### 5.56.1 Description détaillée

classe qui permet de définir des identificateurs de grandeurs quelconque.

BUT: une classe générique qui permet de définir des identificateurs de grandeurs quelconque.

Ces identificateurs sont prévue en particulier pour gérer les sorties de résultats. Le type s'appuie sur un type énuméré pour les gestions rapide de tableau ou de choix (via case). Le type contient une grandeur pointée, mais ne crée pas ni ne détruit la grandeur pointée! Ainsi au niveau de l'affectation, il ne gère que le changement de pointeur.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

29/05/2004

## 5.57 Group\_TypeQuelconque\_enum\_etendu

classe qui permet de définir des identificateurs de grandeurs secondaires associées à des EnumTypeQuelconque.

### Classes

— class [Initialisation\\_tab\\_Daa](#)

*Initialisation\_tab\_Daa*: definition d'une classe pour l'initialisation du tableau tab\_Daa.

— class [TypeQuelconque\\_enum\\_etendu](#)

*TypeQuelconque\_enum\_etendu*: la classe qui gère les types quelconques étendus.

### 5.57.1 Description détaillée

classe qui permet de définir des identificateurs de grandeurs secondaires associées à des EnumTypeQuelconque.

BUT: une classe qui permet de définir des identificateurs de grandeurs secondaires associées des EnumTypeQuelconque. Ces grandeurs ne sont pas vraiment des EnumTypeQuelconque, mais ils s'en déduisent. Ainsi pour ne pas alourdir l'enum de base on crée cette classe associée. Elle est prévue en particulier pour gérer les sorties de résultats. les éléments sont ordonnés selon la règle suivante : si c'est un EnumTypeQuelconque pur -> ordre de l'Enum si c'est un élément de tab\_Daa -> ordre dans tab\_Daa

- maxi des EnumTypeQuelconque lorsqu'il s'agit d'un EnumTypeQuelconque tout simple, dans ce cas Non\_vider == true

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

03/12/2017

## 5.58 Les grandeurs particulières

Définition des grandeurs particulières.

### Classes

- class [Grandeur\\_defaut](#)
- class [Grandeur\\_TenseurHH](#)  
*grandeur TenseurHH: TypeQuelconque::Grandeur::Grandeur\_TenseurHH|*
- class [Grandeur\\_TenseurBB](#)  
*grandeur TenseurBB: TypeQuelconque::Grandeur::Grandeur\_TenseurBB|*
- class [Grandeur\\_TenseurBH](#)  
*grandeur TenseurBH: TypeQuelconque::Grandeur::Grandeur\_TenseurBH|*
- class [Grandeur\\_TenseurHB](#)  
*grandeur TenseurHB: TypeQuelconque::Grandeur::Grandeur\_TenseurHB|*
- class [Tab\\_Grandeur\\_TenseurHH](#)  
*grandeur un tableau de TenseurHH: TypeQuelconque::Grandeur::Tab\_Grandeur\_TenseurHH| tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre*
- class [Tab\\_Grandeur\\_TenseurBH](#)  
*grandeur un tableau de TenseurBH: TypeQuelconque::Grandeur::Tab\_Grandeur\_TenseurBH| tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre*
- class [Tab\\_Grandeur\\_TenseurBB](#)  
*grandeur un tableau de TenseurBB: TypeQuelconque::Grandeur::Tab\_Grandeur\_TenseurBB| tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre*
- class [Tab\\_Grandeur\\_TenseurHB](#)  
*grandeur un tableau de TenseurHB: TypeQuelconque::Grandeur::Tab\_Grandeur\_TenseurHB tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre*
- class [Tab2\\_Grandeur\\_TenseurHH](#)  
*grandeur un tableau à 2 dim de TenseurHH: TypeQuelconque::Grandeur::Tab2\_Grandeur\_TenseurHH*
- class [Grandeur\\_scalaire\\_entier](#)  
*grandeur scalaire réel: TypeQuelconque::Grandeur::Grandeur\_scalaire\_double*
- class [Tab\\_Grandeur\\_scalaire\\_entier](#)

- grandeur tableau de scalaires entière: TypeQuelconque::Grandeur::Tab\_Grandeur\_scalaire\_entiere*
- class [Grandeur\\_scalaire\\_double](#)
- grandeur scalaire réel: TypeQuelconque::Grandeur::Grandeur\_scalaire\_double*
- class [Tab\\_Grandeur\\_scalaire\\_double](#)
- grandeur tableau de scalaires réel: TypeQuelconque::Grandeur::Tab\_Grandeur\_scalaire\_double*
- class [Grandeur\\_Vecteur](#)
- grandeur vecteur: TypeQuelconque::Grandeur::Grandeur\_vecteur*
- class [Tab\\_Grandeur\\_Vecteur](#)
- grandeur un tableau de [Vecteur](#): TypeQuelconque::Grandeur::Tab\_Grandeur\_Vecteur| tous les Vecteurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre ?? à voir*
- class [Grandeur\\_coorдонnee](#)
- grandeur coordonnée: TypeQuelconque::Grandeur::Grandeur\_coorдонnee*
- class [Tab\\_Grandeur\\_Coorдонnee](#)
- grandeur un tableau de [Coordonnee](#): TypeQuelconque::Grandeur::Tab\_Grandeur\_CoorдонneeHH tous les coordonnees doivent avoir la même dimension !! pour la routine GrandeurNumOrdre*
- class [Grandeur\\_Ddl\\_etendu](#)
- grandeur [Ddl\\_etendu](#): TypeQuelconque::Grandeur::Grandeur\_Ddl\_etendu*
- class [Tab\\_Grandeur\\_Ddl\\_etendu](#)
- grandeur un tableau de [Coordonnee](#): TypeQuelconque::Grandeur::Tab\_Grandeur\_CoorдонneeHH tous les coordonnees doivent avoir la même dimension !! pour la routine GrandeurNumOrdre*
- class [Grandeur\\_Vecteur\\_Nommer](#)
- grandeur vecteur nommé : TypeQuelconque::Grandeur::Grandeur\_Vecteur\_Nommer contient un nom d'identificateur (utilisé par exemple pour un nom de fonc*
- class [Tab\\_Grandeur\\_Vecteur\\_Nommer](#)
- grandeur un tableau de [Grandeur\\_Vecteur\\_Nommer](#): TypeQuelconque::Grandeur::Tab\_Grandeur\_Grandeur\_↔ Vecteur\_Nommer*
- class [Grandeur\\_BaseH](#)
- grandeur [BaseH](#): TypeQuelconque::Grandeur::Grandeur\_BaseH*
- class [Tab\\_Grandeur\\_BaseH](#)
- grandeur un tableau de [Grandeur\\_BaseH](#): TypeQuelconque::Grandeur::Tab\_Grandeur\_BaseH tous les coordonnees doivent avoir la même dimension !! pour la routine GrandeurNumOrdre*
- class [Grandeur\\_Double\\_Nommer\\_indicer](#)
- grandeur scalaire double nommé + indice: TypeQuelconque::Grandeur::Grandeur\_Double\_Nommer\_indicer contient un nom d'identificateur et un indice qui est sensé resté fixe pendant les opérations la seule grandeur qui varie c'est le double: c'est donc équivalent à un scalaire double l'indice et le nom\_ref, servent pour la gestion*
- class [Tableau\\_Grandeur\\_quelconque](#)
- un tableau de grandeur quelconque*

### 5.58.1 Description détaillée

Définition des grandeurs particulières.

BUT: classes qui permettent de définir des identificateurs de grandeurs particulières héritants du [TypeQuelconque](#). Ces identificateurs sont prévue en particulier pour gérer les sorties de résultats.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

30/05/2004

## 5.59 Les\_classes\_algo

Algorithmes de base pour la résolution d'équations différentielles, recherche de zéros, intégration.

### Classes

- class [Algo\\_edp](#)  
*BUT: Algorithmes de base pour la résolution d'équations différentielles.*
- class [Algo\\_Integ1D](#)  
*Algorithme d'intégration 1D.*
- class [ErrNonConvergence\\_Newton](#)  
*pour la gestion d'exception pour non convergence*
- class [Algo\\_zero](#)  
*Algorithmes de base pour la recherche de zéro d'une fonction ou d'un ensemble de fonctions.*

### 5.59.1 Description détaillée

Algorithmes de base pour la résolution d'équations différentielles, recherche de zéros, intégration.

BUT: Algorithmes de base pour la résolution d'équations différentielles, recherche de zéros, intégration.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

12/02/2006

## 5.60 Les\_courbes\_1D

Def des courbes 1D.

## Classes

- class [Courbe1D](#)  
*Classe virtuelle permettant le calcul d'une fonction 1D ainsi qu'éventuellement un certain nombre d'information supplémentaires telles que dérivées.*
- class [Courbe\\_cos](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $\text{ampli} \cdot \cos(\alpha \cdot x + \beta)$*
- class [Courbe\\_expo2\\_n](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $(\gamma + \alpha \cdot (x \cdot x)^n)$*
- class [Courbe\\_expo\\_n](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = (\gamma + \alpha \cdot (x)^n)$*
- class [Courbe\\_expoaff](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \gamma + \alpha \cdot (|x|^n)$*
- class [Courbe\\_expression\\_litterale\\_1D](#)  
*Classe permettant le calcul d'une fonction 1D de type : une expression littérale.*
- class [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#)  
*Classe permettant le calcul d'une fonction 1D et de sa dérivée de type : une expression littérale.*
- class [Courbe\\_In\\_cosh](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = a \cdot (b e^{g \cdot x})^n \cdot \ln(\cosh(d e^{(h e + e e \cdot x)})) + k e^{(g e + r e \cdot x)}$*
- class [Courbe\\_relax\\_expo](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = (a - b) \cdot \exp(-c \cdot x) + b$ .*
- class [Courbe\\_sin](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{ampli} \cdot \sin(\alpha \cdot x + \beta)$*
- class [Courbe\\_un\\_moins\\_cos](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = c \cdot (1 - \cos((x - a) \cdot \pi / (b - a)))$*
- class [CourbePolyHermite1D](#)  
*Classe permettant le calcul d'une fonction 1D polynomiale 1D par morceau, avec continuité de la dérivée via une interpolation de type hermite.*
- class [CourbePolyLineaire1D](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{poly-linéaire } c\text{-a-d linéaire par morceaux limités par } 2 \text{ points.}$*
- class [Cpl1D](#)  
*Classe permettant le calcul d'une fonction 1D de type : poly-linéaire 1D simplifiée, qui hérite de la fonction poly-linéaire.*
- class [CourbePolynomiale](#)  
*Classe permettant le calcul d'un polynôme 1D de type :  $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots$*
- class [F1\\_plus\\_F2](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $F(x) = (F1 + F2)(x) = F1(x) + F2(x)$*
- class [F1\\_rond\\_F2](#)  
*Classe permettant le calcul d'une fonction 1D composé :  $F(x) = F1(F2(x)) = F1 \circ F2(x)$*
- class [F\\_cycle\\_add](#)  
*Classe permettant le calcul d'une fonction cyclique avec une amplification additive à chaque cycle.*
- class [F\\_cyclique](#)  
*Classe permettant le calcul d'une fonction cyclique avec une amplification multiplicative à chaque cycle.*
- class [F\\_union\\_1D](#)  
*Classe permettant le calcul d'une fonction égale à l'union d'une suite de fonctions définies sur des segments contigus de manière à couvrir un segment global.*
- class [LesCourbes1D](#)  
*Classe de gestion des différentes courbes 1D enregistrées.*
- class [Poly\\_Lagrange](#)  
*Classe permettant le calcul d'un polynôme 1D à l'aide d'une interpolation de Lagrange.*
- class [SixpodeCos3phi](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = 1 / (1 + \gamma \cdot \cos(3 \cdot \phi)^2)^n$ .*
- class [TangenteHyperbolique](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = a + b \cdot \tanh((x - c) / d)$*
- class [TripodeCos3phi](#)  
*Classe permettant le calcul d'une fonction 1D de type :  $f(x) = 1 / (1 + \gamma \cdot \cos(3 \cdot \phi))^n$ .*

### 5.60.1 Description détaillée

Def des courbes 1D.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

19/01/2001

## 5.61 Les\_fonctions\_nD

Def des fonctions nD.

### Classes

- class [F\\_nD\\_courbe1D](#)  
*Classe permettant d'utiliser une fonction courbe à l'intérieur d'une fonction nD .*
- class [Fonc\\_scal\\_combinees\\_nD](#)  
*Classe permettant le calcul d'une fonction scalaire nD correspondant à une combinaison d'autres fonctions scalaire nD.*
- class [Fonction\\_expression\\_litterale\\_nD](#)  
*Classe permettant le calcul d'une fonction nD de type : une expression littérale.*
- union [Tab\\_car\\_double\\_int\\_1](#)  
*définition d'une union qui lie les réels, les entiers et les caractères*
- class [Fonction\\_externe\\_nD](#)  
*Classe permettant le calcul d'une fonction nD externe, définie par l'utilisateur, via 2 pipes nommés.*
- class [ErrCalculFct\\_nD](#)  
*gestion d'exception pour des erreurs d'appel de fonction nD*
- class [Fonction\\_nD](#)  
*Classe virtuelle d'interface permettant le calcul d'une fonction nD ainsi qu'éventuellement un certain nombre d'informations supplémentaires telles que dérivées.*
- class [LesFonctions\\_nD](#)  
*gestion des différentes fonctions multivariables enregistrées.*

### 5.61.1 Description détaillée

Def des fonctions nD.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

01/06/2016



## 5.62 Classes\_utilitaires\_sur\_vecteurs\_et\_matrices

Def d'utilitaires divers mathematiques sur vecteurs et matrices.

### Classes

- class [ErrMathUtil2](#)  
*cas d'une erreur survenue à cause d'une non convergence*
- class [MathUtil2](#)
- class [Util](#)  
*Util: divers utilitaires sur vecteurs, coordonnées, matrices ...*

### 5.62.1 Description détaillée

Def d'utilitaires divers mathematiques sur vecteurs et matrices.

----- fin gestion des erreurs -----

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

## 5.63 Classes\_gestions\_arret\_Herezh

Def gestion de l'arrêt d'Herezh.

### Classes

- class [ErrSortie](#)  
*gestion d'exception pour Sortie*
- class [ErrSortieFinale](#)  
*gestion d'exception pour Sortie finale*

### Fonctions

- void **Sortie** (int n)  
*fonction pour capter un appel à la sortie*

### 5.63.1 Description détaillée

Def gestion de l'arrêt d'Herezh.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

## 5.64 def\_classes\_suites\_reel

Classe permettant des calculs relatifs à des suites reel.

### Classes

- class [LesSuiteReel](#)  
*gestion des différentes Suites de réels enregistrées.*
- class [Suite\\_arithmetique](#)  
*Classe permettant des calculs relatifs à des suites arithmétique:  $u_n=q U_{(n-1)}$*
- class [Suite\\_equidistante](#)  
*[Suite\\_equidistante](#): Classe permettant des calculs relatifs à des suites equidistante:  $u_n=U_{(n-1)}$*
- class [Suite\\_geometrique](#)  
*[Suite\\_geometrique](#): cas d'une suite géométrique.*
- class [SuiteReel](#)  
*classe d'interface (virtuelle pure) pour la création et l'utilisation de suite*

### 5.64.1 Description détaillée

Classe permettant des calculs relatifs à des suites reel.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

14/11/2007

## 5.65 Groupe\_concernant\_le\_chargement

### Classes

- class [ErrNonConvergence\\_loiDeComportement](#)  
*cas d'une erreur survenue à cause d'une non convergence pour la résolution d'une loi de comportement incrémentale*
- class [Pression\\_appliquee](#)
- class [Force\\_hydroDyna](#)
- class [LesChargeExtSurElement](#)

### 5.65.1 Description détaillée

## 5.66 Groupe\_des\_elements\_finis

### Classes

- class [Biel\\_axi](#)
- class [Biel\\_axiQ](#)
- class [Biellette](#)
- class [BielletteC1](#)
- class [BielletteQ](#)
- class [PoutSimple1](#)
- class [PoutTimo](#)
- class [ElemMeca](#)
- class [ElemPoint\\_CP](#)
- class [ElemPoint\\_CP::ConstrucElemPoint\\_CP](#)
- class [ErrJacobienNegatif\\_ElemMeca](#)
- class [ErrVarJacobienMini\\_ElemMeca](#)
- class [Err\\_inconnue\\_ElemMeca](#)
- class [Hexa](#)
- class [Hexa\\_cm1pti](#)
- class [Hexa\\_cm27pti](#)
- class [Hexa\\_cm64pti](#)
- class [HexaLMemb](#)
- class [HexaMemb](#)
- class [HexaQ](#)
- class [HexaQ\\_cm1pti](#)
- class [HexaQ\\_cm27pti](#)
- class [HexaQ\\_cm64pti](#)
- class [HexaQComp](#)
- class [HexaQComp\\_cm1pti](#)
- class [HexaQComp\\_cm27pti](#)
- class [HexaQComp\\_cm64pti](#)
- class [PentaL](#)
- class [PentaL\\_cm1pti](#)
- class [PentaL\\_cm2pti](#)
- class [PentaL\\_cm6pti](#)
- class [PentaMemb](#)
- class [PentaQ](#)
- class [PentaQ\\_cm12pti](#)
- class [PentaQ\\_cm18pti](#)
- class [PentaQ\\_cm3pti](#)
- class [PentaQ\\_cm9pti](#)
- class [PentaQComp](#)
- class [PentaQComp\\_cm12pti](#)
- class [PentaQComp\\_cm18pti](#)
- class [PentaQComp\\_cm9pti](#)
- class [QuadAxiCCom](#)
- class [QuadAxiCCom\\_cm9pti](#)

- class [QuadAxiL1](#)
- class [QuadAxiL1\\_cm1pti](#)
- class [QuadAxiMemb](#)
- class [QuadAxiQ](#)
- class [QuadAxiQComp](#)
- class [QuadAxiQComp\\_cm4pti](#)
- class [Quad](#)
- class [Quad\\_cm1pti](#)
- class [QuadCCom](#)
- class [QuadCCom\\_cm9pti](#)
- class [QuadQ](#)
- class [QuadQCom](#)
- class [QuadQCom\\_cm4pti](#)
- class [QuadraMemb](#)
- class [SfeMembT](#)
- class [TriaQSfe1](#)
- class [TriaQSfe3](#)
- class [TriaSfe1](#)
- class [TriaSfe1\\_cm5pti](#)
- class [TriaSfe2](#)
- class [TriaSfe3](#)
- class [TriaSfe3\\_3D](#)
- class [TriaSfe3\\_cm12pti](#)
- class [TriaSfe3\\_cm13pti](#)
- class [TriaSfe3\\_cm3pti](#)
- class [TriaSfe3\\_cm4pti](#)
- class [TriaSfe3\\_cm5pti](#)
- class [TriaSfe3\\_cm6pti](#)
- class [TriaSfe3\\_cm7pti](#)
- class [TriaSfe3C](#)
- class [Tetra](#)
- class [TetraMemb](#)
- class [TetraQ](#)
- class [TetraQ\\_15pti](#)
- class [TetraQ\\_cm1pti](#)
- class [TriaAxiL1](#)
- class [TriaAxiMemb](#)
- class [TriaAxiQ3](#)
- class [TriaAxiQ3\\_cm1pti](#)
- class [TriaAxiQ3\\_cmpti1003](#)
- class [TriaCub](#)
- class [TriaCub\\_cm4pti](#)
- class [TriaMemb](#)
- class [TriaMembL1](#)
- class [TriaMembQ3](#)
- class [TriaMembQ3\\_cm1pti](#)
- class [TriaQ3\\_cmpti1003](#)
- class [BielletteThermi](#)
- class [ElemThermi](#)
- class [ErrJacobienNegatif\\_ElemThermi](#)
- class [ErrVarJacobienMini\\_ElemThermi](#)

## Fonctions

- [ElemPoint\\_CP::ElemPoint\\_CP](#) (int num\_mail, int num\_id)
- [ElemPoint\\_CP::ElemPoint\\_CP](#) (const [ElemPoint\\_CP](#) &elem)
- Element \* [ElemPoint\\_CP::Nevez\\_copie](#) () const
- [ElemPoint\\_CP](#) & [ElemPoint\\_CP::operator=](#) ([ElemPoint\\_CP](#) &biel)
- virtual void [ElemPoint\\_CP::InitialisationUmatAbaqus](#) ()
- static const [ElemPoint\\_CP::inNeNpti](#) & [ElemPoint\\_CP::Lecture\\_Abaqus](#) (bool utilisation\_umat\_interne)
- static const [ElemPoint\\_CP::inNeNpti](#) & [ElemPoint\\_CP::IncreElemPtint\\_encours](#) ()
- int [ElemPoint\\_CP::Dim\\_sig\\_eps](#) () const
- Element \* [ElemPoint\\_CP::ConstrucElemPoint\\_CP::NouvelElement](#) (int num\_maill, int num)
- bool [ElemPoint\\_CP::ConstrucElemPoint\\_CP::Element\\_possible](#) ()

## Variables

- static [ElemPoint::DonneeCommune](#) \* [ElemPoint\\_CP::doCo\\_CP](#) = NULL
- static [ElemPoint::UneFois](#) [ElemPoint\\_CP::unefois\\_CP](#)
- static [ConstrucElemPoint\\_CP](#) [ElemPoint\\_CP::construcElemPoint\\_CP](#)

### 5.66.1 Description détaillée

### 5.66.2 Documentation des fonctions

#### 5.66.2.1 Dim\_sig\_eps()

```
int ElemPoint_CP::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Réimplémentée à partir de [ElemPoint](#).

#### 5.66.2.2 InitialisationUmatAbaqus()

```
virtual void ElemPoint_CP::InitialisationUmatAbaqus ( ) [inline], [virtual]
```

Réimplémentée à partir de [ElemPoint](#).



## Chapitre 6

# Documentation des classes

### 6.1 Référence de la structure `__CLPK_complex`

#### Attributs publics

- `__CLPK_real r`
- `__CLPK_real i`

La documentation de cette structure a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/matrices_lapack/Include_lapack/clapack.h`

### 6.2 Référence de la structure `__CLPK_doublecomplex`

#### Attributs publics

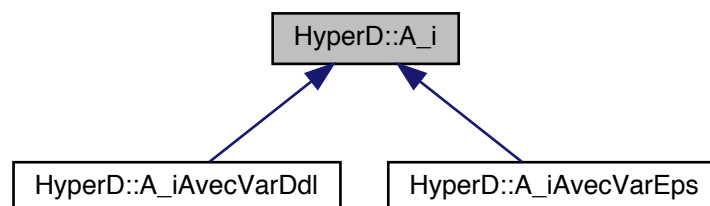
- `__CLPK_doublereal r`
- `__CLPK_doublereal i`

La documentation de cette structure a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/matrices_lapack/Include_lapack/clapack.h`

### 6.3 Référence de la classe `HyperD::A_i`

Graphe d'héritage de `HyperD::A_i`:



## Fonctions membres publiques

- [A\\_i](#) (const [A\\_i](#) &a)
- [A\\_i](#) & **operator=** (const [A\\_i](#) &a)

## Attributs publics

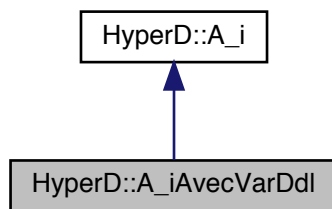
- double **a\_0**
- double **a\_1**
- double **a\_2**

La documentation de cette classe a été générée à partir du fichier suivant :

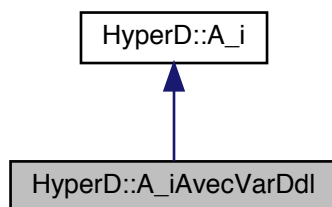
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.4 Référence de la classe HyperD::A\_iAvecVarDdl

Graphe d'héritage de HyperD::A\_iAvecVarDdl:



Graphe de collaboration de HyperD::A\_iAvecVarDdl:





## Fonctions membres publiques

- [A\\_iAvecVarDdl](#) (int nddl=0)
- [A\\_iAvecVarDdl](#) (const [A\\_iAvecVarDdl](#) &a)
- [A\\_iAvecVarDdl](#) & **operator=** (const [A\\_iAvecVarDdl](#) &a)

## Attributs publics

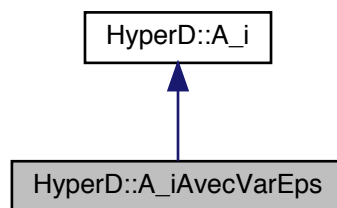
- [Tableau](#)< double > **da\_0**
- [Tableau](#)< double > **da\_1**
- [Tableau](#)< double > **da\_2**

La documentation de cette classe a été générée à partir du fichier suivant :

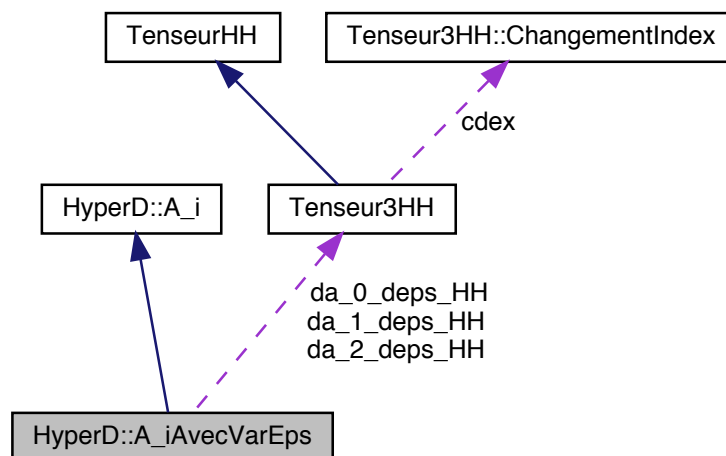
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.5 Référence de la classe HyperD::A\_iAvecVarEps

Graphe d'héritage de HyperD::A\_iAvecVarEps:



Graphe de collaboration de HyperD::A\_iAvecVarEps:



## Fonctions membres publiques

- **A\_iAvecVarEps** (const [A\\_iAvecVarEps](#) &a)
- [A\\_iAvecVarEps](#) & **operator=** (const [A\\_iAvecVarEps](#) &a)

## Attributs publics

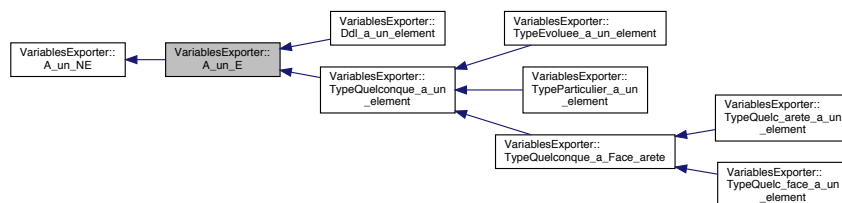
- [Tenseur3HH](#) **da\_0\_deps\_HH**
- [Tenseur3HH](#) **da\_1\_deps\_HH**
- [Tenseur3HH](#) **da\_2\_deps\_HH**

La documentation de cette classe a été générée à partir du fichier suivant :

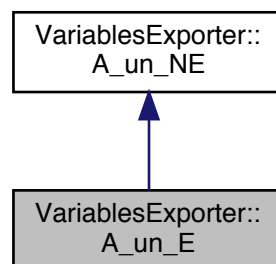
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.6 Référence de la classe VariablesExporter::A\_un\_E

Graphe d'héritage de VariablesExporter::A\_un\_E:



Graphe de collaboration de VariablesExporter::A\_un\_E:



## Fonctions membres publiques

- **A\_un\_E** (int absolu\_, string ref\_, string nom\_mail\_, string nom\_var\_, int nbpti)
- **A\_un\_E** (const **A\_un\_E** &a)
- **A\_un\_E** & **operator=** (const **A\_un\_E** &a)
- bool **operator==** (const **A\_un\_E** &a) const
- bool **operator!=** (const **A\_un\_E** &a) const
- bool **operator<** (const **A\_un\_E** &a) const
- bool **operator<=** (const **A\_un\_E** &a) const
- bool **operator>** (const **A\_un\_E** &a) const
- bool **operator>=** (const **A\_un\_E** &a) const
- void **Affiche** ()
- const **A\_un\_E** \* **PointeurClass\_E\_const** () const
- **A\_un\_E** \* **PointeurClass\_E** ()
- const int & **NBpti\_const** () const
- int & **NBpti** ()
- const int & **Absolue\_const** () const
- int & **Absolue** ()

## Attributs protégés

- int **num\_pti**
- int **absolu**

## Amis

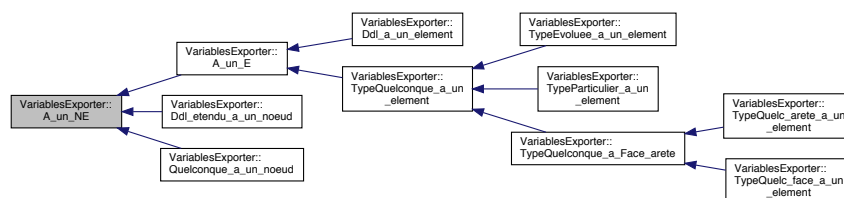
- istream & **operator>>** (istream &, **A\_un\_E** &)
- ostream & **operator<<** (ostream &, const **A\_un\_E** &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

## 6.7 Référence de la classe VariablesExporter::A\_un\_NE

Graphe d'héritage de VariablesExporter::A\_un\_NE:



## Fonctions membres publiques

- **A\_un\_NE** (string ref, string nom\_mail\_, string nom\_var\_)
- **A\_un\_NE** (const **A\_un\_NE** &a)
- **A\_un\_NE** & **operator=** (const **A\_un\_NE** &a)
- bool **operator==** (const **A\_un\_NE** &a) const
- bool **operator!=** (const **A\_un\_NE** &a) const
- bool **operator<** (const **A\_un\_NE** &a) const
- bool **operator<=** (const **A\_un\_NE** &a) const
- bool **operator>** (const **A\_un\_NE** &a) const
- bool **operator>=** (const **A\_un\_NE** &a) const
- void **Affiche** ()
- const string & **Ref\_const** () const
- const string & **Nom\_mail\_const** () const
- const string & **Nom\_var\_const** () const
- string & **Ref\_NE** ()
- string \* **Pointeur\_Ref\_NE** ()
- string & **Nom\_mail** ()
- string \* **Pointeur\_Nom\_mail** ()
- string & **Nom\_var** ()
- const **A\_un\_NE** \* **PointeurClass\_const** () const
- **A\_un\_NE** \* **PointeurClass** ()

## Attributs protégés

- string **ref**
- string **nom\_mail**
- string **nom\_var**

## Amis

- istream & **operator>>** (istream &, **A\_un\_NE** &)
- ostream & **operator<<** (ostream &, const **A\_un\_NE** &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

## 6.8 Référence de la classe Algo\_edp

BUT: Algorithmes de base pour la résolution d'équations différentielles.

```
#include <Algo_edp.h>
```

## Fonctions membres publiques

- **Algo\_edp** (const [Algo\\_edp](#) &a)
- template<class T >  
 void [Runge\\_Kutta\\_step23](#) (T &instance, [Vecteur](#) &(T::\*Ptder\_fonc)(const double &t, const [Vecteur](#) &f, [Vecteur](#) &df, int &erreur), void(T::\*Ptverif\_fonc)(const double &t, const [Vecteur](#) &f, int &erreur\_final), const [Vecteur](#) &val\_initiale, const [Vecteur](#) &der\_initiale, const double &t0, const double &deltat, [Vecteur](#) &val\_finale, [Vecteur](#) &estime\_erreur)  
*résolution par runge kutta imbriqué --> estimation d'erreur il y a 3 calcul de la fonction dérivée 2 et 3 ième ordre imbriquée calcul de la solution à t+dt en fonction de la solution à t, en entrée: \*Ptder\_fonc : pointeur de la fonction de calcul de la dérivées en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t erreur : si diff de 0, indique qu'il y a eu une erreur \*Ptverif\_fonc : pointeur de fonction, pour tester val\_finale après la prédiction explicite finale*
- template<class T >  
 void [Runge\\_Kutta\\_step34](#) (T &instance, [Vecteur](#) &(T::\*Ptder\_fonc)(const double &t, const [Vecteur](#) &f, [Vecteur](#) &df, int &erreur), void(T::\*Ptverif\_fonc)(const double &t, const [Vecteur](#) &f, int &erreur\_final), const [Vecteur](#) &val\_initiale, const [Vecteur](#) &der\_initiale, const double &t0, const double &deltat, [Vecteur](#) &val\_finale, [Vecteur](#) &estime\_erreur)  
*résolution par runge kutta imbriqué --> estimation d'erreur il y a 4 calcul de la fonction dérivée 3 et 4 ième ordre imbriquée (Fehlberg) calcul de la solution à t+dt en fonction de la solution à t, en entrée: \*Ptder\_fonc : pointeur de la fonction de calcul de la dérivées en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t erreur : si diff de 0, indique qu'il y a eu une erreur \*Ptverif\_fonc : pointeur de fonction, pour tester val\_finale après la prédiction explicite finale*
- template<class T >  
 void [Runge\\_Kutta\\_step45](#) (T &instance, [Vecteur](#) &(T::\*Ptder\_fonc)(const double &t, const [Vecteur](#) &f, [Vecteur](#) &df, int &erreur), void(T::\*Ptverif\_fonc)(const double &t, const [Vecteur](#) &f, int &erreur\_final), const [Vecteur](#) &val\_initiale, const [Vecteur](#) &der\_initiale, const double &t0, const double &deltat, [Vecteur](#) &val\_finale, [Vecteur](#) &estime\_erreur)  
*résolution par runge kutta imbriqué --> estimation d'erreur il y a 5 calcul de la fonction dérivée l'algorithme s'appuie sur la présentation de numerical recipes fifth-order Cash-Karp Runge-Kutta calcul de la solution à t+dt en fonction de la solution à t, en entrée: \*Ptder\_fonc : pointeur de la fonction de calcul de la dérivées en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t erreur : si diff de 0, indique qu'il y a eu une erreur \*Ptverif\_fonc : pointeur de fonction, pour tester val\_finale après la prédiction explicite finale*
- template<class T >  
 int [Pilotage\\_kutta](#) (int cas\_kutta, T &instance, [Vecteur](#) &(T::\*Ptder\_fonc)(const double &t, const [Vecteur](#) &f, [Vecteur](#) &df, int &erreur), void(T::\*Ptverif\_fonc)(const double &t, const [Vecteur](#) &f, int &erreur\_final), const [Vecteur](#) &val\_initiale, const [Vecteur](#) &der\_initiale, const double &tdebut, const double &tfin, const double &erreurAbsolue, double &erreurRelative, [Vecteur](#) &val\_finale, [Vecteur](#) &der\_finale, double &dernierTemps, double &dernierdeltat, int &nombreAppelF, int &nb\_step, double &erreur\_maxi\_global)  
*un programme de pilotage de l'intégration d'un pas (on s'inspire du programme libre rkf45) on distingue une erreur globale et une erreur relative l'erreur réelle utilisée sur chaque step est: erreurRelative \* |f| + erreurAbsolue ==== en entrée: ===== cas\_kutta : donne le type de routine kutta imbriqué a employer: =3 pour kutta23, =4 pour kutta34, =5 pour kutta45 tdebut et tfin : temps de début et de fin du calcul, demandé val\_initiale : valeur initiale de la fonction (donc à tdebut) der\_initiale : dérivée initiale de la fonction (donc à tdebut) erreurAbsolue : erreur absolue demandée erreurRelative : erreur relative (à la fonction) demandée Ptder\_fonc : pointeur de fonction en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t erreur : si diff de 0, indique qu'il y a eu une erreur Ptverif\_fonc : pointeur de fonction, pour tester l'intégrité du résultat cela signifie que val\_finale est testée avec cette fonction, systématiquement*
- void [Affiche\\_Explication\\_erreur\\_RG](#) (int type\_erreur)  
*explication erreur affiche à l'écran l'explication de l'erreur de Pilotage\_kutta*
- void [Modif\\_nbMaxiAppel](#) (int nb)  
*—méthodes pour modifier les différents paramètres nombre maxi d'appel de fonction permis dans le cas du runge 4-5 -> 6 appels par step (à la louche)*
- void [Init\\_param\\_val\\_defaut](#) ()  
*initialisation de tous les paramètres à leurs valeurs par défaut*
- void [Change\\_niveau\\_affichage](#) (int niveau)  
*modifie le niveau d'affichage par exemple pour le debug*

## 6.8.1 Description détaillée

BUT: Algorithmes de base pour la résolution d'équations différentielles.

## 6.8.2 Documentation des fonctions membres

### 6.8.2.1 Pilotage\_kutta()

```

template<class T >
int Algo_edp::Pilotage_kutta (
    int cas_kutta,
    T & instance,
    Vecteur &(T::*)(const double &t, const Vecteur &f, Vecteur &df, int &erreur)
    Ptder_fonc,
    void(T::*)(const double &t, const Vecteur &f, int &erreur_final) Ptverif_fonc,
    const Vecteur & val_initiale,
    const Vecteur & der_initiale,
    const double & tdebut,
    const double & tfin,
    const double & erreurAbsolue,
    double & erreurRelative,
    Vecteur & val_finale,
    Vecteur & der_finale,
    double & dernierTemps,
    double & dernierdeltat,
    int & nombreAppelF,
    int & nb_step,
    double & erreur_maxi_global )

```

un programme de pilotage de l'intégration d'un pas (on s'inspire du programme libre rkf45) on distingue une erreur globale et une erreur relative l'erreur réelle utilisée sur chaque step est:  $\text{erreurRelative} * |f| + \text{erreurAbsolue}$  en entrée: ===== cas\_kutta : donne le type de routine kutta imbriqué a employer: =3 pour kutta23, =4 pour kutta34, =5 pour kutta45 tdebut et tfin : temps de début et de fin du calcul, demandé val\_initiale : valeur initiale de la fonction (donc à tdebut) der\_initiale : dérivée initiale de la fonction (donc à tdebut) erreurAbsolue : erreur absolue demandée erreurRelative : erreur relative (à la fonction) demandée Ptder\_fonc : pointeur de fonction en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t erreur : si diff de 0, indique qu'il y a eu une erreur Ptverif\_fonc : pointeur de fonction, pour tester l'intégrité du résultat cela signifie que val\_finale est testée avec cette fonction, systématiquement

- à la fin de chaque prédiction des kuttas imbriqués
- au retour du pilotage — en sortie:
  - dernierTemps : temps final utilisé = tfin si calcul ok, sinon = le dernier temps calculé val\_finale : valeur de la fonction à "dernierTemps" der\_finale : dérivée finale dernierdeltat : le dernier deltat utilisé nombreAppelF : nombre d'appel de la fonction réellement utilisé dans le programme
  - erreurRelative : si retour = 3 ==> erreur relative (à la fonction) minimum possible, nb\_step : nombre de step utilisé pour le calcul (non compté les steps avec trop d'erreur) erreur\_maxi\_global : une approximation de l'erreur maxi global cumulant les erreurs sur tous les steps === en retour un entier qui donne le résultat du calcul : ===== =2 : tout est ok : seul cas où toutes les valeurs de retour sont valides =3 : erreur: la précision relative demandée est trop petite, en retour la précision mini possible =4 : erreur: on a dépassé le nombre maxi d'appel fixé, d'où a peu près à (nombreAppelF/6) step de calcul. =6 : erreur: l'intégration pas possible, due aux précisions demandées, on doit augmenter ces précisions souvent du à une solution qui varie très rapidement --> problème potentiel =8 : erreur: cas\_kutta ne correspond pas à une routine de base existante = 0 : erreur inconnue

### 6.8.2.2 Runge\_Kutta\_step23()

```
template<class T >
void Algo_edp::Runge_Kutta_step23 (
    T & instance,
    Vecteur &(T::*)(const double &t, const Vecteur &f, Vecteur &df, int &erreur)
    Ptder_fonc,
    void(T::*)(const double &t, const Vecteur &f, int &erreur_final) Ptverif_fonc,
    const Vecteur & val_initiale,
    const Vecteur & der_initiale,
    const double & t0,
    const double & deltat,
    Vecteur & val_finale,
    Vecteur & estime_erreur )
```

résolution par runge kutta imbriqué --> estimation d'erreur il y a 3 calcul de la fonction dérivée 2 et 3 ième ordre imbriquée calcul de la solution à t+dt en fonction de la solution à t, en entrée: \*Ptder\_fonc : pointeur de la fonction de calcul de la dérivées en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t erreur : si diff de 0, indique qu'il y a eu une erreur \*Ptverif\_fonc : pointeur de fonction, pour tester val\_finale après la prédiction explicite finale

val\_initiale : valeur des fonctions en t0 der\_initiale : valeur de la dérivée pour t0 t0 : temps initiale deltat : incrément de temps demandé en sortie : val\_finale : valeur des fonctions à tdt estime\_erreur : estimation d'erreur pour chaque fonction --> si estime\_erreur(i) est >= à ConstMath::tresgrand, la valeur finale = la valeur initiale (c-a-d pas de calcul valide)

### 6.8.2.3 Runge\_Kutta\_step34()

```
template<class T >
void Algo_edp::Runge_Kutta_step34 (
    T & instance,
    Vecteur &(T::*)(const double &t, const Vecteur &f, Vecteur &df, int &erreur)
    Ptder_fonc,
    void(T::*)(const double &t, const Vecteur &f, int &erreur_final) Ptverif_fonc,
    const Vecteur & val_initiale,
    const Vecteur & der_initiale,
    const double & t0,
    const double & deltat,
    Vecteur & val_finale,
    Vecteur & estime_erreur )
```

résolution par runge kutta imbriqué --> estimation d'erreur il y a 4 calcul de la fonction dérivée 3 et 4 ième ordre imbriquée (Fehlberg) calcul de la solution à t+dt en fonction de la solution à t, en entrée: \*Ptder\_fonc : pointeur de la fonction de calcul de la dérivées en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t erreur : si diff de 0, indique qu'il y a eu une erreur \*Ptverif\_fonc : pointeur de fonction, pour tester val\_finale après la prédiction explicite finale

val\_initiale : valeur des fonctions en t0 der\_initiale : valeur de la dérivée pour t0 t0 : temps initiale deltat : incrément de temps demandé en sortie : val\_finale : valeur des fonctions à tdt estime\_erreur : estimation d'erreur pour chaque fonction --> si estime\_erreur(i) est >= à ConstMath::tresgrand, la valeur finale = la valeur initiale (c-a-d pas de calcul valide)

### 6.8.2.4 Runge\_Kutta\_step45()

```
template<class T >
void Algo_edp::Runge_Kutta_step45 (
    T & instance,
    Vecteur &(T::*)(const double &t, const Vecteur &f, Vecteur &df, int &erreur)
    Ptder_fonc,
    void(T::*)(const double &t, const Vecteur &f, int &erreur_final) Ptverif_fonc,
    const Vecteur & val_initiale,
    const Vecteur & der_initiale,
    const double & t0,
    const double & deltat,
    Vecteur & val_finale,
    Vecteur & estime_erreur )
```

résolution par runge kutta imbriqué --> estimation d'erreur il y a 5 calcul de la fonction dérivée l'algorithme s'appuie sur la présentation de numerical recipes fifth-order Cash-Karp Runge-Kutta calcul de la solution à t+dt en fonction de la solution à t, en entrée: \*Ptder\_fonc : pointeur de la fonction de calcul de la dérivées en entrée: t et f : temps et la valeur de la fonction a t en sortie: df : dérivée de la fonction a t erreur : si diff de 0, indique qu'il y a eu une erreur \*Ptverif\_fonc : pointeur de fonction, pour tester val\_finale après la prédiction explicite finale

val\_initiale : valeur des fonctions en t0 der\_initiale : valeur de la dérivée pour t0 t0 : temps initiale deltat : incrément de temps demandé en sortie : val\_finale : valeur des fonctions à tdt estime\_erreur : estimation d'erreur pour chaque fonction --> si estime\_erreur(i) est >= à ConstMath::tresgrand, la valeur finale = la valeur initiale (c-a-d pas de calcul valide)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo\_edp.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo\_edp.cc

## 6.9 Référence de la classe Algo\_Integ1D

Algorithme d'intégration 1D.

```
#include <Algo_Integ1D.h>
```

### Fonctions membres publiques

- **Algo\_Integ1D** (int nbptGauss=2)  
*constructeur par défaut nbptGauss : nombre de point de Gauss, utilisé pour l'intégration*
- **Algo\_Integ1D** (const [Algo\\_Integ1D](#) &a)  
*constructeur de copie*
- **~Algo\_Integ1D** ()  
*DESTRUCTEUR :*
- template<class T >  
double **IntegGauss** (const double &tfin, T &instance, double(T::\*Pt\_fonc)(const double &t), const double &deltat)  
*intégration d'une fonction à l'aide de la méthode de Gauss*
- template<class T >  
double **IntegGauss** (T &instance, double(T::\*Pt\_fonc)(const double &theta), const double &deltat)  
*intégration d'une fonction à l'aide de la méthode de Gauss mais ici avec un intervalle fixé de -1 à 1 en entrée: \*Pt\_fonc : pointeur de la fonction, qui est fonction d'une valeur qui doit varier de -1 à 1 (donc il faut faire les interpolations nécessaires) deltat : plage d'intégration en sortie : renvoie la valeur de l'intégrale de t0 à t0+deltat*



## 6.9.1 Description détaillée

Algorithme d'intégration 1D.

BUT: Algorithme d'intégration 1D

### Auteur

Gérard Rio

### Version

1.0

### Date

14/03/2008

## 6.9.2 Documentation des fonctions membres

### 6.9.2.1 IntegGauss()

```
template<class T >
double Algo_Integ1D::IntegGauss (
    const double & tfin,
    T & instance,
    double(T::*)(const double &t) Pt_fonc,
    const double & deltat )
```

intégration d'une fonction à l'aide de la méthode de Gauss

en entrée: \*Pt\_fonc : pointeur de la fonction, tfin : temps finale deltat : plage d'intégration en sortie : renvoie la valeur de l'intégrale de t0 à t0+deltat

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo\_Integ1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo\_Integ1D.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo\_Integ1D\_2.cc

## 6.10 Référence de la classe Algo\_zero

Algorithmes de base pour la recherche de zéro d'une fonction ou d'un ensemble de fonctions.

```
#include <Algo_zero.h>
```

## Fonctions membres publiques

- **Algo\_zero** ()  
*constructeur par défaut*
- **Algo\_zero** (const [Algo\\_zero](#) &a)  
*constructeur de copie*
- **~Algo\_zero** ()  
*DESTRUCTEUR :*
- void **SecondDegre** (double a, double b, double c, double &racine1, double &racine2, int &cas) const  
*recherche du zéro d'une équation du second degré dans l'espace des réels:  $ax^2+bx+c=0$ . cas indique les différents cas: = 1 : il y a deux racines ,  $racine2 > racine1 = 2$  : il y a une racine double = 0 : pas de racine = -1 : pas de racine car a,b,c sont tous inférieur à la précision de traitement = -2 : pas de racine car a,b sont inférieur à la précision de traitement tandis que c est non nul = -3 : une racine simple car a est inférieur à la précision de traitement, donc considéré comme nul b et c sont non nul -> equa du premier degré, solution: racine1*
- void **TroisiemeDegre** (double a, double b, double c, double d, double &racine1, double &racine2, double &racine3, int &cas)  
*recherche du zéro d'une équation du troisième degré dans l'espace des réels:  $ax^3+bx^2+cx+d=0$ . cas indique les différents cas: = 1 : il y a deux racines ,  $racine2 > racine1 = 2$  : il y a une racine double = 3 : il y a une seule racine réelle racine1 (et deux complexes non calculées) = 4 : il y a une racine simple: racine1, et une racine double: racine2 = 5 : il y a 3 racines réelles: racine1, racine2, racine3 = 0 : pas de racine = -1 : pas de racine car a,b,c sont tous inférieur à la précision de traitement = -2 : pas de racine car a,b sont inférieur à la précision de traitement tandis que c est non nul = -3 : une racine simple car a est inférieur à la précision de traitement, donc considéré comme nul b et c sont non nul -> equa du premier degré, solution: racine1*
- template<class T >  
bool **Newton\_raphson** (T &instance, double(T::\*Ptfonc)(double &alpha, double &x, int &test), double(T::\*Ptder\_fonc)(double &alpha, double &x, int &test), double val\_initiale, double &racine, double &der\_at\_racine, int &nb\_incr\_total, int &nb\_iter\_total, double max\_delta\_x) const  
*recherche du zéro d'une fonction en utilisant la méthode de Newton-Raphson incrémentale ici il s'agit d'une fonction à une variable \*Ptfonc : le pointeur de la fonction dont il faut chercher le zéro \*Ptder\_fonc : pointeur de la dérivée de la fonction pour ces deux fonctions, la variable alpha est un facteur de charge qui varie de 0 à 1. Lorsque alpha=0, la racine vaut val\_initiale, et ce que l'on cherche c'est la racine pour alpha = 1. Lorsque alpha varie progressivement de 0 à 1, la racine est sensée varier progressivement de val\_initiale à résidu l'utilisation d'alpha permet de faire du Newton incrémentale. pour ces deux fonctions, l'argument test ramène . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb fatal, qui invalide le calcul de la fonction et/ou de la dérivée val\_initiale : une valeur initiale de x pour la recherche de zéro max\_delta\_x : si > 0 donne le norme maxi permise sur delta\_x au cours d'une itération si ||delta\_x|| est plus grand on fait delta\_x = max\_delta\_x \* delta\_x / || delta\_x|| ramène en sortie: un booléen qui indique si la résolution est correcte ou non racine : la racine trouvée der\_at\_racine : contient en retour la valeur de la dérivée pour la racine trouvée nb\_incr\_total : le nombre total d'incrément qui a été nécessaire nb\_iter\_total : le nombre total d'itération, qui cumule les iter de tous les incréments*
- template<class T >  
bool **Newton\_raphson** (T &instance, [Vecteur](#) &(T::\*Ptfonc)(const double &alpha, const [Vecteur](#) &x, int &test), [Mat\\_abstraite](#) &(T::\*Ptder\_fonc)(const double &alpha, const [Vecteur](#) &x, int &test), const [Vecteur](#) &val\_initiale, [Vecteur](#) &racine, [Mat\\_abstraite](#) &der\_at\_racine, int &nb\_incr\_total, int &nb\_iter\_total, double max\_delta\_x) const  
*idem précédemment, mais pour une fonction à valeur vectorielle les matrices sont telles que:  $der\_at\_racine(i,j) = df(i)/d(x(j))$ , soit:  $df = der\_at\_racine * dx$*
- template<class T >  
bool **Newton\_raphson** (T &instance, [Vecteur](#) &(T::\*Ptfonc)(const double &alpha, const [Vecteur](#) &x, int &test), [Mat\\_abstraite](#) &(T::\*Ptder\_fonc)(const double &alpha, const [Vecteur](#) &x, [Vecteur](#) &res, int &test), const [Vecteur](#) &val\_initiale, [Vecteur](#) &racine, [Mat\\_abstraite](#) &der\_at\_racine, int &nb\_incr\_total, int &nb\_iter\_total, double max\_delta\_x) const  
*idem précédemment, mais pour une fonction à valeur vectorielle et .. la méthode externe calcul la valeur de la fonction et de la dérivée en même temps mais on utilise également la fonction résidu toute seule les matrices sont telles que:  $der\_at\_racine(i,j) = df(i)/d(x(j))$ , soit:  $df = der\_at\_racine * dx$*
- void **Modif\_prec\_res\_abs** (double eps)  
*méthodes pour modifier les différents paramètres précision sur le résidu à convergence*
- void **Modif\_prec\_res\_rel** (double eps)  
*précision absolue*
- void **Modif\_iter\_max** (double eps)  
*précision relative nombre d'itération maxi pour un incrément*
- void **Modif\_coef\_mini\_delta\_x** (double eps)

- *précision relative sur le mini de l'incrément de x*
- void **Modif\_mini\_delta\_x** (double eps)  
*minimum de delta x permis*
- void **Modif\_maxi\_delta\_x** (double eps)  
*maximum de delta x permis*
- void **Modif\_nbMaxilncre** (int nb)  
*nombre maxi d'incrément permis*
- void **Modif\_nbMaxi\_test\_moins1** (int nb)  
*nombre maxi de test -1 permis*
- void **Init\_param\_val\_defaut** ()  
*initialisation de tous les paramètres à leurs valeurs par défaut*
- void **Modif\_affichage** (int niveau)  
*le niveau d'affichage*
- const double & **Prec\_res\_abs** () const  
*récup en lecture des différents paramètres*
- const double & **Prec\_res\_rel** () const  
*précision absolue sur le résidu à convergence*
- const int & **Iter\_max** () const  
*précision relative sur le résidu à convergence*
- const double & **Coef\_mini\_delta\_x** () const  
*nombre d'itération maxi pour un incrément*
- const double & **Mini\_delta\_x** () const  
*précision relative sur le mini de l'incrément de x*
- const double & **Maxi\_delta\_x** () const  
*minimum de delta x permis*
- const int & **NbMaxilncre** () const  
*maximum de delta x permis*
- const int & **Maxi\_test\_moins1** () const  
*nombre maxi d'incrément permis*
- const int & **Permet\_affichage** () const  
*nombre maxi de test -1 permis*
- void **Affiche** () const  
*le niveau d'affichage*
- void **Lecture\_base\_info** (ifstream &ent, const int cas)  
*--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*

### 6.10.1 Description détaillée

Algorithmes de base pour la recherche de zéro d'une fonction ou d'un ensemble de fonctions.

BUT: Algorithmes de base pour la recherche de zéro d'une fonction ou d'un ensemble de fonctions.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

11/10/2003

## 6.10.2 Documentation des fonctions membres

### 6.10.2.1 Affiche()

```
void Algo_zero::Affiche ( ) const
```

le niveau d'affichage

affichage à l'écran des infos

La documentation de cette classe a été générée à partir du fichier suivant :

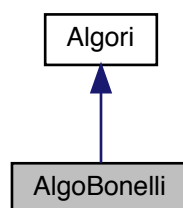
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo\_zero.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo\_zero.cc

## 6.11 Référence de la classe AlgoBonelli

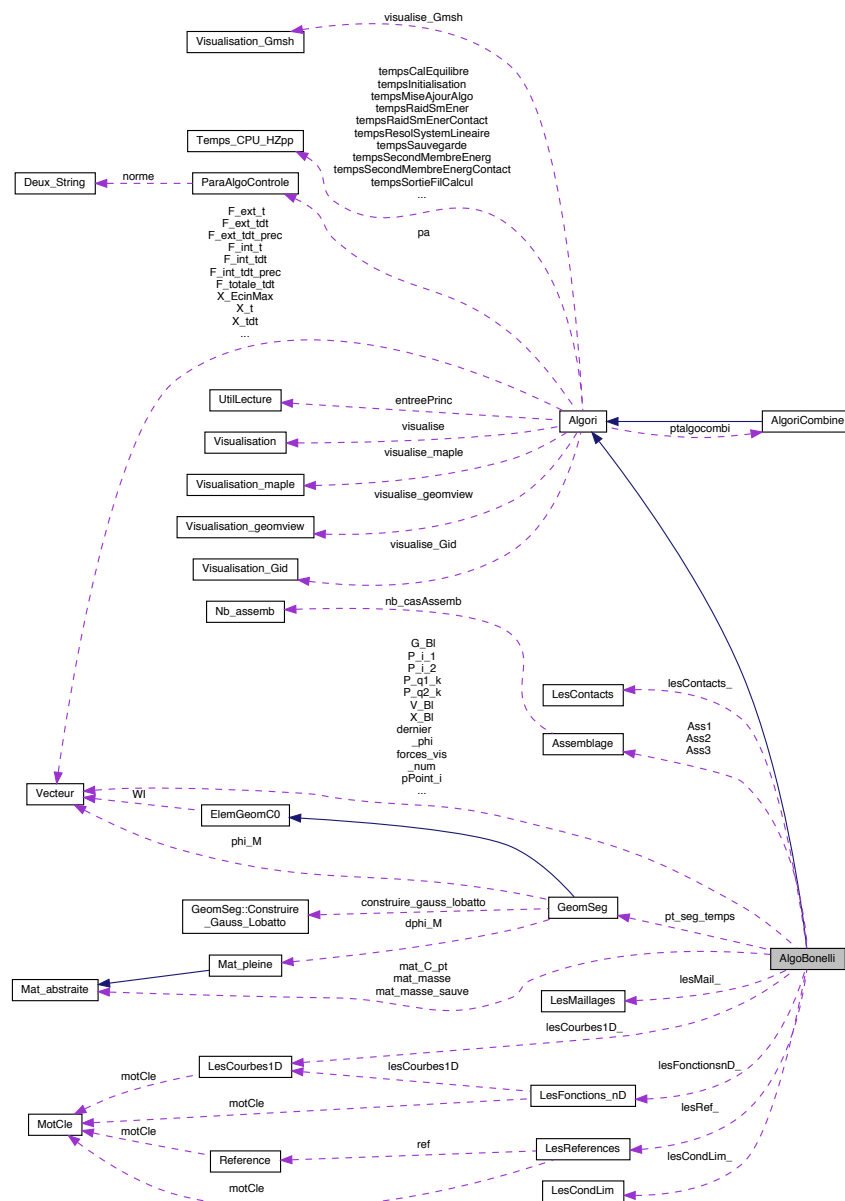
BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées. Correspond à l'implantation de l'algorithme de Bonelli et Bursi. Le modèle est de type Galerkin discontinu en temps P1-P1, pour la discrétisation. L'algorithme est de type prédiction suivi de 1 ou plusieurs cycles de correction. Il est explicite, dans le sens où on n'utilise pas de comportement tangent du matériau et que l'on contrôle exactement le nombre d'itération du processus.

```
#include <AlgoBonelli.h>
```

Graphique d'héritage de AlgoBonelli:



Graphe de collaboration de AlgoBonelli:



## Fonctions membres publiques

- `AlgoBonelli` (const bool avec `_typeDeCal`, const list< `EnumSousTypeCalcul` > & `soustype`, const list< bool > & `avec_soustypeDeCal`, `UtilLecture` & `entreePrinc`)
- `AlgoBonelli` (const `AlgoBonelli` & `algo`)
- `Algori` \* `New_idem` (const `Algori` \* `algo`) const
- void `Execution` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \* `varExp`, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `InitAlgorithme` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)

- void `MiseAJourAlgo` (`ParaGlob *`, `LesMaillages *`, `LesReferences *`, `LesCourbes1D *`, `LesFonctions_nD *`, `VariablesExporter *`, `LesLoisDeComp *`, `DiversStockage *`, `Charge *`, `LesCondLim *`, `LesContacts *`, `Resultats *`)
- void `CalEquilibre` (`ParaGlob *`, `LesMaillages *`, `LesReferences *`, `LesCourbes1D *`, `LesFonctions_nD *`, `VariablesExporter *`, `LesLoisDeComp *`, `DiversStockage *`, `Charge *`, `LesCondLim *`, `LesContacts *`, `Resultats *`, `Tableau< Fonction_nD * > *tb_combiner`)
- void `FinCalcul` (`ParaGlob *`, `LesMaillages *`, `LesReferences *`, `LesCourbes1D *`, `LesFonctions_nD *`, `VariablesExporter *`, `LesLoisDeComp *`, `DiversStockage *`, `Charge *`, `LesCondLim *`, `LesContacts *`, `Resultats *`)
- void `SchemaXML_Algori` (`ofstream &sort`, `const Enum_IO_XML enu`) `const`

## Types protégés

- enum `enuTypePhase` { `PREDICTION_bonelli`, `INTEGRATION_bonelli`, `CORRECTION_bonelli`, `FIN_↔ DELTA_T_bonelli` }

## Fonctions membres protégées

- void `lecture_Parametres` (`UtilLecture &entreePrinc`)
- void `Ecrit_Base_info_Parametre` (`UtilLecture &entreePrinc`, `const int &cas`)
- void `Lecture_Base_info_Parametre` (`UtilLecture &entreePrinc`, `const int &cas`, `bool choix`)
- void `Info_commande_parametres` (`UtilLecture &entreePrinc`)
- void `Gestion_pas_de_temps` (`LesMaillages *lesMail`, `int cas`, `int nbstep`)
- void `Modif_transi_pas_de_temps` (`double delta_tau`)
- void `AvanceDDL_avec_CL` (`const Vecteur &phii`, `enuTypePhase phasage`)
- void `CalEnergieAffichageBonelli` (`const double &coef_mass`, `int icharge`)
- bool `ActionInteractiveAlgo` ()

## Attributs protégés

- double `a_a`
- double `b_b`
- int `k_max`
- double `omega_b`
- double `rho_b`
- `GeomSeg * pt_seg_temps`
- int `nb_pt_int_t`
- double `delta_t`
- double `deltat2`
- double `unsurdeltat`
- double `deltatSurDeux`
- double `delta_t_total`
- double `deltat2_total`
- double `unsurdeltat_total`
- double `deltatSurDeux_total`
- `LesMaillages * lesMail_`
- `LesReferences * lesRef_`
- `LesCourbes1D * lesCourbes1D_`
- `LesFonctions_nD * lesFonctionsnD_`
- `Charge * charge_`
- `LesCondLim * lesCondLim_`
- `LesContacts * lesContacts_`
- `Assemblage * Ass1`
- `Assemblage * Ass2`
- `Assemblage * Ass3`
- double `maxPuissExt`
- double `maxPuissInt`
- double `maxReaction`

- int **inReaction**
- int **inSol**
- double **maxDeltaDdl**
- int **cas\_combi\_ddl**
- int **icas**
- bool **erreurSecondMembre**
- bool **prepa\_avec\_remont**
- bool **brestart**
- `OrdreVisu::EnumTypeIncre` **type\_incre**
- `Vecteur` **q\_0**
- `Vecteur` **p\_0**
- `Vecteur` **q\_1**
- `Vecteur` **p\_1**
- `Vecteur` **q\_2**
- `Vecteur` **p\_2**
- `Vecteur` **pPoint\_i**
- `Vecteur` **P\_i\_1**
- `Vecteur` **P\_i\_2**
- `Vecteur` **P\_q1\_k**
- `Vecteur` **P\_q2\_k**
- `Vecteur` **r\_1\_k**
- `Vecteur` **r\_2\_k**
- `Vecteur` **vec\_travail1**
- `Vecteur` **vec\_travail2**
- `Vecteur` **dernier\_phi**
- `Vecteur` **vglobin**
- `Vecteur` **vglobex**
- `Vecteur` **vglobaal**
- `Vecteur` **vcontact**
- `Vecteur` **vitesse\_tplus**
- `Vecteur` **X\_BI**
- `Vecteur` **V\_BI**
- `Vecteur` **G\_BI**
- `Vecteur` **forces\_vis\_num**
- `list< LesCondLim::Gene_asso >` **li\_gene\_asso**
- `Tableau< Nb_assemb >` **t\_assemb**
- `Tableau< Enum_ddl >` **tenuXVG**
- `Mat_abstraite *` **mat\_masse**
- `Mat_abstraite *` **mat\_masse\_sauve**
- `Mat_abstraite *` **mat\_C\_pt**

## Membres hérités additionnels

### 6.11.1 Description détaillée

BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées. Correspond à l'implantation de l'algorithme de Bonelli et Bursi. Le modèle est de type galerkin discontinu en temps P1-P1, pour la discrétisation. L'algorithme est de type prédiction suivi de 1 ou plusieurs cycles de correction. Il est explicite, dans le sens où on n'utilise pas de comportement tangent du matériau et que l'on contrôle exactement le nombre d'itération du processus.

### 6.11.2 Documentation des fonctions membres

### 6.11.2.1 CalEquilibre()

```
void AlgoBonelli::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

Implémente [Algori](#).

### 6.11.2.2 Ecrit\_Base\_info\_Parametre()

```
void AlgoBonelli::Ecrit_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.11.2.3 Execution()

```
void AlgoBonelli::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).



#### 6.11.2.4 FinCalcul()

```
void AlgoBonelli::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.11.2.5 Info\_commande\_parametres()

```
void AlgoBonelli::Info_commande_parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Implémente [Algori](#).

#### 6.11.2.6 InitAlgorithme()

```
void AlgoBonelli::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.11.2.7 Lecture\_Base\_info\_Parametre()

```
void AlgoBonelli::Lecture_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas,
    bool choix ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.11.2.8 lecture\_Parametres()

```
void AlgoBonelli::lecture_Parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Réimplémentée à partir de [Algori](#).

### 6.11.2.9 MiseAJourAlgo()

```
void AlgoBonelli::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.11.2.10 New\_idem()

```
Algori * AlgoBonelli::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.11.2.11 SchemaXML\_Algori()

```
void AlgoBonelli::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

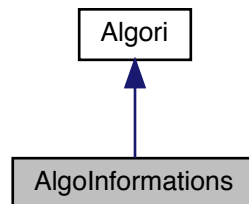
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinDiscontinu/DG\_DynaExplicite/Algo↔Bonelli.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinDiscontinu/DG\_DynaExplicite/Algo↔Bonelli.cc

## 6.12 Référence de la classe AlgoInformations

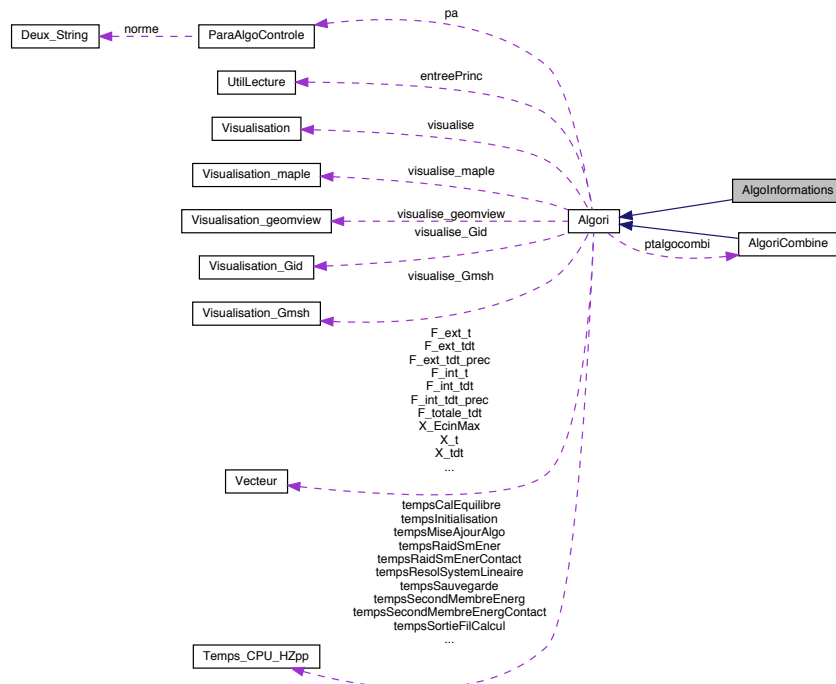
BUT: Calcul d'informations générales types, géométriques par exemple, ou de visualisation.

```
#include <AlgoInformations.h>
```

Grphe d'héritage de AlgoInformations:



Graphe de collaboration de AlgoInformations:



## Fonctions membres publiques

- **AlgoInformations** (const bool avec\_typeDeCal, const list< EnumSousTypeCalcul > &soustype, const list< bool > &avec\_soustypeDeCal, UtilLecture &entreePrinc)
- **AlgoInformations** (const AlgoInformations &algo)
- **Algori \* New\_idem** (const Algori \*algo) const
- void **Execution** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*varExpor, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **InitAlgorithme** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **MiseAJourAlgo** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **CalEquilibre** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*, Tableau< Fonction\_nD \* > \*tb\_combiner)
- void **FinCalcul** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **SchemaXML\_Algori** (ofstream &sort, const Enum\_IO\_XML enu) const

## Membres hérités additionnels

### 6.12.1 Description détaillée

BUT: Calcul d'informations générales types, géométriques par exemple, ou de visualisation.

## 6.12.2 Documentation des fonctions membres

### 6.12.2.1 CalEquilibre()

```
void AlgoInformations::CalEquilibre (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    Tableau< Fonction_nD * > * tb_combiner ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.12.2.2 Execution()

```
void AlgoInformations::Execution (
    ParaGlob * p,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.12.2.3 FinCalcul()

```
void AlgoInformations::FinCalcul (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.12.2.4 InitAlgorithme()

```
void AlgoInformations::InitAlgorithme (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.12.2.5 MiseAJourAlgo()

```
void AlgoInformations::MiseAJourAlgo (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.12.2.6 New\_idem()

```
Algori * AlgoInformations::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.12.2.7 SchemaXML\_Algori()

```
void AlgoInformations::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [inline], [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

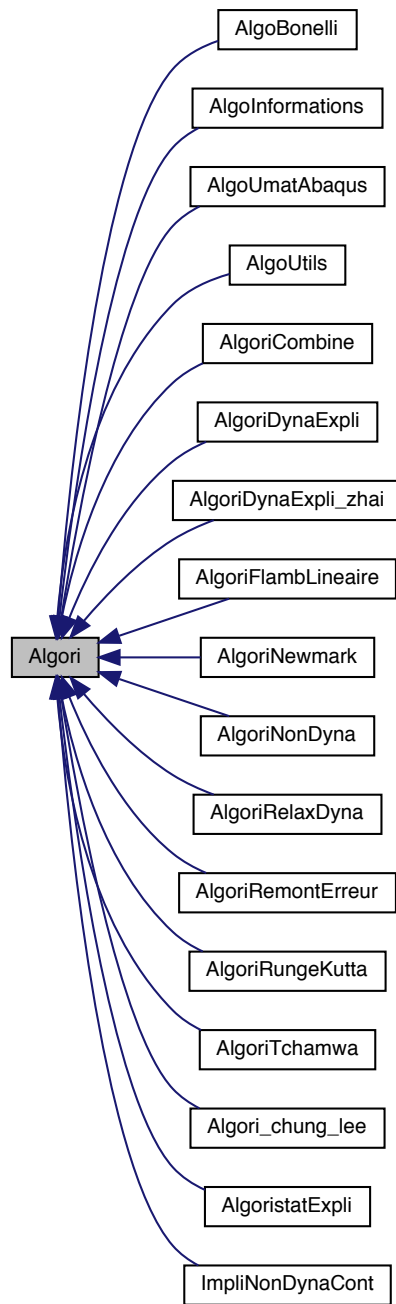
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/AlgoInformations.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/AlgoInformations.cc

## 6.13 Référence de la classe Algori

BUT: Classe de base de Defintion des differents algoritmes de resolution.

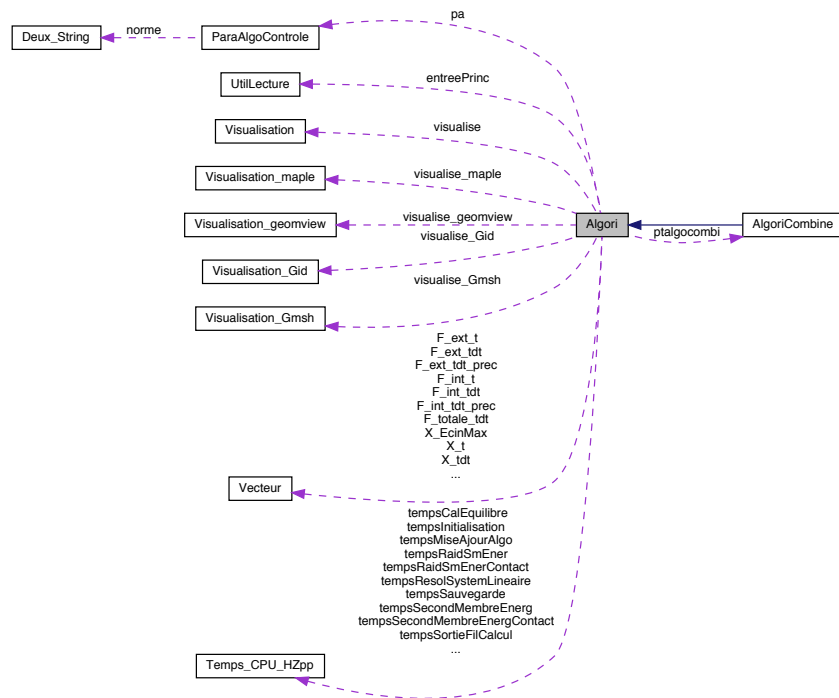
```
#include <Algori.h>
```

Graphe d'héritage de Algori:





Graphe de collaboration de Algori:



## Classes

- class [DeuxString](#)
- class [ListDeuxString](#)

## Fonctions membres publiques

- **Algori** ([EnumTypeCalcul](#) type, const bool avec\_typeDeCalcul, const list< [EnumSousTypeCalcul](#) > &sous-type, const list< bool > &avec\_soustypeDeCalcul, [UtilLecture](#) &entreePrinc)
- **Algori** (const [Algori](#) &algo)
- virtual [Algori](#) \* **New\_idem** (const [Algori](#) \*algo) const =0
- virtual void **Lecture** ([UtilLecture](#) &entreePrinc, [ParaGlob](#) &paraGlob, [LesMaillages](#) &lesMail)
- void **Coherence\_Algo\_typeConvergence** ()
- int **NbCasAssemblage** () const
- virtual void **Execution** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)=0
- virtual void **InitAlgorithme** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)=0
- virtual void **MiseAJourAlgo** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)=0
- virtual void **CalEquilibre** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [Tableau](#)< [Fonction\\_nD](#) \* > \*tb\_combiner)=0
- virtual void **FinCalcul** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)=0

- virtual void [SchemaXML\\_Alogri](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const =0
- void [Affiche](#) () const
- void [Affiche1](#) () const
- void [Affiche2](#) () const
- void [Info\\_commande\\_ParaAlgoControle](#) ([UtilLecture](#) &lec)
- void [InfoIncrementDdl](#) ([LesMaillages](#) \*lesMail, int inSol, double maxDeltaDdl, const [Nb\\_assemb](#) &nb\_↵  
casAssemb)
- void [InfoIncrementReac](#) ([LesMaillages](#) \*lesMail, int compteur, int inreaction, double maxreaction, const  
[Nb\\_assemb](#) &nb\_casAssemb)
- void [InfoIncrementReac](#) ([LesMaillages](#) \*lesMail, int inreaction, double maxreaction, const [Nb\\_assemb](#)  
&nb\_casAssemb)
- [EnumTypeCalcul TypeDeCalcul](#) () const
- void [DefFlotExterne](#) ([UtilLecture](#) \*entreePrinc)
- bool [ExisteDonneesExternes](#) () const
- int [NbTypeFlotExt](#) () const
- const [ListDeuxString](#) & [Noms\\_fichier](#) (int nbf) const
- const int [TypeDonneesExternes](#) (int nbf) const
- const int [Num\\_restart](#) () const
- const int [Num\\_increment](#) () const
- void [Affectation\\_icharge](#) (int icharge\_)
- const bool [Active\\_sauvegarde](#) () const
- virtual void [Ecriture\\_base\\_info](#) (int cas, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences,  
[LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD, [LesLoisDeComp](#) \*lesLoisDeComp,  
[DiversStockage](#) \*diversStockage, [Charge](#) \*charge, [LesCondLim](#) \*lesCondlim, [LesContacts](#) \*lesContacts,  
[Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre=0)
- virtual void [Lecture\\_base\\_info](#) (int cas, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences,  
[LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD, [LesLoisDeComp](#) \*lesLoisDeComp,  
[DiversStockage](#) \*diversStockage, [Charge](#) \*charge, [LesCondLim](#) \*lesCondlim, [LesContacts](#) \*lesContacts,  
[Resultats](#) \*resultats, int inc\_voulu=0)
- virtual void [Visu\\_vrml](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#)  
\*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- virtual void [Visu\\_maple](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#)  
\*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- virtual void [Visu\\_geomview](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*,  
[LesFonctions\\_nD](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*,  
[Resultats](#) \*)
- virtual void [Visu\\_Gid](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*,  
[LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- virtual void [Visu\\_Gmsh](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#)  
\*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void [LectureCommandeVisu](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*,  
[LesFonctions\\_nD](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*,  
[Resultats](#) \*)
- void [VisuAuFiIDuCalcul](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#)  
\*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [OrdreVisu](#)↵  
::[EnumTypeIncre](#) &type\_incre, int num\_incre)
- void [EcritureCommandeVisu](#) ()
- virtual void [Sortie\\_temps\\_cpu](#) (const [LesCondLim](#) &lesCondLim, const [Charge](#) &charge, const [LesContacts](#)  
&contact)
- virtual void [Arret\\_du\\_comptage\\_CPU](#) ()

## Fonctions membres publiques statiques

- static [Alogri](#) \* [New\\_Agori](#) ([EnumTypeCalcul](#) type, const bool avec\_typeDeCalcul, const list<  
[EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCalcul, [UtilLecture](#) &entree↵  
Princ)
- static [Tableau](#)< [Alogri](#) \* > [New\\_tous\\_les\\_Algo](#) (const bool avec\_typeDeCalcul, const list<  
[EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCalcul, [UtilLecture](#) &entree↵  
Princ)

## Fonctions membres protégées

- void **Preparation\_conteneurs\_interne** ([LesMaillages](#) &lesMail)
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)
- virtual void **Ecrit\_Base\_info\_Parametre** ([UtilLecture](#) &entreePrinc, const int &cas)=0
- virtual void **Lecture\_Base\_info\_Parametre** ([UtilLecture](#) &entreePrinc, const int &cas, bool choix)=0
- virtual void **Info\_commande\_parametres** ([UtilLecture](#) &entreePrinc)=0
- void **Change\_affectation\_pointeur\_sous\_type** (const list< [EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCal)
- void **Init\_ParaAlgoControle** ([ParaAlgoControle](#) &paa)
- [ParaAlgoControle](#) & **ParaAlgoControle\_de\_lalgo** ()
- streampos **Debut\_increment** () const
- int **PhaseDeConvergence** () const
- void **Change\_PhaseDeConvergence** (int phase)
- int **Arret\_A\_Equilibre\_Statique** () const
- double **Precision\_equilibre** () const
- void **Temps\_CPU\_HZpp\_to\_lesTempsCpu** (const [LesCondLim](#) &lesCondLim, const Charge &charge, const [LesContacts](#) &contact)
- void **Change\_ptalgocombi** ([AlgoriCombine](#) \*ptal)
- [Tableau](#)< [Temps\\_CPU\\_HZpp](#) > & **Ajout\_Temps\_CPU\_HZpp\_to\_lesTempsCpu** ([Tableau](#)< [Temps\\_CPU\\_HZpp](#) > &lesTsCpu)
- const int & **ArretEquilibreStatique** () const
- void **InitRemontSigma** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*, [DiversStockage](#) \*, Charge \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **InitRemontEps** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*, [DiversStockage](#) \*, Charge \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **InitErreur** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [DiversStockage](#) \*toto, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*titi, [Resultats](#) \*tutu)
- void **RemontSigma** ([LesMaillages](#) \*lesMail)
- void **RemontEps** ([LesMaillages](#) \*lesMail)
- void **RemontErreur** ([LesMaillages](#) \*lesMail)
- bool **InitRemont** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [DiversStockage](#) \*, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*, [Resultats](#) \*)
- bool **CalculRemont** ([LesMaillages](#) \*lesMail, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre)
- void **LectureUnTypeExterne** ([UtilLecture](#) \*entreePrinc, const int type)
- virtual void **lecture\_Parametres** ([UtilLecture](#) &entreePrinc)
- void **Info\_com\_parametres** ([UtilLecture](#) &entreePrinc)
- bool **Line\_search1** ([Vecteur](#) &sauve\_deltadept, double &puis\_precedente, [Vecteur](#) &Vres, [LesMaillages](#) \*lesMail, [Vecteur](#) \*sol, int &compteur, [Vecteur](#) &sauve\_dept\_a\_tdt, Charge \*charge, [Vecteur](#) &vglobex, [Assemblage](#) &Ass, [Vecteur](#) &v\_travail, [LesCondLim](#) \*lesCondLim, [Vecteur](#) &vglobal, [LesReferences](#) \*lesRef, [Vecteur](#) &vglobin, const [Nb\\_assemb](#) &nb\_casAssemb, int cas\_combi\_ddl, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- bool **Line\_search2** ([Vecteur](#) &sauve\_deltadept, double &puis\_precedente, [Vecteur](#) &Vres, [LesMaillages](#) \*lesMail, [Vecteur](#) \*sol, int &compteur, [Vecteur](#) &sauve\_dept\_a\_tdt, Charge \*charge, [Vecteur](#) &vglobex, [Assemblage](#) &Ass, [Vecteur](#) &v\_travail, [LesCondLim](#) \*lesCondLim, [Vecteur](#) &vglobal, [LesReferences](#) \*lesRef, [Vecteur](#) &vglobin, const [Nb\\_assemb](#) &nb\_casAssemb, int cas\_combi\_ddl, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- bool **RaidSmEner** ([LesMaillages](#) \*lesMail, [Assemblage](#) &Ass, [Vecteur](#) &vglobin, [Mat\\_abstraite](#) &matglob)
- bool **SecondMembreEnergy** ([LesMaillages](#) \*lesMail, [Assemblage](#) &Ass, [Vecteur](#) &vglobin)
- bool **RaidSmEnerContact** ([LesContacts](#) \*lesCont, [Assemblage](#) &Ass, [Vecteur](#) &vglobin, [Mat\\_abstraite](#) &matglob)
- bool **SecondMembreEnergyContact** ([LesContacts](#) \*lesContacts, [Assemblage](#) &Ass, [Vecteur](#) &vglobin, bool aff\_iteration)
- [Tableau](#)< [Mat\\_abstraite](#) \* > **Choix\_matriciel** (int nbddl, [Tableau](#)< [Mat\\_abstraite](#) \* > &tab\_matglob, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, const [Nb\\_assemb](#) &nb\_casAssemb, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lescontacts=NULL)
- [Tableau](#)< [Mat\\_abstraite](#) \* > **Mise\_a\_jour\_Choix\_matriciel\_contact** ([Tableau](#)< [Mat\\_abstraite](#) \* > &tab\_matglob, const [Nb\\_assemb](#) &nb\_casAssemb, [LesContacts](#) \*lescontacts, int niveau\_substitution, [TroisEntiers](#) \*nouvelle\_largeur\_imposee=NULL)
- [Mat\\_abstraite](#) \* **Choix\_matrice\_masse** (int nbddl, [Mat\\_abstraite](#) \*matglob, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, const [Nb\\_assemb](#) &nb\_casAssemb, [LesContacts](#) \*lescontacts, [LesCondLim](#) \*lesCondLim)

- `Mat_abstraite * Mise_a_jour_type_et_taille_matrice_masse_en_explicite` (int nbddl, `Mat_abstraite` \*matglob, `LesMaillages` \*lesMail, `LesReferences` \*lesRef, const `Nb_assemb` &nb\_casAssemb, `LesContacts` \*lescontacts)
- void `Cal_matrice_masse` (`LesMaillages` \*lesMail, `Assemblage` &Ass, `Mat_abstraite` &matglob, const `DiversStockage` \*diversStockage, `LesReferences` \*lesRef, const `Enum_ddl` &N\_ddl, `LesFonctions_nD` \*lesFonctionsnD)
- void `Ajout_masses_ponctuelles` (`LesMaillages` \*lesMail, `Assemblage` &Ass, `Mat_abstraite` &matglob, const `DiversStockage` \*diversStockage, `LesReferences` \*lesRef, const `Enum_ddl` &N\_ddl, `LesFonctions_nD` \*lesFonctionsnD)
- bool `Pilotage_du_temps` (`Charge` \*charge, bool &arret)
- bool `Pilotage_fin_iteration_implicit` (int compteur)
- bool `Convergence` (bool affiche, double last\_var\_ddl\_max, `Vecteur` &residu, double maxPuissExt, double maxPuissInt, double maxReaction, int itera, bool &arret)
- bool `Pilotage_fin_relaxation_et_ou_residu` (const int &relax\_vit\_acce, const int &iter\_ou\_incr, const int &compteur, const bool &arretResidu, bool &arret)
- void `Pilotage_chaque_iteration` (`Vecteur` \*sol, double &maxDeltaDdl, const int &itera, `LesMaillages` \*lesMail)
- bool `Pilotage_init_Xtdt` ()
- void `Pilotage_maxi_X_V` (const `Vecteur` &X\_t, `Vecteur` &X\_tdt, const `Vecteur` &V\_t, `Vecteur` &V\_tdt)
- void `CalEnergieAffichage` (const double &coef\_mass, const `Vecteur` &V, const `Mat_abstraite` &mat\_mass, const `Vecteur` &delta\_ddl, int icharge, bool brestart, const `Vecteur` &gamma, const `Vecteur` &forces\_vis\_num)
- void `CalEnergieAffichage` (const `Vecteur` &delta\_ddl, int icharge, bool brestart, const `Vecteur` &forces\_vis\_num)
- void `Affiche_RaidSM` (const `Vecteur` &vglobin, const `Mat_abstraite` &matglob) const
- int `AmortissementCinetique` (const `Vecteur` &delta\_ddl, const double &coef\_mass, const `Vecteur` &X, const `Mat_abstraite` &mat\_mass, int icharge, `Vecteur` &V)
- void `InitialiseAmortissementCinetique` ()
- `Mat_abstraite * Cal_mat_visqueux_num_expli` (const `Mat_abstraite` &mat\_mass, `Mat_abstraite` \*mat\_Cpt, const `Vecteur` &delta\_X, bool inita, const `Vecteur` &vitesse)
- void `Cal_mat_visqueux_num_stat` (`Mat_abstraite` &mat\_glob, `Vecteur` &forces\_vis\_num)
- void `MiseAJourAlgoMere` (`ParaGlob` \*paraGlob, `LesMaillages` \*lesMail, `LesReferences` \*lesRef, `LesCourbes1D` \*lesCourbes1D, `LesFonctions_nD` \*lesFonctionsnD, `VariablesExporter` \*varExpor, `LesLoisDeComp` \*lesLoisDeComp, `DiversStockage` \*diversStockage, `Charge` \*charge, `LesCondLim` \*lesCondLim, `LesContacts` \*lesContacts, `Resultats` \*resultats)
- bool `Gestion_stockage_et_renumerotation_avec_contact` (bool premier\_calcul, `LesMaillages` \*lesMail, bool &nouvelle\_situation\_contact, `LesCondLim` \*lesCondLim, `LesReferences` \*lesRef, `Tableau`<`Mat_abstraite` \* > &tab\_matglob, const `Nb_assemb` &nb\_casAssemb, `LesContacts` \*lescontacts, int niveau\_substitution)
- bool `Gestion_stockage_et_renumerotation_sans_contact` (`LesContacts` \*lescontacts, bool premier\_calcul, `LesMaillages` \*lesMail, bool &nouvelle\_situation\_CLL, `LesCondLim` \*lesCondLim, `LesReferences` \*lesRef, `Tableau`<`Mat_abstraite` \* > &tab\_matglob, const `Nb_assemb` &nb\_casAssemb, int niveau\_substitution)
- void `TdtversT` ()
- void `Cal_Transfert_delta_et_var_X` (double &max\_delta\_X, double &max\_var\_delta\_X)
- void `VerifSingulariteRaideurMeca` (int nbddl, const `LesMaillages` &lesMail) const
- virtual void `Repercussion_algo_sur_cinematique` (`LesContacts` \*lesContacts, `Vecteur` &Xtdt, `Vecteur` &Vtdt)
- bool `Controle_retour_sur_un_increment_enregistre` (int nb\_incr\_en\_arriere, int &icharge)
- int `AmortissementCinetique_individuel_aux_noeuds` (const `Vecteur` &delta\_ddl, const double &coef\_mass, const `Vecteur` &X, const `Mat_abstraite` &mat\_mass, int icharge, `Vecteur` &V)
- void `Passage_aux_noeuds_grandeurs_globales` (`LesMaillages` \*lesMail)
- void `Passage_aux_noeuds_F_int_t_et_F_ext_t` (`LesMaillages` \*lesMail)
- void `Passage_aux_noeuds_grandeur_globale_particuliere` (const `TypeQuelconque` &typeGenerique, `LesMaillages` \*lesMail)
- void `Passage_de_grandeurs_globales_vers_noeuds_pour_variables_globales` (`LesMaillages` \*lesMail, `VariablesExporter` \*varExpor, const `Nb_assemb` &nb\_casAssemb, const `LesReferences` &lesRef)
- void `Transfert_ParaGlob_COMPTEUR_ITERATION_ALGO_GLOBAL` (int compteur) const
- void `Transfert_ParaGlob_NORME_CONVERGENCE` (double laNorme) const
- void `Transfert_ParaGlob_COMPTEUR_INCREMENT_CHARGE_ALGO_GLOBAL` (int incre) const
- void `Transfert_ParaGlob_ALGO_GLOBAL_ACTUEL` (`EnumTypeCalcul` type\_algo) const
- void `Transfert_ParaGlob_energies_internesLoisComp` () const
- void `Transfert_ParaGlob_energies_contact` () const
- void `Transfert_ParaGlob_energies_hourglass_bulk_stab` () const
- void `Transfert_ParaGlob_volume_entre_plans` () const

## Attributs protégés

- int **mode\_debug**
- int **permet\_affichage**
- [EnumTypeCalcul](#) **typeCalcul**
- bool **avec\_typeDeCalcul**
- int **nb\_CasAssemblage**
- [Tableau](#)< double > **paraTypeCalcul**
- list< [EnumSousTypeCalcul](#) > const \* **soustypeDeCalcul**
- list< bool > const \* **avec\_soustypeDeCalcul**
- [ParaAlgoControle](#) **pa**
- double **temps\_derniere\_sauvegarde**
- [List\\_io](#)< [Entier\\_et\\_Double](#) > **list\_incre\_temps\_sauvegarder**
- [List\\_io](#)< [Entier\\_et\\_Double](#) > **list\_incre\_temps\_calculer**
- double **tempsDerniereSortieFilCalcul**
- [Tableau](#)< [ListDeuxString](#) > **noms\_fichier**
- [Tableau](#)< int > **typeFlotExterne**
- [UtilLecture](#) \* **entreePrinc**
- streampos **debut\_increment**
- [Visualisation](#) **visualise**
- [Visualisation\\_maple](#) **visualise\_maple**
- [Visualisation\\_geomview](#) **visualise\_geomview**
- [Visualisation\\_Gid](#) **visualise\_Gid**
- [Visualisation\\_Gmsh](#) **visualise\_Gmsh**
- int **icharge**
- double **E\_cin\_0**
- double **E\_cin\_t**
- double **E\_cin\_tdt**
- double **E\_int\_t**
- double **E\_int\_tdt**
- double **E\_ext\_t**
- double **E\_ext\_tdt**
- double **bilan\_E**
- double **q\_mov\_t**
- double **q\_mov\_tdt**
- double **P\_acc\_tdt**
- double **P\_int\_tdt**
- double **P\_ext\_tdt**
- double **bilan\_P\_tdt**
- double **P\_acc\_t**
- double **P\_int\_t**
- double **P\_ext\_t**
- double **bilan\_P\_t**
- double **E\_visco\_numerique\_t**
- double **E\_visco\_numerique\_tdt**
- [Vecteur](#) **F\_int\_t**
- [Vecteur](#) **F\_ext\_t**
- [Vecteur](#) **F\_int\_tdt**
- [Vecteur](#) **F\_ext\_tdt**
- [Vecteur](#) **F\_totale\_tdt**
- [Vecteur](#) **residu\_final**
- [Vecteur](#) **X\_t**
- [Vecteur](#) **X\_tdt**
- [Vecteur](#) **delta\_X**
- [Vecteur](#) **var\_delta\_X**
- double **delta\_t\_precedent\_a\_convergence**
- [Vecteur](#) **vitesse\_t**
- [Vecteur](#) **vitesse\_tdt**
- [Vecteur](#) **acceleration\_t**
- [Vecteur](#) **acceleration\_tdt**
- double **E\_bulk**
- double **P\_bulk**
- [List\\_io](#)< double > **E\_cin\_ind\_t**
- [List\\_io](#)< double > **E\_cin\_ind\_tdt**
- [List\\_io](#)< double > **E\_int\_ind\_t**

- `List_io`< double > `E_int_ind_tdt`
- `List_io`< double > `E_ext_ind_t`
- `List_io`< double > `E_ext_ind_tdt`
- `List_io`< double > `bilan_ind_E`
- `List_io`< double > `q_mov_ind_t`
- `List_io`< double > `q_mov_ind_tdt`
- `List_io`< double > `P_acc_ind_tdt`
- `List_io`< double > `P_int_ind_tdt`
- `List_io`< double > `P_ext_ind_tdt`
- `List_io`< double > `bilan_P_ind_tdt`
- `List_io`< double > `P_acc_ind_t`
- `List_io`< double > `P_int_ind_t`
- `List_io`< double > `P_ext_ind_t`
- `List_io`< double > `bilan_P_ind_t`
- `List_io`< double > `E_visco_numerique_ind_t`
- `List_io`< double > `E_visco_numerique_ind_tdt`
- `List_io`< Vecteur > `F_int_ind_t`
- `List_io`< Vecteur > `F_ext_ind_t`
- `List_io`< Vecteur > `F_int_ind_tdt`
- `List_io`< Vecteur > `F_ext_ind_tdt`
- `List_io`< double > `E_bulk_ind`
- `List_io`< double > `P_bulk_ind`
- `map`< string, const double \*, std::less< string > > `listeVarGlob`
- `List_io`< TypeQuelconque > `listeVecGlob`
- int `deja_lue_entete_parametre`
- bool `amortissement_cinetique`
- Vecteur `X_EcinMax`
- Vecteur `F_int_tdt_prec`
- Vecteur `F_ext_tdt_prec`
- Vecteur \* `vglob_stat`
- Vecteur `vec_trav`
- Temps\_CPU\_HZpp `tempsInitialisation`
- Temps\_CPU\_HZpp `tempsMiseAJourAlgo`
- Temps\_CPU\_HZpp `tempsCalEquilibre`
- Temps\_CPU\_HZpp `tempsRaidSmEner`
- Temps\_CPU\_HZpp `tempsSecondMembreEnerg`
- Temps\_CPU\_HZpp `tempsResolSystemLineaire`
- Temps\_CPU\_HZpp `tempsSauvegarde`
- Temps\_CPU\_HZpp `tempsSortieFilCalcul`
- Temps\_CPU\_HZpp `tempsRaidSmEnerContact`
- Temps\_CPU\_HZpp `tempsSecondMembreEnergContact`
- Temps\_CPU\_HZpp `temps_CL`
- Temps\_CPU\_HZpp `temps_CLL`
- Temps\_CPU\_HZpp `temps_lois_comportement`
- Temps\_CPU\_HZpp `temps_metrique_K_SM`
- Temps\_CPU\_HZpp `temps_chargement`
- Temps\_CPU\_HZpp `temps_rech_contact`
- Tableau< Coordonnee3 > `lesTempsCpu`
- AlgoriCombine \* `ptalgocombi`

## Amis

- class `Charge`
- class `AlgoriCombine`
- ostream & `operator`<< (ostream &sort, const `Algori` &)

### 6.13.1 Description détaillée

BUT: Classe de base de Defintion des differents algorithmes de resolution.

## 6.13.2 Documentation des fonctions membres

### 6.13.2.1 AmortissementCinetique()

```
int Algori::AmortissementCinetique (
    const Vecteur & delta_ddl,
    const double & coef_mass,
    const Vecteur & X,
    const Mat_abstraite & mat_mass,
    int ichearge,
    Vecteur & V ) [protected]
```

if ((ichearge > nb\_deb\_test\_amort\_cinetique\_noe) && (!amortissement\_cinetique\_au\_noeud) && amortissement\_←  
\_cinetique )

### 6.13.2.2 CalEquilibre()

```
virtual void Algori::CalEquilibre (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    Tableau< Fonction_nD * > * tb_combiner ) [pure virtual]
```

Implémenté dans [AlgoriRungeKutta](#), et [AlgoriNewmark](#).

### 6.13.2.3 Info\_commande\_parametres()

```
virtual void Algori::Info_commande_parametres (
    UtilLecture & entreePrinc ) [protected], [pure virtual]
```

Implémenté dans [AlgoristatExpli](#).

### 6.13.2.4 lecture\_Parametres()

```
void Algori::lecture_Parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Réimplémentée dans [AlgoristatExpli](#).



### 6.13.2.5 SchemaXML\_Algori()

```
virtual void Algori::SchemaXML_Algori (
    ostream & sort,
    const Enum_IO_XML enu ) const [pure virtual]
```

Implémenté dans [AlgoristatExpli](#).

La documentation de cette classe a été générée à partir du fichier suivant :

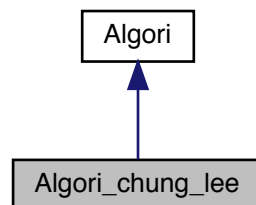
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoRef/Algori.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoRef/Algori.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoRef/Algori2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoRef/Algori3.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoRef/Algori4.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/RemontErreur.cc

## 6.14 Référence de la classe Algori\_chung\_lee

BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées. On ne prend pas en compte les phénomènes de contact. Ici il s'agit de l'algorithme proposé par Chung Lee.

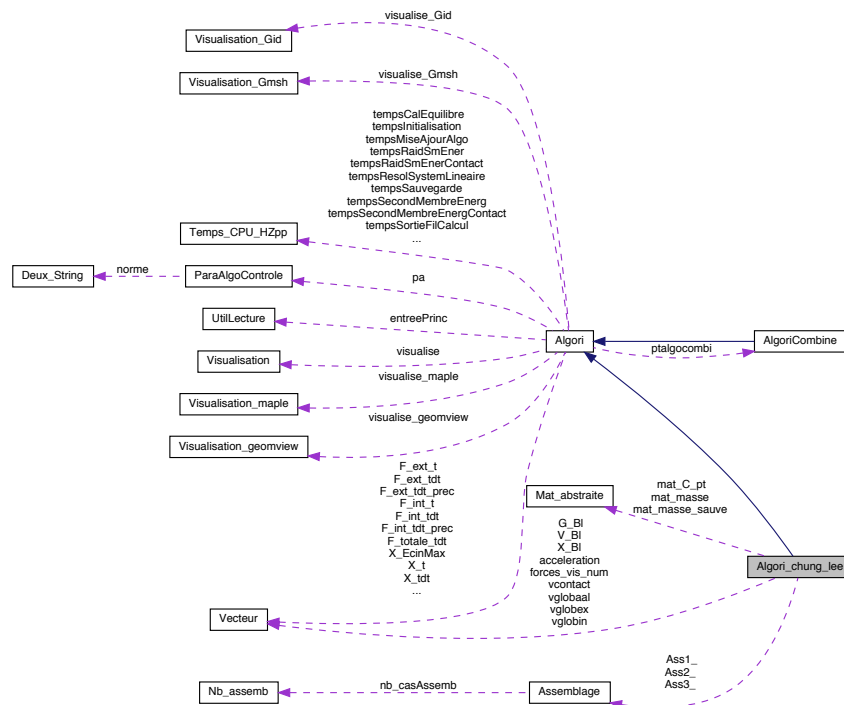
```
#include <Algori_chung_lee.h>
```

Graphes d'héritage de Algori\_chung\_lee:





Graphe de collaboration de `Algori_chung_lee`:



## Fonctions membres publiques

- `Algori_chung_lee` (const bool avec\_typeDeCal, const list< [EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCal, [UtilLecture](#) &entreePrinc)
- `Algori_chung_lee` (const `Algori_chung_lee` &algo)
- `Algori * New_idem` (const `Algori` \*algo) const
- void `Execution` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*varExpor, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `InitAlgorithme` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `MiseAJourAlgo` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `CalEquilibre` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*, `Tableau`< `Fonction_nD` > \*tb\_combiner)
- void `FinCalcul` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `SchemaXML_Algori` (ofstream &sort, const `Enum_IO_XML` enu) const

## Fonctions membres protégées

- void `Calcul_Equilibre` (`ParaGlob` \*paraGlob, `LesMaillages` \*lesMail, `LesReferences` \*lesRef, `LesCourbes1D` \*lesCourbes1D, `LesFonctions_nD` \*lesFonctionsnD, `LesLoisDeComp` \*lesLoisDeComp, `DiversStockage` \*diversStockage, `Charge` \*charge, `LesCondLim` \*lesCondLim, `LesContacts` \*lesContacts, `Resultats` \*resultats)

- bool **ActionInteractiveAlgo** ()
- void **lecture\_Parametres** ([UtilLecture](#) &entreePrinc)
- void **Ecrit\_Base\_info\_Parametre** ([UtilLecture](#) &entreePrinc, const int &cas)
- void **Lecture\_Base\_info\_Parametre** ([UtilLecture](#) &entreePrinc, const int &cas, bool choix)
- void **Info\_commande\_parametres** ([UtilLecture](#) &entreePrinc)
- bool **Gestion\_pas\_de\_temps** (bool modif\_pas\_de\_temps, [LesMaillages](#) \*lesMail, int cas)

### Attributs protégés

- double \* **beta\_cl**
- double **delta\_t**
- double **deltat2**
- double **unsurdeltat**
- double **deltatSurDeux**
- double **betachapeau**
- double **gammaa**
- double **gammachapeau**
- [Assemblage](#) \* **Ass1\_**
- [Assemblage](#) \* **Ass2\_**
- [Assemblage](#) \* **Ass3\_**
- int **cas\_combi\_ddl**
- int **icas**
- bool **prepa\_avec\_remont**
- bool **brestart**
- [OrdreVisu::EnumTypeIncre](#) **type\_incre**
- [Vecteur](#) **vglobin**
- [Vecteur](#) **vglobex**
- [Vecteur](#) **vglobaal**
- [Vecteur](#) **vcontact**
- [Vecteur](#) **acceleration**
- [Vecteur](#) **X\_BI**
- [Vecteur](#) **V\_BI**
- [Vecteur](#) **G\_BI**
- [Vecteur](#) **forces\_vis\_num**
- [list](#)< [LesCondLim::Gene\\_asso](#) > **li\_gene\_asso**
- [Tableau](#)< [Nb\\_assemb](#) > **t\_assemb**
- [Tableau](#)< [Enum\\_ddl](#) > **tenuXVG**
- [Mat\\_abstraite](#) \* **mat\_masse**
- [Mat\\_abstraite](#) \* **mat\_masse\_sauve**
- [Mat\\_abstraite](#) \* **mat\_C\_pt**

### Membres hérités additionnels

#### 6.14.1 Description détaillée

BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées. On ne prend pas en compte les phénomènes de contact. Ici il s'agit de l'algorithme proposé par Chung Lee.

#### 6.14.2 Documentation des fonctions membres

### 6.14.2.1 `CalEquilibre()`

```
void Algori_chung_lee::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

Implémente [Algori](#).

### 6.14.2.2 `Ecrit_Base_info_Parametre()`

```
void Algori_chung_lee::Ecrit_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.14.2.3 `Execution()`

```
void Algori_chung_lee::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.14.2.4 FinCalcul()

```
void Algori_chung_lee::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.14.2.5 Info\_commande\_parametres()

```
void Algori_chung_lee::Info_commande_parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Implémente [Algori](#).

#### 6.14.2.6 InitAlgorithme()

```
void Algori_chung_lee::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.14.2.7 `Lecture_Base_info_Parametre()`

```
void Algori_chung_lee::Lecture_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas,
    bool choix ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.14.2.8 `lecture_Parametres()`

```
void Algori_chung_lee::lecture_Parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Réimplémentée à partir de [Algori](#).

### 6.14.2.9 `MiseAJourAlgo()`

```
void Algori_chung_lee::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.14.2.10 `New_idem()`

```
Algori * Algori_chung_lee::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.14.2.11 SchemaXML\_Algori()

```
void Algori_chung_lee::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [inline], [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

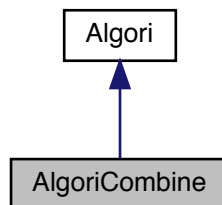
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori\_↔chung\_lee.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori\_↔chung\_lee.cc

## 6.15 Référence de la classe AlgoriCombine

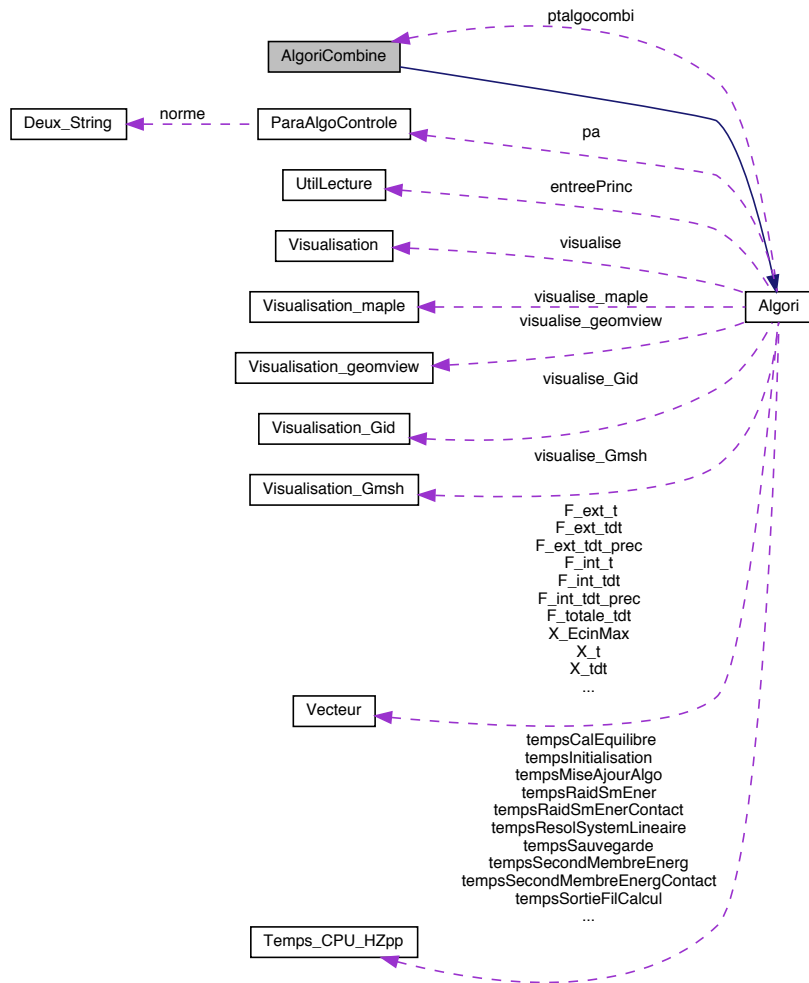
BUT: Algorithme combiné, l'objectif est de mettre en oeuvre plusieurs algorithmes existants, suivant une stratégie que l'utilisateur impose au travers de fonction nD particulières.

```
#include <AlgoriCombine.h>
```

Graphe d'héritage de AlgoriCombine:



Graphe de collaboration de AlgoriCombine:



## Fonctions membres publiques

- **AlgoriCombine** (const bool avec\_typeDeCal, const list< [EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCal, [UtilLecture](#) &entreePrinc)
- **AlgoriCombine** (const [AlgoriCombine](#) &algo)
- [Algori](#) \* **New\_idem** (const [Algori](#) \*algo) const
- virtual void **Lecture** ([UtilLecture](#) &entreePrinc, [ParaGlob](#) &paraGlob, [LesMaillages](#) &lesMail)
- void **Execution** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*varExpor, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **InitAlgorithme** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **MiseAJourAlgo** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **CalEquilibre** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [Tableau](#)< [Fonction\\_nD](#) \* > \*tb\_combiner)

- void `FinCalcul` (`ParaGlob *`, `LesMaillages *`, `LesReferences *`, `LesCourbes1D *`, `LesFonctions_nD *`, `VariablesExporter *`, `LesLoisDeComp *`, `DiversStockage *`, `Charge *`, `LesCondLim *`, `LesContacts *`, `Resultats *`)
- void `SchemaXML_Algori` (`ofstream &`, `const Enum_IO_XML`) `const`
- void `AutreSortieTempsCPU` (`ofstream &sort`, `const int cas`) `const`
- `list< EnumTypeCalcul > List_Sous_Algo` () `const`
- virtual void `Sortie_temps_cpu` (`const LesCondLim &lesCondLim`, `const Charge &charge`, `const LesContacts &contact`)
- virtual void `Arret_du_comptage_CPU` ()

## Membres hérités additionnels

### 6.15.1 Description détaillée

BUT: Algorithme combiné, l'objectif est de mettre en oeuvre plusieurs algorithmes existants, suivant une stratégie que l'utilisateur impose au travers de fonction nD particulières.

### 6.15.2 Documentation des fonctions membres

#### 6.15.2.1 Arret\_du\_comptage\_CPU()

```
void AlgoriCombine::Arret_du_comptage_CPU ( ) [virtual]
```

Réimplémentée à partir de [Algori](#).

#### 6.15.2.2 AutreSortieTempsCPU()

```
void AlgoriCombine::AutreSortieTempsCPU (
    ofstream & sort,
    const int cas ) const
```

—> pour rappel:

#### 6.15.2.3 CalEquilibre()

```
void AlgoriCombine::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

Implémente [Algori](#).



#### 6.15.2.4 `Execution()`

```
void AlgoriCombine::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.15.2.5 `FinCalcul()`

```
void AlgoriCombine::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.15.2.6 `InitAlgorithme()`

```
void AlgoriCombine::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.15.2.7 Lecture()

```
void AlgoriCombine::Lecture (
    UtilLecture & entreePrinc,
    ParaGlob & paraGlob,
    LesMaillages & lesMail ) [virtual]
```

Réimplémentée à partir de [Algori](#).

### 6.15.2.8 MiseAJourAlgo()

```
void AlgoriCombine::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.15.2.9 New\_idem()

```
Algori * AlgoriCombine::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.15.2.10 SchemaXML\_Algori()

```
void AlgoriCombine::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [virtual]
```

Implémente [Algori](#).

### 6.15.2.11 Sortie\_temps\_cpu()

```
void AlgoriCombine::Sortie_temps_cpu (
    const LesCondLim & lesCondLim,
    const Charge & charge,
    const LesContacts & contact ) [virtual]
```

Réimplémentée à partir de [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

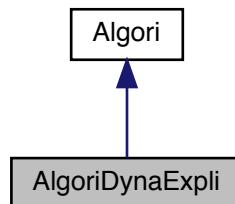
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgorithmeCombiner/AlgoriCombine.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgorithmeCombiner/AlgoriCombine.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgorithmeCombiner/AlgoriCombine2.cc

## 6.16 Référence de la classe AlgoriDynaExpli

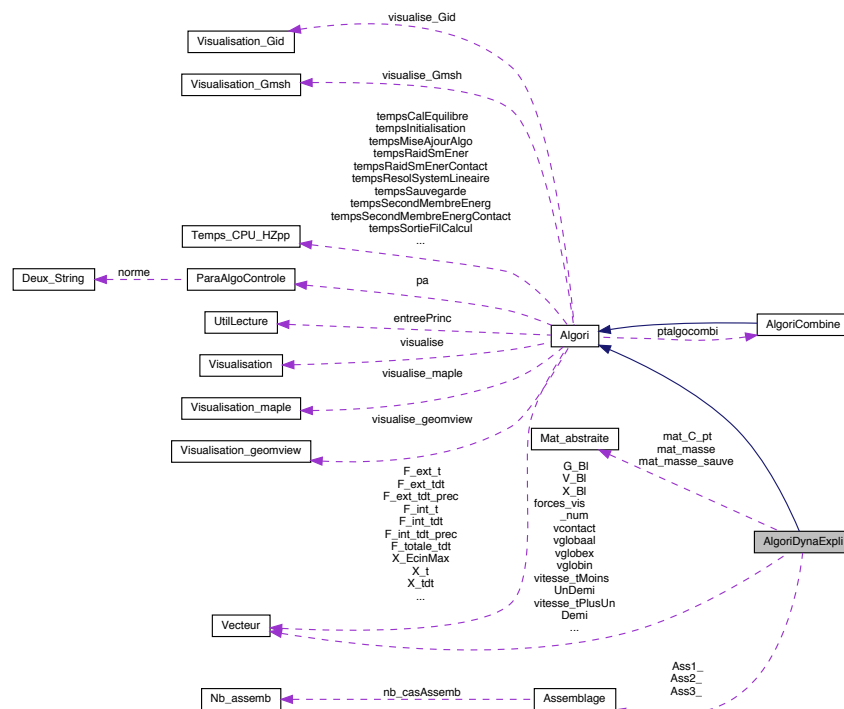
BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. On ne prend pas en compte les phénomènes de contact.

```
#include <AlgoriDynaExpli.h>
```

Graphe d'héritage de AlgoriDynaExpli:



Graphe de collaboration de AlgoriDynaExpli:



## Fonctions membres publiques

- **AlgoriDynaExpli** (const bool avec\_typeDeCal, const list< [EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCal, [UtilLecture](#) &entreePrinc)
- **AlgoriDynaExpli** (const [AlgoriDynaExpli](#) &algo)
- **Algori \* New\_idem** (const [Algori](#) \*algo) const
- void **Execution** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*varExpOr, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **InitAlgorithme** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **MiseAJourAlgo** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **CalEquilibre** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [Tableau](#)< [Fonction\\_nD](#) \* > \*tb\_combiner)
- void **FinCalcul** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **SchemaXML\_Algori** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const

## Fonctions membres protégées

- void **Calcul\_Equilibre** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD, [VariablesExporter](#) \*varExpOr, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, [Charge](#) \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats)

- void **Calcul\_Equilibre2** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD, [VariablesExporter](#) \*varExpor, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats)
- void [lecture\\_Parametres](#) ([UtilLecture](#) &entreePrinc)
- void [Ecrit\\_Base\\_info\\_Parametre](#) ([UtilLecture](#) &entreePrinc, const int &cas)
- void [Lecture\\_Base\\_info\\_Parametre](#) ([UtilLecture](#) &entreePrinc, const int &cas, bool choix)
- void [Info\\_commande\\_parametres](#) ([UtilLecture](#) &entreePrinc)
- bool **Gestion\_pas\_de\_temps** (bool modif\_pas\_de\_temps, [LesMaillages](#) \*lesMail, int cas)
- bool **ActionInteractiveAlgo** ()

### Attributs protégés

- [Vecteur](#) [vitesse\\_tMoinsUnDemi](#)
- [Vecteur](#) [vitesse\\_tPlusUnDemi](#)
- double [delta\\_t](#)
- double [unsurdeltat](#)
- double [deltatSurDeux](#)
- double [deltat2](#)
- double [deltat2SurDeux](#)
- int [type\\_cal\\_equilibre](#)
- [Assemblage](#) \* [Ass1\\_](#)
- [Assemblage](#) \* [Ass2\\_](#)
- [Assemblage](#) \* [Ass3\\_](#)
- int [cas\\_combi\\_ddl](#)
- int [icas](#)
- bool [prepa\\_avec\\_remont](#)
- bool [brestart](#)
- [OrdreVisu::EnumTypeIncre](#) [type\\_incre](#)
- int [compteur\\_demarrage](#)
- [Vecteur](#) [vglobin](#)
- [Vecteur](#) [vglobex](#)
- [Vecteur](#) [vglobaal](#)
- [Vecteur](#) [vcontact](#)
- [Vecteur](#) [X\\_BI](#)
- [Vecteur](#) [V\\_BI](#)
- [Vecteur](#) [G\\_BI](#)
- [Vecteur](#) [forces\\_vis\\_num](#)
- list< [LesCondLim::Gene\\_asso](#) > [li\\_gene\\_asso](#)
- Tableau< [Nb\\_assemb](#) > [t\\_assemb](#)
- Tableau< [Enum\\_ddl](#) > [tenuXVG](#)
- [Mat\\_abstraite](#) \* [mat\\_masse](#)
- [Mat\\_abstraite](#) \* [mat\\_masse\\_sauve](#)
- [Mat\\_abstraite](#) \* [mat\\_C\\_pt](#)

### Membres hérités additionnels

#### 6.16.1 Description détaillée

BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées. On ne prend pas en compte les phénomènes de contact.

#### 6.16.2 Documentation des fonctions membres

### 6.16.2.1 CalEquilibre()

```
void AlgoriDynaExpli::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

Implémente [Algori](#).

### 6.16.2.2 Ecrit\_Base\_info\_Parametre()

```
void AlgoriDynaExpli::Ecrit_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.16.2.3 Execution()

```
void AlgoriDynaExpli::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.16.2.4 `FinCalcul()`

```
void AlgoriDynaExpli::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.16.2.5 `Info_commande_parametres()`

```
void AlgoriDynaExpli::Info_commande_parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Implémente [Algori](#).

#### 6.16.2.6 `InitAlgorithme()`

```
void AlgoriDynaExpli::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.16.2.7 Lecture\_Base\_info\_Parametre()

```
void AlgoriDynaExpli::Lecture_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas,
    bool choix ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.16.2.8 lecture\_Parametres()

```
void AlgoriDynaExpli::lecture_Parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Réimplémentée à partir de [Algori](#).

### 6.16.2.9 MiseAJourAlgo()

```
void AlgoriDynaExpli::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.16.2.10 New\_idem()

```
Algori * AlgoriDynaExpli::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).



### 6.16.2.11 SchemaXML\_Algori()

```
void AlgoriDynaExpli::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

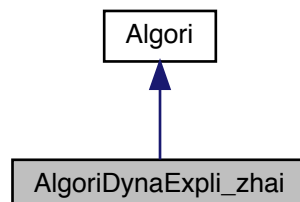
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori↔DynaExpli.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori↔DynaExpli.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori↔DynaExpli2.cc

## 6.17 Référence de la classe AlgoriDynaExpli\_zhai

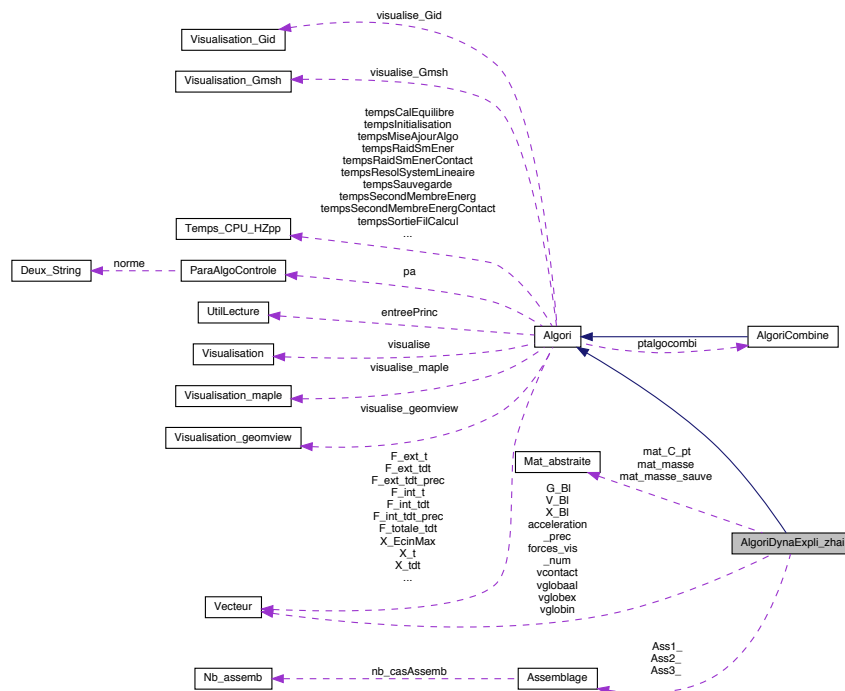
BUT: Algorithme de calcul dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees. On ne prend pas en compte les phénomènes de contact. La méthode implantée est celle de zhai.

```
#include <AlgoriDynaExpli_zhai.h>
```

Graphe d'héritage de AlgoriDynaExpli\_zhai:



Graphe de collaboration de AlgoriDynaExpli\_zhai:



## Fonctions membres publiques

- **AlgoriDynaExpli\_zhai** (const bool avec\_typeDeCal, const list< [EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCal, [UtilLecture](#) &entreePrinc)
- **AlgoriDynaExpli\_zhai** (const [AlgoriDynaExpli\\_zhai](#) &algo)
- **Algori \* New\_idem** (const [Algori](#) \*algo) const
- void **Execution** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*varExpor, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **InitAlgorithme** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **MiseAJourAlgo** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **CalEquilibre** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [Tableau](#)< [Fonction\\_nD](#) \* > \*tb\_combiner)
- void **FinCalcul** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **SchemaXML\_Algori** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const

## Fonctions membres protégées

- void **Calcul\_Equilibre** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD, [VariablesExporter](#) \*varExpor, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, [Charge](#) \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats)

- bool **ActionInteractiveAlgo** ()
- void **lecture\_Parametres** (UtilLecture &entreePrinc)
- void **Ecrit\_Base\_info\_Parametre** (UtilLecture &entreePrinc, const int &cas)
- void **Lecture\_Base\_info\_Parametre** (UtilLecture &entreePrinc, const int &cas, bool choix)
- void **Info\_commande\_parametres** (UtilLecture &entreePrinc)
- bool **Gestion\_pas\_de\_temps** (bool modif\_pas\_de\_temps, LesMaillages \*lesMail, int cas)

### Attributs protégés

- double \* **grand\_phi**
- double \* **phi\_minus**
- double \* **gamma\_pt**
- double \* **beta**
- double **delta\_t**
- double **deltat2**
- double **unsurdeltat**
- double **undemiplusgrandphi\_deltat2**
- double **grandphi\_deltat2**
- double **unpluphi\_delta\_t**
- double **phi\_delta\_t**
- double **un\_moins\_gamma\_delta\_t**
- double **gamma\_delta\_t**
- double **undemi\_moins\_beta\_deltat2**
- double **beta\_deltat2**
- [Assemblage](#) \* **Ass1\_**
- [Assemblage](#) \* **Ass2\_**
- [Assemblage](#) \* **Ass3\_**
- int **cas\_combi\_ddl**
- int **icas**
- bool **prepa\_avec\_remont**
- bool **brestart**
- [OrdreVisu::EnumTypeIncre](#) **type\_incre**
- [Vecteur](#) **vglobin**
- [Vecteur](#) **vglobex**
- [Vecteur](#) **vglobaal**
- [Vecteur](#) **vcontact**
- [Vecteur](#) **acceleration\_prec**
- [Vecteur](#) **X\_BI**
- [Vecteur](#) **V\_BI**
- [Vecteur](#) **G\_BI**
- [Vecteur](#) **forces\_vis\_num**
- [list](#)< [LesCondLim::Gene\\_asso](#) > **li\_gene\_asso**
- [Tableau](#)< [Nb\\_assemb](#) > **t\_assemb**
- [Tableau](#)< [Enum\\_ddl](#) > **tenuXVG**
- [Mat\\_abstraite](#) \* **mat\_masse**
- [Mat\\_abstraite](#) \* **mat\_masse\_sauve**
- [Mat\\_abstraite](#) \* **mat\_C\_pt**

### Membres hérités additionnels

#### 6.17.1 Description détaillée

BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées. On ne prend pas en compte les phénomènes de contact. La méthode implantée est celle de zhai.

#### 6.17.2 Documentation des fonctions membres

### 6.17.2.1 CalEquilibre()

```
void AlgoriDynaExpli_zhai::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

Implémente [Algori](#).

### 6.17.2.2 Ecrit\_Base\_info\_Parametre()

```
void AlgoriDynaExpli_zhai::Ecrit_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.17.2.3 Execution()

```
void AlgoriDynaExpli_zhai::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.17.2.4 FinCalcul()

```
void AlgoriDynaExpli_zhai::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.17.2.5 Info\_commande\_parametres()

```
void AlgoriDynaExpli_zhai::Info_commande_parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Implémente [Algori](#).

#### 6.17.2.6 InitAlgorithme()

```
void AlgoriDynaExpli_zhai::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.17.2.7 Lecture\_Base\_info\_Parametre()

```
void AlgoriDynaExpli_zhai::Lecture_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas,
    bool choix ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.17.2.8 lecture\_Parametres()

```
void AlgoriDynaExpli_zhai::lecture_Parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Réimplémentée à partir de [Algori](#).

### 6.17.2.9 MiseAJourAlgo()

```
void AlgoriDynaExpli_zhai::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.17.2.10 New\_idem()

```
Algori * AlgoriDynaExpli_zhai::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.17.2.11 SchemaXML\_Algori()

```
void AlgoriDynaExpli_zhai::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [inline], [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

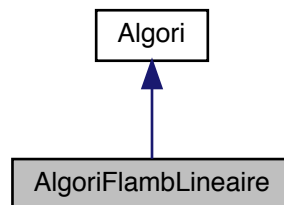
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori↔  
DynaExpli\_zhai.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori↔  
DynaExpli\_zhai.cc

## 6.18 Référence de la classe AlgoriFlambLineaire

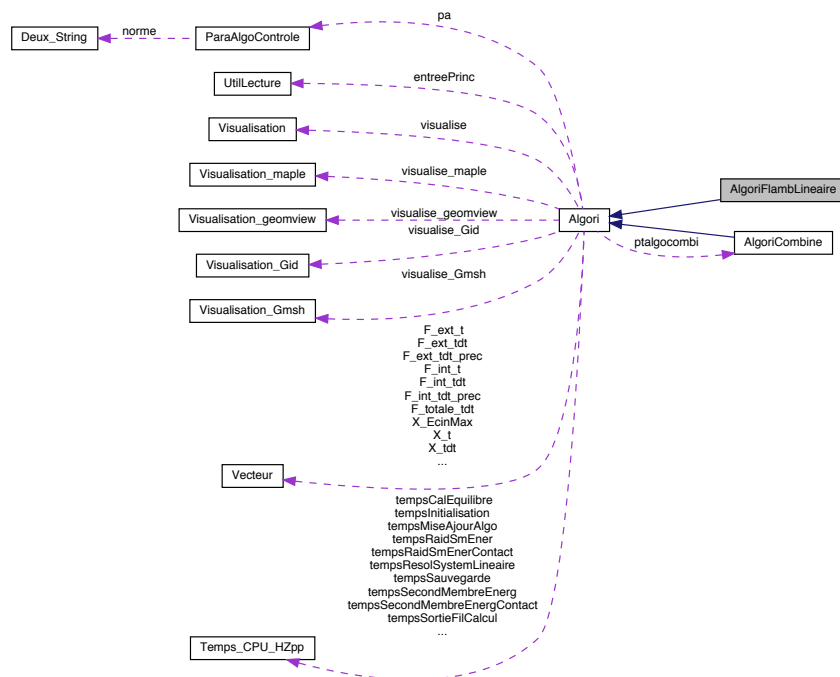
BUT: Algorithme de calcul pour le flambement linéaire, le calcul préliminaire est non dynamique, pour de la mécanique en coordonnées matérielles entraînées.

```
#include <AlgoriFlambLineaire.h>
```

Grappe d'héritage de AlgoriFlambLineaire:



Graphe de collaboration de `AlgoriFlambLineaire`:



## Fonctions membres publiques

- `AlgoriFlambLineaire` (const bool avec `_typeDeCal`, const list< `EnumSousTypeCalcul` > & `soustype`, const list< bool > & `avec_soustypeDeCal`, `UtilLecture` & `entreePrinc`)
- `AlgoriFlambLineaire` (const `AlgoriFlambLineaire` & `algo`)
- `Algori` \* `New_idem` (const `Algori` \* `algo`) const
- void `Execution` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \* `varExp`, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `InitAlgorithme` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `MiseAJourAlgo` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `CalEquilibre` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*, `Tableau`< `Fonction_nD` \* > \* `tb_combiner`)
- void `FinCalcul` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `SchemaXML_Algori` (`ofstream` & `sort`, const `Enum_IO_XML` enu) const

## Membres hérités additionnels

### 6.18.1 Description détaillée

BUT: Algorithme de calcul pour le flambement linéaire, le calcul préliminaire est non dynamique, pour de la mécanique en coordonnées matérielles entraînées.



## 6.18.2 Documentation des fonctions membres

### 6.18.2.1 `CalEquilibre()`

```
void AlgoriFlambLineaire::CalEquilibre (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    Tableau< Fonction_nD * > * tb_combiner ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.18.2.2 `Execution()`

```
void AlgoriFlambLineaire::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.18.2.3 FinCalcul()

```
void AlgoriFlambLineaire::FinCalcul (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.18.2.4 InitAlgorithme()

```
void AlgoriFlambLineaire::InitAlgorithme (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.18.2.5 MiseAJourAlgo()

```
void AlgoriFlambLineaire::MiseAJourAlgo (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.18.2.6 New\_idem()

```
Algori * AlgoriFlambLineaire::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.18.2.7 SchemaXML\_Algori()

```
void AlgoriFlambLineaire::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [inline], [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

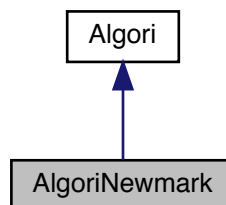
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/AlgoriFlambLineaire.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/AlgoriFlambLineaire.cc

## 6.19 Référence de la classe AlgoriNewmark

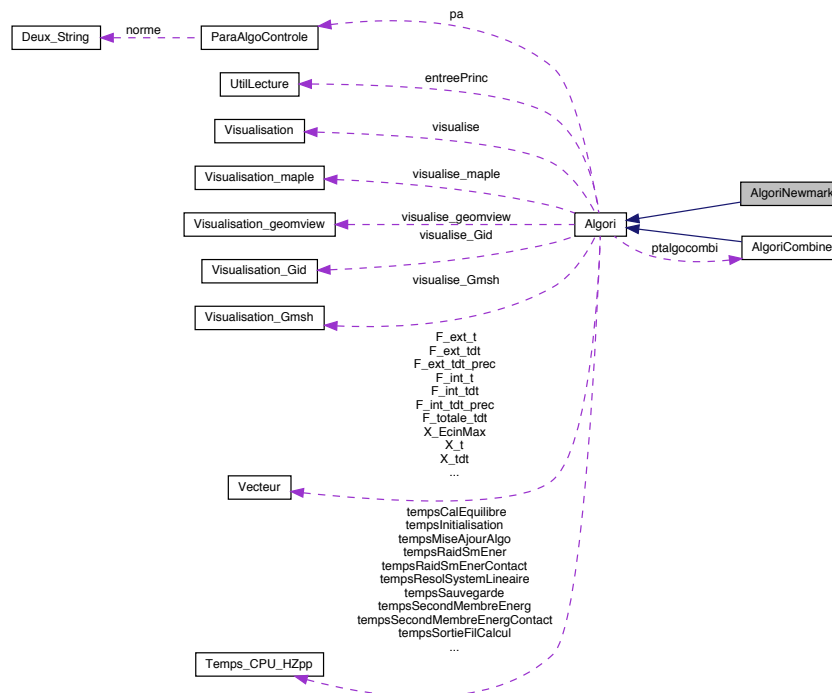
BUT: Algorithme de calcul dynamique, pour de la mecanique du solide déformable en coordonnees materielles entrainees, en utilisant la discrétisation temporelle de newmark .

```
#include <AlgoriNewmark.h>
```

Graphe d'héritage de AlgoriNewmark:



Graphe de collaboration de AlgoriNewmark:



## Fonctions membres publiques

- **AlgoriNewmark** (const bool avec\_typeDeCal, const list< EnumSousTypeCalcul > &soustype, const list< bool > &avec\_soustypeDeCal, UtilLecture &entreePrinc)
- **AlgoriNewmark** (const AlgoriNewmark &algo)
- **Algori \* New\_idem** (const Algori \*algo) const
- void **Execution** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*varExpor, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **InitAlgorithme** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **MiseAJourAlgo** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **CalEquilibre** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*, Tableau< Fonction\_nD \* > \*tb\_combiner)
- void **FinCalcul** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **SchemaXML\_Algori** (ofstream &sort, const Enum\_IO\_XML enu) const

## Membres hérités additionnels

### 6.19.1 Description détaillée

BUT: Algorithme de calcul dynamique, pour de la mecanique du solide déformable en coordonnees materielles entrainees, en utilisant la discrétisation temporelle de newmark .

## 6.19.2 Documentation des fonctions membres

### 6.19.2.1 CalEquilibre()

```
void AlgoriNewmark::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

\*\*\* dans le cas de contact qui impose la position de ddl, il faudrait en tenir compte en dynamique, au niveau des statuts \*\*\* donc pour l'instant, ce n'est pas pris en comptes !!! \*\*\*

Implémente [Algori](#).

### 6.19.2.2 Execution()

```
void AlgoriNewmark::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.19.2.3 FinCalcul()

```
void AlgoriNewmark::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.19.2.4 InitAlgorithme()

```
void AlgoriNewmark::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.19.2.5 MiseAJourAlgo()

```
void AlgoriNewmark::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.19.2.6 New\_idem()

```
Algori * AlgoriNewmark::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.19.2.7 SchemaXML\_Algori()

```
void AlgoriNewmark::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [inline], [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

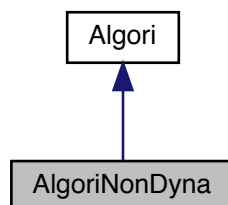
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaImplicite/Algori↔Newmark.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaImplicite/Algori↔Newmark.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaImplicite/Algori↔Newmark2.cc

## 6.20 Référence de la classe AlgoriNonDyna

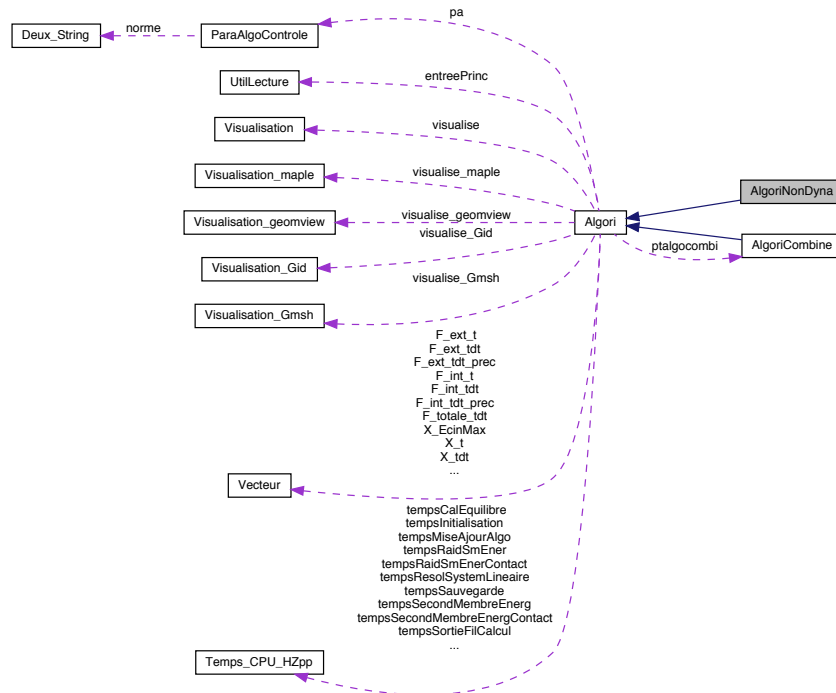
BUT: Algorithme de calcul non dynamique, pour de la mecanique du solide déformable en coordonnees materielles entrainees.

```
#include <AlgoriNonDyna.h>
```

Grappe d'héritage de AlgoriNonDyna:



Graphe de collaboration de AlgoriNonDyna:



## Fonctions membres publiques

- **AlgoriNonDyna** (const bool avec\_typeDeCal, const list< EnumSousTypeCalcul > &soustype, const list< bool > &avec\_soustypeDeCal, UtilLecture &entreePrinc)
- **AlgoriNonDyna** (const AlgoriNonDyna &algo)
- **Algori \* New\_idem** (const Algori \*algo) const
- void **Execution** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*varExpor, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **InitAlgorithme** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **MiseAJourAlgo** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **CalEquilibre** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*, Tableau< Fonction\_nD \* > \*tb\_combiner)
- void **FinCalcul** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **SchemaXML\_Algori** (ofstream &, const Enum\_IO\_XML) const

## Membres hérités additionnels

### 6.20.1 Description détaillée

BUT: Algorithme de calcul non dynamique, pour de la mecanique du solide déformable en coordonnees materielles entrainees.



## 6.20.2 Documentation des fonctions membres

### 6.20.2.1 `CalEquilibre()`

```
void AlgoriNonDyna::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

Implémente [Algori](#).

### 6.20.2.2 `Execution()`

```
void AlgoriNonDyna::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.20.2.3 FinCalcul()

```
void AlgoriNonDyna::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.20.2.4 InitAlgorithme()

```
void AlgoriNonDyna::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.20.2.5 MiseAJourAlgo()

```
void AlgoriNonDyna::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.20.2.6 New\_idem()

```
Algori * AlgoriNonDyna::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.20.2.7 SchemaXML\_Algori()

```
void AlgoriNonDyna::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

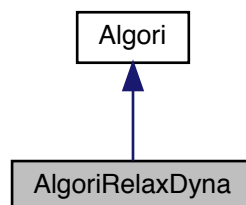
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/AlgoriNon↔  
Dyna.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/AlgoriNon↔  
Dyna.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/AlgoriNon↔  
Dyna2.cc

## 6.21 Référence de la classe AlgoriRelaxDyna

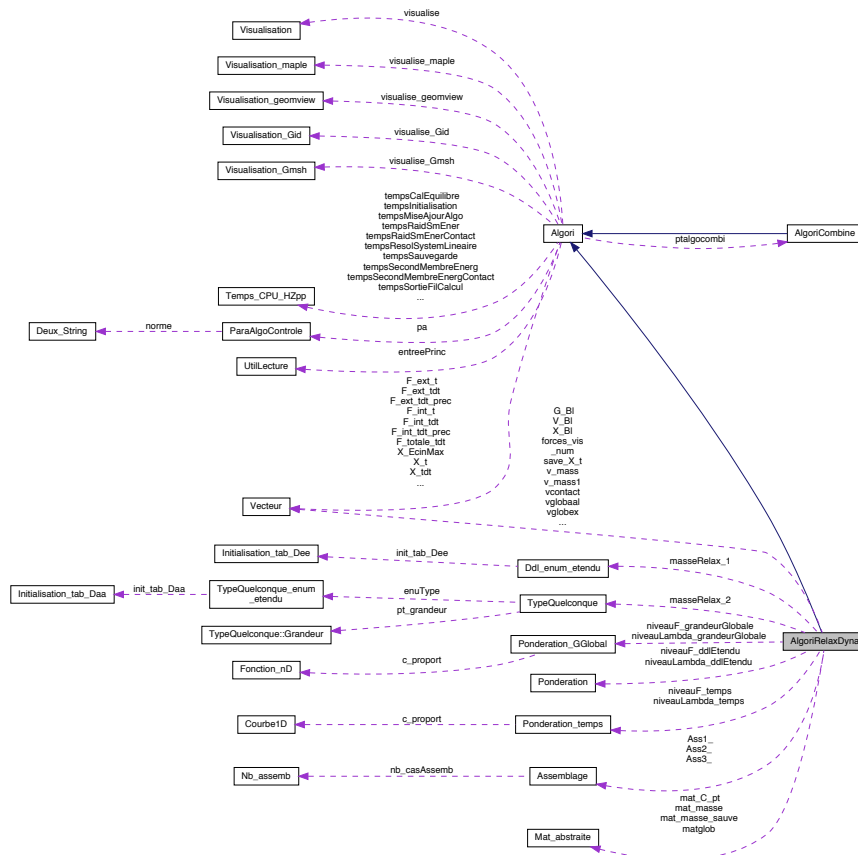
BUT: Algorithme de relaxation dynamique selon la méthode de Barnes, modifiée par Julien Troufflard puis généralisée.

```
#include <Algori_relax_dyna.h>
```

Grappe d'héritage de AlgoriRelaxDyna:



Graphe de collaboration de AlgoriRelaxDyna:



## Fonctions membres publiques

- `AlgoriRelaxDyna` (const bool avec\_typeDeCal, const list< `EnumSousTypeCalcul` > &soustype, const list< bool > &avec\_soustypeDeCal, `UtilLecture` &entreePrinc)
- `AlgoriRelaxDyna` (const `AlgoriRelaxDyna` &algo)
- `Algori * New_idem` (const `Algori` \*algo) const
- void `Execution` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*varExp, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `InitAlgorithme` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `MiseAJourAlgo` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `CalEquilibre` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*, `Tableau`< `Fonction_nD` > \*tb\_combiner)
- void `FinCalcul` (`ParaGlob` \*, `LesMaillages` \*, `LesReferences` \*, `LesCourbes1D` \*, `LesFonctions_nD` \*, `VariablesExporter` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*)
- void `SchemaXML_Algori` (ofstream &sort, const `Enum_IO_XML` enu) const

## Fonctions membres protégées

- void `lecture_Parametres` (`UtilLecture` &entreePrinc)
- void `Ecrit_Base_info_Parametre` (`UtilLecture` &entreePrinc, const int &cas)
- void `Lecture_Base_info_Parametre` (`UtilLecture` &entreePrinc, const int &cas, bool choix)
- void `Info_commande_parametres` (`UtilLecture` &entreePrinc)
- bool `Gestion_pas_de_temps` (bool modif\_pas\_de\_temps, `LesMaillages` \*lesMail, int cas)
- bool `ActionInteractiveAlgo` ()
- void `InitCalculMatriceMasse` (`LesMaillages` \*lesMail, `Mat_abstraite` &mat\_mass, `Assemblage` &Ass, `Mat_abstraite` &mat\_mass\_sauve, `LesFonctions_nD` \*lesFonctionsnD)
- void `CalculMatriceMasse` (const int &relax\_vit\_acce, `LesMaillages` \*lesMail, `Assemblage` &Ass, int compteur, const `DiversStockage` \*diversStockage, `LesReferences` \*lesRef, const Enum\_ddl &N\_ddl, `LesContacts` \*lescontacts, `LesFonctions_nD` \*lesFonctionsnD)
- void `CalculEnContinuMatriceMasse` (const int &relax\_vit\_acce, `LesMaillages` \*lesMail, `Assemblage` &Ass, int compteur, const `DiversStockage` \*diversStockage, `LesReferences` \*lesRef, const Enum\_ddl &N\_ddl, bool premier\_calcul, `LesContacts` \*lescontacts, bool &force\_recalcul\_masse, `LesFonctions_nD` \*lesFonctionsnD)
- bool `Cinetique_ou_visqueux` (bool &force\_recalcul\_masse)
- double `Calcul_Lambda` ()
- virtual void `Repercussion_algo_sur_cinematique` (`LesContacts` \*lesContacts, `Vecteur` &Xtdt, `Vecteur` &Vtdt)
- void `Transfert_ParaGlob_AMOR_CINET_VISQUEUX` () const

## Attributs protégés

- int `cL_a_chaque_iteration`
- double `delta_t`
- int `typeCalRelaxation`
- double `lambda`
- double `lambda_initial`
- `Ponderation_GGlobal` \* `niveauLambda_grandeurGlobale`
- `Ponderation` \* `niveauLambda_ddlEtendu`
- `Ponderation_temps` \* `niveauLambda_temps`
- double `lambda_min`
- double `lambda_max`
- double `delta_lambda`
- bool `pilotage_auto_lambda`
- `List_io`< int > `list_iter_relax`
- int `casMass_relax`
- `Tableau`< int > `option_recalcul_masse`
- `Tableau`< `Fonction_nD` \* > `fct_nD_option_recalcul_masse`
- `Tableau`< string > `nom_fct_nD_option_recalcul_masse`
- double `alph`
- double `beta`
- double `gamma`
- double `theta`
- int `ncycle_calcul`
- int `type_calcul_masse`
- double `fac_epsilon`
- double `proportion_cinetique`
- bool `visqueux_activer`
- int `et_recalcul_masse_a_la_transition`
- `Ponderation_GGlobal` \* `niveauF_grandeurGlobale`
- `Ponderation` \* `niveauF_ddlEtendu`
- `Ponderation_temps` \* `niveauF_temps`
- int `type_activation_contact`
- int `choix_mini_masse_nul`
- `Assemblage` \* `Ass1_`
- `Assemblage` \* `Ass2_`
- `Assemblage` \* `Ass3_`
- int `cas_combi_ddl`
- int `icas`
- int `compteur`
- bool `prepa_avec_remont`

- bool **brestart**
- `OrdreVisu::EnumTypeIncre` **type\_incre**
- int **compteur\_demarrage**
- Vecteur **vglobin**
- Vecteur **vglobex**
- Vecteur **vglobaal**
- Vecteur **vcontact**
- Vecteur **save\_X\_t**
- Vecteur **X\_BI**
- Vecteur **V\_BI**
- Vecteur **G\_BI**
- Vecteur **forces\_vis\_num**
- `list< LesCondLim::Gene_asso >` **li\_gene\_asso**
- `Tableau< Nb_assemb >` **t\_assemb**
- `Tableau< Enum_ddl >` **tenuXVG**
- `Mat_abstraite * mat_masse`
- `Mat_abstraite * mat_masse_sauve`
- `Mat_abstraite * mat_C_pt`
- Vecteur **v\_mass**
- Vecteur **v\_mass1**
- `Tableau< Coordonnee >` **sortie\_mass\_noeud**
- `Mat_abstraite * matglob`

### Attributs protégés statiques

- static `Ddl_enum_etendu` **masseRelax\_1**
- static `TypeQuelconque` **masseRelax\_2**

### Membres hérités additionnels

#### 6.21.1 Description détaillée

BUT: Algorithme de relaxation dynamique selon la méthode de Barnes, modifiée par Julien Troufflard puis généralisée.

#### 6.21.2 Documentation des fonctions membres

##### 6.21.2.1 CalEquilibre()

```
void AlgoriRelaxDyna::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

Implémente [Algori](#).

### 6.21.2.2 Ecrit\_Base\_info\_Parametre()

```
void AlgoriRelaxDyna::Ecrit_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.21.2.3 Execution()

```
void AlgoriRelaxDyna::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.21.2.4 FinCalcul()

```
void AlgoriRelaxDyna::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.21.2.5 Info\_commande\_parametres()

```
void AlgoriRelaxDyna::Info_commande_parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.21.2.6 InitAlgorithme()

```
void AlgoriRelaxDyna::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.21.2.7 Lecture\_Base\_info\_Parametre()

```
void AlgoriRelaxDyna::Lecture_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas,
    bool choix ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.21.2.8 lecture\_Parametres()

```
void AlgoriRelaxDyna::lecture_Parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Réimplémentée à partir de [Algori](#).



### 6.21.2.9 MiseAJourAlgo()

```
void AlgoriRelaxDyna::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.21.2.10 New\_idem()

```
Algori * AlgoriRelaxDyna::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.21.2.11 Repercussion\_algo\_sur\_cinematique()

```
void AlgoriRelaxDyna::Repercussion_algo_sur_cinematique (
    LesContacts * lesContacts,
    Vecteur & Xtdt,
    Vecteur & Vtdt ) [protected], [virtual]
```

Réimplémentée à partir de [Algori](#).

### 6.21.2.12 SchemaXML\_Algori()

```
void AlgoriRelaxDyna::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

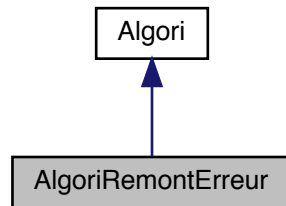
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori\_↔relax\_dyna.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori\_↔relax\_dyna.cc

## 6.22 Référence de la classe AlgoriRemontErreur

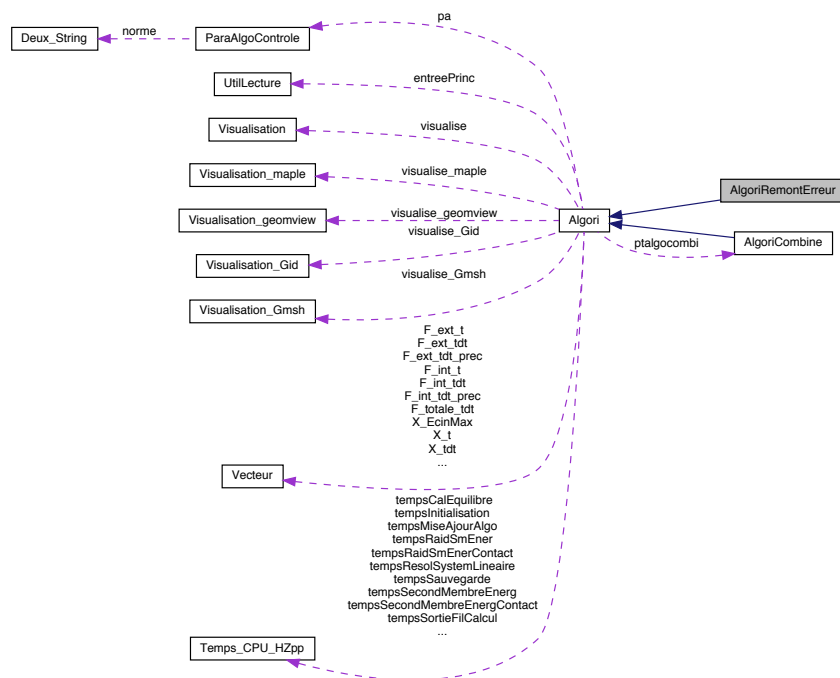
BUT: Algorithme de calcul de la remontée des contraintes aux noeuds, puis de calcul d'erreur, après un calcul de mécanique.

```
#include <AlgoriRemontErreur.h>
```

Graphe d'héritage de AlgoriRemontErreur:



Graphe de collaboration de AlgoriRemontErreur:



### Fonctions membres publiques

- **AlgoriRemontErreur** (**EnumTypeCalcul** type, **UtilLecture** &entreePrinc)
- void **Execution** (**ParaGlob** \*, **LesMaillages** \*, **LesReferences** \*, **LesCourbes1D** \*, **LesFonctions\_nD** \*, **VariablesExporter** \*, **LesLoisDeComp** \*, **DiversStockage** \*, **Charge** \*, **LesCondLim** \*, **LesContacts** \*, **Resultats** \*)

## Membres hérités additionnels

### 6.22.1 Description détaillée

BUT: Algorithme de calcul de la remontée des contraintes aux noeuds, puis de calcul d'erreur, après un calcul de mécanique.

### 6.22.2 Documentation des fonctions membres

#### 6.22.2.1 Execution()

```
void AlgoriRemontErreur::Execution (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

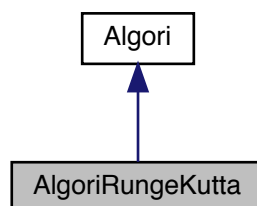
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/AlgoriRemontErreur.h

## 6.23 Référence de la classe AlgoriRungeKutta

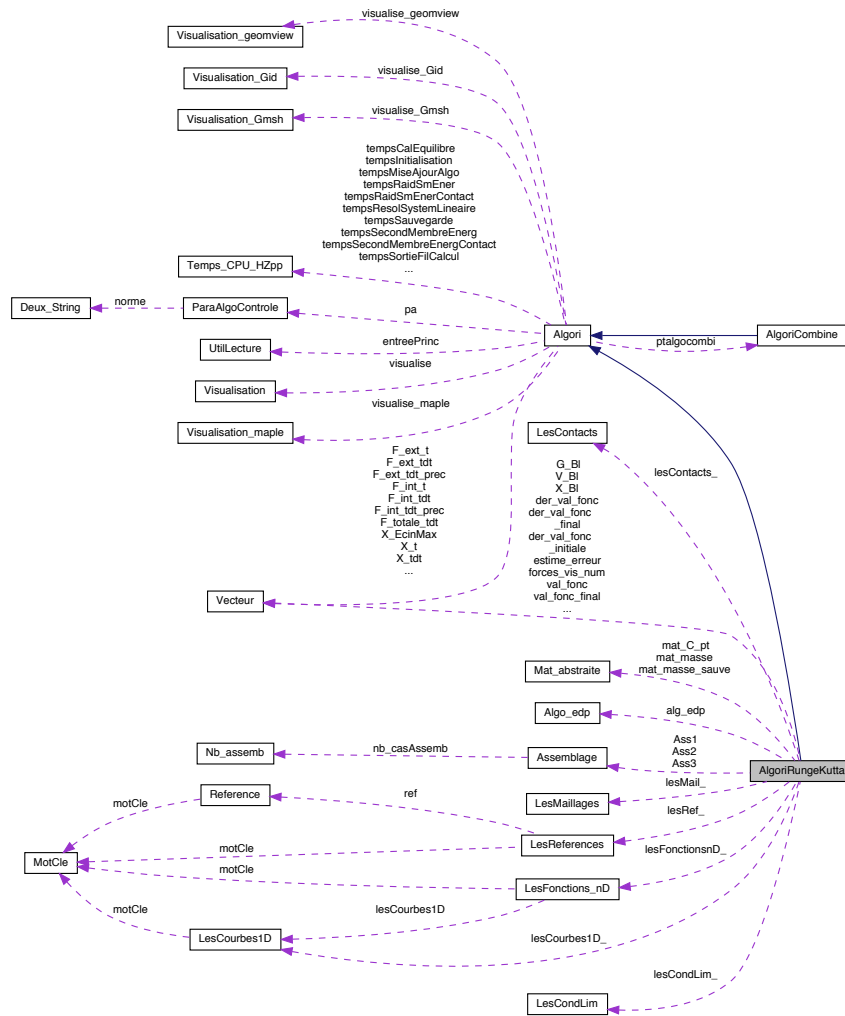
BUT: Algorithme de calcul dynamique, méthode de Runge Kutta, pour de la mécanique du solide.

```
#include <AlgoRungeKutta.h>
```

Graphe d'héritage de AlgoriRungeKutta:



Graphe de collaboration de AlgoriRungeKutta:



## Fonctions membres publiques

- `AlgoriRungeKutta` (const bool avec\_typeDeCal, const list< [EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCal, [UtilLecture](#) &entreePrinc)
- `AlgoriRungeKutta` (const [AlgoriRungeKutta](#) &algo)
- `Algori * New_idem` (const [Algori](#) \*algo) const
- void `Execution` ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*varExpor, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void `InitAlgorithme` ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void `MiseAJourAlgo` ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void `CalEquilibre` ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [Tableau](#)< [Fonction\\_nD](#) \* > \*tb\_combiner)

- void `FinCalcul` (`ParaGlob *`, `LesMaillages *`, `LesReferences *`, `LesCourbes1D *`, `LesFonctions_nD *`, `VariablesExporter *`, `LesLoisDeComp *`, `DiversStockage *`, `Charge *`, `LesCondLim *`, `LesContacts *`, `Resultats *`)
- void `SchemaXML_Algori` (`ofstream &sort`, `const Enum_IO_XML enu`) `const`

### Fonctions membres protégées

- void `lecture_Parametres` (`UtilLecture &entreePrinc`)
- void `Ecrit_Base_info_Parametre` (`UtilLecture &entreePrinc`, `const int &cas`)
- void `Lecture_Base_info_Parametre` (`UtilLecture &entreePrinc`, `const int &cas`, `bool choix`)
- void `Info_commande_parametres` (`UtilLecture &entreePrinc`)
- bool `Gestion_pas_de_temps` (`bool modif_pas_de_temps`, `LesMaillages *lesMail`, `int cas`, `int nbstep`)
- void `Modif_transi_pas_de_temps` (`const double &delta_tau`)
- bool `ActionInteractiveAlgo` ()
- `Vecteur & Dyna_point` (`const double &tau`, `const Vecteur &val_fonc`, `Vecteur &der_val_fonc`, `int &erreur`)
- void `Verif_integrite_Solution` (`const double &tau`, `const Vecteur &val_fonc`, `int &erreur`)

### Attributs protégés

- double `delta_t`
- double `unsurdeltat`
- double `deltatSurDeux`
- double `deltat2`
- `LesMaillages * lesMail_`
- `LesReferences * lesRef_`
- `LesCourbes1D * lesCourbes1D_`
- `LesFonctions_nD * lesFonctionsnD_`
- `Charge * charge_`
- `LesCondLim * lesCondLim_`
- `LesContacts * lesContacts_`
- `Assemblage * Ass1`
- `Assemblage * Ass2`
- `Assemblage * Ass3`
- double `maxPuissExt`
- double `maxPuissInt`
- double `maxReaction`
- int `inReaction`
- int `inSol`
- double `maxDeltaDdl`
- int `cas_combi_ddl`
- int `icas`
- bool `erreurSecondMembre`
- bool `prepa_avec_remont`
- bool `brestart`
- `OrdreVisu::EnumTypeIncre type_incre`
- `Vecteur vglobin`
- `Vecteur vglobex`
- `Vecteur vglobaal`
- `Vecteur vcontact`
- `Vecteur X_BI`
- `Vecteur V_BI`
- `Vecteur G_BI`
- `Vecteur forces_vis_num`
- `list< LesCondLim::Gene_asso > li_gene_asso`
- `Tableau< Nb_assemb > t_assemb`
- `Tableau< Enum_ddl > tenuXVG`
- `Mat_abstraite * mat_masse`
- `Mat_abstraite * mat_masse_sauve`
- `Mat_abstraite * mat_C_pt`
- `Algo_edp alg_edp`
- int `cas_kutta`

- double **erreurAbsolue**
- double **erreurRelative**
- double **erreur\_maxi\_global**
- int **nbMaxiAppel**
- bool **pilotage\_un\_step**
- [Vecteur](#) **estime\_erreur**
- [Vecteur](#) **val\_fonc\_initiale**
- [Vecteur](#) **der\_val\_fonc\_initiale**
- [Vecteur](#) **val\_fonc**
- [Vecteur](#) **der\_val\_fonc**
- [Vecteur](#) **val\_fonc\_final**
- [Vecteur](#) **der\_val\_fonc\_final**
- double **scale\_fac**

## Membres hérités additionnels

### 6.23.1 Description détaillée

BUT: Algorithme de calcul dynamique, méthode de Runge Kutta, pour de la mecanique du solide.

### 6.23.2 Documentation des fonctions membres

#### 6.23.2.1 CalEquilibre()

```
void AlgoriRungeKutta::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

!!!!!! n'est prévu pour l'instant que pour l'arrêt en déplacement, !!!!!

Implémente [Algori](#).

#### 6.23.2.2 Ecrit\_Base\_info\_Parametre()

```
void AlgoriRungeKutta::Ecrit_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.23.2.3 Execution()

```
void AlgoriRungeKutta::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.23.2.4 FinCalcul()

```
void AlgoriRungeKutta::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.23.2.5 Info\_commande\_parametres()

```
void AlgoriRungeKutta::Info_commande_parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.23.2.6 InitAlgorithme()

```
void AlgoriRungeKutta::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.23.2.7 Lecture\_Base\_info\_Parametre()

```
void AlgoriRungeKutta::Lecture_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas,
    bool choix ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.23.2.8 lecture\_Parametres()

```
void AlgoriRungeKutta::lecture_Parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Réimplémentée à partir de [Algori](#).

### 6.23.2.9 MiseAJourAlgo()

```
void AlgoriRungeKutta::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).



### 6.23.2.10 New\_idem()

```
Algori * AlgoriRungeKutta::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.23.2.11 SchemaXML\_Algori()

```
void AlgoriRungeKutta::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

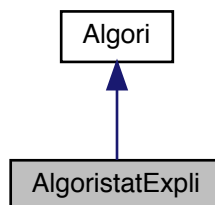
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algo↔  
RungeKutta.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algo↔  
RungeKutta.cc

## 6.24 Référence de la classe AlgoristatExpli

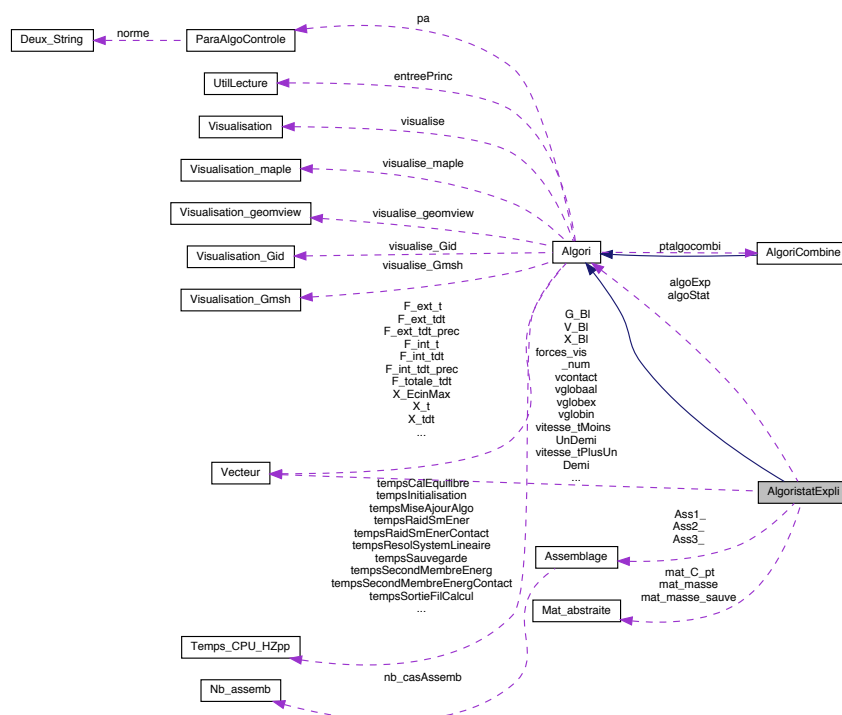
BUT: Algorithme de calcul mixte: statique - pseudo-dynamique explicite, pour de la mecanique du solide déformable en coordonnees materielles entrainees.

```
#include <AlgoriMixte.h>
```

Graphe d'héritage de AlgoristatExpli:



Graphe de collaboration de AlgoristatExpli:



## Fonctions membres publiques

- **AlgoristatExpli** (const bool avec\_typeDeCal, const list< EnumSousTypeCalcul > &soustype, const list< bool > &avec\_soustypeDeCal, UtilLecture &entreePrinc)
- **AlgoristatExpli** (const AlgoristatExpli &algo)
- **Algori** \* **New\_idem** (const Algori \*algo) const
- void **Execution** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*varExpor, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **InitAlgorithme** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **MiseAJourAlgo** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **CalEquilibre** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*, Tableau< Fonction\_nD > \*tb\_combiner)
- void **FinCalcul** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **SchemaXML\_Algori** (ofstream &sort, const Enum\_IO\_XML enu) const  
*\*\*\*\* inexploitable pour l'instant*

## Fonctions membres protégées

- void **Calcul\_Equilibre** (ParaGlob \*paraGlob, LesMaillages \*lesMail, LesReferences \*lesRef, LesCourbes1D \*lesCourbes1D, LesFonctions\_nD \*lesFonctionsnD, VariablesExporter \*varExpor, LesLoisDeComp \*lesLoisDeComp, DiversStockage \*diversStockage, Charge \*charge, LesCondLim \*lesCondLim, LesContacts \*lesContacts, Resultats \*resultats)

- void **Calcul\_Equilibre2** (**ParaGlob** \*paraGlob, **LesMaillages** \*lesMail, **LesReferences** \*lesRef, **LesCourbes1D** \*lesCourbes1D, **LesFonctions\_nD** \*lesFonctionsnD, **VariablesExporter** \*varExpor, **LesLoisDeComp** \*lesLoisDeComp, **DiversStockage** \*diversStockage, **Charge** \*charge, **LesCondLim** \*lesCondLim, **LesContacts** \*lesContacts, **Resultats** \*resultats)
  - \*\*\*\* *inexploitable pour l'instant*
- void **lecture\_Parametres** (**UtilLecture** &entreePrinc)
  - \*\*\*\* *inexploitable pour l'instant*
- void **Ecrit\_Base\_info\_Parametre** (**UtilLecture** &entreePrinc, const int &cas)
- void **Lecture\_Base\_info\_Parametre** (**UtilLecture** &entreePrinc, const int &cas, bool choix)
- void **Info\_commande\_parametres** (**UtilLecture** &entreePrinc)
  - \*\*\*\* *inexploitable pour l'instant*
- void **Gestion\_pas\_de\_temps** (**LesMaillages** \*lesMail, int cas)
- bool **ActionInteractiveAlgo** ()
  - \*\*\*\* *inexploitable pour l'instant*

## Attributs protégés

- **Algori** \* **algoStat**
- **Algori** \* **algoExp**
- **Vecteur** **vitesse\_tMoinsUnDemi**
- **Vecteur** **vitesse\_tPlusUnDemi**
- double **delta\_t**
- double **unsurdeltat**
- double **deltatSurDeux**
- double **deltat2**
- double **deltat2SurDeux**
- int **type\_cal\_equilibre**
- **Assemblage** \* **Ass1\_**
- **Assemblage** \* **Ass2\_**
- **Assemblage** \* **Ass3\_**
- int **cas\_combi\_ddl**
- int **icas**
- bool **prepa\_avec\_remont**
- bool **brestart**
- **OrdreVisu::EnumTypeIncre** **type\_incre**
- **Vecteur** **vglobin**
- **Vecteur** **vglobex**
- **Vecteur** **vglobaal**
- **Vecteur** **vcontact**
- **Vecteur** **X\_BI**
- **Vecteur** **V\_BI**
- **Vecteur** **G\_BI**
- **Vecteur** **forces\_vis\_num**
- list< **LesCondLim::Gene\_asso** > **li\_gene\_asso**
- **Tableau**< **Nb\_assemb** > **t\_assemb**
- **Tableau**< **Enum\_ddl** > **tenuXVG**
- **Mat\_abstraite** \* **mat\_masse**
- **Mat\_abstraite** \* **mat\_masse\_sauve**
- **Mat\_abstraite** \* **mat\_C\_pt**

## Membres hérités additionnels

### 6.24.1 Description détaillée

BUT: Algorithme de calcul mixte: statique - pseudo-dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées.

## 6.24.2 Documentation des fonctions membres

### 6.24.2.1 CalEquilibre()

```
void AlgoristatExpli::CalEquilibre (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    Tableau< Fonction_nD * > * tb_combiner ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.24.2.2 Ecrit\_Base\_info\_Parametre()

```
void AlgoristatExpli::Ecrit_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.24.2.3 Execution()

```
void AlgoristatExpli::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.24.2.4 FinCalcul()

```
void AlgoristatExpli::FinCalcul (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

#### 6.24.2.5 Info\_commande\_parametres()

```
void AlgoristatExpli::Info_commande_parametres (
    UtilLecture & entreePrinc ) [inline], [protected], [virtual]
```

\*\*\*\* inexploitable pour l'instant

Implémente [Algori](#).

#### 6.24.2.6 InitAlgorithme()

```
void AlgoristatExpli::InitAlgorithme (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.24.2.7 Lecture\_Base\_info\_Parametre()

```
void AlgoristatExpli::Lecture_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas,
    bool choix ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.24.2.8 lecture\_Parametres()

```
void AlgoristatExpli::lecture_Parametres (
    UtilLecture & entreePrinc ) [inline], [protected], [virtual]
```

\*\*\*\* inexploitable pour l'instant

Réimplémentée à partir de [Algori](#).

### 6.24.2.9 MiseAJourAlgo()

```
void AlgoristatExpli::MiseAJourAlgo (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.24.2.10 New\_idem()

```
Algori * AlgoristatExpli::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.24.2.11 SchemaXML\_Algori()

```
void AlgoristatExpli::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [inline], [virtual]
```

\*\*\*\* inexploitable pour l'instant

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

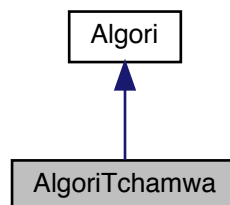
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoMixte/AlgoriMixte.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoMixte/AlgoriMixte.cc

## 6.25 Référence de la classe AlgoriTchamwa

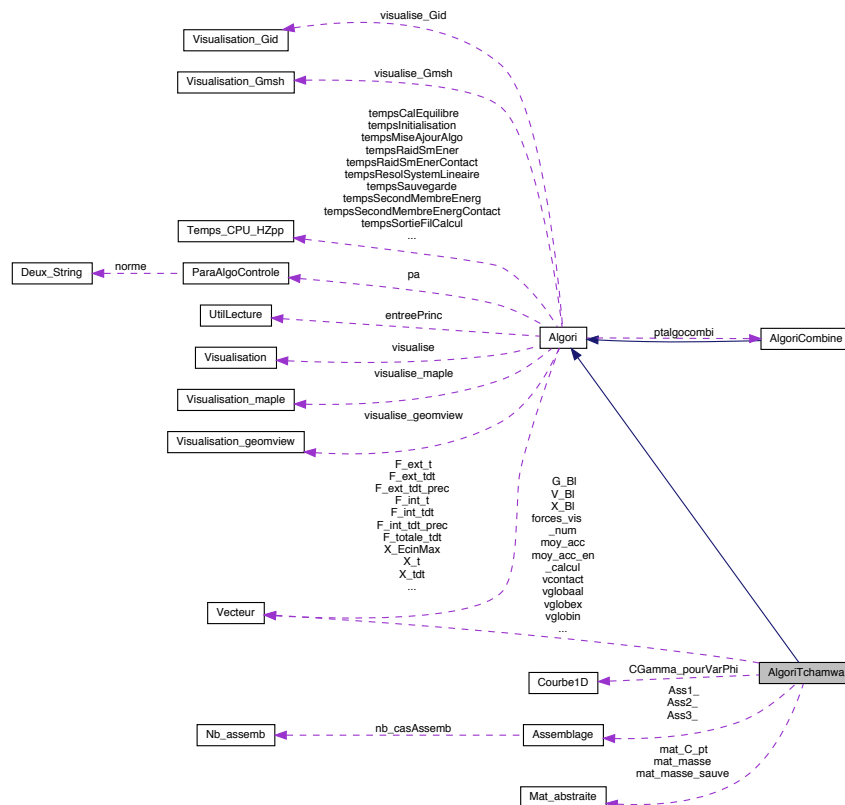
BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées. On ne prend pas en compte les phénomènes de contact. Correspond à l'implantation de l'algo de wielgosz tchamwa.

```
#include <Algori_tchamwa.h>
```

Graphe d'héritage de AlgoriTchamwa:



Graphe de collaboration de AlgoriTchamwa:



## Fonctions membres publiques

- **AlgoriTchamwa** (const bool avec\_typeDeCal, const list< EnumSousTypeCalcul > &soustype, const list< bool > &avec\_soustypeDeCal, UtilLecture &entreePrinc)
- **AlgoriTchamwa** (const AlgoriTchamwa &algo)
- **Algori** \* New\_idem (const Algori \*algo) const
- void Execution (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*varExpor, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void InitAlgorithme (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void MiseAJourAlgo (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void CalEquilibre (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*, Tableau< Fonction\_nD > \*tb\_combiner)
- void FinCalcul (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void SchemaXML\_Algori (ofstream &sort, const Enum\_IO\_XML enu) const

## Fonctions membres protégées

- void Calcul\_Equilibre2 (ParaGlob \*paraGlob, LesMaillages \*lesMail, LesReferences \*lesRef, LesCourbes1D \*lesCourbes1D, LesFonctions\_nD \*lesFonctionsnD, VariablesExporter \*varExpor,



- LesLoisDeComp \*lesLoisDeComp, DiversStockage \*diversStockage, Charge \*charge, LesCondLim \*les←  
CondLim, LesContacts \*lesContacts, Resultats \*resultats)
- void **Calcul\_Equilibre4** (ParaGlob \*paraGlob, LesMaillages \*lesMail, LesReferences \*lesRef,  
LesCourbes1D \*lesCourbes1D, LesFonctions\_nD \*lesFonctionsnD, VariablesExporter \*varExpor,  
LesLoisDeComp \*lesLoisDeComp, DiversStockage \*diversStockage, Charge \*charge, LesCondLim \*les←  
CondLim, LesContacts \*lesContacts, Resultats \*resultats)
- void **lecture\_Parametres** (UtilLecture &entreePrinc)
- void **Ecrit\_Base\_info\_Parametre** (UtilLecture &entreePrinc, const int &cas)
- void **Lecture\_Base\_info\_Parametre** (UtilLecture &entreePrinc, const int &cas, bool choix)
- void **Info\_commande\_parametres** (UtilLecture &entreePrinc)
- bool **Gestion\_pas\_de\_temps** (bool modif\_pas\_de\_temps, LesMaillages \*lesMail, int cas, double nouveau←  
\_dt)
- bool **ActionInteractiveAlgo** ()

## Attributs protégés

- double \* **phi\_1**
- const double **alphaa**
- const double **gammaa**
- const double **lambdaa**
- double **betaa**
- double **delta\_t**
- double **delta\_t\_critique**
- double **deltat2**
- double **unsurdeltat**
- double **lambdaDeltat**
- double **alphaDelta\_t**
- double **betaDeltat2**
- double **gammaDelta\_t**
- int **type\_cal\_equilibre**
- Courbe1D \* **CGamma\_pourVarPhi**
- string **nom\_courbe\_CGamma\_pourVarPhi**
- int **npas\_moyacc**
- int **npas\_effectue**
- double **valmin**
- double **valmax**
- Vecteur **moy\_acc**
- Vecteur **moy\_acc\_en\_calcul**
- int **degre\_num**
- int **degre\_deno**
- double **type4\_inc\_deb**
- double **type4\_inc\_deplace**
- Assemblage \* **Ass1\_**
- Assemblage \* **Ass2\_**
- Assemblage \* **Ass3\_**
- int **cas\_combi\_ddl**
- int **icas**
- bool **prepa\_avec\_remont**
- bool **brestart**
- OrdreVisu::EnumTypeIncre **type\_incre**
- Vecteur **vglobin**
- Vecteur **vglobex**
- Vecteur **vglobaal**
- Vecteur **vcontact**
- Vecteur **X\_BI**
- Vecteur **V\_BI**
- Vecteur **G\_BI**
- Vecteur **forces\_vis\_num**
- list< LesCondLim::Gene\_asso > **li\_gene\_asso**
- Tableau< Nb\_assemb > **t\_assemb**
- Tableau< Enum\_ddl > **tenuXVG**
- Mat\_abstraite \* **mat\_masse**
- Mat\_abstraite \* **mat\_masse\_sauve**
- Mat\_abstraite \* **mat\_C\_pt**

## Membres hérités additionnels

### 6.25.1 Description détaillée

BUT: Algorithme de calcul dynamique explicite, pour de la mécanique du solide déformable en coordonnées matérielles entraînées. On ne prend pas en compte les phénomènes de contact. Correspond à l'implantation de l'algo de wielgosz tchamwa.

### 6.25.2 Documentation des fonctions membres

#### 6.25.2.1 CalEquilibre()

```
void AlgoriTchamwa::CalEquilibre (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    Tableau< Fonction_nD * > * tb_combiner ) [virtual]
```

Implémente [Algori](#).

#### 6.25.2.2 Ecrit\_Base\_info\_Parametre()

```
void AlgoriTchamwa::Ecrit_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.25.2.3 Execution()

```
void AlgoriTchamwa::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.25.2.4 FinCalcul()

```
void AlgoriTchamwa::FinCalcul (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.25.2.5 Info\_commande\_parametres()

```
void AlgoriTchamwa::Info_commande_parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.25.2.6 InitAlgorithme()

```
void AlgoriTchamwa::InitAlgorithme (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.25.2.7 Lecture\_Base\_info\_Parametre()

```
void AlgoriTchamwa::Lecture_Base_info_Parametre (
    UtilLecture & entreePrinc,
    const int & cas,
    bool choix ) [protected], [virtual]
```

Implémente [Algori](#).

### 6.25.2.8 lecture\_Parametres()

```
void AlgoriTchamwa::lecture_Parametres (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

Réimplémentée à partir de [Algori](#).

### 6.25.2.9 MiseAJourAlgo()

```
void AlgoriTchamwa::MiseAJourAlgo (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lescontacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.25.2.10 New\_idem()

```
Algori * AlgoriTchamwa::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.25.2.11 SchemaXML\_Algori()

```
void AlgoriTchamwa::SchemaXML_Algori (
    ofstream & sort,
    const Enum_IO_XML enu ) const [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

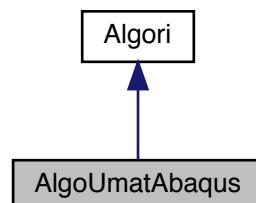
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori\_↔  
tchamwa.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori\_↔  
tchamwa.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoDynaExplicite/Algori\_↔  
tchamwa2.cc

## 6.26 Référence de la classe AlgoUmatAbaqus

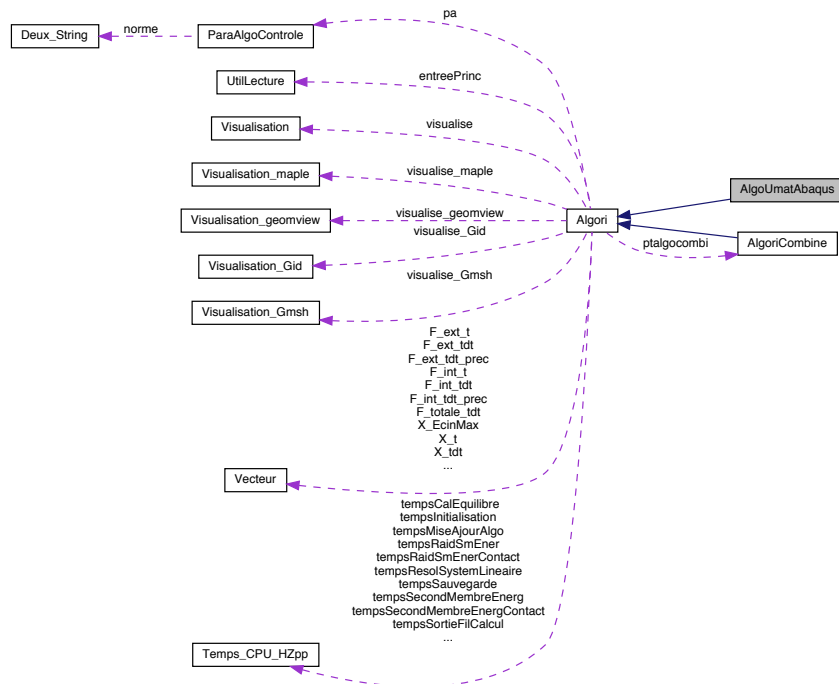
BUT: Algorithme de calcul permettant l'utilisation d'herezh++ comme Umat pour Abaqus.

```
#include <AlgoUmatAbaqus.h>
```

Grappe d'héritage de AlgoUmatAbaqus:



Graphe de collaboration de AlgoUmatAbaqus:



## Fonctions membres publiques

- **AlgoUmatAbaqus** (const bool avec\_typeDeCal, const list< EnumSousTypeCalcul > &soustype, const list< bool > &avec\_soustypeDeCal, UtilLecture &entreePrinc)
- **AlgoUmatAbaqus** (const AlgoUmatAbaqus &algo)
- **Algori \* New\_idem** (const Algori \*algo) const
- void **Execution** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*varExpor, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **InitAlgorithme** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **MiseAJourAlgo** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **CalEquilibre** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*, Tableau< Fonction\_nD \* > \*tb\_combiner)
- void **FinCalcul** (ParaGlob \*, LesMaillages \*, LesReferences \*, LesCourbes1D \*, LesFonctions\_nD \*, VariablesExporter \*, LesLoisDeComp \*, DiversStockage \*, Charge \*, LesCondLim \*, LesContacts \*, Resultats \*)
- void **SchemaXML\_Algori** (ofstream &, const Enum\_IO\_XML) const

## Membres hérités additionnels

### 6.26.1 Description détaillée

BUT: Algorithme de calcul permettant l'utilisation d'herezh++ comme Umat pour Abaqus.

## 6.26.2 Documentation des fonctions membres

### 6.26.2.1 CalEquilibre()

```
void AlgoUmatAbaqus::CalEquilibre (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    Tableau< Fonction_nD * > * tb_combiner ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.26.2.2 Execution()

```
void AlgoUmatAbaqus::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.26.2.3 FinCalcul()

```
void AlgoUmatAbaqus::FinCalcul (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.26.2.4 InitAlgorithme()

```
void AlgoUmatAbaqus::InitAlgorithme (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.26.2.5 MiseAJourAlgo()

```
void AlgoUmatAbaqus::MiseAJourAlgo (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).



### 6.26.2.6 New\_idem()

```
Algori * AlgoUmatAbaqus::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.26.2.7 SchemaXML\_Algori()

```
void AlgoUmatAbaqus::SchemaXML_Algori (
    ofstream & ,
    const Enum_IO_XML ) const [inline], [virtual]
```

Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

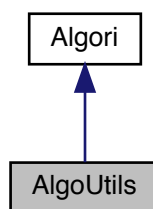
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/AlgoUmatAbaqus.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/AlgoUmatAbaqus.cc

## 6.27 Référence de la classe AlgoUtils

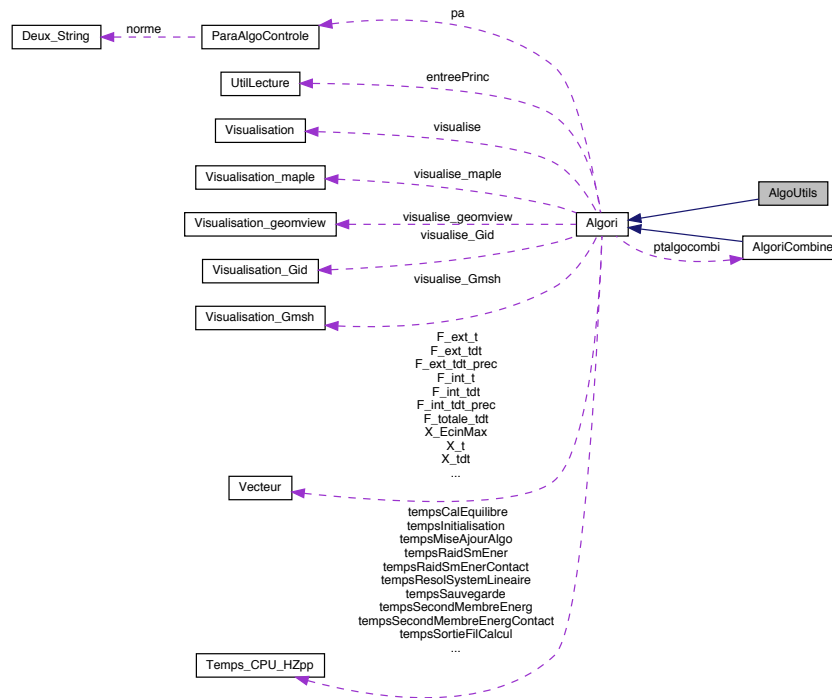
BUT: Utilitaires divers. ex: transformation de maillage, quadratique incomplet en quadratique complet.

```
#include <AlgoUtils.h>
```

Graphe d'héritage de AlgoUtils:



Graphe de collaboration de AlgoUtils:



## Fonctions membres publiques

- **AlgoUtils** (const bool avec\_typeDeCal, const list< [EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCal, [UtilLecture](#) &entreePrinc)
- **AlgoUtils** (const [AlgoUtils](#) &algo)
- **Algori** \* **New\_idem** (const [Algori](#) \*algo) const
- void **Execution** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*varExpor, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **InitAlgorithme** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **MiseAJourAlgo** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **CalEquilibre** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [Tableau](#)< [Fonction\\_nD](#) > \*tb\_combiner)
- void **FinCalcul** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **SchemaXML\_Algori** (ofstream &, const [Enum\\_IO\\_XML](#)) const

## Membres hérités additionnels

### 6.27.1 Description détaillée

BUT: Utilitaires divers. ex: transformation de maillage, quadratique incomplet en quadratique complet.

## 6.27.2 Documentation des fonctions membres

### 6.27.2.1 CalEquilibre()

```
void AlgoUtils::CalEquilibre (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    Tableau< Fonction_nD * > * tb_combiner ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.27.2.2 Execution()

```
void AlgoUtils::Execution (
    ParaGlob * p,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

### 6.27.2.3 FinCalcul()

```
void AlgoUtils::FinCalcul (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.27.2.4 InitAlgorithme()

```
void AlgoUtils::InitAlgorithme (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.27.2.5 MiseAJourAlgo()

```
void AlgoUtils::MiseAJourAlgo (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]
```

Implémente [Algori](#).

### 6.27.2.6 New\_idem()

```
Algori * AlgoUtils::New_idem (
    const Algori * algo ) const [inline], [virtual]
```

Implémente [Algori](#).

### 6.27.2.7 SchemaXML\_Algori()

```
void AlgoUtils::SchemaXML_Algori (
    ofstream & ,
    const Enum_IO_XML ) const [inline], [virtual]
```

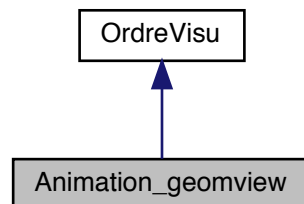
Implémente [Algori](#).

La documentation de cette classe a été générée à partir du fichier suivant :

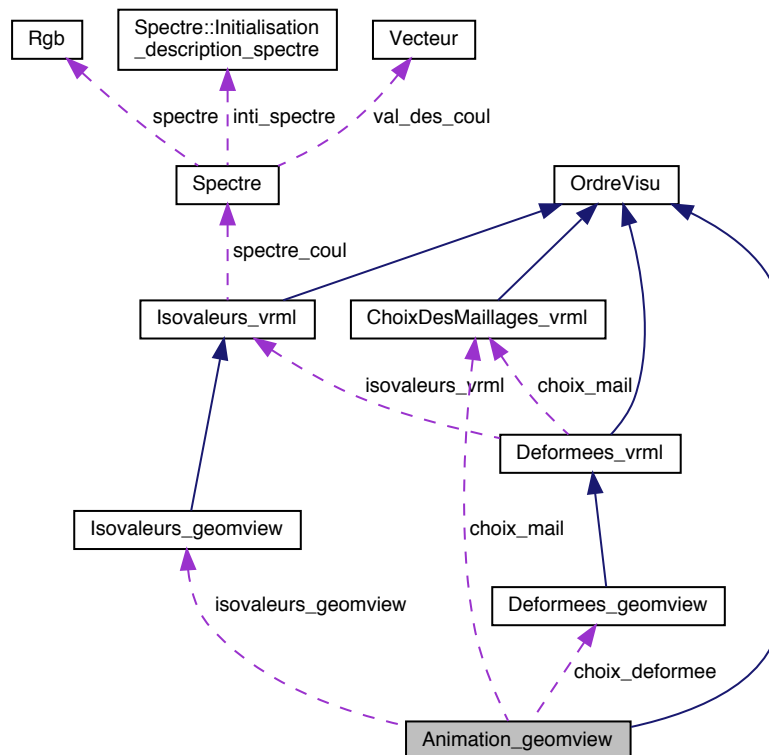
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/AlgoUtils.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoUtilitaires/AlgoUtils.cc

## 6.28 Référence de la classe Animation\_geomview

Grappe d'héritage de Animation\_geomview:



Graphe de collaboration de Animation\_geomview:



## Fonctions membres publiques

- **Animation\_geomview** (const [Animation\\_geomview](#) &ord)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [EnumTypeIncre](#) type\_incre, int incre, bool animation, const [map](#)< string, const double \*, [std::less](#)< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **ChoixOrdre** ()
- void **Jonction\_isovaleur** (const [Isovaleurs\\_geomview](#) \*iso)
- void **Jonction\_ChoixDesMaillages** (const [ChoixDesMaillages\\_vrml](#) \*choix\_m)
- void **Jonction\_Deformees\_geomview** (const [Deformees\\_geomview](#) \*choix\_def)
- void **Init\_liste\_cordonnee** ()
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

## Attributs protégés

- double **cycleInterval**
- bool **loop**
- double **startTime**
- double **stopTime**
- const [Isovaleurs\\_geomview](#) \* **isovaleurs\_geomview**
- const [ChoixDesMaillages\\_vrml](#) \* **choix\_mail**
- const [Deformees\\_geomview](#) \* **choix\_deformee**

## Membres hérités additionnels

### 6.28.1 Documentation des fonctions membres

#### 6.28.1.1 ChoixOrdre()

```
void Animation_geomview::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.28.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Animation_geomview::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.28.1.3 ExeOrdre()

```
void Animation_geomview::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * ,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.28.1.4 Lecture\_parametres\_OrdreVisu()

```
void Animation_geomview::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

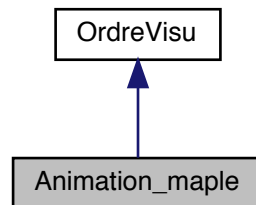
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

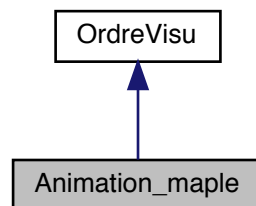
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Animation\_geomview.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Animation\_geomview.cc

## 6.29 Référence de la classe Animation\_maple

Graphe d'héritage de Animation\_maple:



Graphe de collaboration de Animation\_maple:





## Fonctions membres publiques

- **Animation\_maple** (const [Animation\\_maple](#) &ord)
- void **Initialisation** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesmail, [LesReferences](#) \*lesRefer, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, [Charge](#) \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, EnumTypeIncre type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob, bool fil\_calcul)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, EnumTypeIncre type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **ChoixOrdre** ()
- void **Jonction\_choix\_grandeurs\_maple** ([Choix\\_grandeurs\\_maple](#) \*choix)
- void **Jonction\_ChoixDesMaillages** (const [ChoixDesMaillages\\_vrml](#) \*choix\_m)
- void **Ajout\_courbe** ([List\\_io](#)< [DeuxDoubles](#) > &courbe)
- void **Entete\_fichier\_maple** (const list< int > &list\_mail, ostream &sort)
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

## Membres hérités additionnels

### 6.29.1 Documentation des fonctions membres

#### 6.29.1.1 ChoixOrdre()

```
void Animation_maple::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.29.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Animation_maple::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.29.1.3 ExeOrdre()

```
void Animation_maple::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * ,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.29.1.4 Initialisation()

```
void Animation_maple::Initialisation (
    ParaGlob * paraGlob,
    LesMaillages * lesmail,
    LesReferences * lesRefer,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    EnumTypeIncre type_incre,
    int incre,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob,
    bool fil_calcul ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.29.1.5 Lecture\_parametres\_OrdreVisu()

```
void Animation_maple::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

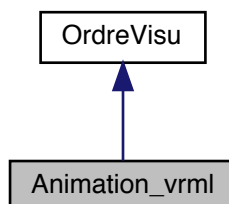
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

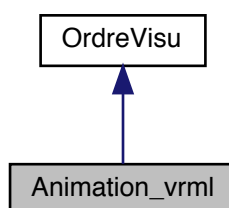
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Animation\_maple.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Animation\_maple.cc

## 6.30 Référence de la classe Animation\_vrml

Grphe d'héritage de Animation\_vrml:



Grphe de collaboration de Animation\_vrml:



### Fonctions membres publiques

- **Animation\_vrml** (const [Animation\\_vrml](#) &ord)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [EnumTypeIncre](#) type\_incre, int incre, bool animation, const map< string, const double >, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **ChoixOrdre** ()
- void **Jonction\_isevaleur** (const [Isovaleurs\\_vrml](#) \*iso)
- void **Jonction\_ChoixDesMaillages** (const [ChoixDesMaillages\\_vrml](#) \*choix\_m)
- void **Jonction\_Deformees\_vrml** (const [Deformees\\_vrml](#) \*choix\_def)
- void **Init\_liste\_cordonnee** ()
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

### Membres hérités additionnels

#### 6.30.1 Documentation des fonctions membres

### 6.30.1.1 ChoixOrdre()

```
void Animation_vrml::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.30.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Animation_vrml::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.30.1.3 ExeOrdre()

```
void Animation_vrml::ExeOrdre (
    ParaGlob * paraGlob,
    const Tableau< int > & tab_mail,
    LesMaillages * ,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.30.1.4 Lecture\_parametres\_OrdreVisu()

```
void Animation_vrml::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

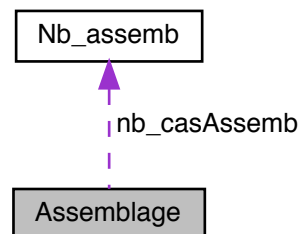
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Animation\_vrml.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Animation\_vrml.cc

## 6.31 Référence de la classe Assemblage

Graphe de collaboration de Assemblage:



### Fonctions membres publiques

- **Assemblage** ([Nb\\_assemb](#) nb\_cas)
- **Assemblage** (const [Assemblage](#) &a)
- void **AssemSM** ([Vecteur](#) &vecglob, const [Vecteur](#) &vecloc, const [DdlElement](#) &tab\_ddl, const [Tableau](#)<[Noeud](#) \* > &tab\_noeud)
- void **AssemSM** ([Tableau](#)< [Vecteur](#) > &vecglob, const [Tableau](#)< [Vecteur](#) \* > &vecloc, const [DdlElement](#) &tab\_ddl, const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- void **AssembMatSym** ([Mat\\_abstraite](#) &matglob, const [Mat\\_abstraite](#) &matloc, const [DdlElement](#) &tab\_ddl, const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- void **AssembMatnonSym** ([Mat\\_abstraite](#) &matglob, const [Mat\\_abstraite](#) &matloc, const [DdlElement](#) &tab\_ddl, const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- void **AssembDiagonale** ([Vecteur](#) &vecglob, const [Mat\\_abstraite](#) &matloc, const [DdlElement](#) &tab\_ddl, const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- void **AssembDiagoMajorValPropre** ([Vecteur](#) &vecglob, const [Mat\\_abstraite](#) &matloc, const [DdlElement](#) &tab\_ddl, const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- [Nb\\_assemb](#) **Nb\_cas\_assemb** ()
- void **Change\_cas\_assemblage** ([Nb\\_assemb](#) nb\_cas)

### Attributs protégés

- [Nb\\_assemb](#) **nb\_casAssemb**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution\_Condi/Assemblage.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution\_Condi/Assemblage.cc

## 6.32 Référence de la classe Banniere

### Fonctions membres publiques statiques

- static void **Sortie\_banniere** (ofstream &sort)
- static void **Sortie\_banniere** ()
- static void **Passage\_lecture\_banniere** (ifstream &entr)
- static string **CopiPirate** ()

## Amis

- class **Construc\_banniere**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/Banniere.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/banniere.cc

## 6.33 Référence de la classe Base3D3B

Les base covariantes et absolus.

```
#include <Base3D3.h>
```

### Fonctions membres publiques

- **Base3D3B** ()  
*CONSTRUCTEURS : par défaut, défini une Base de coordonnées nulles.*
- **Base3D3B** (bool)  
*defini une Base unitaire en dimension 3 :100,010,001*
- **Base3D3B** (const [Tableau](#)< [Coordonnee3B](#) > &v1)  
*defini une Base donnée a partir d'un tableau*
- **Base3D3B** (const [Coordonnee3B](#) &v1, const [Coordonnee3B](#) &v2, const [Coordonnee3B](#) &v3)  
*defini une Base donnée a partir de trois vecteur*
- **Base3D3B** (const [Base3D3B](#) &)  
*constructeur de copie*
- **~Base3D3B** ()  
*DESTRUCTEUR :*
- [Base3D3B](#) & **operator=** (const [Base3D3B](#) &aB)  
*METHODES PUBLIQUES : surcharge de l'affectation.*
- const int **Dimension** ()  
*retourne la dimension des Coordonnee3s*
- const int **NbCoordonnee** ()  
*retourne le nombre de Coordonnee3s de la base*
- [Coordonnee3B](#) & **operator()** (int i)  
*retourne le i ieme vecteur en I/O*
- [Coordonnee3B](#) **operator()** (int i) const  
*retourne le i ieme vecteur en lecture only*
- double & **operator()** (int i, int j)  
*retourne la j ieme composante du i ieme vecteur en I/O*
- double **operator()** (int i, int j) const  
*retourne la j ieme composante du i ieme vecteur en lecture only*

## Amis

- istream & **operator>>** (istream &, [Base3D3B](#) &)  
*surcharge de l'operator de lecture avec le type*
- ostream & **operator<<** (ostream &, const [Base3D3B](#) &)  
*surcharge de l'operator d'écriture*

### 6.33.1 Description détaillée

Les base covariantes et absolus.

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Reperes\_bases/Base3D3.h

## 6.34 Référence de la classe Base3D3H

Les base duales.

```
#include <Base3D3.h>
```

### Fonctions membres publiques

- **Base3D3H** ()  
*CONSTRUCTEURS : par défaut, défini une Base de coordonnées nulles.*
- **Base3D3H** (bool)  
*défini une Base unitaire en dimension 3 :100,010,001*
- **Base3D3H** (const [Tableau](#)< [Coordonnee3H](#) > &v1)  
*défini une Base donnée à partir d'un tableau*
- **Base3D3H** (const [Coordonnee3H](#) &v1, const [Coordonnee3H](#) &v2, const [Coordonnee3H](#) &v3)  
*défini une Base donnée à partir de trois vecteur*
- **Base3D3H** (const [Base3D3H](#) &)  
*constructeur de copie*
- **~Base3D3H** ()  
*DESTRUCTEUR :*
- [Base3D3H](#) & **operator=** (const [Base3D3H](#) &aB)  
*surcharge de l'affectation*
- const int **Dimension** ()  
*retourne la dimension des Coordonnee3s*
- const int **NbCoordonnee** ()  
*retourne le nombre de Coordonnee3s de la base*
- [Coordonnee3H](#) & **operator()** (int i)  
*retourne le i ieme vecteur en I/O*
- [Coordonnee3H](#) **operator()** (int i) const  
*retourne le i ieme vecteur en lecture only*
- double & **operator()** (int i, int j)  
*retourne la j ieme composante du i ieme vecteur en I/O*
- double **operator()** (int i, int j) const  
*retourne la j ieme composante du i ieme vecteur en lecture only*

### Amis

- istream & **operator>>** (istream &, [Base3D3H](#) &)  
*surcharge de l'operator de lecture avec le type*
- ostream & **operator<<** (ostream &, const [Base3D3H](#) &)  
*surcharge de l'operator d'écriture*

### 6.34.1 Description détaillée

Les base duales.

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Reperes\_bases/Base3D3.h

## 6.35 Référence de la classe BaseB

Les base covariantes et absolus.

```
#include <Base.h>
```

### Fonctions membres publiques

- **BaseB** ()  
*constructeur par défaut, defini la Base absolu en dimension 3 les vecteurs sont unitaires, ex en dim 2 -> la base : 1,0 et 0,1*
- **BaseB** (int dim)  
*defini la Base absolu en dimension dim*
- **BaseB** (int dim, const [Tableau](#)< [CoordonneeB](#) > &v1)  
*defini une Base relative v1(dim) c-a-d differente de la base triviale absolu*
- **BaseB** (int dim, int n)  
*defini une Base locale absolue (triviale) de n vecteurs de dimension dim*
- **BaseB** (int dim, int n, double s)  
*idem dessus mais avec toutes les composantes = s*
- **BaseB** (int dim, int n, const [Tableau](#)< [CoordonneeB](#) > &v1)  
*defini une Base locale relative v1(dim) de n vecteurs de dimension dim*
- **BaseB** (const [BaseB](#) &)  
*constructeur de copie*
- **~BaseB** ()  
*DESTRUCTEUR :*
- **BaseB & operator=** (const [BaseB](#) &aB)  
*surcharge de l'affectation*
- const int **Dimension** () const
- const int **NbVecteur** () const  
*retourne le nombre de vecteurs de la base*
- const [CoordonneeB](#) & **operator()** (int i) const  
*retourne le i ieme vecteurs en lecture only*
- [CoordonneeB](#) & **CoordoB** (int i)  
*retourne le i ieme vecteurs en I/O*
- const [Coordonnee](#) & **Coordo** (int i) const  
*acces directe contrôlé à des vecteurs sans variance: uniquement en lecture*
- void **Affiche** () const  
*affichage à l'écran des infos*
- void **Affectation\_trans\_variance** (const [BaseH](#) &aH)  
*affectation trans\_variance: utile pour une recopie de mêmes valeurs mais ici l'appel est explicite donc a priori on sait ce que l'on fait il n'y a pas de redimensionnement, donc la dimension et le nombre des vecteurs doivent être identiques*
- void **Change\_repere** (const [BaseH](#) &lpH, [BaseB](#) &apB)  
*changement de base on suppose que this(i) correspond aux coordonnées dans un premier repère L\_a lpH correspond aux coordonnées dans L\_a d'un nouveau repère en sortie: apB(i) correspond aux coordonnées dans ipB de this(i)*
- void **ChangeBase\_theta\_vers\_Xi** ([BaseB](#) &apB, const [BaseH](#) &gammaH)  
*la méthode qui suit a pour objectif de calculer les vecteurs de la base naturelle finale associée à un paramétrage cartésien initial*



- void `ChangeBase_curviligne` (const `Mat_pleine` &gamma, `BaseB` &apB, `BaseH` &apH, `BaseB` &IpB) const  
la méthode calcule à partir de this qui correspond à  $g\_alpha$  dans  $I\_a$ :  $apB : g\_alpha$  dans  $I^{alpha}$   $apH : g^{alpha}$  dans  $I^{alpha}$   $IpB : I\_beta$  dans  $I\_a$
- `Coordonnee` & `BaseAbsolue` (`Coordonnee` &A, const `CoordonneeH` &B) const  
calcul des composantes de coordonnées locales dans la base absolue en argument :  $A \rightarrow$  une référence sur les coordonnées résultat qui peut avoir une dimension différente des coordonnées locale, retour d'une référence sur A
- void `Affectation_partielle` (int nb\_vecteur\_a\_affecter, const `BaseB` &B, bool plusZero)  
une partie des vecteurs de B est affectée à this, si this contient plus de vecteur que B, les autres vecteurs sont mis à 0 ou non suivant la valeur de plusZero: = false: les autres vecteurs sont inchangées, plusZero = true: les autres vecteurs sont mis à 0

## Amis

- `istream` & `operator>>` (`istream` &, `BaseB` &)  
surcharge de l'operator de lecture avec le type
- `ostream` & `operator<<` (`ostream` &, const `BaseB` &)  
surcharge de l'operator d'écriture

### 6.35.1 Description détaillée

Les base covariantes et absolus.

### 6.35.2 Documentation des fonctions membres

#### 6.35.2.1 ChangeBase\_curviligne()

```
void BaseB::ChangeBase_curviligne (
    const Mat_pleine & gamma,
    BaseB & apB,
    BaseH & apH,
    BaseB & IpB ) const
```

la méthode calcule à partir de this qui correspond à  $g\_alpha$  dans  $I\_a$ :  $apB : g\_alpha$  dans  $I^{alpha}$   $apH : g^{alpha}$  dans  $I^{alpha}$   $IpB : I\_beta$  dans  $I\_a$

soient une base globale orthonormée :  $I\_a$  et une base locale orthonormée  $I\_alpha$  pour l'instant:  $I\_a$  est en 3 dimensions et  $alpha$  varie de 1 à 2 soient une base naturelle  $g\_alpha$  et duale associée  $g^{beta}$  telles que :  $I\_alpha$  appartient à l'espace des  $g\_alpha$  ou  $g^{beta}$

A) on a:  $g^{alpha} = gamma(alpha,beta) * I^{beta}$  c-a-d :  $gamma^{alpha\_beta}$

NB: 1) comme l'est orthonormée:  $I^{beta} = I\_beta$

2) chaque ligne de  $gamma$  représente un  $g^{alpha}$

B) dans ce contexte on a: inverse de  $gamma \rightarrow$  une matrice  $beta$  et chaque ligne de  $beta$  représente les coordonnées de  $g\_alpha$  dans  $I^{alpha}$  c-a-d  $g\_alpha = alpha(delta,alpha) * I\_delta$

C) on peut également calculer les coordonnées de  $I\_beta$  dans le repère globale

$I\_beta = gamma(alpha,beta) * g\_alpha$

### 6.35.2.2 ChangeBase\_theta\_vers\_Xi()

```
void BaseB::ChangeBase_theta_vers_Xi (
    BaseB & apB,
    const BaseH & gammaH )
```

la méthode qui suit a pour objectif de calculer les vecteurs de la base naturelle finale associée à un paramétrage cartésien initial

donc soit connue une base naturelle  $\hat{g}_i$  associée à un paramétrage  $\theta^i$   $\hat{g}_i$  est représenté par les vecteurs de this, et représente la situation déformée soit la base duale de  $\hat{g}_i$  :  $\gamma^i = g^i$  c-à-d la base duale en situation non déformée maintenant: si on considère les coordonnées initiales  $X^i$  on cherche les vecteurs des bases naturelles associées à  $X^i$  avant  $\hat{g}_i$  et après déformation  $\hat{g}'_i$  on a (cf. annexe dans le document théorique d'Herezh)

$\hat{g}'_i = l_i$  par définition et

$\hat{g}'_i = (\text{vec } l_i \cdot \text{vec } g^j) \sim \hat{g}_j$  c-a-d

$\hat{g}'_i = \gamma^j(i) * (*this)(j)$

c'est le résultat de la méthode qui suit

changement de base: retourne  $\text{apB}(a) = (*this)(j) \cdot \gamma^j_a$  il faut que le nombre de vecteur de apB soit identique à la dimension de gammaH et que le nombre de vecteur de gammaH soit identique au nombre de vecteur de this et que la dimension de apB soit identique à la dimension de this

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Reperes\_bases/Base.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Reperes\_bases/Base\_1.cc

## 6.36 Référence de la classe BaseB\_0\_t\_tdt

un groupe de 3 bases covariantes et absolus à 0, t et tdt

```
#include <Base.h>
```

### Fonctions membres publiques

- **BaseB\_0\_t\_tdt** ()  
*par défaut, défini les Bases en absolu en dimension 3 les vecteurs sont unitaires, ex en dim 2 -> la base : 1,0 et 0,1*
- **BaseB\_0\_t\_tdt** (int dim)  
*défini les Bases en absolu en dimension dim*
- **BaseB\_0\_t\_tdt** (int dim, const [Tableau](#)< [CoordonneeB](#) > &v1)  
*défini 3 Bases relatives v1(dim) c-a-d différente de la base triviale absolu*
- **BaseB\_0\_t\_tdt** (int dim, int n)  
*défini les Bases locale absolue (triviale) de n vecteurs de dimension dim*
- **BaseB\_0\_t\_tdt** (int dim, int n, double s)  
*idem dessus mais avec toutes les composantes = s*
- **BaseB\_0\_t\_tdt** (int dim, int n, const [Tableau](#)< [CoordonneeB](#) > &v1)  
*défini les Bases locale relative v1(dim) de n vecteurs de dimension dim*
- **BaseB\_0\_t\_tdt** (const [BaseB\\_0\\_t\\_tdt](#) &b)  
*constructeur de copie*
- **~BaseB\_0\_t\_tdt** ()  
*DESTRUCTEUR :*
- **BaseB\_0\_t\_tdt & operator=** (const [BaseB\\_0\\_t\\_tdt](#) &b)  
*surcharge de l'affectation*
- const [BaseB](#) & **Const\_BaseB\_Noed** () const  
*recupération de la base, a) en constant, la base actuelle*

- const [BaseB](#) & [Const\\_BaseB\\_Noeud\\_t](#) () const  
*b) en constant, la base à t*
- const [BaseB](#) & [Const\\_BaseB\\_Noeud\\_0](#) () const  
*c) en constant, la base initiale*
- [BaseB](#) & [BaseB\\_Noeud](#) ()  
*d) en lecture écriture, la base actuelle*
- [BaseB](#) & [BaseB\\_Noeud\\_t](#) ()  
*e) en lecture écriture, la base à t*
- [BaseB](#) & [BaseB\\_Noeud\\_0](#) ()  
*f) en lecture écriture, la base à 0*

## Amis

- [istream](#) & [operator](#)>> ([istream](#) &, [BaseB\\_0\\_t\\_tdt](#) &)  
*surcharge de l'operator de lecture avec le type*
- [ostream](#) & [operator](#)<< ([ostream](#) &, const [BaseB\\_0\\_t\\_tdt](#) &)  
*surcharge de l'operator d'écriture*

### 6.36.1 Description détaillée

un groupe de 3 bases covariantes et absolus à 0, t et tdt

il s'agit ici essentiellement d'un conteneur pour optimiser le stockage

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Reperes\_bases/Base.h

## 6.37 Référence de la classe BaseH

Les base duales.

```
#include <Base.h>
```

### Fonctions membres publiques

- [BaseH](#) ()  
*par défaut, défini une Base absolu en dimension 3 les vecteurs sont unitaires, ex en dim 2 -> la base : 1,0 et 0,1*
- [BaseH](#) (int dim)  
*défini une Base absolu en dimension dim*
- [BaseH](#) (int dim, const [Tableau](#)< [CoordonneeH](#) > &v1)  
*défini une Base relative v1(dim) c-a-d différente de la base triviale absolue*
- [BaseH](#) (int dim, int n)  
*défini une Base locale absolue (triviale) de n vecteurs de dimension dim*
- [BaseH](#) (int dim, int n, double s)  
*idem dessus mais avec toutes les composantes = s*
- [BaseH](#) (int dim, int n, const [Tableau](#)< [CoordonneeH](#) > &v1)  
*défini une Base locale relative v1(dim) de n vecteurs de dimension dim*
- [BaseH](#) (const [BaseH](#) &)  
*constructeur de copie*
- [~BaseH](#) ()  
*DESTRUCTEUR :*

- **BaseH** & **operator=** (const **BaseH** &aB)  
*surcharge de l'affectation*
- const int **Dimension** () const  
*retourne la dimension des vecteurs*
- const int **NbVecteur** () const  
*retourne le nombre de vecteurs de la base*
- const **CoordonneeH** & **operator()** (int i) const  
*retourne le i ieme vecteurs en lecture only*
- **CoordonneeH** & **CoordoH** (int i)  
*retourne le i ieme vecteurs en I/O*
- const **Coordonnee** & **Coordo** (int i) const  
*acces directe contrôlé à des vecteurs sans variance: uniquement en lecture*
- void **Affiche** () const  
*affichage à l'écran des infos*
- void **Affectation\_trans\_variance** (const **BaseB** &aB)  
*affectation trans\_variance: utile pour une recopie de mêmes valeurs mais ici l'appel est explicite donc a priori on sait ce que l'on fait il n'y a pas de redimensionnement, donc la dimension et le nombre des vecteurs doivent être identiques*
- void **Change\_repere** (const **BaseB** &lpB, **BaseH** &apH)  
*changement de base on suppose que this(i) correspond aux coordonnées dans un premier repère  $I_a$  lpB correspond aux coordonnées dans  $I_a$  d'un nouveau repère en sortie: apH(i) correspond aux coordonnées dans ipH de this(i)*
- void **ChangeBase\_theta\_vers\_Xi** (**BaseH** &apH, const **BaseB** &betaB)  
*la méthode qui suit a pour objectif de calcul les vecteurs de la base naturelle finale associée à un paramétrage cartésien initiale*
- void **ChangeBase\_curviligne** (const **Mat\_pleine** &gamma, **BaseB** &apB, **BaseH** &apH, **BaseH** &lpH) const  
*la méthode calcule à partir de this qui correspond à  $g^\alpha$  dans  $I_a$ : apB :  $g_\alpha$  dans  $I^\alpha$  apH :  $g^\alpha$  dans  $I^\alpha$  lpH :  $I^\alpha$  beta dans  $I_a$*
- **Coordonnee** & **BaseAbsolue** (**Coordonnee** &A, const **CoordonneeB** &B) const  
*calcul des composantes de coordonnées locales dans la base absolue en argument : A -> une référence sur les coordonnées résultat qui peut avoir une dimension différente des coordonnées locale, retour d'une référence sur A*
- void **Affectation\_partielle** (int nb\_vecteur\_a\_affecter, const **BaseH** &B, bool plusZero)  
*une partie des vecteurs de B est affectée à this, si this contient plus de vecteur que B, les autres vecteurs sont mis à 0 ou non suivant la valeur de plusZero: = false: les autres vecteurs sont inchangées, plusZero = true: les autres vecteurs sont mis à 0*

## Amis

- istream & **operator>>** (istream &, **BaseH** &)  
*surcharge de l'operator de lecture avec le type*
- ostream & **operator<<** (ostream &, const **BaseH** &)  
*surcharge de l'operator d'écriture*

### 6.37.1 Description détaillée

Les base duales.

### 6.37.2 Documentation des fonctions membres

### 6.37.2.1 ChangeBase\_curviligne()

```
void BaseH::ChangeBase_curviligne (
    const Mat_pleine & gamma,
    BaseB & apB,
    BaseH & apH,
    BaseH & IpH ) const
```

la méthode calcule à partir de this qui correspond à  $g^\alpha$  dans  $I_a$ :  $apB : g_\alpha$  dans  $I^\alpha$   $apH : g^\alpha$  dans  $I^\alpha$   $IpH : I^\beta$  dans  $I_a$

soient une base globale orthonormée :  $I_a$  et une base locale orthonormée  $I_\alpha$  pour l'instant:  $I_a$  est en 3 dimensions et  $\alpha$  varie de 1 à 2 soient une base naturelle  $g_\alpha$  et duale associée  $g^\beta$  telles que :  $I_\alpha$  appartient à l'espace des  $g_\alpha$  ou  $g^\beta$

A) on a:  $g^\alpha = \gamma(\alpha, \beta) * I^\beta$  c-a-d :  $\gamma^\alpha_\beta$

NB: 1) comme l'est orthonormée:  $I^\beta = I_\beta$  2) chaque ligne de  $\gamma$  représente un  $g^\alpha$

B) dans ce contexte on a: inverse de  $\gamma \rightarrow$  une matrice  $\beta$  et chaque ligne de  $\beta$  représente les coordonnées de  $g_\alpha$  dans  $I^\alpha$  c-a-d  $g_\alpha = \alpha(\delta, \alpha) * I_\delta$

C) on peut également calculer les coordonnées de  $I_\beta$  dans le repère globale

$I_\beta = \gamma(\alpha, \beta) * g_\alpha$

### 6.37.2.2 ChangeBase\_theta\_vers\_Xi()

```
void BaseH::ChangeBase_theta_vers_Xi (
    BaseH & apH,
    const BaseB & betaB )
```

la méthode qui suit a pour objectif de calcul les vecteurs de la base naturelle finale associée à un paramétrage cartésien initiale

donc soit connue une base duale  $\hat{g}^i$  associée à un paramétrage  $\theta^i$   $\hat{g}^i$  est représenté par les vecteurs de this, et représente la situation déformée soit la base naturel de  $g_i$  :  $\beta_{B_i} = g_i$  c-à-d la base naturelle en situation non déformée maintenant: si on considère les coordonnées initiales  $X^i$  on cherche les vecteurs des bases duales associées à  $X^i$  avant  $g^i$  et après déformation  $\hat{g}^i$  on a (cf. annexe dans le document théorique d'Herezh)

$g^i = g'_i = I_i = I^i$  par définition et

$\hat{g}^i = (\text{vec } I_i \cdot \text{vec } g_j) \sim \hat{g}^j$  c-a-d

$\hat{g}^i = \beta_{B^j}(i) * (*this)(j)$

c'est le résultat de la méthode qui suit

changement de base: retourne  $apH(a) = (*this)(j) \cdot \beta_{B_j}(a)$  il faut que le nombre de vecteur de  $apH$  soit identique à la dimension de  $\beta_{B_j}$  et que le nombre de vecteur de  $\beta_{B_j}$  soit identique au nombre de vecteur de this et que la dimension de  $apB$  soit identique à la dimension de this

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Reperes\_bases/Base.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Reperes\_bases/Base\_1.cc

## 6.38 Référence de la classe BaseH\_0\_t\_tdt

un groupe de 3 bases contravariant et absolus à 0, t et tdt

```
#include <Base.h>
```

## Fonctions membres publiques

- **BaseH\_0\_t\_tdt** ()  
*CONSTRUCTEURS : par défaut, défini les Bases en absolu en dimension 3 les vecteurs sont unitaires, ex en dim 2 -> la base : 1,0 et 0,1.*
- **BaseH\_0\_t\_tdt** (int dim)  
*défini les Bases en absolu en dimension dim*
- **BaseH\_0\_t\_tdt** (int dim, const [Tableau](#)< [CoordonneeH](#) > &v1)  
*défini 3 Bases relatives v1(dim) c-a-d différente de la base triviale absolu*
- **BaseH\_0\_t\_tdt** (int dim, int n)  
*défini les Bases locale absolue (triviale) de n vecteurs de dimension dim*
- **BaseH\_0\_t\_tdt** (int dim, int n, double s)  
*idem dessus mais avec toutes les composantes = s*
- **BaseH\_0\_t\_tdt** (int dim, int n, const [Tableau](#)< [CoordonneeH](#) > &v1)  
*défini les Bases locale relative v1(dim) de n vecteurs de dimension dim*
- **BaseH\_0\_t\_tdt** (const [BaseH\\_0\\_t\\_tdt](#) &b)  
*constructeur de copie*
- **~BaseH\_0\_t\_tdt** ()  
*DESTRUCTEUR :*
- **BaseH\_0\_t\_tdt & operator=** (const [BaseH\\_0\\_t\\_tdt](#) &b)  
*surcharge de l'affectation*
- const [BaseH](#) & **Const\_BaseH\_Noeud** () const  
*récupération de la base, a) en constant, la base actuelle*
- const [BaseH](#) & **Const\_BaseH\_Noeud\_t** () const  
*b) en constant, la base à t*
- const [BaseH](#) & **Const\_BaseH\_Noeud\_0** () const  
*c) en constant, la base initiale*
- [BaseH](#) & **BaseH\_Noeud** ()  
*d) en lecture écriture, la base actuelle*
- [BaseH](#) & **BaseH\_Noeud\_t** ()  
*e) en lecture écriture, la base à t*
- [BaseH](#) & **BaseH\_Noeud\_0** ()  
*f) en lecture écriture, la base à 0*

## Amis

- [istream](#) & **operator>>** ([istream](#) &ent, [BaseH\\_0\\_t\\_tdt](#) &)  
*surcharge de l'operator de lecture avec le type*
- [ostream](#) & **operator<<** ([ostream](#) &sort, const [BaseH\\_0\\_t\\_tdt](#) &)  
*surcharge de l'operator d'écriture*

### 6.38.1 Description détaillée

un groupe de 3 bases contravariant et absolus à 0, t et tdt

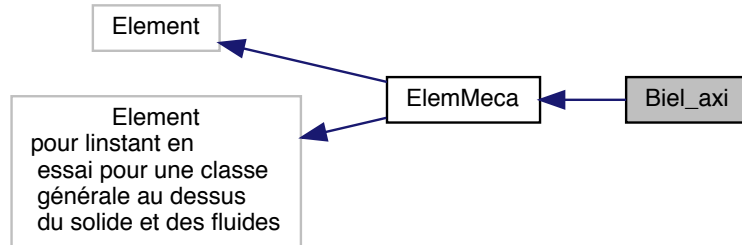
il s'agit ici essentiellement d'un conteneur pour optimiser le stockage

La documentation de cette classe a été générée à partir du fichier suivant :

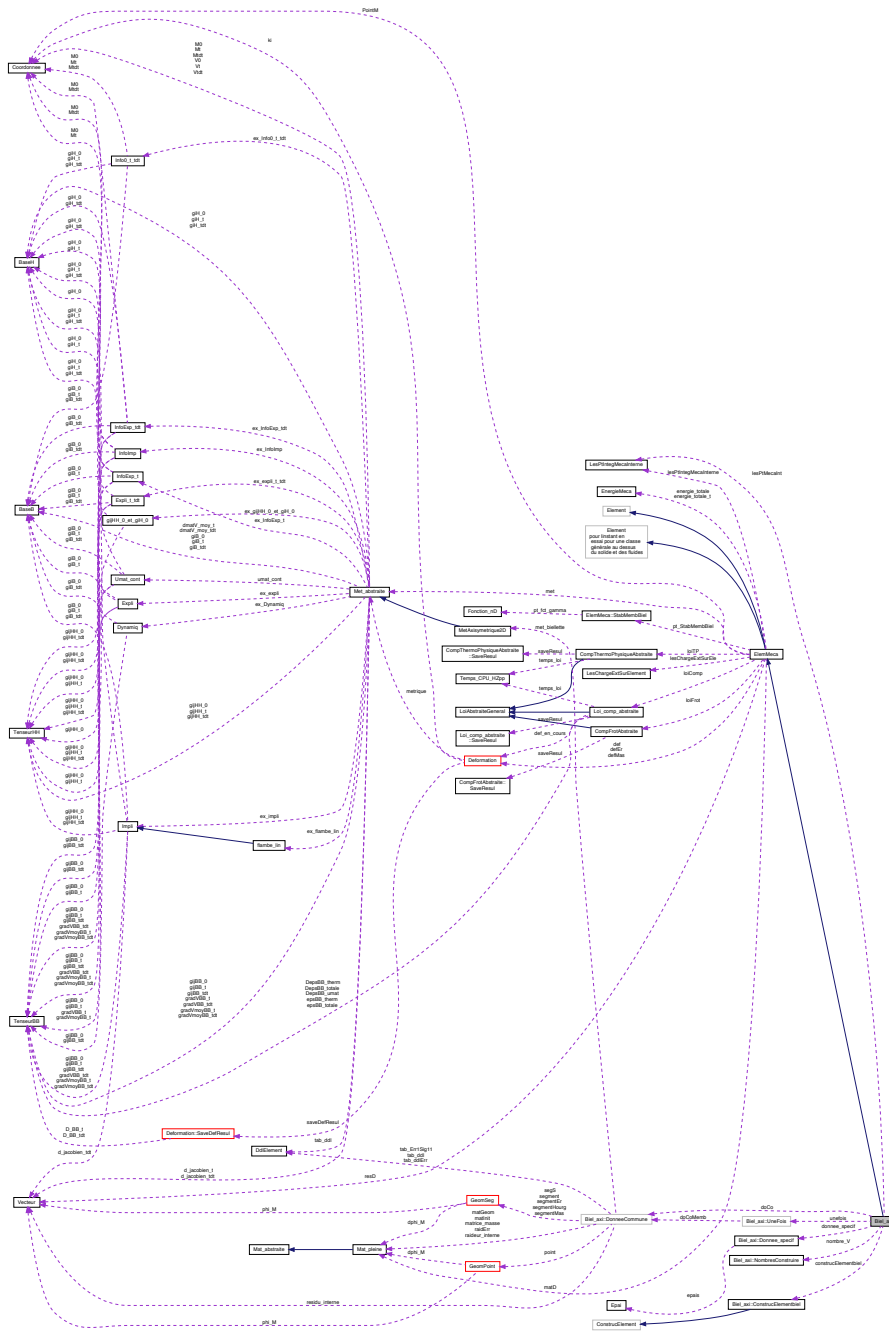
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Reperes\_bases/Base.h

## 6.39 Référence de la classe Biel\_axi

Graphe d'héritage de Biel\_axi:



Graphe de collaboration de Biel\_axi:



### Classes

- class [ConstrucElementbiel](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)

### Fonctions membres publiques

- **Biel\_axi** (double epai, int num\_maill=0, int num\_id=-3)



- `Biel_axi` (int num\_maill, int num\_id)
- `Biel_axi` (double epai, int num\_maill, int num\_id, const `Tableau< Noeud * >` &tab)
- `Biel_axi` (const `Biel_axi` &biel)
- `Element * Nevez_copie` () const
- `Biel_axi & operator=` (`Biel_axi` &biel)
- void `LectureDonneesParticulieres` (`UtilLecture *`, `Tableau< Noeud * > *`)
- `Element::ResRaid Calcul_implicit` (const `ParaAlgoControle` &pa)
- `Vecteur * CalculResidu_t` (const `ParaAlgoControle` &pa)
- `Vecteur * CalculResidu_tdt` (const `ParaAlgoControle` &pa)
- `Mat_pleine * CalculMatriceMasse` (`Enum_calcul_masse` id\_calcul\_masse)
- double `Long_arrete_mini_sur_c` (`Enum_dure` temps)
- `Element::Er_ResRaid ContrainteAuNoeud_ResRaid` ()
- `Element::Er_ResRaid ErreurAuNoeud_ResRaid` ()
- const `DdlElement & TableauDdl` () const
- void `Libere` ()
- void `DefLoi` (`LoiAbstraiteGeneral *NouvelleLoi`)
- int `TestComplet` ()
- `Element * Complete` (`BlocGen` &bloc, `LesFonctions_nD *lesFonctionsnD`)
- `Element * Complet_Hourglass` (`LoiAbstraiteGeneral *NouvelleLoi`, const `BlocGen` &bloc)
- `ElemGeomC0 & ElementGeometrique` () const
- const `ElemGeomC0 & ElementGeometrique_const` () const
- `Coordonnee & Point_physique` (const `Coordonnee` &c\_int, `Coordonnee` &co, `Enum_dure` temps)
- void `Point_physique` (const `Coordonnee` &c\_int, `Tableau< Coordonnee >` &t\_co)
- virtual double `Epaisseurs` (`Enum_dure` enu, const `Coordonnee` &)
- virtual double `EpaisseurMoyenne` (`Enum_dure` enu)
- void `AfficheVarDual` (ofstream &sort, `Tableau< string >` &nom)
- void `Info_com_Element` (`UtilLecture *entreePrinc`, string &ordre, `Tableau< Noeud * > *tabMaillageNoeud`)
- int `PointLePlusPres` (`Enum_dure` temps, `Enum_ddl` enu, const `Coordonnee` &M)
- `Coordonnee CoordPtInteg` (`Enum_dure` temps, `Enum_ddl` enu, int iteg, bool &erreur)
- `Tableau< double > Valeur_a_diff_temps` (bool absolue, `Enum_dure` enu\_t, const `List_io< Ddl_enum_etendu >` &enu, int iteg)
- void `ValTensorielle_a_diff_temps` (bool absolue, `Enum_dure` enu\_t, `List_io< TypeQuelconque >` &enu, int iteg)
- bool `SurfExiste` (int) const
- bool `AreteExiste` (int na) const
- void `Lecture_base_info` (ifstream &ent, const `Tableau< Noeud * > *tabMaillageNoeud`, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- `ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale` (const `ParaAlgoControle` &pa)
- `List_io< TypeQuelconque > Les_types_particuliers_internes` (bool absolue) const
- void `Grandeur_particuliere` (bool absolue, `List_io< TypeQuelconque >` &litQ, int iteg)
- void `Inactive_ddl_primaire` ()
- void `Active_ddl_primaire` ()
- void `Plus_ddl_Sigma` ()
- void `Inactive_ddl_Sigma` ()
- void `Active_ddl_Sigma` ()
- void `Active_premier_ddl_Sigma` ()
- void `LectureContraintes` (`UtilLecture *entreePrinc`)
- bool `ContraintesAbsolues` (`Tableau< Vecteur >` &tabSig)
- `DdlElement & Tableau_de_Sig1` () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void `TdtversT` ()
- void `TversTdt` ()
- void `ErreurElement` (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual const `DeuxCoordonnees & Boite_encombre_element` (`Enum_dure` temps)
- `Vecteur SM_charge_volumique_E_t` (const `Coordonnee` &force, `Fonction_nD *pt_fonct`, const `ParaAlgoControle` &pa, bool sur\_volume\_finale\_)
- `Vecteur SM_charge_volumique_E_tdt` (const `Coordonnee` &force, `Fonction_nD *pt_fonct`, const `ParaAlgoControle` &pa, bool sur\_volume\_finale\_)
- `ResRaid SMR_charge_volumique_I` (const `Coordonnee` &force, `Fonction_nD *pt_fonct`, const `ParaAlgoControle` &pa, bool sur\_volume\_finale\_)
- `Vecteur SM_charge_surfacique_E_t` (const `Coordonnee` &force, `Fonction_nD *pt_fonct`, int numFace, const `ParaAlgoControle` &pa)

- Vecteur **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E\_t** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E\_tdt** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrostatique\_E\_t** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_hydrostatique\_E\_tdt** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_lineique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Tableau< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- double **H** ([Enum\\_dure](#) enu=TEMPS\_tdt)
- void **ConstTabDdl** ()

## Fonctions membres protégées

- int **Dim\_sig\_eps** () const
- virtual [EIFrontiere](#) \* **new\_frontiere\_lin** (int, Tableau< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- virtual [EIFrontiere](#) \* **new\_frontiere\_surf** (int, Tableau< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [Biel\\_axi::DonneeCommune](#) \* **Init** ([Donnee\\_specif](#) donnee\_specif=[Donnee\\_specif](#)(), bool sans\_init←noeud=false)
- void **Def\_DonneeCommune** ()
- void **Destruction** ()
- Vecteur \* **CalculResidu** (bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_volumique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, bool atdt, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- Vecteur **SM\_charge\_surfacique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrostatique\_E** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, bool atdt, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_lineique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef←\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, bool atdt, const [ParaAlgoControle](#) &pa)
- const double & **CalEpaisseurMoyenne\_et\_vol\_pti** (bool atdt)

## Attributs protégés

- `Donnee_specif` `donnee_specif`
- `LesPtIntegMecalInterne` `lesPtMecalInt`

## Attributs protégés statiques

- `static` `DonneeCommune` \* `doCo` = `NULL`
- `static` `UneFois` `unefois`
- `static` `NombresConstruire` `nombre_V`
- `static` `ConstrucElementbiel` `construcElementbiel`

## Membres hérités additionnels

### 6.39.1 Documentation des fonctions membres

#### 6.39.1.1 `Active_ddl_Sigma()`

```
void Biel_axi::Active_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.39.1.2 `Active_premier_ddl_Sigma()`

```
void Biel_axi::Active_premier_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.39.1.3 `ContraintesAbsolues()`

```
bool Biel_axi::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.39.1.4 `Dim_sig_eps()`

```
int Biel_axi::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.39.1.5 EpaisseurMoyenne()

```
virtual double Biel_axi::EpaisseurMoyenne (
    Enum_dure enu ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.39.1.6 Epaisseurs()

```
virtual double Biel_axi::Epaisseurs (
    Enum_dure enu,
    const Coordonnee & ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.39.1.7 ErreurElement()

```
void Biel_axi::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

### 6.39.1.8 Inactive\_ddl\_Sigma()

```
void Biel_axi::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.39.1.9 LectureContraintes()

```
void Biel_axi::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.39.1.10 `Les_types_particuliers_internes()`

```
List_io< TypeQuelconque > Biel_axi::Les_types_particuliers_internes (
    bool absolue ) const [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.39.1.11 `Long_arrete_mini_sur_c()`

```
double Biel_axi::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.39.1.12 `MatricesGeometrique_Et_Initiale()`

```
ElemMeca::MatGeomInit Biel_axi::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.39.1.13 `new_frontiere_lin()`

```
virtual ElFrontiere * Biel_axi::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.39.1.14 `new_frontiere_surf()`

```
virtual ElFrontiere * Biel_axi::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.39.1.15 Plus\_ddl\_Sigma()

```
void Biel_axi::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.39.1.16 Tableau\_de\_Sig1()

```
DdlElement & Biel_axi::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

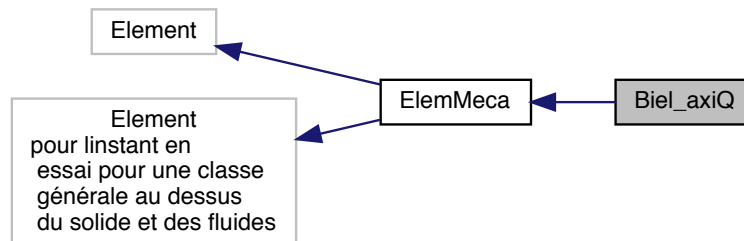
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

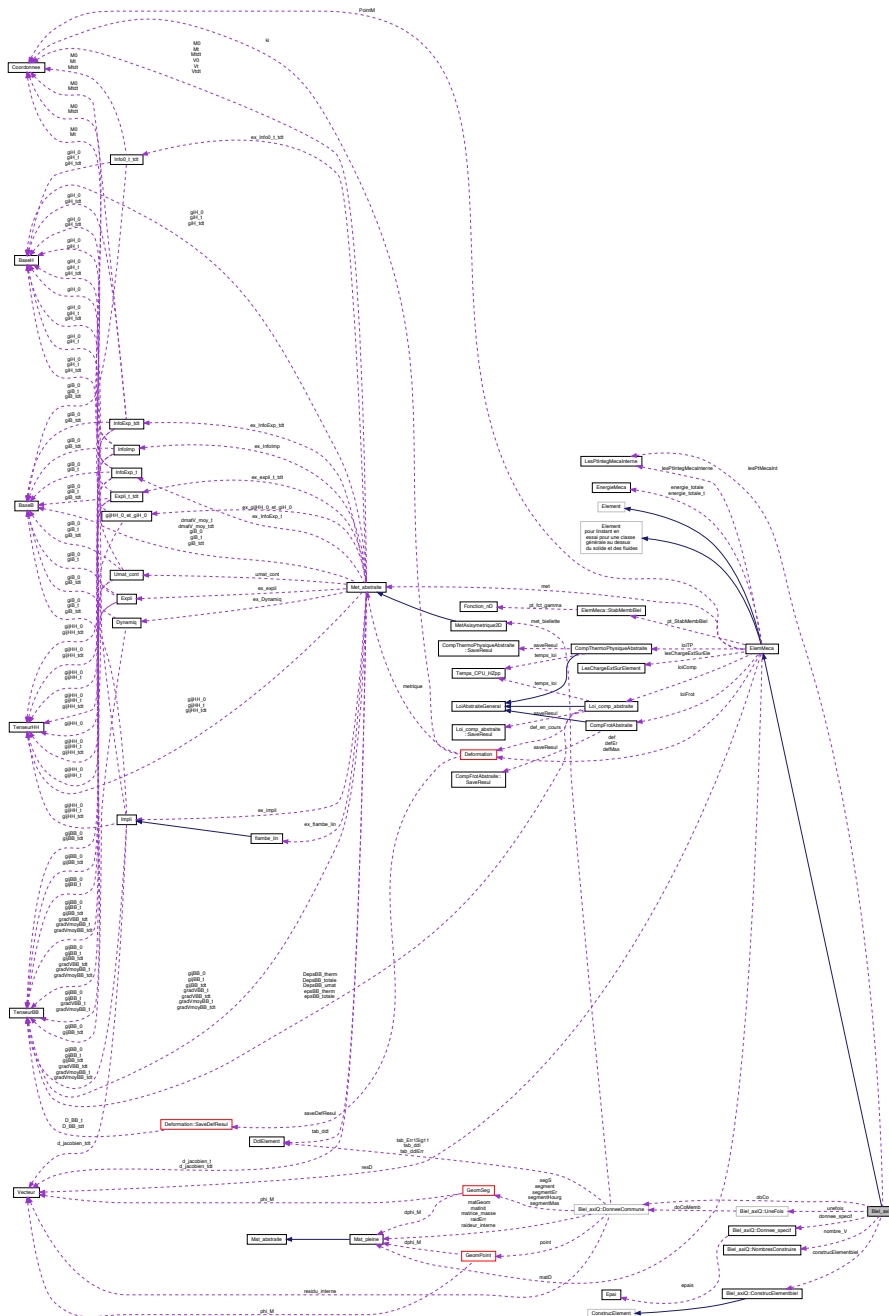
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axi.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axi.cc

## 6.40 Référence de la classe Biel\_axiQ

Graphe d'héritage de Biel\_axiQ:



Graphe de collaboration de Biel\_axiQ:



### Classes

- class [ConstrucElementbiel](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)

### Fonctions membres publiques

- **Biel\_axiQ** (double epai, int num\_maill=0, int num\_id=-3)

- **Biel\_axiQ** (int num\_maill, int num\_id)
- **Biel\_axiQ** (double epai, int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **Biel\_axiQ** (const [Biel\\_axiQ](#) &biel)
- [Element](#) \* **Nevez\_copie** () const
- [Biel\\_axiQ](#) & **operator=** ([Biel\\_axiQ](#) &biel)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- [Element::ResRaid](#) **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- [Element::Er\\_ResRaid](#) **ContrainteAuNoeud\_ResRaid** ()
- [Element::Er\\_ResRaid](#) **ErreurAuNoeud\_ResRaid** ()
- const [DdlElement](#) & **TableauDdl** () const
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComplet** ()
- [Element](#) \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- [Element](#) \* **Complet\_Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- [ElemGeomC0](#) & **ElementGeometrique** () const
- const [ElemGeomC0](#) & **ElementGeometrique\_const** () const
- [Coordonnee](#) & **Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- virtual double **Epaisseurs** ([Enum\\_dure](#) enu, const [Coordonnee](#) &)
- virtual double **EpaisseurMoyenne** ([Enum\\_dure](#) enu)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- bool **SurfExiste** (int) const
- bool **AreteExiste** (int na) const
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- [ElemMeca::MatGeomInit](#) **MatricesGeometrique\_Et\_Initiale** (const [ParaAlgoControle](#) &pa)
- [List\\_io](#)< [TypeQuelconque](#) > **Les\_types\_particuliers\_internes** (bool absolue) const
- void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &litQ, int iteg)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- [DdlElement](#) & **Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual const [DeuxCoordonnees](#) & **Boite\_encombre\_element** ([Enum\\_dure](#) temps)
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [ResRaid](#) **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)



- Vecteur `SM_charge_surfacique_E_tdt` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numFace, const `ParaAlgoControle` &pa)
- ResRaid `SMR_charge_surfacique_I` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numFace, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_pression_E_t` (double pression, `Fonction_nD` \*pt\_fonct, int numFace, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_pression_E_tdt` (double pression, `Fonction_nD` \*pt\_fonct, int numFace, const `ParaAlgoControle` &pa)
- ResRaid `SMR_charge_pression_I` (double pression, `Fonction_nD` \*pt\_fonct, int numFace, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_hydrostatique_E_t` (const `Coordonnee` &dir\_normal\_liquide, const double &poidvol, int numFace, const `Coordonnee` &M\_liquide, const `ParaAlgoControle` &pa, bool sans\_limitation)
- Vecteur `SM_charge_hydrostatique_E_tdt` (const `Coordonnee` &dir\_normal\_liquide, const double &poidvol, int numFace, const `Coordonnee` &M\_liquide, const `ParaAlgoControle` &pa, bool sans\_limitation)
- ResRaid `SMR_charge_hydrostatique_I` (const `Coordonnee` &dir\_normal\_liquide, const double &poidvol, int numFace, const `Coordonnee` &M\_liquide, const `ParaAlgoControle` &pa, bool sans\_limitation)
- Vecteur `SM_charge_lineique_E_t` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numArete, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_lineique_E_tdt` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numArete, const `ParaAlgoControle` &pa)
- ResRaid `SMR_charge_lineique_I` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numArete, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_lineique_Suiv_E_t` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numArete, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_lineique_Suiv_E_tdt` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numArete, const `ParaAlgoControle` &pa)
- ResRaid `SMR_charge_lineique_Suiv_I` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numArete, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_hydrodynamique_E_t` (`Courbe1D` \*frot\_fluid, const double &poidvol, `Courbe1D` \*coef\_aero\_n, int numFace, const double &coef\_mul, `Courbe1D` \*coef\_aero\_t, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_hydrodynamique_E_tdt` (`Courbe1D` \*frot\_fluid, const double &poidvol, `Courbe1D` \*coef\_aero\_n, int numFace, const double &coef\_mul, `Courbe1D` \*coef\_aero\_t, const `ParaAlgoControle` &pa)
- ResRaid `SMR_charge_hydrodynamique_I` (`Courbe1D` \*frot\_fluid, const double &poidvol, `Courbe1D` \*coef\_aero\_n, int numFace, const double &coef\_mul, `Courbe1D` \*coef\_aero\_t, const `ParaAlgoControle` &pa)
- Tableau< `EIFrontiere` \* > const & `Frontiere` (bool force=false)
- double `H` (`Enum_dure` enu=TEMPS\_tdt)
- void `ConstTabDdl` ()

### Fonctions membres protégées

- int `Dim_sig_eps` () const
- virtual `EIFrontiere` \* `new_frontiere_lin` (int, Tableau< `Noeud` \* > &tab, `DdlElement` &ddelem)
- virtual `EIFrontiere` \* `new_frontiere_surf` (int, Tableau< `Noeud` \* > &tab, `DdlElement` &ddelem)
- `Biel_axiQ::DonneeCommune` \* `Init` (`Donnee_specif` donnee\_specif=`Donnee_specif`()), bool sans\_init ← noeud=false)
- void `Def_DonneeCommune` ()
- void `Destruction` ()
- Vecteur \* `CalculResidu` (bool atdt, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_volumique_E` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, bool atdt, const `ParaAlgoControle` &pa, bool sur\_volume\_finale\_)
- Vecteur `SM_charge_surfacique_E` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numFace, bool atdt, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_pression_E` (double pression, `Fonction_nD` \*pt\_fonct, int numFace, bool atdt, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_hydrostatique_E` (const `Coordonnee` &dir\_normal\_liquide, const double &poidvol, int numFace, const `Coordonnee` &M\_liquide, bool atdt, const `ParaAlgoControle` &pa, bool sans\_limitation)
- Vecteur `SM_charge_lineique_E` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numArete, bool atdt, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_lineique_Suiv_E` (const `Coordonnee` &force, `Fonction_nD` \*pt\_fonct, int numArete, bool atdt, const `ParaAlgoControle` &pa)
- Vecteur `SM_charge_hydrodynamique_E` (`Courbe1D` \*frot\_fluid, const double &poidvol, `Courbe1D` \*coef\_aero\_n, int numFace, const double &coef\_mul, `Courbe1D` \*coef\_aero\_t, bool atdt, const `ParaAlgoControle` &pa)
- const double & `CalEpaisseurMoyenne_et_vol_pti` (bool atdt)

## Attributs protégés

- [Donnee\\_specif](#) `donnee_specif`
- [LesPtIntegMecalInterne](#) `lesPtMecalInt`

## Attributs protégés statiques

- `static` `DonneeCommune` \* `doCo` = NULL
- `static` `UneFois` `unefois`
- `static` [NombresConstruire](#) `nombre_V`
- `static` [ConstrucElementbiel](#) `construcElementbiel`

## Membres hérités additionnels

### 6.40.1 Documentation des fonctions membres

#### 6.40.1.1 `Active_ddl_Sigma()`

```
void Biel_axiQ::Active_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.40.1.2 `Active_premier_ddl_Sigma()`

```
void Biel_axiQ::Active_premier_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.40.1.3 `ContraintesAbsolues()`

```
bool Biel_axiQ::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.40.1.4 `Dim_sig_eps()`

```
int Biel_axiQ::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.40.1.5 `EpaisseurMoyenne()`

```
virtual double Biel_axiQ::EpaisseurMoyenne (
    Enum_dure enu ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.40.1.6 `Epaisseurs()`

```
virtual double Biel_axiQ::Epaisseurs (
    Enum_dure enu,
    const Coordonnee & ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.40.1.7 `ErreurElement()`

```
void Biel_axiQ::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

#### 6.40.1.8 `Inactive_ddl_Sigma()`

```
void Biel_axiQ::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.40.1.9 `LectureContraintes()`

```
void Biel_axiQ::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.40.1.10 Les\_types\_particuliers\_internes()

```
List_io< TypeQuelconque > Biel_axiQ::Les_types_particuliers_internes (
    bool absolue ) const [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.40.1.11 Long\_arrete\_mini\_sur\_c()

```
double Biel_axiQ::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.40.1.12 MatricesGeometrique\_Et\_Initiale()

```
ElemMeca::MatGeomInit Biel_axiQ::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.40.1.13 new\_frontiere\_lin()

```
virtual ElFrontiere * Biel_axiQ::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.40.1.14 new\_frontiere\_surf()

```
virtual ElFrontiere * Biel_axiQ::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.40.1.15 Plus\_ddl\_Sigma()

```
void Biel_axiQ::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.40.1.16 Tableau\_de\_Sig1()

```
DdlElement & Biel_axiQ::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

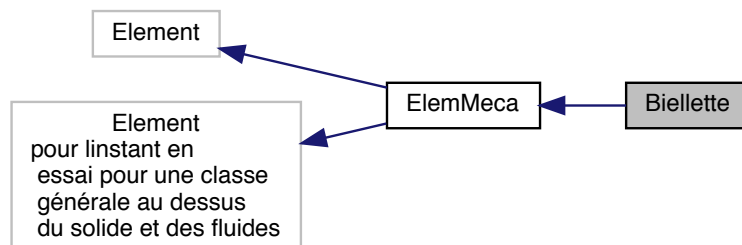
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

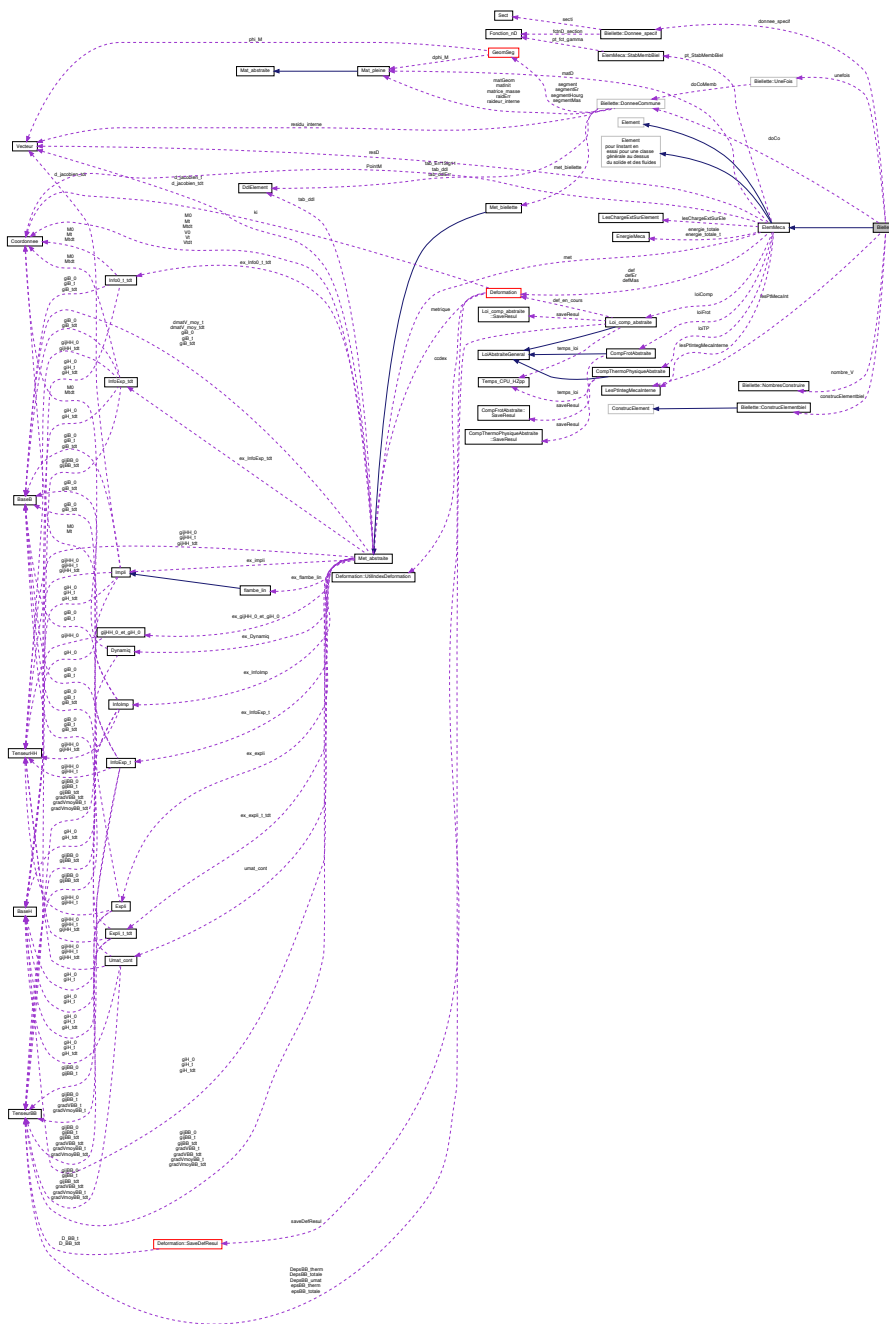
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axiQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axiQ.cc

## 6.41 Référence de la classe Biellette

Graphe d'héritage de Biellette:



Graphe de collaboration de Biellette:



### Classes

- class [ConstrucElementbiel](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)

### Fonctions membres publiques

- **Biellette** (double sect, int num\_maill=0, int num\_id=-3)

- **Bielle** (int num\_maill, int num\_id)
- **Bielle** (double sect, int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **Bielle** (const [Bielle](#) &biel)
- [Element](#) \* **Nevez\_copie** () const
- [Bielle](#) & **operator=** ([Bielle](#) &biel)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- [Element](#)::[ResRaid](#) **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- [Element](#)::[Er\\_ResRaid](#) **ContrainteAuNoeud\_ResRaid** ()
- [Element](#)::[Er\\_ResRaid](#) **ErreurAuNoeud\_ResRaid** ()
- const [DdlElement](#) & **TableauDdl** () const
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComplet** ()
- [Element](#) \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- [Element](#) \* **Complet\_Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- [ElemGeomC0](#) & **ElementGeometrique** () const
- const [ElemGeomC0](#) & **ElementGeometrique\_const** () const
- [Coordonnee](#) & **Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- virtual double **Section** ([Enum\\_dure](#) enu, const [Coordonnee](#) &)
- virtual double **SectionMoyenne** ([Enum\\_dure](#) enu)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- bool **SurfExiste** (int) const
- bool **AreteExiste** (int na) const
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- [ElemMeca](#)::[MatGeomInit](#) **MatricesGeometrique\_Et\_Initiale** (const [ParaAlgoControle](#) &pa)
- [List\\_io](#)< [TypeQuelconque](#) > **Les\_types\_particuliers\_internes** (bool absolue) const
- void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &litQ, int iteg)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- [DdlElement](#) & **Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual const [DeuxCoordonnees](#) & **Boite\_encombre\_element** ([Enum\\_dure](#) temps)
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [ResRaid](#) **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_lineique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)

- Vecteur **SM\_charge\_lineique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Tableau](#)< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- double **S** ([Enum\\_dure](#) enu=TEMPS\_tdt)
- void **ConstTabDdl** ()

## Fonctions membres protégées

- int **Dim\_sig\_eps** () const
- virtual [EIFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- virtual [EIFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [Biellette::DonneeCommune](#) \* **Init** ([Donnee\\_specif](#) donnee\_specif=[Donnee\\_specif](#)(), bool sans\_init\_↔ noeud=false)
- void **Def\_DonneeCommune** ()
- void **Destruction** ()
- Vecteur \* **CalculResidu** (bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_volumique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, bool atdt, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- Vecteur **SM\_charge\_lineique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_↔\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, bool atdt, const [ParaAlgoControle](#) &pa)
- const double & **CalSectionMoyenne\_et\_vol\_pti** (const bool atdt)

## Attributs protégés

- [Donnee\\_specif](#) donnee\_specif
- [LesPtIntegMecalInterne](#) lesPtMecalnt

## Attributs protégés statiques

- static [DonneeCommune](#) \* **doCo** = NULL
- static UneFois **unefois**
- static [NombresConstruire](#) nombre\_V
- static [ConstrucElementbiel](#) construcElementbiel

## Membres hérités additionnels

### 6.41.1 Documentation des fonctions membres



#### 6.41.1.1 Active\_ddl\_Sigma()

```
void Bielle::Active_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.2 Active\_premier\_ddl\_Sigma()

```
void Bielle::Active_premier_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.3 ContraintesAbsolues()

```
bool Bielle::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.4 Dim\_sig\_eps()

```
int Bielle::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.5 ErreurElement()

```
void Bielle::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

#### 6.41.1.6 Inactive\_ddl\_Sigma()

```
void Bielle::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.7 LectureContraintes()

```
void Bielle::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.8 Les\_types\_particuliers\_internes()

```
List_io< TypeQuelconque > Bielle::Les_types_particuliers_internes (
    bool absolue ) const [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.41.1.9 Long\_arrete\_mini\_sur\_c()

```
double Bielle::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.10 MatricesGeometrique\_Et\_Initiale()

```
ElemMeca::MatGeomInit Bielle::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.41.1.11 new\_frontiere\_lin()

```
virtual ElFrontiere * Bielle::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.12 new\_frontiere\_surf()

```
virtual ElFrontiere * Bielle::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.13 Plus\_ddl\_Sigma()

```
void Bielle::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.41.1.14 Section()

```
virtual double Bielle::Section (
    Enum_dure enu,
    const Coordonnee & ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.41.1.15 SectionMoyenne()

```
virtual double Bielle::SectionMoyenne (
    Enum_dure enu ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.41.1.16 Tableau\_de\_Sig1()

```
DdlElement & Bielle::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

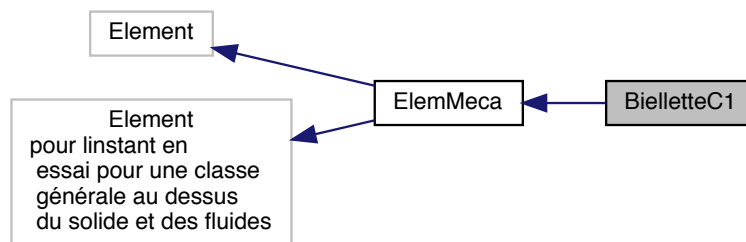
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

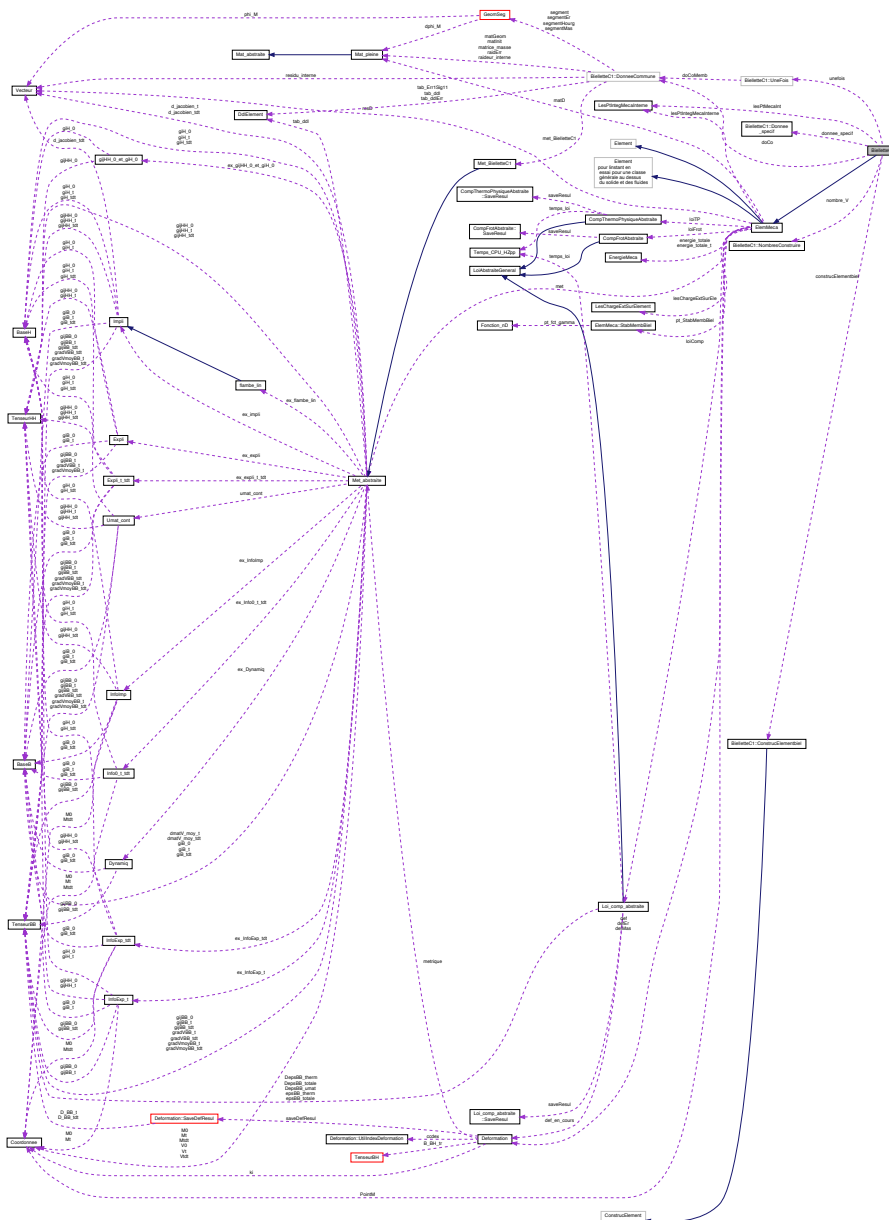
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Bielle/Bielle.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Bielle/Bielle.cc

## 6.42 Référence de la classe BielleC1

Grphe d'héritage de BielleC1:



Graphe de collaboration de `BielletteC1`:



## Classes

- class [Constructeur](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)

## Fonctions membres publiques

- `BielletteC1` (double sect, int num\_maill=0, int num\_id=-3)
- `BielletteC1` (int num\_maill, int num\_id)
- `BielletteC1` (double sect, int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- `BielletteC1` (const [BielletteC1](#) &biel)
- `Element` \* [Nevez\\_copie](#) () const

```

— BielleC1 & operator= (BielleC1 &biel)
— void LectureDonneesParticulieres (UtilLecture *, Tableau< Noeud * > *)
— Element::ResRaid Calcul_implicit (const ParaAlgoControle &pa)
— Vecteur * CalculResidu_t (const ParaAlgoControle &pa)
— Vecteur * CalculResidu_tdt (const ParaAlgoControle &pa)
— Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse)
— double Long_arrete_mini_sur_c (Enum_dure temps)
— Element::Er_ResRaid ContrainteAuNoeud_ResRaid ()
— Element::Er_ResRaid ErreurAuNoeud_ResRaid ()
— const DdlElement & TableauDdl () const
— void Libere ()
— void DefLoi (LoiAbstraiteGeneral *NouvelleLoi)
— int TestComplet ()
— Element * Complete (BlocGen &bloc, LesFonctions_nD *lesFonctionsnD)
— Element * Complet_Hourglass (LoiAbstraiteGeneral *NouvelleLoi, const BlocGen &bloc)
— ElemGeomC0 & ElementGeometrique () const
— const ElemGeomC0 & ElementGeometrique_const () const
— Coordonnee & Point_physique (const Coordonnee &c_int, Coordonnee &co, Enum_dure temps)
— void Point_physique (const Coordonnee &c_int, Tableau< Coordonnee > &t_co)
— void AfficheVarDual (ofstream &sort, Tableau< string > &nom)
— void Info_com_Element (UtilLecture *entreePrinc, string &ordre, Tableau< Noeud * > *tabMaillageNoeud)
— int PointLePlusPres (Enum_dure temps, Enum_ddl enu, const Coordonnee &M)
— Coordonnee CoordPtInteg (Enum_dure temps, Enum_ddl enu, int iteg, bool &erreur)
— Tableau< double > Valeur_a_diff_temps (bool absolue, Enum_dure enu_t, const List_io< Ddl_enum_etendu
> &enu, int iteg)
— void ValTensorielle_a_diff_temps (bool absolue, Enum_dure enu_t, List_io< TypeQuelconque > &enu, int
iteg)
— bool SurfExiste (int) const
— bool AreteExiste (int na) const
— void Lecture_base_info (ifstream &ent, const Tableau< Noeud * > *tabMaillageNoeud, const int cas)
— void Ecriture_base_info (ofstream &sort, const int cas)
— ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle &pa)
— void Inactive_ddl_primaire ()
— void Active_ddl_primaire ()
— void Plus_ddl_Sigma ()
— void Inactive_ddl_Sigma ()
— void Active_ddl_Sigma ()
— void Active_premier_ddl_Sigma ()
— void LectureContraintes (UtilLecture *entreePrinc)
— bool ContraintesAbsolues (Tableau< Vecteur > &tabSig)
— DdlElement & Tableau_de_Sig1 () const
      !!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en
— void TdtversT ()
— void TversTdt ()
— void ErreurElement (int type, double &errElemRelative, double &numérateur, double &denominateur)
      !!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en
— Vecteur SM_charge_volumique_E_t (const Coordonnee &force, Fonction_nD *pt_fonct, const
ParaAlgoControle &pa, bool sur_volume_finale_)
— Vecteur SM_charge_volumique_E_tdt (const Coordonnee &force, Fonction_nD *pt_fonct, const
ParaAlgoControle &pa, bool sur_volume_finale_)
— ResRaid SMR_charge_volumique_I (const Coordonnee &force, Fonction_nD *pt_fonct, const
ParaAlgoControle &pa, bool sur_volume_finale_)
— Vecteur SM_charge_lineique_E_t (const Coordonnee &force, Fonction_nD *pt_fonct, int numArete, const
ParaAlgoControle &pa)
— Vecteur SM_charge_lineique_E_tdt (const Coordonnee &force, Fonction_nD *pt_fonct, int numArete, const
ParaAlgoControle &pa)
— ResRaid SMR_charge_lineique_I (const Coordonnee &force, Fonction_nD *pt_fonct, int numArete, const
ParaAlgoControle &pa)
— Vecteur SM_charge_lineique_Suiv_E_t (const Coordonnee &force, Fonction_nD *pt_fonct, int numArete,
const ParaAlgoControle &pa)
— Vecteur SM_charge_lineique_Suiv_E_tdt (const Coordonnee &force, Fonction_nD *pt_fonct, int numArete,
const ParaAlgoControle &pa)
— ResRaid SMR_charge_lineique_Suiv_I (const Coordonnee &force, Fonction_nD *pt_fonct, int numArete,
const ParaAlgoControle &pa)

```

- Vecteur **SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Tableau< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- double **Section** ([Enum\\_dure](#), const [Coordonnee](#) &)
- double **SectionMoyenne** ([Enum\\_dure](#))
- void **ConstTabDdl** ()

## Fonctions membres protégées

- int **Dim\_sig\_eps** () const
- virtual [EIFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- virtual [EIFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- void **Init** ([Donnee\\_specif](#) donnee\_specif=[Donnee\\_specif](#)(), bool sans\_init\_noeud=false)
- void **Def\_DonneeCommune** ()
- void **Destruction** ()
- Vecteur \* **CalculResidu** (bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_volumique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, bool atdt, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- Vecteur **SM\_charge\_lineique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, bool atdt, const [ParaAlgoControle](#) &pa)

## Attributs protégés

- [Donnee\\_specif](#) donnee\_specif
- [LesPtIntegMecalInterne](#) lesPtMecalnt

## Attributs protégés statiques

- static [DonneeCommune](#) \* **doCo** = NULL
- static UneFois **unefois**
- static [NombresConstruire](#) nombre\_V
- static [ConstrucElementbiel](#) construcElementbiel

## Membres hérités additionnels

### 6.42.1 Documentation des fonctions membres

#### 6.42.1.1 Active\_ddl\_Sigma()

```
void BielleC1::Active_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.42.1.2 Active\_premier\_ddl\_Sigma()

```
void BielleC1::Active_premier_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.42.1.3 ContraintesAbsolues()

```
bool BielleC1::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.42.1.4 Dim\_sig\_eps()

```
int BielleC1::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.42.1.5 ErreurElement()

```
void BielleC1::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

#### 6.42.1.6 Inactive\_ddl\_Sigma()

```
void BielleC1::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).



### 6.42.1.7 LectureContraintes()

```
void BielleC1::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.42.1.8 Long\_arrete\_mini\_sur\_c()

```
double BielleC1::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.42.1.9 MatricesGeometrique\_Et\_Initiale()

```
ElemMeca::MatGeomInit BielleC1::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.42.1.10 new\_frontiere\_lin()

```
virtual ElFrontiere * BielleC1::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.42.1.11 new\_frontiere\_surf()

```
virtual ElFrontiere * BielleC1::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.42.1.12 Plus\_ddl\_Sigma()

```
void BielleC1::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.42.1.13 Section()

```
double BielleC1::Section (
    Enum_dure ,
    const Coordonnee & ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.42.1.14 SectionMoyenne()

```
double BielleC1::SectionMoyenne (
    Enum_dure ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.42.1.15 Tableau\_de\_Sig1()

```
DdlElement & BielleC1::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

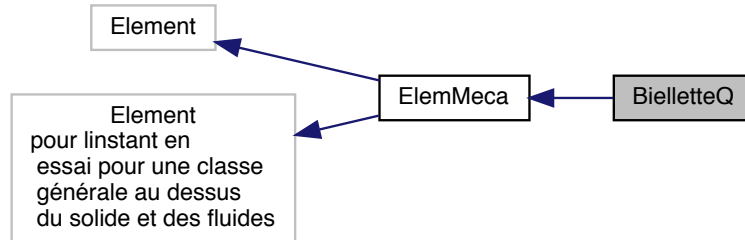
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

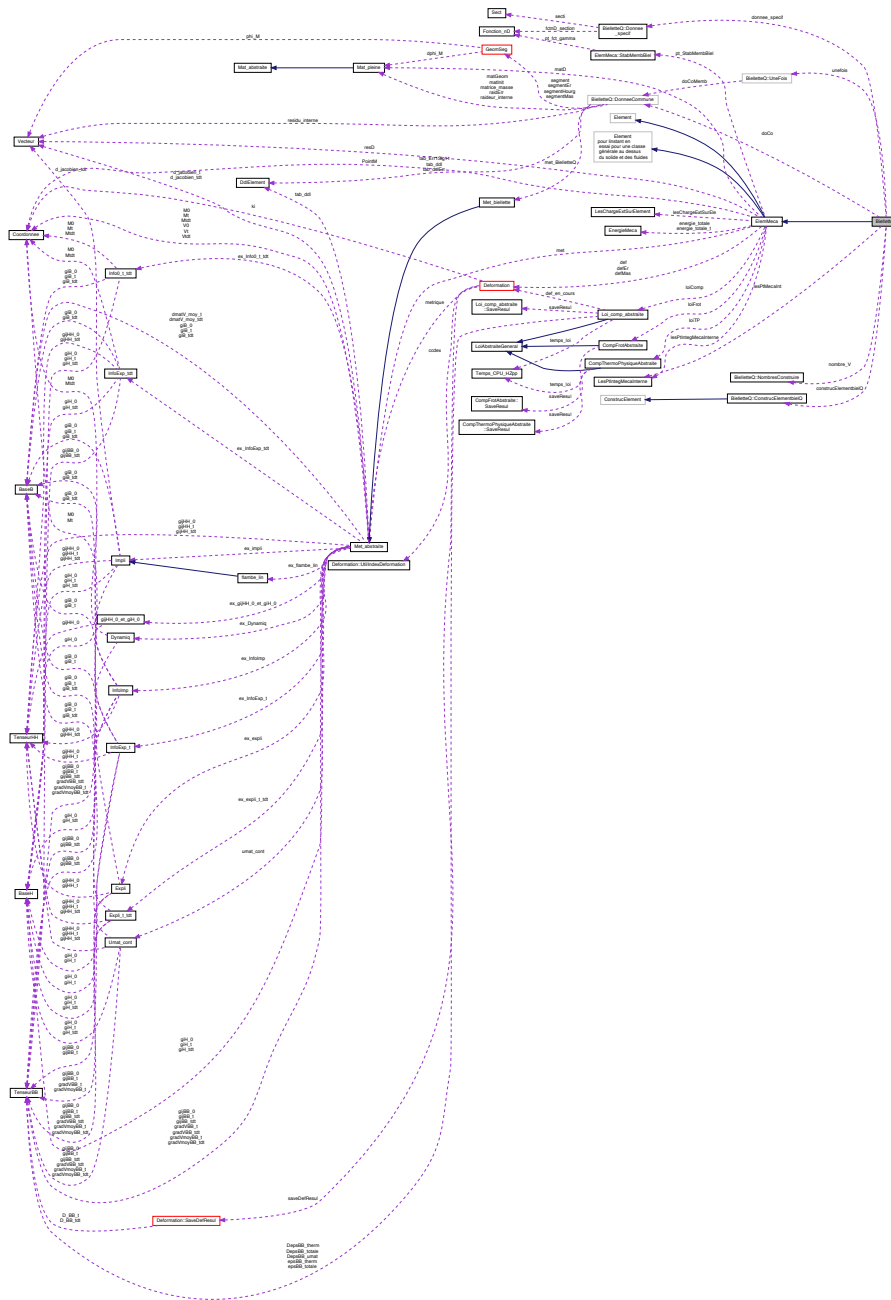
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/BielleC1/BielleC1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/BielleC1/BielleC1.cc

## 6.43 Référence de la classe BielletteQ

Graphe d'héritage de BielletteQ:



Graphe de collaboration de BielletteQ:



**Classes**

- class [ConstrucElementbielQ](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)

**Fonctions membres publiques**

- **BielletteQ** (double sect, int num\_maill=0, int num\_id=-3)
- **BielletteQ** (int num\_maill, int num\_id)

- **BielletteQ** (double sect, int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **BielletteQ** (const [BielletteQ](#) &biel)
- Element \* **Nevez\_copie** () const
- [BielletteQ](#) & **operator=** ([BielletteQ](#) &biel)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- Element::ResRaid **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- const [DdlElement](#) & **TableauDdl** () const
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComplet** ()
- Element \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- Element \* **Complet\_Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- [ElemGeomC0](#) & **ElementGeometrique** () const
- const [ElemGeomC0](#) & **ElementGeometrique\_const** () const
- [Coordonnee](#) & **Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- bool **SurfExiste** (int) const
- bool **AreteExiste** (int na) const
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- [ElemMeca::MatGeomInit](#) **MatricesGeometrique\_Et\_Initiale** (const [ParaAlgoControle](#) &pa)
- [List\\_io](#)< [TypeQuelconque](#) > **Les\_types\_particuliers\_internes** (bool absolue) const
- void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, int iteg)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- [DdlElement](#) & **Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual const [DeuxCoordonnees](#) & **Boite\_encombre\_element** ([Enum\\_dure](#) temps)
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [ResRaid](#) **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_lineique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_lineique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [ResRaid](#) **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)

- Vecteur **SM\_charge\_lineique\_Suiv\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Tableau](#)< [ElFrontiere](#) \* > const & **Frontiere** (bool force=false)
- double **S** ([Enum\\_dure](#) enu=TEMPS\_tdt)
- void **ConstTabDdl** ()

## Fonctions membres protégées

- int **Dim\_sig\_eps** () const
- virtual [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- virtual [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [BiellletteQ::DonneeCommune](#) \* **Init** ([Donnee\\_specif](#) donnee\_specif=[Donnee\\_specif](#)(), bool sans\_init\_↔ noeud=false)
- void **Def\_DonneeCommune** ()
- void **Destruction** ()
- [Vecteur](#) \* **CalculResidu** (bool atdt, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_volumique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, bool atdt, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_lineique\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_lineique\_Suiv\_E** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_hydrodynamique\_E** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_↔\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, bool atdt, const [ParaAlgoControle](#) &pa)
- const double & **CalSectionMoyenne\_et\_vol\_pti** (const bool atdt)

## Attributs protégés

- [Donnee\\_specif](#) **donnee\_specif**
- [LesPtIntegMecalInterne](#) **lesPtMecalnt**

## Attributs protégés statiques

- static [DonneeCommune](#) \* **doCo** = NULL
- static UneFois **unefois**
- static [NombresConstruire](#) **nombre\_V**
- static [ConstrucElementbielQ](#) **construcElementbielQ**

## Membres hérités additionnels

### 6.43.1 Documentation des fonctions membres

#### 6.43.1.1 Active\_ddl\_Sigma()

```
void BielleQ::Active_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.2 Active\_premier\_ddl\_Sigma()

```
void BielleQ::Active_premier_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.3 ContraintesAbsolues()

```
bool BielleQ::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.4 Dim\_sig\_eps()

```
int BielleQ::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.5 ErreurElement()

```
void BielleQ::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

#### 6.43.1.6 Inactive\_ddl\_Sigma()

```
void BielleQ::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.7 LectureContraintes()

```
void BielleQ::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.8 Les\_types\_particuliers\_internes()

```
List_io< TypeQuelconque > BielleQ::Les_types_particuliers_internes (
    bool absolue ) const [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.43.1.9 Long\_arrete\_mini\_sur\_c()

```
double BielleQ::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.10 MatricesGeometrique\_Et\_Initiale()

```
ElemMeca::MatGeomInit BielleQ::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).



#### 6.43.1.11 new\_frontiere\_lin()

```
virtual ElFrontiere * BiелletteQ::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.12 new\_frontiere\_surf()

```
virtual ElFrontiere * BiелletteQ::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.13 Plus\_ddl\_Sigma()

```
void BiелletteQ::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.43.1.14 Tableau\_de\_Sig1()

```
DdlElement & BiелletteQ::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

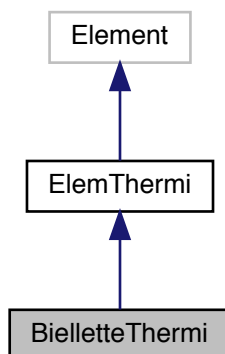
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

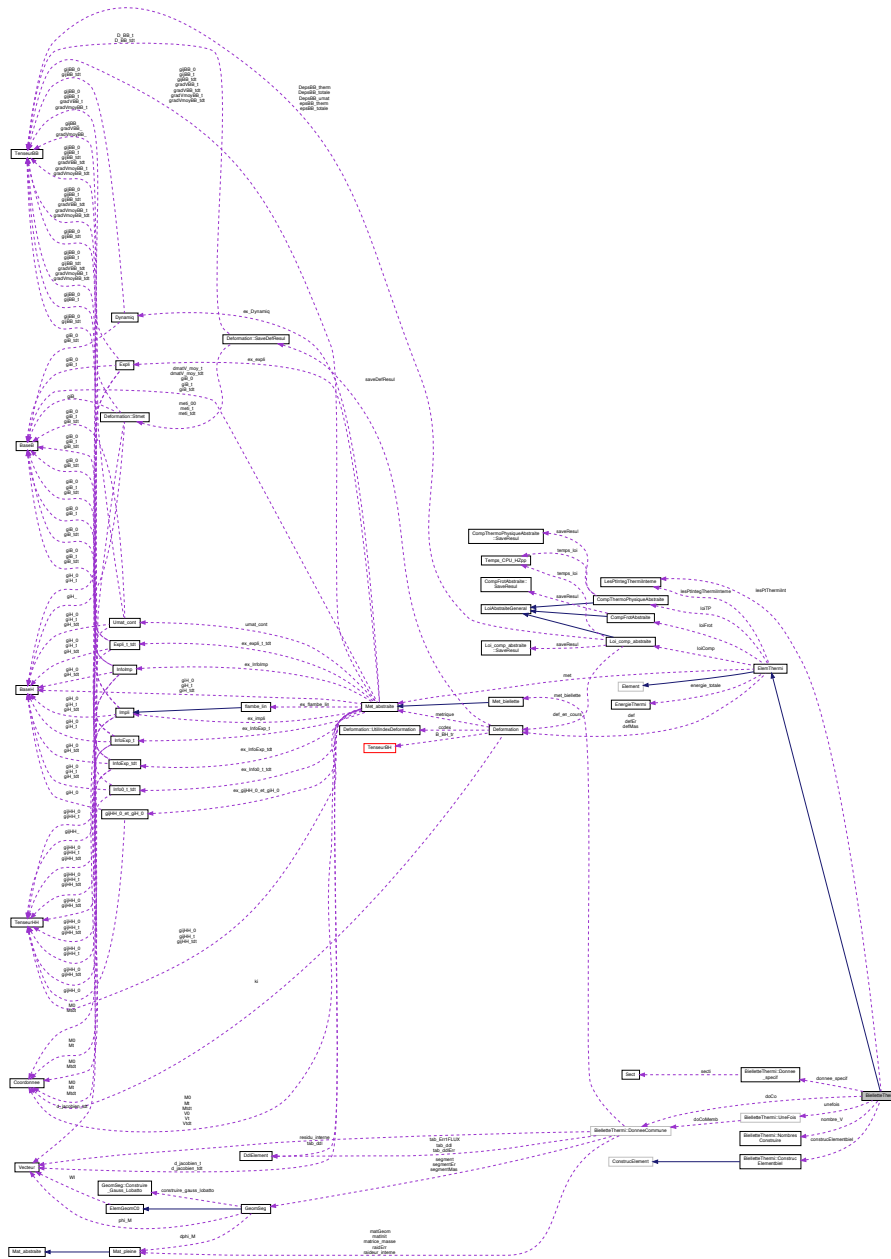
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/BielletteQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/BielletteQ.cc

## 6.44 Référence de la classe BielleThermi

Graphe d'héritage de BielleThermi:



Graphe de collaboration de **BielletteThermi**:



### Classes

- class [ConstrucElementbiel](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)

### Fonctions membres publiques

- **BielletteThermi** (double sect, int num\_maill=0, int num\_id=-3)
- **BielletteThermi** (int num\_maill, int num\_id)
- **BielletteThermi** (double sect, int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **BielletteThermi** (const [BielletteThermi](#) &biel)

- Element \* **Nevez\_copie** () const
- **BielletteThermi** & **operator=** (**BielletteThermi** &biel)
- void **LectureDonneesParticulieres** (**UtilLecture** \*, **Tableau**< **Noeud** \* > \*)
- Element::ResRaid **Calcul\_implicit** (const **ParaAlgoControle** &pa)
- **Vecteur** \* **CalculResidu\_t** (const **ParaAlgoControle** &pa)
- **Vecteur** \* **CalculResidu\_tdt** (const **ParaAlgoControle** &pa)
- **Mat\_pleine** \* **CalculMatriceMasse** (**Enum\_calcul\_masse** id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** (**Enum\_dure** temps)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- const **DdlElement** & **TableauDdl** () const
- void **Libere** ()
- void **DefLoi** (**LoiAbstraiteGeneral** \*NouvelleLoi)
- int **TestComplet** ()
- Element \* **Complete** (**BlocGen** &bloc, **LesFonctions\_nD** \*lesFonctionsnD)
- Element \* **Complet\_Hourglass** (**LoiAbstraiteGeneral** \*NouvelleLoi, const **BlocGen** &bloc)
- **ElemGeomC0** & **ElementGeometrique** () const
- const **ElemGeomC0** & **ElementGeometrique\_const** () const
- **Coordonnee** & **Point\_physique** (const **Coordonnee** &c\_int, **Coordonnee** &co, **Enum\_dure** temps)
- void **Point\_physique** (const **Coordonnee** &c\_int, **Tableau**< **Coordonnee** > &t\_co)
- virtual double **Section** (**Enum\_dure** enu, const **Coordonnee** &)
- virtual double **SectionMoyenne** (**Enum\_dure** enu)
- void **AfficheVarDual** (ofstream &sort, **Tableau**< string > &nom)
- void **Info\_com\_Element** (**UtilLecture** \*entreePrinc, string &ordre, **Tableau**< **Noeud** \* > \*tabMaillageNoeud)
- int **PointLePlusPres** (**Enum\_dure** temps, **Enum\_ddl** enu, const **Coordonnee** &M)
- **Coordonnee** **CoordPtInteg** (**Enum\_dure** temps, **Enum\_ddl** enu, int iteg, bool &erreur)
- **Tableau**< double > **Valeur\_a\_diff\_temps** (bool absolue, **Enum\_dure** enu\_t, const **List\_io**< **Ddl\_enum\_etendu** > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, **Enum\_dure** enu\_t, **List\_io**< **TypeQuelconque** > &enu, int iteg)
- bool **SurfExiste** (int) const
- bool **AreteExiste** (int na) const
- void **Lecture\_base\_info** (ifstream &ent, const **Tableau**< **Noeud** \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- **ElemThermi::MatGeomInit** **MatricesGeometrique\_Et\_Initiale** (const **ParaAlgoControle** &pa)
- **List\_io**< **TypeQuelconque** > **Les\_types\_particuliers\_internes** (bool absolue) const
- void **Grandeur\_particuliere** (bool absolue, **List\_io**< **TypeQuelconque** > &liTQ, int iteg)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Flux** ()
- void **Inactive\_ddl\_Flux** ()
- void **Active\_ddl\_Flux** ()
- void **Active\_premier\_ddl\_Flux** ()
- void **LectureFlux** (**UtilLecture** \*entreePrinc)
- bool **FluxAbsolues** (**Tableau**< **Vecteur** > &tabFlux)
- **DdlElement** & **Tableau\_de\_Flux1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual const **DeuxCoordonnees** & **Boite\_encombre\_element** (**Enum\_dure** temps)
- **Vecteur** **SM\_charge\_volumique\_E\_t** (const **Coordonnee** &force, const **ParaAlgoControle** &pa)
- **Vecteur** **SM\_charge\_volumique\_E\_tdt** (const **Coordonnee** &force, const **ParaAlgoControle** &pa)
- ResRaid **SMR\_charge\_volumique\_I** (const **Coordonnee** &force, const **ParaAlgoControle** &pa)
- **Vecteur** **SM\_charge\_lineique\_E\_t** (const **Coordonnee** &force, int numArete, const **ParaAlgoControle** &pa)
- **Vecteur** **SM\_charge\_lineique\_E\_tdt** (const **Coordonnee** &force, int numArete, const **ParaAlgoControle** &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const **Coordonnee** &force, int numArete, const **ParaAlgoControle** &pa)
- **Vecteur** **SM\_charge\_lineique\_Suiv\_E\_t** (const **Coordonnee** &force, int numArete, const **ParaAlgoControle** &pa)
- **Vecteur** **SM\_charge\_lineique\_Suiv\_E\_tdt** (const **Coordonnee** &force, int numArete, const **ParaAlgoControle** &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const **Coordonnee** &force, int numArete, const **ParaAlgoControle** &pa)

- Vecteur **SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Tableau< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- double **S** ([Enum\\_dure](#) enu=TEMPS\_tdt)
- void **ConstTabDdl** ()

### Fonctions membres protégées

- int **Dim\_flux\_gradT** () const
- virtual [EIFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- virtual [EIFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [BielletteThermi::DonneeCommune](#) \* **Init** ([Donnee\\_specif](#) donnee\_specif=[Donnee\\_specif](#)()), bool sans\_init←\_noeud=false)
- void **Def\_DonneeCommune** ()
- void **Destruction** ()
- [Vecteur](#) \* **CalculResidu** (bool atdt, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_volumique\_E** (const [Coordonnee](#) &force, bool atdt, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_lineique\_E** (const [Coordonnee](#) &force, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_lineique\_Suiv\_E** (const [Coordonnee](#) &force, int numArete, bool atdt, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_hydrodynamique\_E** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef←\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, bool atdt, const [ParaAlgoControle](#) &pa)
- const double & **CalSectionMoyenne\_et\_vol\_pti** (const bool atdt)

### Attributs protégés

- [Donnee\\_specif](#) donnee\_specif
- [LesPtIntegThermiInterne](#) lesPtThermiInt

### Attributs protégés statiques

- static [DonneeCommune](#) \* **doCo** = NULL
- static UneFois **unefois**
- static [NombresConstruire](#) nombre\_V
- static [ConstrucElementbiel](#) construcElementbiel

### Membres hérités additionnels

#### 6.44.1 Documentation des fonctions membres

##### 6.44.1.1 Active\_ddl\_Flux()

```
void BielletteThermi::Active_ddl_Flux ( ) [inline], [virtual]
```

Implémente [ElemThermi](#).

#### 6.44.1.2 Active\_premier\_ddl\_Flux()

```
void BielleTteThermi::Active_premier_ddl_Flux ( ) [inline], [virtual]
```

Implémente [ElemThermi](#).

#### 6.44.1.3 Dim\_flux\_gradT()

```
int BielleTteThermi::Dim_flux_gradT ( ) const [inline], [protected], [virtual]
```

Implémente [ElemThermi](#).

#### 6.44.1.4 ErreurElement()

```
void BielleTteThermi::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemThermi](#).

#### 6.44.1.5 FluxAbsolues()

```
bool BielleTteThermi::FluxAbsolues (
    Tableau< Vecteur > & tabFlux ) [inline], [virtual]
```

Implémente [ElemThermi](#).

#### 6.44.1.6 Inactive\_ddl\_Flux()

```
void BielleTteThermi::Inactive_ddl_Flux ( ) [inline], [virtual]
```

Implémente [ElemThermi](#).

#### 6.44.1.7 `LectureFlux()`

```
void BiелletteThermi::LectureFlux (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemThermi](#).

#### 6.44.1.8 `Les_types_particuliers_internes()`

```
List_io< TypeQuelconque > BiелletteThermi::Les_types_particuliers_internes (
    bool absolue ) const [virtual]
```

Réimplémentée à partir de [ElemThermi](#).

#### 6.44.1.9 `Long_arrete_mini_sur_c()`

```
double BiелletteThermi::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemThermi](#).

#### 6.44.1.10 `MatricesGeometrique_Et_Initiale()`

```
ElemThermi::MatGeomInit BiелletteThermi::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

Réimplémentée à partir de [ElemThermi](#).

#### 6.44.1.11 `new_frontiere_lin()`

```
virtual ElFrontiere * BiелletteThermi::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemThermi](#).

**6.44.1.12 new\_frontiere\_surf()**

```
virtual ElFrontiere * BiелletteThermi::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemThermi](#).

**6.44.1.13 Plus\_ddl\_Flux()**

```
void BiелletteThermi::Plus_ddl_Flux ( ) [inline], [virtual]
```

Implémente [ElemThermi](#).

**6.44.1.14 Section()**

```
virtual double BiелletteThermi::Section (
    Enum_dure enu,
    const Coordonnee & ) [inline], [virtual]
```

Réimplémentée à partir de [ElemThermi](#).

**6.44.1.15 SectionMoyenne()**

```
virtual double BiелletteThermi::SectionMoyenne (
    Enum_dure enu ) [inline], [virtual]
```

Réimplémentée à partir de [ElemThermi](#).

**6.44.1.16 Tableau\_de\_Flux1()**

```
DdlElement & BiелletteThermi::Tableau_de_Flux1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemThermi](#).

La documentation de cette classe a été générée à partir du fichier suivant :

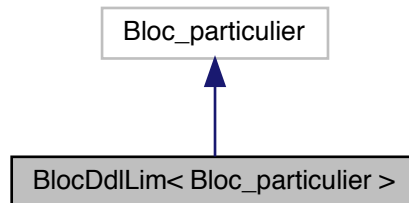
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/Biellette/BielletteThermi.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/Biellette/BielletteThermi.cc



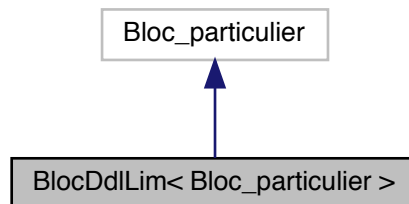
## 6.45 Référence du modèle de la classe BlocDdlLim< Bloc\_particulier >

```
#include <BlocDdlLim.h>
```

Graphe d'héritage de BlocDdlLim< Bloc\_particulier > :



Graphe de collaboration de BlocDdlLim< Bloc\_particulier > :



### Fonctions membres publiques

- **BlocDdlLim** (const Bloc\_particulier &a)
- **BlocDdlLim** (const string &nom\_mail, const Bloc\_particulier &a)
- **BlocDdlLim** (const string \*nom\_mail, const Bloc\_particulier &a)
- **BlocDdlLim** (const [BlocDdlLim](#) &a)
- const string \* **NomMaillage** () const
- void **Change\_nom\_maillage** (const string &nouveau)  
*change le nom de maillage ATTENTION : Les modifications liées au changement du nom de maillage sont a la charge de l'utilisateur. le nouveau nom ne doit pas être vide !! sinon erreur*
- void **Lecture** ([UtilLecture](#) &entreePrinc)  
*lecture d'un bloc*
- void **Affiche** () const  
*affichage des infos*
- bool **operator==** (const [BlocDdlLim](#) &a) const  
*surcharge de l'operateur ==*
- [BlocDdlLim](#) & **operator=** (const [BlocDdlLim](#) &a)  
*surcharge de l'operateur =*
- bool **operator!=** (const [BlocDdlLim](#) &a) const

## Attributs protégés

- string \* **nom\_maillage**

## Amis

- istream & **operator**>> (istream &entree, [BlocDdlLim](#)< Bloc\_particulier > &coo)
- ostream & **operator**<< (ostream &sort, const [BlocDdlLim](#)< Bloc\_particulier > &coo)

### 6.45.1 Description détaillée

```
template<class Bloc_particulier>
class BlocDdlLim< Bloc_particulier >
```

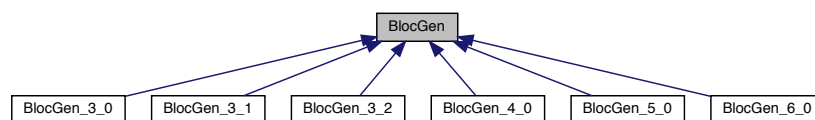
### 6.45.2 cas d'un bloc template avec conditions limites

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/[BlocDdlLim.h](#)

## 6.46 Référence de la classe BlocGen

Graphe d'héritage de BlocGen:



## Fonctions membres publiques

- **BlocGen** (int n=1, int m=1)
- **BlocGen** (const [BlocGen](#) &a)
- string & **NomRef** () const
- const string & **Nom** (int i) const
- const double & **Val** (int i) const
- int **DimNom** () const
- int **DimVal** () const
- bool **MemeCibleMaisDataDifferentes** ([BlocGen](#) &) const
- void **Affiche** () const
- bool **operator==** (const [BlocGen](#) &a) const
- bool **operator!=** (const [BlocGen](#) &a) const
- void **Change\_val** (const int i, double val)
- void **Change\_nom** (const int i, const string &nom)
- void **Lecture** ([UtilLecture](#) &entreePrinc)
- string \* **NomMaillage** ()
- [BlocGen](#) & **operator=** (const [BlocGen](#) &a)

## Attributs protégés

- [Tableau](#)< double > **pt**
- [Tableau](#)< string > **ptnom**

## Amis

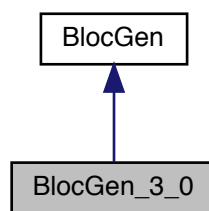
- `istream & operator>>` (`istream &`, [BlocGen &](#))
- `ostream & operator<<` (`ostream &`, `const BlocGen &`)

La documentation de cette classe a été générée à partir du fichier suivant :

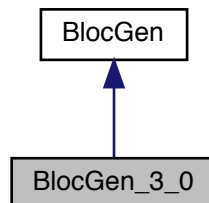
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.cc`

## 6.47 Référence de la classe BlocGen\_3\_0

Graphe d'héritage de BlocGen\_3\_0:



Graphe de collaboration de BlocGen\_3\_0:



## Amis

- `istream & operator>>` (`istream &entree`, `BlocGen\_3\_0 &coo`)
- `ostream & operator<<` (`ostream &sort`, `const BlocGen\_3\_0 &coo`)

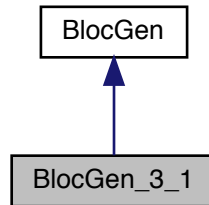
## Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

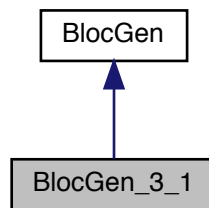
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`

## 6.48 Référence de la classe BlocGen\_3\_1

Grphe d'héritage de BlocGen\_3\_1:



Grphe de collaboration de BlocGen\_3\_1:



### Amis

- `istream & operator>> (istream &entree, BlocGen\_3\_1 &coo)`
- `ostream & operator<< (ostream &sort, const BlocGen\_3\_1 &coo)`

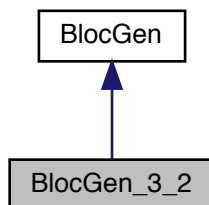
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

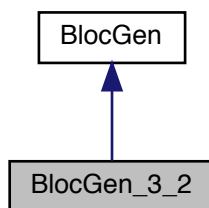
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`

## 6.49 Référence de la classe BlocGen\_3\_2

Grphe d'héritage de BlocGen\_3\_2:



Grphe de collaboration de BlocGen\_3\_2:



### Amis

- `istream & operator>>` (`istream &entree`, [BlocGen\\_3\\_2 &coo](#))
- `ostream & operator<<` (`ostream &sort`, `const BlocGen\_3\_2 &coo`)

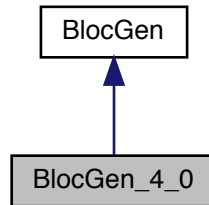
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

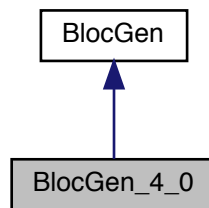
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`

## 6.50 Référence de la classe BlocGen\_4\_0

Grphe d'héritage de BlocGen\_4\_0:



Grphe de collaboration de BlocGen\_4\_0:



### Amis

- `istream & operator>>` (`istream &entree`, [BlocGen\\_4\\_0 &coo](#))
- `ostream & operator<<` (`ostream &sort`, `const BlocGen\_4\_0 &coo`)

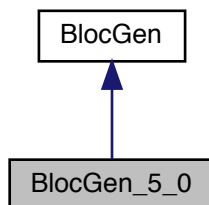
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

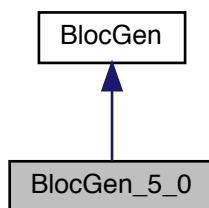
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`

## 6.51 Référence de la classe BlocGen\_5\_0

Grphe d'héritage de BlocGen\_5\_0:



Grphe de collaboration de BlocGen\_5\_0:



### Amis

- `istream & operator>>` (`istream &entree`, [BlocGen\\_5\\_0 &coo](#))
- `ostream & operator<<` (`ostream &sort`, `const BlocGen\_5\_0 &coo`)

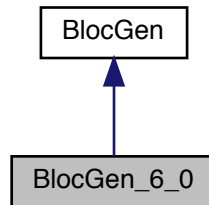
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

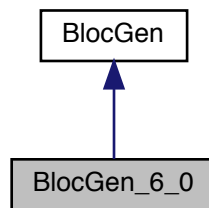
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`

## 6.52 Référence de la classe BlocGen\_6\_0

Graphe d'héritage de BlocGen\_6\_0:



Graphe de collaboration de BlocGen\_6\_0:



### Amis

- `istream & operator>>` (`istream &entree`, [BlocGen\\_6\\_0 &coo](#))
- `ostream & operator<<` (`ostream &sort`, `const BlocGen\_6\_0 &coo`)

### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`

## 6.53 Référence de la classe BlocGeneEtVecMultType

### Fonctions membres publiques

- `BlocGeneEtVecMultType` (`int n=1`, `int m=1`)
- `BlocGeneEtVecMultType` (`const BlocGeneEtVecMultType &a`)
- `const string & NomRef` () const
- `const string & Mot_clef` () const
- `bool MemeCibleMaisDataDiffere`nts (`BlocGeneEtVecMultType &d`) const
- `void Affiche` () const
- `bool operator==` (`const BlocGeneEtVecMultType &a`) const
- `bool operator!=` (`const BlocGeneEtVecMultType &a`) const
- `const Coordonnee & Vect_de_coor`donnee (`int i`) const



- const double & **Val** (int i) const
- int **DimVect** () const
- int **DimVal** () const
- const string & **Nom\_vect** (int i, int j) const
- const string & **Nom\_val** (int i) const
- **BlocGeneEtVecMultType** & **operator=** (const **BlocGeneEtVecMultType** &a)
- void **Lecture** (**UtilLecture** &entreePrinc)
- void **Lecture\_entete** (**UtilLecture** &entreePrinc)
- void **Change\_vect\_de\_cooronnee** (int i, const **Coordonnee** &a)
- string \* **NomMaillage** ()
- void **Change\_val** (const int i, double val)
- void **Change\_nom\_val** (int i, const string &nom)
- void **Change\_vect\_coorpt** (const **Tableau**< **Coordonnee** > &vect)
- void **Change\_vect\_coorpt** (const list< **Coordonnee** > &livect)
- void **Change\_tab\_val** (const **Tableau**< double > &t\_val)
- void **Change\_tab\_val** (const list< double > &li)
- void **Change\_ptnom\_vect** (const **Tableau**< **Tableau**< string > > &t\_ptnom\_vect)
- void **Change\_nom\_vect** (int i, int j, const string &nom)
- void **Change\_ptnom\_val** (const **Tableau**< string > &t\_ptnom\_val)
- void **Change\_ptnom\_val** (const list< string > &li)

### Attributs protégés

- **Tableau**< **Coordonnee** > **vect\_coorpt**
- **Tableau**< **Tableau**< string > > **ptnom\_vect**
- **Tableau**< double > **tab\_val**
- **Tableau**< string > **ptnom\_tab\_val**
- string **nomref**
- string **motClef**

### Amis

- istream & **operator**>> (istream &entree, **BlocGeneEtVecMultType** &)
- ostream & **operator**<< (ostream &sort, const **BlocGeneEtVecMultType** &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.cc

## 6.54 Référence de la classe BlocScal

### Fonctions membres publiques

- **BlocScal** (const string nom, const double valeur)
- **BlocScal** (const **BlocScal** &a)
- const string & **NomRef** () const
- bool **MemeCibleMaisDataDifferentes** (**BlocScal** &) const
- void **Affiche** () const
- bool **operator==** (const **BlocScal** &a) const
- const double **Val** () const
- bool **operator!=** (const **BlocScal** &a) const
- bool **operator<** (const **BlocScal** &a) const
- void **Change\_Nomref** (const string newnom)
- void **Lecture** (**UtilLecture** &entreePrinc)
- string \* **NomMaillage** ()
- **BlocScal** & **operator=** (const **BlocScal** &a)
- void **Change\_val** (const double a)

### Attributs protégés

- double **val**
- string **nomref**

## Amis

- `istream & operator>>` (`istream &`, [BlocScal &](#))
- `ostream & operator<<` (`ostream &`, `const` [BlocScal &](#))

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.cc`

## 6.55 Référence de la classe BlocScal\_ou\_fctnD

### Fonctions membres publiques

- `BlocScal_ou_fctnD` (`const string nom`, `const double valeur`)
- `BlocScal_ou_fctnD` (`const` [BlocScal\\_ou\\_fctnD &a](#))
- `const string & NomRef` () `const`
- `bool MemeCibleMaisDataDifferentes` ([BlocScal\\_ou\\_fctnD &](#)) `const`
- `void Affiche` () `const`
- `bool operator==` (`const` [BlocScal\\_ou\\_fctnD &a](#)) `const`
- `const double * Val` () `const`
- `const string * Fct_nD` () `const`
- `bool operator!=` (`const` [BlocScal\\_ou\\_fctnD &a](#)) `const`
- `bool operator<` (`const` [BlocScal\\_ou\\_fctnD &a](#)) `const`
- `void Change_Nomref` (`const string newnom`)
- `void Lecture` ([UtilLecture &entreePrinc](#))
- `string * NomMaillage` ()
- [BlocScal\\_ou\\_fctnD & operator=](#) (`const` [BlocScal\\_ou\\_fctnD &a](#))
- `void Change_val` (`const double a`)
- `void Change_fctnD` (`const string a`)

### Attributs protégés

- `double * val`
- `string * fctnD`
- `string nomref`

## Amis

- `istream & operator>>` (`istream &`, [BlocScal\\_ou\\_fctnD &](#))
- `ostream & operator<<` (`ostream &`, `const` [BlocScal\\_ou\\_fctnD &](#))

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.cc`

## 6.56 Référence de la classe BlocScalType

### Fonctions membres publiques

- `BlocScalType` (`const` [BlocScalType &a](#))
- `const string & NomRef` () `const`
- `const string & Mot_clef` () `const`
- `bool MemeCibleMaisDataDifferentes` ([BlocScalType &d](#)) `const`
- `void Affiche` () `const`
- `bool operator==` (`const` [BlocScalType &a](#)) `const`
- `bool operator!=` (`const` [BlocScalType &a](#)) `const`
- `const double Val` () `const`
- `string * NomMaillage` ()
- `void Change_Nomref` (`const string newnom`)
- `void Change_Mot_clef` (`const string newnom`)
- `void Lecture` ([UtilLecture &entreePrinc](#))
- [BlocScalType & operator=](#) (`const` [BlocScalType &a](#))
- `void Change_val` (`const double a`)

### Attributs protégés

- double **val**
- string **nomref**
- string **motClef**

### Amis

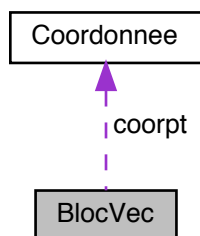
- istream & **operator**>> (istream &, [BlocScalType](#) &)
- ostream & **operator**<< (ostream &, const [BlocScalType](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.cc

## 6.57 Référence de la classe BlocVec

Graphe de collaboration de BlocVec:



### Fonctions membres publiques

- **BlocVec** (const [BlocVec](#) &a)
- const string & **NomRef** () const
- bool **MemeCibleMaisDataDifferentes** ([BlocVec](#) &) const
- void **Affiche** () const
- bool **operator==** (const [BlocVec](#) &a) const
- bool **operator!=** (const [BlocVec](#) &a) const
- const [Coordonnee](#) & **Coord** () const
- double **Coord** (int i) const
- string \* **NomMaillage** ()
- void **Lecture** ([UtilLecture](#) &entreePrinc)
- [BlocVec](#) & **operator=** (const [BlocVec](#) &a)
- void **Change\_val** ([Coordonnee](#) &a)

### Attributs protégés

- [Coordonnee](#) **coorpt**
- string **nomref**

### Amis

- istream & **operator**>> (istream &, [BlocVec](#) &)
- ostream & **operator**<< (ostream &, const [BlocVec](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.cc

## 6.58 Référence de la classe BlocVecMultType

### Fonctions membres publiques

- **BlocVecMultType** (int n=1, int m=1)
- **BlocVecMultType** (const [BlocVecMultType](#) &a)
- const string & **NomRef** () const
- const string & **Mot\_clef** () const
- bool **MemeCibleMaisDataDifferentes** ([BlocVecMultType](#) &d) const
- void **Affiche** () const
- bool **operator==** (const [BlocVecMultType](#) &a) const
- bool **operator!=** (const [BlocVecMultType](#) &a) const
- const [Coordonnee](#) & **Vect\_de\_cooronnee** (int i) const
- const double & **Val** (int i) const
- int **DimVect** () const
- int **DimVal** () const
- string \* **NomMaillage** ()
- void **Lecture** ([UtilLecture](#) &entreePrinc)
- [BlocVecMultType](#) & **operator=** (const [BlocVecMultType](#) &a)
- void **Change\_vect\_de\_cooronnee** (int i, const [Coordonnee](#) &a)
- void **Change\_val** (const int i, double val)

### Attributs protégés

- [Tableau](#)< [Coordonnee](#) > **vect\_coorpt**
- [Tableau](#)< double > **pt**
- string **nomref**
- string **motClef**

### Amis

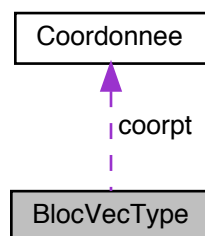
- istream & **operator>>** (istream &, [BlocVecMultType](#) &)
- ostream & **operator<<** (ostream &, const [BlocVecMultType](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.cc

## 6.59 Référence de la classe BlocVecType

Graphe de collaboration de BlocVecType:



## Fonctions membres publiques

- **BlocVecType** (const [BlocVecType](#) &a)
- const string & **NomRef** () const
- const string & **Mot\_clef** () const
- bool **MemeCibleMaisDataDifferent** ([BlocVecType](#) &d) const
- void **Affiche** () const
- bool **operator==** (const [BlocVecType](#) &a) const
- bool **operator!=** (const [BlocVecType](#) &a) const
- const [Coordonnee](#) & **Coord** () const
- double **Coord** (int i) const
- string \* **NomMaillage** ()
- void **Lecture** ([UtilLecture](#) &entreePrinc)
- void **Change\_val** ([Coordonnee](#) &a)
- [BlocVecType](#) & **operator=** (const [BlocVecType](#) &a)

## Attributs protégés

- [Coordonnee](#) **coorpt**
- string **nomref**
- string **motClef**

## Amis

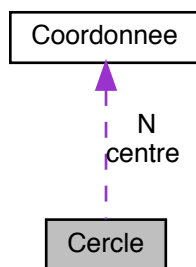
- istream & **operator>>** (istream &, [BlocVecType](#) &)
- ostream & **operator<<** (ostream &, const [BlocVecType](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/Bloc.cc

## 6.60 Référence de la classe Cercle

Graphe de collaboration de Cercle:



## Fonctions membres publiques

- **Cercle** (const [Coordonnee](#) &B, const double &r, const [Coordonnee](#) \*N)
- **Cercle** (int dim)
- **Cercle** (const [Cercle](#) &a)
- [Cercle](#) & **operator=** (const [Cercle](#) &P)
- const [Coordonnee](#) & **CentreCercle** () const
- double **RayonCercle** () const
- const [Coordonnee](#) \* **NormaleCercle** () const

- void **Change\_centre** (const [Coordonnee](#) &B)
- void **Change\_rayon** (const double &r)
- void **Change\_Normale** (const [Coordonnee](#) &N)
- void **change\_donnees** (const [Coordonnee](#) &B, const double &r, const [Coordonnee](#) \*N)
- int **Intersection** (const [Droite](#) &D, [Coordonnee](#) &M1, [Coordonnee](#) &M2)
- double **Distance\_au\_Cercle** (const [Coordonnee](#) &M) const
- [Coordonnee](#) **Projete** (const [Coordonnee](#) &M) const
- bool **ProjeteDedans** (const [Coordonnee](#) &M) const
- bool **Dedans** (const [Coordonnee](#) &M) const

### Attributs protégés

- [Coordonnee](#) **centre**
- double **rayon**
- [Coordonnee](#) \* **N**

### Amis

- istream & **operator**>> (istream &, [Cercle](#) &)
- ostream & **operator**<< (ostream &, const [Cercle](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Cercle.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Cercle.cc

## 6.61 Référence de la classe [Tenseur1BBB::ChangementIndex](#)

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-1.h

## 6.62 Référence de la classe [Tenseur1BBHH::ChangementIndex](#)

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-1.h

## 6.63 Référence de la classe [Tenseur1HHBB::ChangementIndex](#)

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-1.h

## 6.64 Référence de la classe Tenseur1HHHH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-1.h

## 6.65 Référence de la classe Tenseur2BB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h

## 6.66 Référence de la classe Tenseur2BBBB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-2.h

## 6.67 Référence de la classe Tenseur2BBHH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-2.h

## 6.68 Référence de la classe Tenseur2BH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h

## 6.69 Référence de la classe Tenseur2HB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h

## 6.70 Référence de la classe Tenseur2HH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h

## 6.71 Référence de la classe Tenseur2HHBB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-2.h

## 6.72 Référence de la classe Tenseur2HHHH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-2.h

## 6.73 Référence de la classe Tenseur3BB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h



## 6.74 Référence de la classe Tenseur3BBBB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3.h.old.h

## 6.75 Référence de la classe Tenseur3BBHH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3.h.old.h

## 6.76 Référence de la classe Tenseur3BH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h

## 6.77 Référence de la classe Tenseur3HB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h

## 6.78 Référence de la classe Tenseur3HH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h

## 6.79 Référence de la classe Tenseur3HHBB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3.h.old.h

## 6.80 Référence de la classe Tenseur3HHHH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3.h.old.h

## 6.81 Référence de la classe Tenseur\_ns2BB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h

## 6.82 Référence de la classe Tenseur\_ns2HH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h

## 6.83 Référence de la classe Tenseur\_ns3BB::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h

## 6.84 Référence de la classe Tenseur\_ns3HH::ChangementIndex

### Attributs publics

- [Tableau](#)< int > **idx\_i**
- [Tableau](#)< int > **idx\_j**
- [Tableau2](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h

## 6.85 Référence de la classe TenseurQ3\_troisSym\_BBBB::ChangementIndex

### Attributs publics

- [Tableau4](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3\_TroisSym.h

## 6.86 Référence de la classe TenseurQ3\_troisSym\_HHHH::ChangementIndex

### Attributs publics

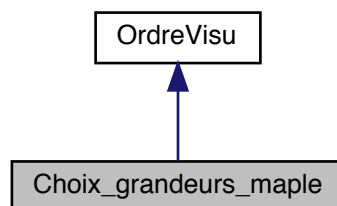
- [Tableau4](#)< int > **odVect**

La documentation de cette classe a été générée à partir du fichier suivant :

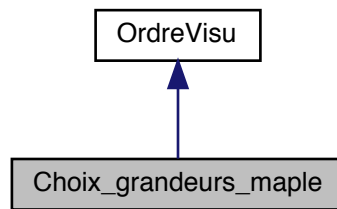
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3\_TroisSym.h

## 6.87 Référence de la classe Choix\_grandeurs\_maple

Grappe d'héritage de Choix\_grandeurs\_maple:



Graphe de collaboration de Choix\_grandeurs\_maple:



## Fonctions membres publiques

- **Choix\_grandeurs\_maple** (const [Choix\\_grandeurs\\_maple](#) &algo)
- void [Initialisation](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, EnumTypeIncre type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob, bool fil\_calcul)
- void [ExeOrdre](#) ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, OrdreVisu::EnumTypeIncre type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob)
- void [ChoixOrdre](#) ()
- void [Init\\_liste\\_grandeurs](#) ([LesMaillages](#) \*lesMail, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, bool fil\_calcul)
- void [Entete\\_fichier\\_maple](#) (const list< int > &list\_mail, ostream &sort)
- bool [Choix\\_deux\\_grandeurs](#) (int num\_mail)
- void [Jonction\\_Animation\\_maple](#) ([Animation\\_maple](#) \*choix)
- void [Jonction\\_ChoixDesMaillages](#) (const [ChoixDesMaillages\\_vrml](#) \*choix\_m)
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)

## Membres hérités additionnels

### 6.87.1 Documentation des fonctions membres

#### 6.87.1.1 ChoixOrdre()

```
void Choix_grandeurs_maple::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.87.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Choix_grandeurs_maple::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.87.1.3 ExeOrdre()

```
void Choix_grandeurs_maple::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.87.1.4 Initialisation()

```
void Choix_grandeurs_maple::Initialisation (
    ParaGlob * paraGlob,
    LesMaillages * lesmail,
    LesReferences * lesRefer,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    EnumTypeIncre type_incre,
    int incre,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob,
    bool fil_calcul ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.87.1.5 Lecture\_parametres\_OrdreVisu()

```
void Choix_grandeurs_maple::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

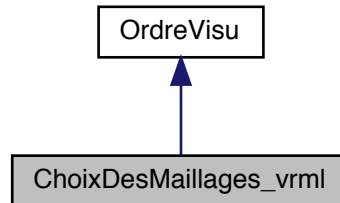
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

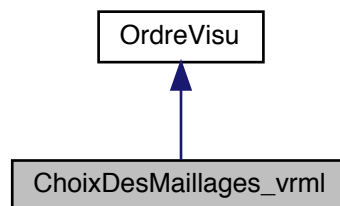
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Choix\_grandeurs\_maple.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Choix\_grandeurs\_maple.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Choix\_grandeurs\_maple2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Choix\_grandeurs\_maple3.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Copie\_de\_Choix\_grandeurs\_maple.cc ↵

## 6.88 Référence de la classe ChoixDesMaillages\_vrml

Graphe d'héritage de ChoixDesMaillages\_vrml:



Graphe de collaboration de ChoixDesMaillages\_vrml:



### Fonctions membres publiques

- **ChoixDesMaillages\_vrml** (const [ChoixDesMaillages\\_vrml](#) &ord)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#) ←  
::EnumTypeIncre, int, bool, const map< string, const double \* , std::less< string > > &, const [List\\_io](#)<  
[TypeQuelconque](#) > &listeVecGlob)
- void **ChoixOrdre** ()
- void **Init\_nb\_maill** (int &nb\_maillage\_oss)
- const list< int > & **List\_choisit** () const
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

### Membres hérités additionnels

#### 6.88.1 Documentation des fonctions membres

##### 6.88.1.1 ChoixOrdre()

```
void ChoixDesMaillages_vrml::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.88.1.2 Ecriture\_parametres\_OrdreVisu()

```
void ChoixDesMaillages_vrml::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.88.1.3 ExeOrdre()

```
void ChoixDesMaillages_vrml::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
    const map< string, const double *, std::less< string > > & ,
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.88.1.4 Lecture\_parametres\_OrdreVisu()

```
void ChoixDesMaillages_vrml::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

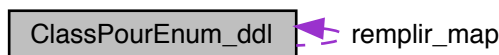
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/ChoixDesMaillages\_vrml.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/ChoixDesMaillages\_vrml.cc

## 6.89 Référence de la classe ClassPourEnum\_ddl

classe utilitaire entre Enum\_ddl et une map

```
#include <Enum_ddl.h>
```

Graphes de collaboration de ClassPourEnum\_ddl:



### Fonctions membres publiques

- [ClassPourEnum\\_ddl](#) ()

*le constructeur qui remplit effectivement la map*

### Attributs publics statiques

- static map< string, Enum\_ddl, std::less< string > > [map\\_Enum\\_ddl](#)  
*def de la map qui fait la liaison entre les string et les énumérés*
- static [ClassPourEnum\\_ddl](#) [remplir\\_map](#)  
*def de la grandeur statique qui permet de remplir la map*

### Attributs protégés

- map< string, Enum\_ddl, std::less< string > >::iterator **il**  
*variable de travail*
- map< string, Enum\_ddl, std::less< string > >::iterator **ilfin**

### Amis

- Enum\_ddl **Id\_nom\_ddl** (const string &nom)  
*Retourne l'identificateur de type enumere associe au nom du degre de liberte nom\_ddl Enum\_ddl Id\_nom\_ddl (const char\* nom\_ddl);*
- bool **ExisteEnum\_ddl** (const string &nom)  
*retourne true si l'identificateur existe, false sinon bool ExisteEnum\_ddl(const char\* nom\_ddl);*

#### 6.89.1 Description détaillée

classe utilitaire entre Enum\_ddl et une map

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/[Enum\\_ddl.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/[Enum\\_ddl\\_var\\_static.cc](#)

## 6.90 Référence de la classe ClassPourEnumTypeGrandeur

def de map qui fait la liaison entre les string et les énumérés

```
#include <EnumTypeGrandeur.h>
```

Graphe de collaboration de ClassPourEnumTypeGrandeur:



### Attributs publics statiques

- static map< string, EnumTypeGrandeur, std::less< string > > [map\\_EnumTypeGrandeur](#)  
*def de map qui fait la liaison entre les string et les énumérés*
- static map< string, EnumType2Niveau, std::less< string > > [map\\_EnumType2Niveau](#)
- static [ClassPourEnumTypeGrandeur](#) [remplir\\_map](#)

#### 6.90.1 Description détaillée

def de map qui fait la liaison entre les string et les énumérés

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/[EnumTypeGrandeur.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/[EnumTypeGrandeur.cc](#)

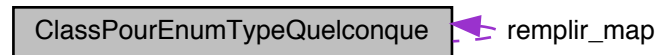


## 6.91 Référence de la classe ClassPourEnumTypeQuelconque

def de la map qui fait la liaison entre les string et les énumérés

```
#include <Enum_TypeQuelconque.h>
```

Graphe de collaboration de ClassPourEnumTypeQuelconque:



### Fonctions membres publiques statiques

- static int **NombreEnumTypeQuelconque** ()

### Attributs publics statiques

- static map< string, [EnumTypeQuelconque](#), std::less< string > > **map\_EnumTypeQuelconque**
- static [ClassPourEnumTypeQuelconque](#) **remplir\_map**
- static [Tableau](#)< int > **tt\_GLOB**
- static [Tableau](#)< [Enum\\_dure](#) > **tt\_TQ\_temps**

#### 6.91.1 Description détaillée

def de la map qui fait la liaison entre les string et les énumérés

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/[Enum\\_TypeQuelconque.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/[Enum\\_TypeQuelconque.cc](#)

## 6.92 Référence de la classe ClassPourEnuTypeCL

def de la map qui fait la liaison entre les string et les énumérés

```
#include <EnuTypeCL.h>
```

Graphe de collaboration de ClassPourEnuTypeCL:



### Attributs publics statiques

- static map< string, [EnuTypeCL](#), std::less< string > > **map\_EnuTypeCL**  
*def de la map qui fait la liaison entre les string et les énumérés*
- static [ClassPourEnuTypeCL](#) **remplir\_map**

### 6.92.1 Description détaillée

def de la map qui fait la liaison entre les string et les énumérés

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/[EnuTypeCL.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnuTypeCL.cc

## 6.93 Référence de la classe ClassPourEnuTypeQuelParticulier

def de la map qui fait la liaison entre les string et les énumérés

```
#include <EnuTypeQuelParticulier.h>
```

Graphe de collaboration de ClassPourEnuTypeQuelParticulier:



### Attributs publics statiques

- static map< string, [EnuTypeQuelParticulier](#), std::less< string > > **map\_EnuTypeQuelParticulier**  
*def de la map qui fait la liaison entre les string et les énumérés*
- static [ClassPourEnuTypeQuelParticulier](#) **remplir\_map**

### 6.93.1 Description détaillée

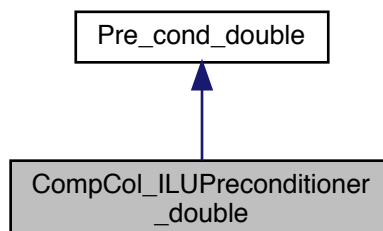
def de la map qui fait la liaison entre les string et les énumérés

La documentation de cette classe a été générée à partir du fichier suivant :

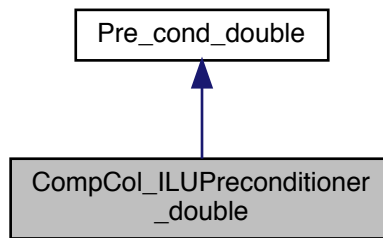
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/[EnuTypeQuelParticulier.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnuTypeQuelParticulier.cc

## 6.94 Référence de la classe CompCol\_ILUPreconditioner\_double

Graphe d'héritage de CompCol\_ILUPreconditioner\_double:



Graphe de collaboration de CompCol\_ILUPreconditioner\_double:



### Fonctions membres publiques

- **CompCol\_ILUPreconditioner\_double** (const CompCol\_Mat\_double &A)
- **CompCol\_ILUPreconditioner\_double** (const [Mat\\_abstraite](#) &A)
- VECTOR\_double [solve](#) (const VECTOR\_double &x) const
- VECTOR\_double [trans\\_solve](#) (const VECTOR\_double &x) const

### Fonctions membres protégées

- void **Init\_CompCol\_Mat\_double** (const CompCol\_Mat\_double &A)
- void **Init\_Mat\_creuse\_CompCol** (const [Mat\\_abstraite](#) &A)

## 6.94.1 Documentation des fonctions membres

### 6.94.1.1 solve()

```
VECTOR_double CompCol_ILUPreconditioner_double::solve (  
    const VECTOR_double & x ) const [virtual]
```

Implémente [Pre\\_cond\\_double](#).

### 6.94.1.2 trans\_solve()

```
VECTOR_double CompCol_ILUPreconditioner_double::trans_solve (  
    const VECTOR_double & x ) const [virtual]
```

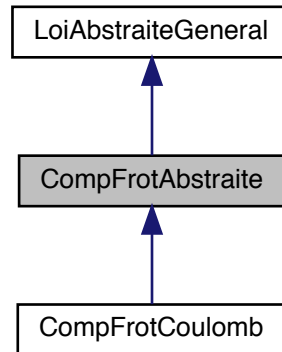
Implémente [Pre\\_cond\\_double](#).

La documentation de cette classe a été générée à partir du fichier suivant :

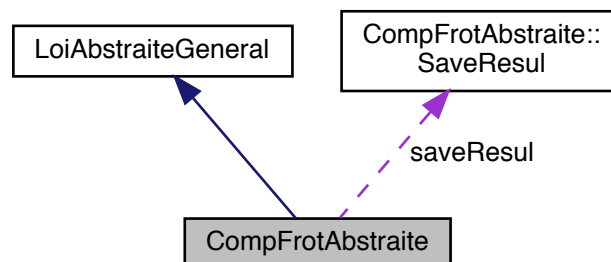
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/ilupre\_double\_GR.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/ilupre\_double\_GR.cc

## 6.95 Référence de la classe CompFrotAbstraite

Graphe d'héritage de CompFrotAbstraite:



Graphe de collaboration de CompFrotAbstraite:



### Classes

- class [SaveResul](#)

### Fonctions membres publiques

- **CompFrotAbstraite** ([Enum\\_comp](#) id\_compor, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension)
- **CompFrotAbstraite** (char \*nom, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension)
- **CompFrotAbstraite** (const [CompFrotAbstraite](#) &a)
- virtual [SaveResul](#) \* **New\_et\_Initialise** ()
- virtual void **AfficheDataSpecif** (ofstream &, [SaveResul](#) \*) const
- virtual bool **Cal\_explicit\_contact\_tdt** (const [Coordonnee](#) &vit\_T, const [Coordonnee](#) &normale, const [Coordonnee](#) &force\_normale, const [Coordonnee](#) &force\_tangente, [EnergieMeca](#) &energie\_frottement, const double delta\_t, [Coordonnee](#) &F\_frot, [CompFrotAbstraite::SaveResul](#) \*!=NULL)
- virtual bool **Cal\_implicit** (const [Coordonnee](#) &vit\_T, const [Coordonnee](#) &normale, const [Coordonnee](#) &force\_normale, const [Coordonnee](#) &force\_tangente, [EnergieMeca](#) &energie\_frottement, const double

- delta\_t, [Coordonnee](#) &F\_frot, [Tableau](#)< [Coordonnee](#) > &d\_F\_frot\_vit, [CompFrotAbstraite::SaveResul](#) \*=NULL)
- virtual void **Activation\_donnees** ()
- void **Modif\_comp\_tangent\_simplifie** (bool modif)
- bool **Test\_loi\_simplifie** ()
- virtual void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &, [CompFrotAbstraite::SaveResul](#) \*, [list](#)< int > &)
- virtual void **ListeGrandeurs\_particulieres** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &)
- virtual [CompFrotAbstraite](#) \* **Nouvelle\_loi\_identique** () const =0

### Fonctions membres protégées

- void **Info\_commande\_don\_LoisDeComp** ([UtilLecture](#) &) const
- void **Lecture\_don\_base\_info** (ifstream &, const int, [LesReferences](#) &, [LesCourbes1D](#) &, [LesFonctions\\_nD](#) &)
- void **Ecriture\_don\_base\_info** (ofstream &, const int) const
- void **Activ\_donnees** ()
- virtual bool **Calcul\_Frottement** (const [Coordonnee](#) &vit\_T, const [Coordonnee](#) &normale, const [Coordonnee](#) &force\_normale, const [Coordonnee](#) &force\_tangente, [EnergieMeca](#) &energie\_frottement, const double delta\_t, [Coordonnee](#) &F\_frot)=0
- virtual bool **Calcul\_DFrottement\_tdt** (const [Coordonnee](#) &vit\_T, const [Coordonnee](#) &normale, const [Coordonnee](#) &force\_normale, const [Coordonnee](#) &force\_tangente, [EnergieMeca](#) &energie\_frottement, const double delta\_t, [Coordonnee](#) &F\_frot, [Tableau](#)< [Coordonnee](#) > &d\_F\_frot\_vit)=0

### Attributs protégés

- [SaveResul](#) \* **saveResul**
- bool **comp\_tangent\_simplifie**

## 6.95.1 Documentation des fonctions membres

### 6.95.1.1 `Modif_comp_tangent_simplifie()`

```
void CompFrotAbstraite::Modif_comp_tangent_simplifie (
    bool modif ) [inline], [virtual]
```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

### 6.95.1.2 `Test_loi_simplifie()`

```
bool CompFrotAbstraite::Test_loi_simplifie ( ) [inline], [virtual]
```

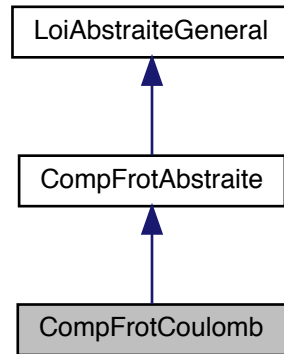
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

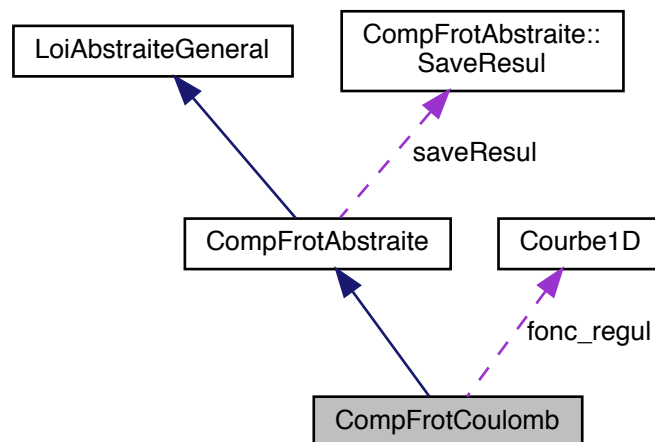
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/CompFrotAbstraite.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/CompFrotAbstraite.cc`

## 6.96 Référence de la classe CompFrotCoulomb

Grphe d'héritage de CompFrotCoulomb:



Grphe de collaboration de CompFrotCoulomb:



### Fonctions membres publiques

- **CompFrotCoulomb** ([Enum\\_comp](#) id\_compor, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension)
- **CompFrotCoulomb** (char \*nom, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension)
- **CompFrotCoulomb** (const [CompFrotCoulomb](#) &a)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)

- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- `CompFrotAbstraite * Nouvelle_loi_identique` () const
- void `Info_commande_LoisDeComp` (UtilLecture &lec)

### Fonctions membres protégées

- void `Info_commande_don_LoisDeComp` (UtilLecture &) const
- void `Lecture_don_base_info` (ifstream &, const int, LesReferences &, LesCourbes1D &)
- void `Ecriture_don_base_info` (ofstream &, const int) const
- bool `Calcul_Frottement` (const Coordonnee &vit\_T, const Coordonnee &normale, const Coordonnee &force←\_normale, const Coordonnee &force\_tangente, EnergieMeca &energie\_frottement, const double delta\_←t, Coordonnee &F\_frot)
- bool `Calcul_DFrottement_tdt` (const Coordonnee &vit\_T, const Coordonnee &normale, const Coordonnee &force\_normale, const Coordonnee &force\_tangente, EnergieMeca &energie\_frottement, const double delta\_t, Coordonnee &F\_frot, Tableau< Coordonnee > &d\_F\_frot\_vit)

### Attributs protégés

- double `mu_statique`
- double \* `mu_cine`
- double \* `x_c`
- int `regularisation`
- `Courbe1D * fonc_regul`
- double `epsil`

## 6.96.1 Documentation des fonctions membres

### 6.96.1.1 Affiche()

```
void CompFrotCoulomb::Affiche ( ) const [virtual]
Implémente LoiAbstraiteGeneral.
```

### 6.96.1.2 Calcul\_DFrottement\_tdt()

```
bool CompFrotCoulomb::Calcul_DFrottement_tdt (
    const Coordonnee & vit_T,
    const Coordonnee & normale,
    const Coordonnee & force_normale,
    const Coordonnee & force_tangente,
    EnergieMeca & energie_frottement,
    const double delta_t,
    Coordonnee & F_frot,
    Tableau< Coordonnee > & d_F_frot_vit ) [protected], [virtual]
```

Implémente [CompFrotAbstraite](#).

### 6.96.1.3 Calcul\_Frottement()

```
bool CompFrotCoulomb::Calcul_Frottement (
    const Coordonnee & vit_T,
    const Coordonnee & normale,
    const Coordonnee & force_normale,
    const Coordonnee & force_tangente,
    EnergieMeca & energie_frottement,
    const double delta_t,
    Coordonnee & F_frot ) [protected], [virtual]
```

Implémente [CompFrotAbstraite](#).

#### 6.96.1.4 Ecriture\_base\_info\_loi()

```
void CompFrotCoulomb::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.96.1.5 Info\_commande\_LoisDeComp()

```
void CompFrotCoulomb::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.96.1.6 Lecture\_base\_info\_loi()

```
void CompFrotCoulomb::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.96.1.7 LectureDonneesParticulieres()

```
void CompFrotCoulomb::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.96.1.8 Nouvelle\_loi\_identique()

```
CompFrotAbstraite * CompFrotCoulomb::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [CompFrotAbstraite](#).

#### 6.96.1.9 TestComplet()

```
int CompFrotCoulomb::TestComplet ( ) [virtual]
```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

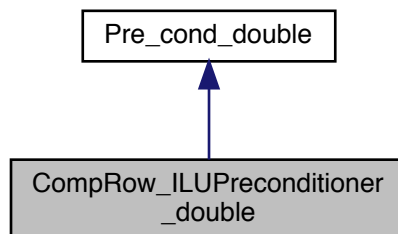
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Frottement/CompFrotCoulomb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Frottement/CompFrotCoulomb.cc

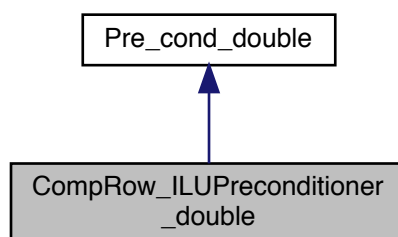


## 6.97 Référence de la classe CompRow\_ILUPreconditioner\_double

Grappe d'héritage de CompRow\_ILUPreconditioner\_double:



Grappe de collaboration de CompRow\_ILUPreconditioner\_double:



### Fonctions membres publiques

- **CompRow\_ILUPreconditioner\_double** (const CompRow\_Mat\_double &A)
- **CompRow\_ILUPreconditioner\_double** (const [Mat\\_abstraite](#) &A)
- VECTOR\_double [solve](#) (const VECTOR\_double &x) const
- VECTOR\_double [trans\\_solve](#) (const VECTOR\_double &x) const

### Fonctions membres protégées

- void [Init\\_CompRow\\_Mat\\_double](#) (const CompRow\_Mat\_double &A)

### 6.97.1 Documentation des fonctions membres

#### 6.97.1.1 solve()

```
VECTOR_double CompRow_ILUPreconditioner_double::solve (  
    const VECTOR_double & x ) const [virtual]
```

Implémente [Pre\\_cond\\_double](#).

### 6.97.1.2 trans\_solve()

```
VECTOR_double CompRow_ILUPreconditioner_double::trans_solve (
    const VECTOR_double & x ) const [virtual]
```

Implémente [Pre\\_cond\\_double](#).

La documentation de cette classe a été générée à partir du fichier suivant :

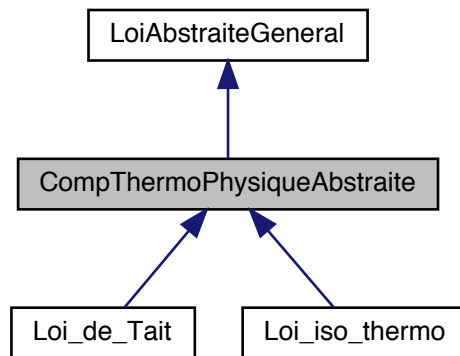
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/ilupre\_double\_GR.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/ilupre\_double\_GR.cc

## 6.98 Référence de la classe CompThermoPhysiqueAbstraite

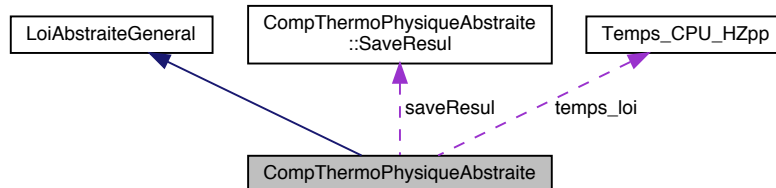
pour les comportements thermophysiques

```
#include <CompThermoPhysiqueAbstraite.h>
```

Grappe d'héritage de CompThermoPhysiqueAbstraite:



Grappe de collaboration de CompThermoPhysiqueAbstraite:



### Classes

- class [SaveResul](#)
- class [StockParalnt](#)

### Fonctions membres publiques

- **CompThermoPhysiqueAbstraite** ([Enum\\_comp](#) id\_compor, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension)

- `CompThermoPhysiqueAbstraite` (char \*nom, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension)
- `CompThermoPhysiqueAbstraite` (const [CompThermoPhysiqueAbstraite](#) &a)
- virtual `SaveResul * New_et_Initialise` ()
- virtual void `AfficheDataSpecif` (ofstream &, [SaveResul \\*](#)) const
- virtual void `Cal_donnees_thermiques` (const double &P\_t, [CompThermoPhysiqueAbstraite::SaveResul](#) \*saveTP, const [Deformation](#) &def, const double &P, [Enum\\_dure](#) temps, [ThermoDonnee](#) &donnee↔  
`Thermique`)=0
- void `Def_type_deformation` ([Deformation](#) &def)
- void `Cal_cinematique_thermique` (bool premier\_calcul, [PtIntegThermiInterne](#) &ptIntegThermi, [Tableau](#)<  
[CoordonneeB](#) > &d\_gradTB, [Deformation](#) &def, const [Met\\_abstraite::Impli](#) &ex)
- virtual const [Met\\_abstraite::Impli](#) & `Cal_implicit` (const double &P\_t, [CompThermoPhysiqueAbstraite::SaveResul](#)  
\*saveDon, const double &P, [Deformation](#) &def, [DdlElement](#) &tab\_ddl, [PtIntegThermiInterne](#) &ptIntegThermi,  
[Tableau](#)< [CoordonneeB](#) > &d\_gradTB, [Tableau](#)< [CoordonneeH](#) > &d\_fluxH, const [ParaAlgoControle](#)  
&pa, [CompThermoPhysiqueAbstraite](#) \*loiTP, bool dilatation, [EnergieThermi](#) &energ, const [EnergieThermi](#)  
&energ\_t, bool premier\_calcul)
- virtual void `Activation_donnees` ([Tableau](#)< [Noeud \\*](#) > &tabnoeud)
- void `Modif_comp_tangent_simplifie` (bool modif)
- bool `Test_loi_simplife` ()
- virtual void `Grandeur_particuliere` (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &, [CompThermoPhysiqueAbstraite::SaveResul](#)  
\*, [list](#)< int > &)
- virtual void `ListeGrandeurs_particulieres` (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &)
- virtual [CompThermoPhysiqueAbstraite \\*](#) `Nouvelle_loi_identique` () const =0

### Fonctions membres protégées

- virtual void `Calcul_DfluxH_tdt` (const double &P\_t, [PtIntegThermiInterne](#) &ptIntegThermi, const double  
&P, [DdlElement](#) &tab\_ddl, const [Deformation](#) &def, [Tableau](#)< [CoordonneeB](#) > &d\_gradTB, [Tableau](#)<  
[CoordonneeH](#) > &d\_flux, [ThermoDonnee](#) &dTP, [EnergieThermi](#) &energ, const [EnergieThermi](#) &energ\_↔  
t, const [Met\\_abstraite::Impli](#) &ex)=0
- void `Info_commande_don LoisDeComp` ([UtilLecture](#) &) const
- void `Lecture_don_base_info` (ifstream &, const int, [LesReferences](#) &, [LesCourbes1D](#) &, [LesFonctions\\_nD](#)  
&)
- void `Ecriture_don_base_info` (ofstream &, const int) const
- void `Activ_donnees` ([Tableau](#)< [Noeud \\*](#) > &tabnoeud)
- virtual void `CalculGrandeurTravail` (const [PtIntegThermiInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#))

### Attributs protégés

- [SaveResul \\*](#) `saveResul`
- bool `comp_tangent_simplifie`
- bool `thermo_dependant`
- double `temperature`
- [Temps\\_CPU\\_HZpp](#) `temps_loi`

## 6.98.1 Description détaillée

pour les comportements thermophysiques

## 6.98.2 Documentation des fonctions membres

### 6.98.2.1 `Modif_comp_tangent_simplifie()`

```
void CompThermoPhysiqueAbstraite::Modif_comp_tangent_simplifie (
    bool modif ) [inline], [virtual]
```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

### 6.98.2.2 Test\_loi\_simplife()

```
bool CompThermoPhysiqueAbstraite::Test_loi_simplife ( ) [inline], [virtual]
```

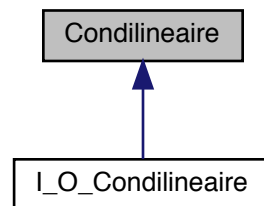
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

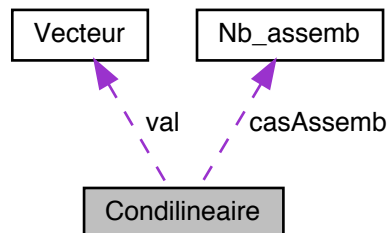
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/CompThermoPhysiqueAbstraite.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/CompThermoPhysiqueAbstraite.cc

## 6.99 Référence de la classe Condilinaire

Grphe d'héritage de Condilinaire:



Grphe de collaboration de Condilinaire:



### Fonctions membres publiques

- **Condilinaire** ([Tableau](#)< Enum\_ddl > &t\_enuu, const [Tableau](#)< int > &ptt, const [Vecteur](#) &vall, double betar, int posiddl, const [Tableau](#)< [Noeud](#) \* > &t\_n)
- **Condilinaire** ([Tableau](#)< Enum\_ddl > &t\_enuu, const [Tableau](#)< [Noeud](#) \* > &t\_n)
- **Condilinaire** (const [Condilinaire](#) &a)
- [Condilinaire](#) & **operator=** (const [Condilinaire](#) &cond)
- const [Tableau](#)< int > & **Pt\_t** () const
- [Tableau](#)< int > & **ChangePt** ()
- const [Vecteur](#) & **Val** () const
- [Vecteur](#) & **Valchange** ()
- void **ChangeCoeff** (const [Vecteur](#) &v)
- double **Beta** () const
- double & **BetaChange** ()

- void **ChangeBeta** (const double &x)
- void **ChangeUk\_impose** (double Uk\_new)
- const double & **Val\_Uk\_impose** () const
- **Noeud \* Noe** ()
- void **Change\_tab\_enum** (const **Tableau**< Enum\_ddl > &t\_enuu)
- const **Tableau**< **Noeud \* >** & **TabNoeud** () const
- void **ChangeTabNoeud** (const **Tableau**< **Noeud \* >** &t\_n)
- const int & **iddl** () const
- int & **Changeiddl** ()
- void **Change\_taille** (int taille)
- const **Condilinaire** & **ConditionPourPointeursAssemblage** (const **Nb\_assemb** &nb\_casAssemb)
- const **Nb\_assemb** & **CasAssemb** () const
- const **Tableau**< Enum\_ddl > & **Tab\_Enum** () const
- int **DiffMaxiNumeroNoeud** () const
- void **Largeur\_Bande** (int &demi, int &total, const **Nb\_assemb** &casAssemb)
- void **Affiche** () const
- void **Lecture\_base\_info** (**Tableau**< int > &numMaillage, ifstream &ent, **Tableau**< int > &numNoeud)
- void **Ecriture\_base\_info** (ofstream &sort)

### Attributs protégés

- **Tableau**< int > **pt**
- **Vecteur val**
- double **beta**
- double **Uk\_impose**
- **Nb\_assemb casAssemb**
- **Tableau**< Enum\_ddl > **t\_enu**
- int **iddl**
- **Tableau**< **Noeud \* >** **t\_noeud**

### Amis

- istream & **operator**>> (istream &ent, **Condilinaire** &)
- ostream & **operator**<< (ostream &sort, const **Condilinaire** &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution\_Condi/Condilinaire.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution\_Condi/Condilinaire.cc

## 6.100 Référence de la classe CondLim

### Fonctions membres publiques

- void **Val\_imposee\_Sm** (**Vecteur** &vecglob, int i, double val, **Vecteur \*vec2**)
- void **Val\_imposee\_Mat** (**Mat\_abstraite** &matglob, **Vecteur** &vecglob, int i, double val, **Vecteur \*vec2**)
- void **Val\_imposSimple\_Mat** (**Mat\_abstraite** &matglob, int i, double val)
- double **RemonteDdlBloqueMat** (**Mat\_abstraite** &matglob, **Vecteur** &solution, int i)
- double **MaxEffort** (int &ili)
- double **ValReact** (int &ili)
- void **CondilinaireCHRepere** (**Mat\_abstraite** &matglob, **Vecteur** &vecglob, const **Tableau**< int > &pt, const **Vecteur** &val, double valeur, **Vecteur \*vec2**)
- void **CondilinaireImpose** (**Mat\_abstraite** &matglob, **Vecteur** &vecglob, **Vecteur \*vec2**)
- void **Replinitiaux** (**Vecteur** &sol)
- void **CondiLineaireImposeComple** (**Mat\_abstraite** &matglob, **Vecteur** &vecglob, const **Tableau**< int > &pt, const **Vecteur** &val, double valeur, **Vecteur \*vec2**)
- void **EffaceSauvegarde** ()
- void **EffaceCoLin** ()

### Amis

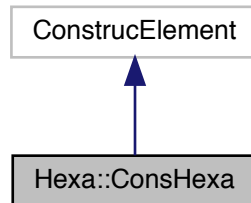
- istream & **operator**>> (istream &ent, **CondLim** &)
- ostream & **operator**<< (ostream &sort, const **CondLim** &)

La documentation de cette classe a été générée à partir du fichier suivant :

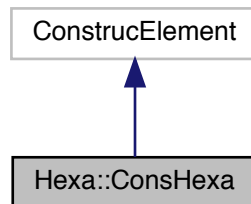
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution\_Condi/CondLim.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Resolution\_Condi/CondLim.cc

## 6.101 Référence de la classe Hexa::ConsHexa

Grphe d'héritage de Hexa::ConsHexa:



Grphe de collaboration de Hexa::ConsHexa:



### Fonctions membres publiques

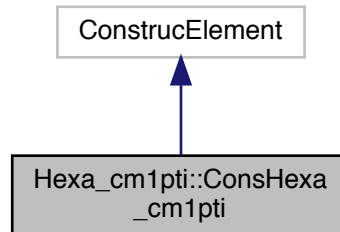
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

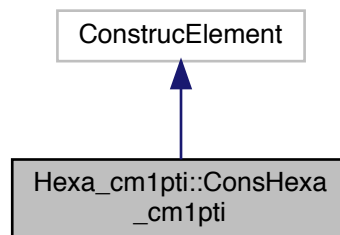
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa.h`

## 6.102 Référence de la classe Hexa\_cm1pti::ConsHexa\_cm1pti

Grphe d'héritage de Hexa\_cm1pti::ConsHexa\_cm1pti:



Grphe de collaboration de Hexa\_cm1pti::ConsHexa\_cm1pti:



### Fonctions membres publiques

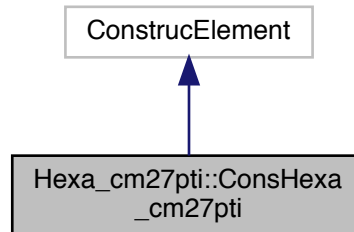
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

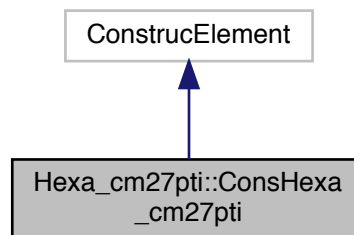
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa_cm1pti.h`

### 6.103 Référence de la classe Hexa\_cm27pti::ConsHexa\_cm27pti

Grphe d'héritage de Hexa\_cm27pti::ConsHexa\_cm27pti:



Grphe de collaboration de Hexa\_cm27pti::ConsHexa\_cm27pti:



#### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

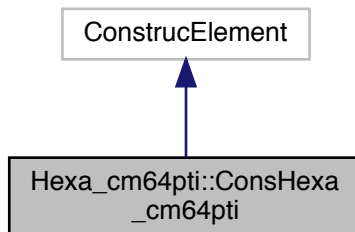
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm27pti.h

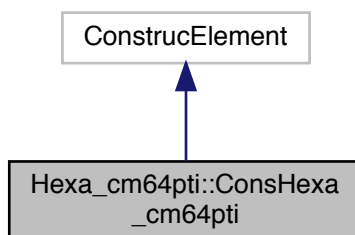


## 6.104 Référence de la classe Hexa\_cm64pti::ConsHexa\_cm64pti

Grphe d'héritage de Hexa\_cm64pti::ConsHexa\_cm64pti:



Grphe de collaboration de Hexa\_cm64pti::ConsHexa\_cm64pti:



### Fonctions membres publiques

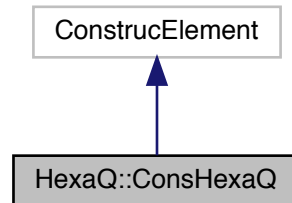
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

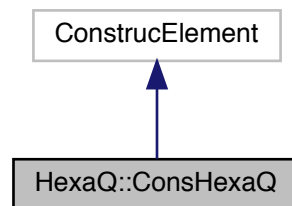
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm64pti.h

## 6.105 Référence de la classe HexaQ::ConsHexaQ

Grphe d'héritage de HexaQ::ConsHexaQ:



Grphe de collaboration de HexaQ::ConsHexaQ:



### Fonctions membres publiques

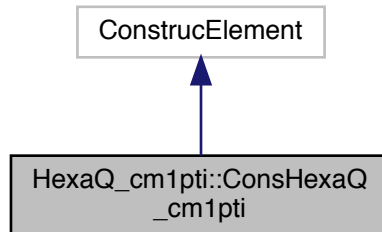
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

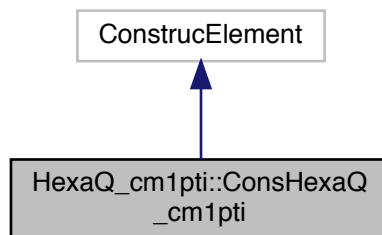
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ.h`

## 6.106 Référence de la classe HexaQ\_cm1pti::ConsHexaQ\_cm1pti

Graphe d'héritage de HexaQ\_cm1pti::ConsHexaQ\_cm1pti:



Graphe de collaboration de HexaQ\_cm1pti::ConsHexaQ\_cm1pti:



### Fonctions membres publiques

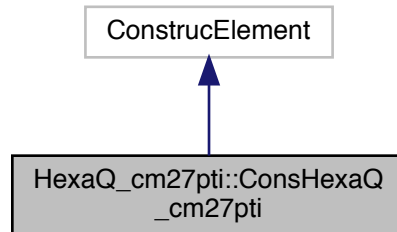
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

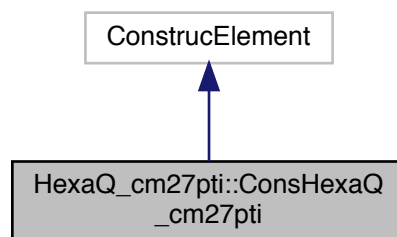
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm1pti.h

## 6.107 Référence de la classe HexaQ\_cm27pti::ConsHexaQ\_cm27pti

Grphe d'héritage de HexaQ\_cm27pti::ConsHexaQ\_cm27pti:



Grphe de collaboration de HexaQ\_cm27pti::ConsHexaQ\_cm27pti:



### Fonctions membres publiques

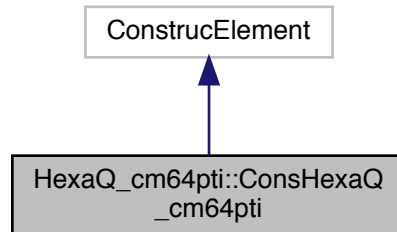
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

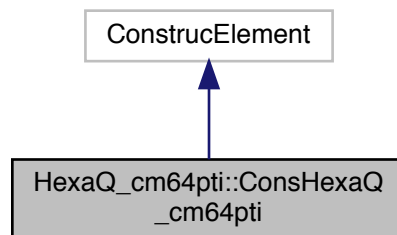
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm27pti.h

## 6.108 Référence de la classe HexaQ\_cm64pti::ConsHexaQ\_cm64pti

Grphe d'héritage de HexaQ\_cm64pti::ConsHexaQ\_cm64pti:



Grphe de collaboration de HexaQ\_cm64pti::ConsHexaQ\_cm64pti:



### Fonctions membres publiques

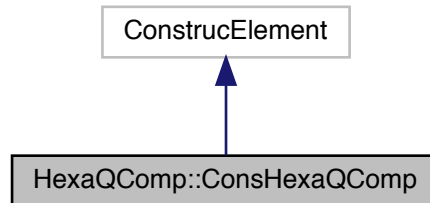
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

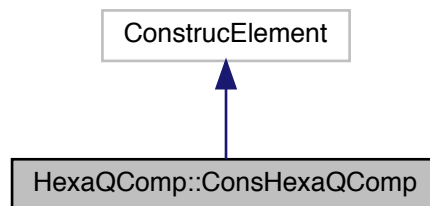
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm64pti.h

## 6.109 Référence de la classe HexaQComp::ConsHexaQComp

Grphe d'héritage de HexaQComp::ConsHexaQComp:



Grphe de collaboration de HexaQComp::ConsHexaQComp:



### Fonctions membres publiques

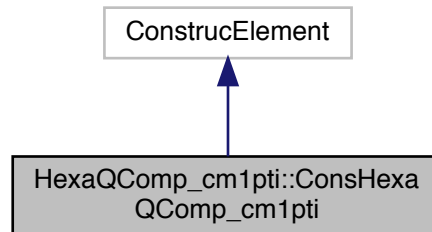
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

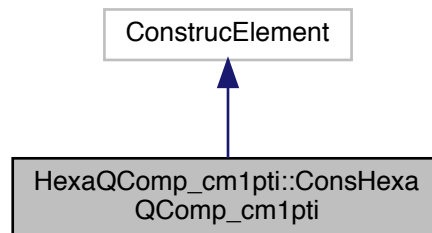
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp.h`

## 6.110 Référence de la classe HexaQComp\_cm1pti::ConsHexaQComp\_cm1pti

Graphe d'héritage de HexaQComp\_cm1pti::ConsHexaQComp\_cm1pti:



Graphe de collaboration de HexaQComp\_cm1pti::ConsHexaQComp\_cm1pti:



### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

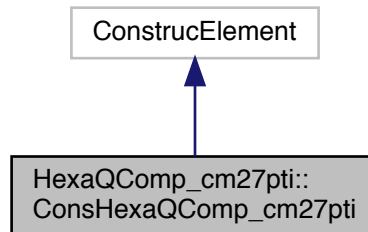
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm1pti.h

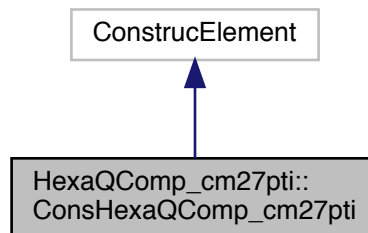
## 6.111 Référence de la classe

### HexaQComp\_cm27pti::ConsHexaQComp\_cm27pti

Grphe d'héritage de HexaQComp\_cm27pti::ConsHexaQComp\_cm27pti:



Grphe de collaboration de HexaQComp\_cm27pti::ConsHexaQComp\_cm27pti:



### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

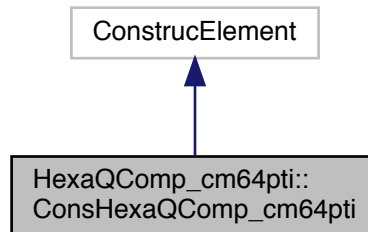
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm27pti.h

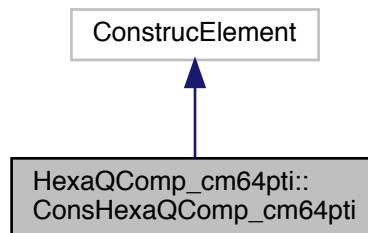


## 6.112 Référence de la classe HexaQComp\_cm64pti::ConsHexaQComp\_cm64pti

Graphe d'héritage de HexaQComp\_cm64pti::ConsHexaQComp\_cm64pti:



Graphe de collaboration de HexaQComp\_cm64pti::ConsHexaQComp\_cm64pti:



### Fonctions membres publiques

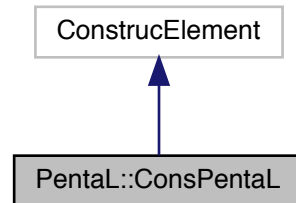
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

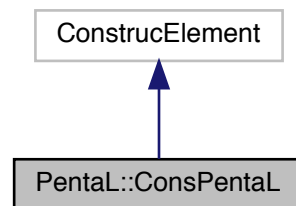
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm64pti.h

## 6.113 Référence de la classe PentaL::ConsPentaL

Grphe d'héritage de PentaL::ConsPentaL:



Grphe de collaboration de PentaL::ConsPentaL:



### Fonctions membres publiques

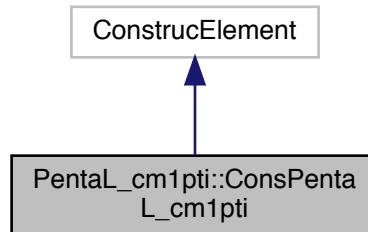
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

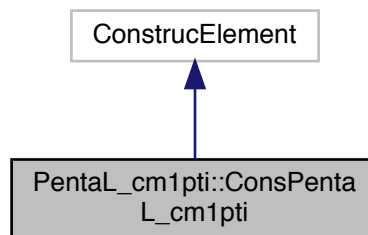
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL.h

## 6.114 Référence de la classe PentaL\_cm1pti::ConsPentaL\_cm1pti

Grphe d'héritage de PentaL\_cm1pti::ConsPentaL\_cm1pti:



Grphe de collaboration de PentaL\_cm1pti::ConsPentaL\_cm1pti:



### Fonctions membres publiques

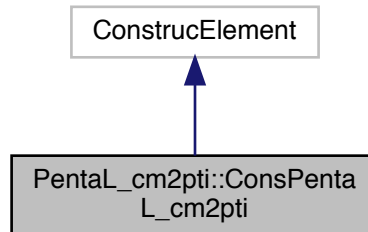
- `Element *` **NouvelElement** (`int num_maill, int num`)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

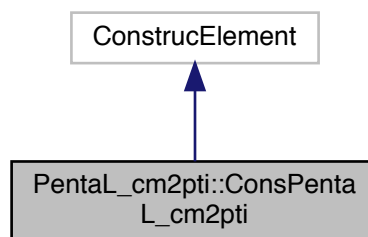
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL_cm1pti.h`

## 6.115 Référence de la classe PentaL\_cm2pti::ConsPentaL\_cm2pti

Grphe d'héritage de PentaL\_cm2pti::ConsPentaL\_cm2pti:



Grphe de collaboration de PentaL\_cm2pti::ConsPentaL\_cm2pti:



### Fonctions membres publiques

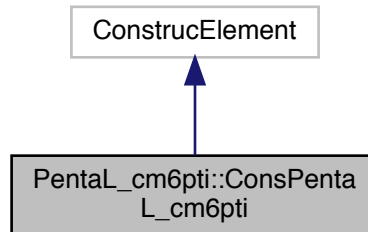
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

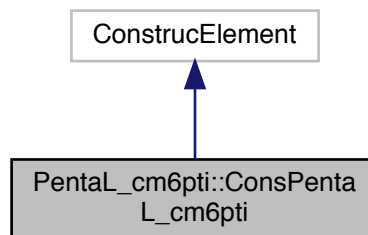
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm2pti.h

## 6.116 Référence de la classe PentaL\_cm6pti::ConsPentaL\_cm6pti

Grphe d'héritage de PentaL\_cm6pti::ConsPentaL\_cm6pti:



Grphe de collaboration de PentaL\_cm6pti::ConsPentaL\_cm6pti:



### Fonctions membres publiques

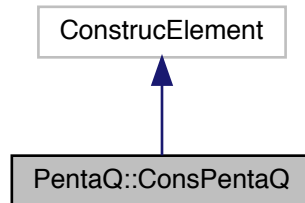
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

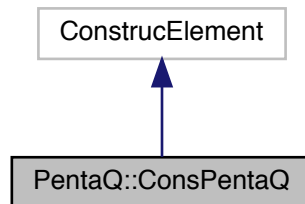
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm6pti.h

## 6.117 Référence de la classe PentaQ::ConsPentaQ

Grphe d'héritage de PentaQ::ConsPentaQ:



Grphe de collaboration de PentaQ::ConsPentaQ:



### Fonctions membres publiques

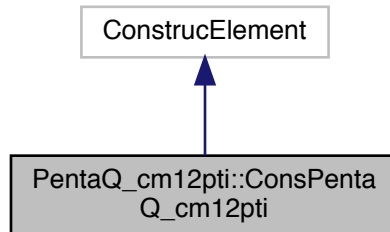
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

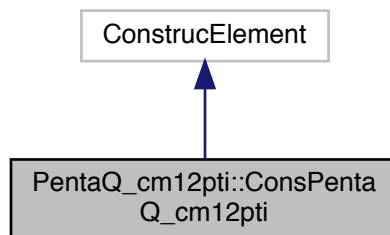
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ.h`

## 6.118 Référence de la classe PentaQ\_cm12pti::ConsPentaQ\_cm12pti

Grphe d'héritage de PentaQ\_cm12pti::ConsPentaQ\_cm12pti:



Grphe de collaboration de PentaQ\_cm12pti::ConsPentaQ\_cm12pti:



### Fonctions membres publiques

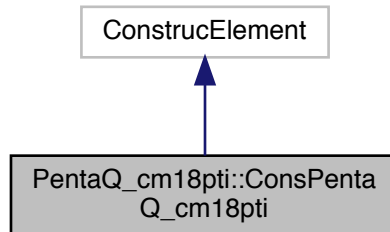
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

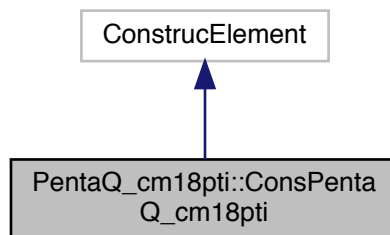
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm12pti.h

## 6.119 Référence de la classe PentaQ\_cm18pti::ConsPentaQ\_cm18pti

Grphe d'héritage de PentaQ\_cm18pti::ConsPentaQ\_cm18pti:



Grphe de collaboration de PentaQ\_cm18pti::ConsPentaQ\_cm18pti:



### Fonctions membres publiques

- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

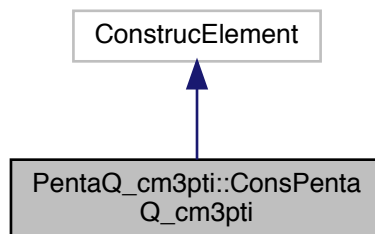
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm18pti.h`

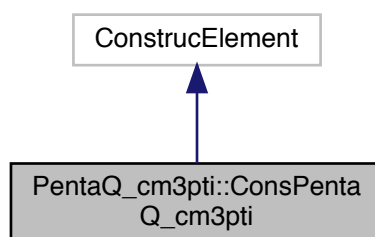


## 6.120 Référence de la classe PentaQ\_cm3pti::ConsPentaQ\_cm3pti

Grphe d'héritage de PentaQ\_cm3pti::ConsPentaQ\_cm3pti:



Grphe de collaboration de PentaQ\_cm3pti::ConsPentaQ\_cm3pti:



### Fonctions membres publiques

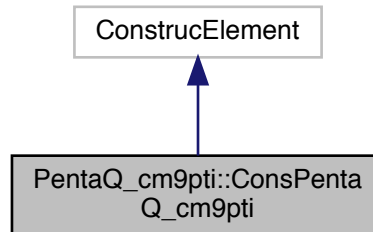
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

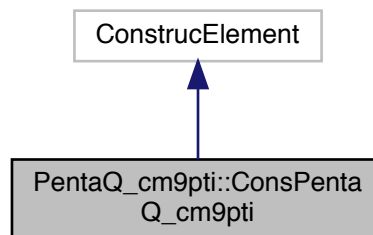
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm3pti.h

## 6.121 Référence de la classe PentaQ\_cm9pti::ConsPentaQ\_cm9pti

Grphe d'héritage de PentaQ\_cm9pti::ConsPentaQ\_cm9pti:



Grphe de collaboration de PentaQ\_cm9pti::ConsPentaQ\_cm9pti:



### Fonctions membres publiques

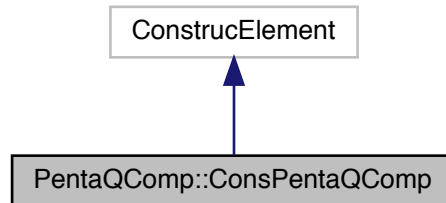
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

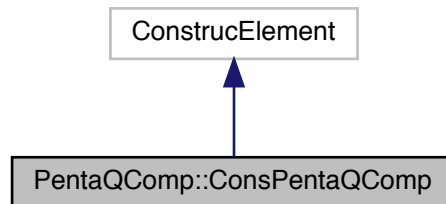
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm9pti.h

## 6.122 Référence de la classe PentaQComp::ConsPentaQComp

Grappe d'héritage de PentaQComp::ConsPentaQComp:



Grappe de collaboration de PentaQComp::ConsPentaQComp:



### Fonctions membres publiques

- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

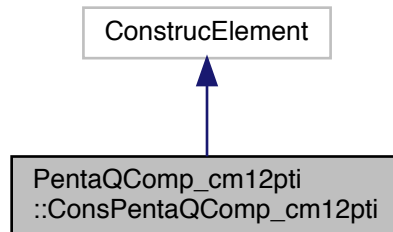
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp.h`

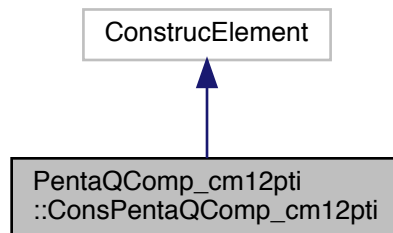
## 6.123 Référence de la classe

### PentaQComp\_cm12pti::ConsPentaQComp\_cm12pti

Graphe d'héritage de PentaQComp\_cm12pti::ConsPentaQComp\_cm12pti:



Graphe de collaboration de PentaQComp\_cm12pti::ConsPentaQComp\_cm12pti:



### Fonctions membres publiques

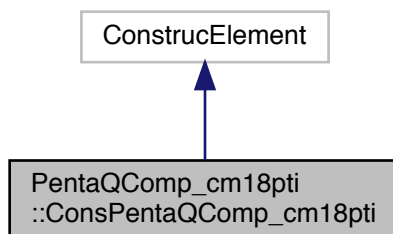
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

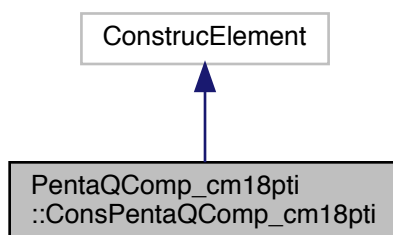
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm12pti.h

## 6.124 Référence de la classe PentaQComp\_cm18pti::ConsPentaQComp\_cm18pti

Graphe d'héritage de PentaQComp\_cm18pti::ConsPentaQComp\_cm18pti:



Graphe de collaboration de PentaQComp\_cm18pti::ConsPentaQComp\_cm18pti:



### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

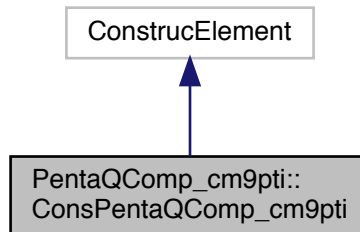
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm18pti.h

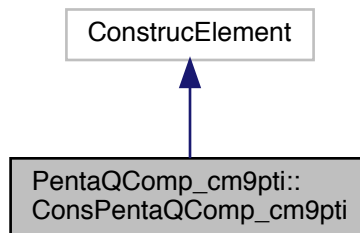
## 6.125 Référence de la classe

### PentaQComp\_cm9pti::ConsPentaQComp\_cm9pti

Graphe d'héritage de PentaQComp\_cm9pti::ConsPentaQComp\_cm9pti:



Graphe de collaboration de PentaQComp\_cm9pti::ConsPentaQComp\_cm9pti:



### Fonctions membres publiques

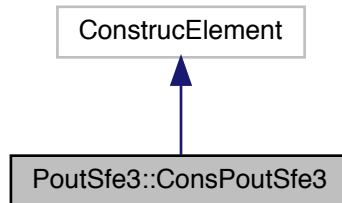
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

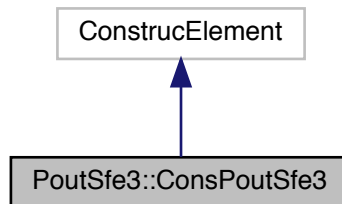
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm9pti.h

## 6.126 Référence de la classe PoutSfe3::ConsPoutSfe3

Grphe d'héritage de PoutSfe3::ConsPoutSfe3:



Grphe de collaboration de PoutSfe3::ConsPoutSfe3:



### Fonctions membres publiques

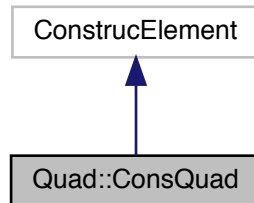
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

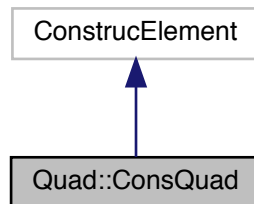
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/PoutSfe3.h`

## 6.127 Référence de la classe Quad::ConsQuad

Grphe d'héritage de Quad::ConsQuad:



Grphe de collaboration de Quad::ConsQuad:



### Fonctions membres publiques

- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

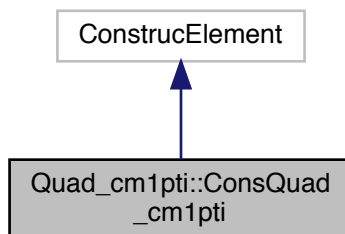
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad.h`

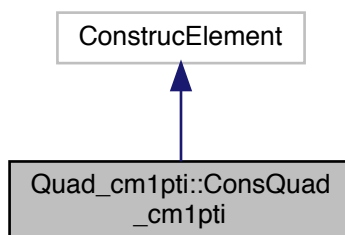


## 6.128 Référence de la classe Quad\_cm1pti::ConsQuad\_cm1pti

Grappe d'héritage de Quad\_cm1pti::ConsQuad\_cm1pti:



Grappe de collaboration de Quad\_cm1pti::ConsQuad\_cm1pti:



### Fonctions membres publiques

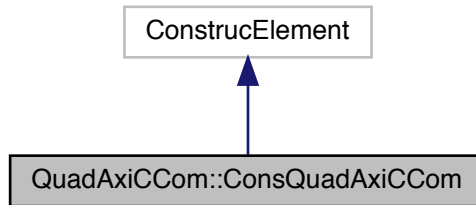
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

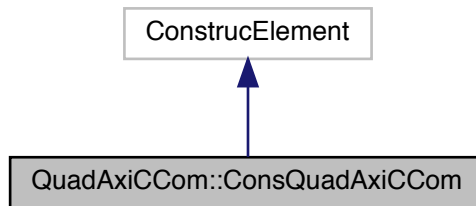
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad_cm1pti.h`

## 6.129 Référence de la classe QuadAxiCCom::ConsQuadAxiCCom

Grphe d'héritage de QuadAxiCCom::ConsQuadAxiCCom:



Grphe de collaboration de QuadAxiCCom::ConsQuadAxiCCom:



### Fonctions membres publiques

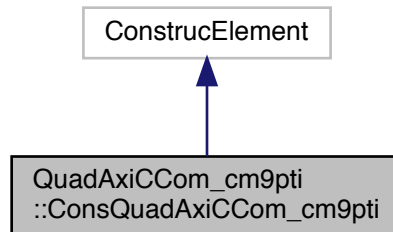
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

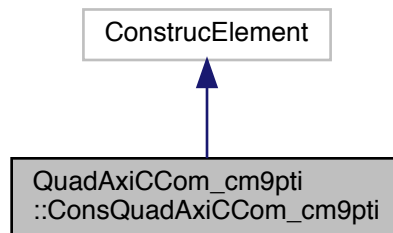
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom.h

## 6.130 Référence de la classe QuadAxiCCom\_cm9pti::ConsQuadAxiCCom\_cm9pti

Graphe d'héritage de QuadAxiCCom\_cm9pti::ConsQuadAxiCCom\_cm9pti:



Graphe de collaboration de QuadAxiCCom\_cm9pti::ConsQuadAxiCCom\_cm9pti:



### Fonctions membres publiques

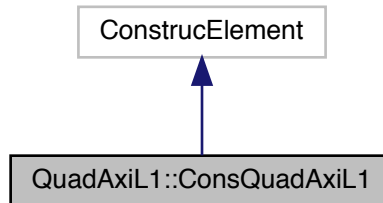
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

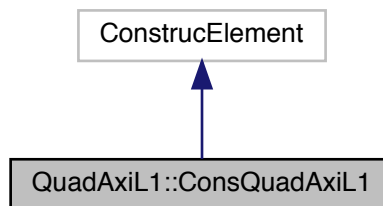
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom\_cm9pti.h

## 6.131 Référence de la classe QuadAxiL1::ConsQuadAxiL1

Grphe d'héritage de QuadAxiL1::ConsQuadAxiL1:



Grphe de collaboration de QuadAxiL1::ConsQuadAxiL1:



### Fonctions membres publiques

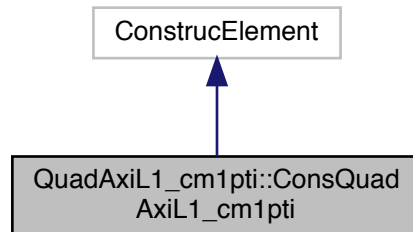
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

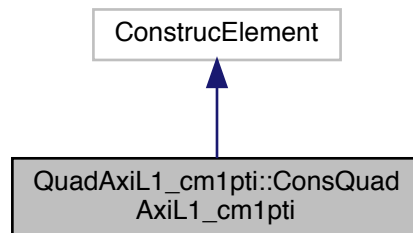
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1.h

## 6.132 Référence de la classe QuadAxiL1\_cm1pti::ConsQuadAxiL1\_cm1pti

Graphe d'héritage de QuadAxiL1\_cm1pti::ConsQuadAxiL1\_cm1pti:



Graphe de collaboration de QuadAxiL1\_cm1pti::ConsQuadAxiL1\_cm1pti:



### Fonctions membres publiques

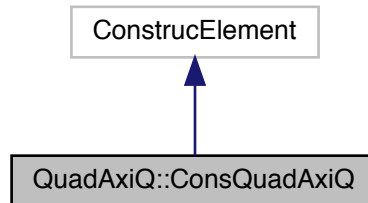
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

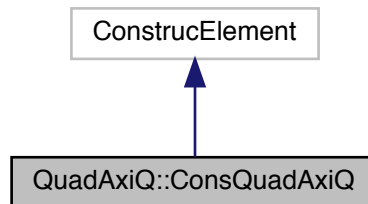
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1\_cm1pti.h

### 6.133 Référence de la classe QuadAxiQ::ConsQuadAxiQ

Grphe d'héritage de QuadAxiQ::ConsQuadAxiQ:



Grphe de collaboration de QuadAxiQ::ConsQuadAxiQ:



#### Fonctions membres publiques

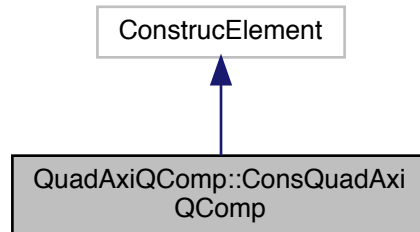
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

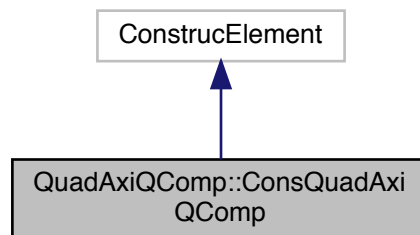
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQ.h

## 6.134 Référence de la classe QuadAxiQComp::ConsQuadAxiQComp

Graphe d'héritage de QuadAxiQComp::ConsQuadAxiQComp:



Graphe de collaboration de QuadAxiQComp::ConsQuadAxiQComp:



### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

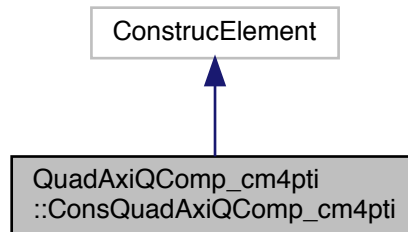
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQComp.h

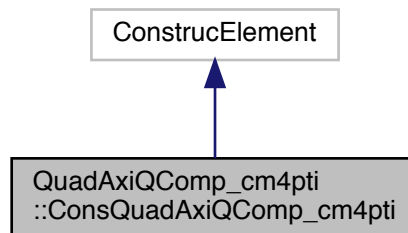
## 6.135 Référence de la classe

### QuadAxiQComp\_cm4pti::ConsQuadAxiQComp\_cm4pti

Grphe d'héritage de QuadAxiQComp\_cm4pti::ConsQuadAxiQComp\_cm4pti:



Grphe de collaboration de QuadAxiQComp\_cm4pti::ConsQuadAxiQComp\_cm4pti:



### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

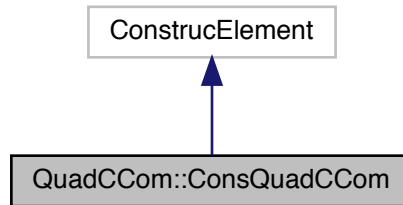
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQComp\_cm4pti.h

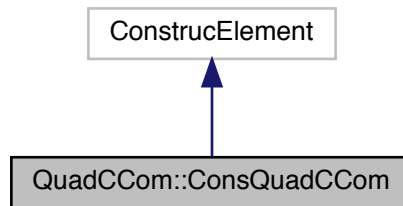


## 6.136 Référence de la classe QuadCCom::ConsQuadCCom

Grphe d'héritage de QuadCCom::ConsQuadCCom:



Grphe de collaboration de QuadCCom::ConsQuadCCom:



### Fonctions membres publiques

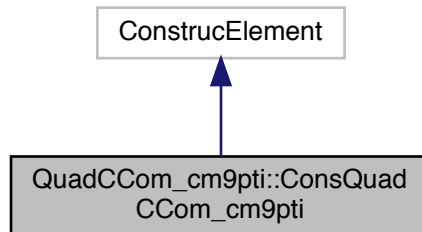
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

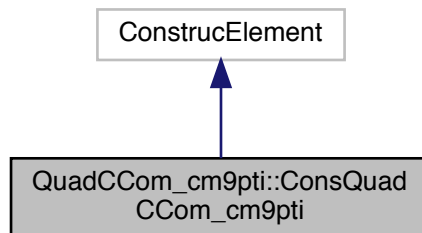
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom.h`

## 6.137 Référence de la classe QuadCCom\_cm9pti::ConsQuadCCom\_cm9pti

Graphe d'héritage de QuadCCom\_cm9pti::ConsQuadCCom\_cm9pti:



Graphe de collaboration de QuadCCom\_cm9pti::ConsQuadCCom\_cm9pti:



### Fonctions membres publiques

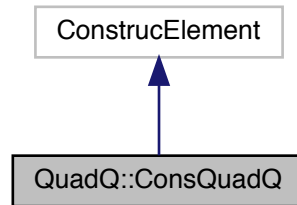
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

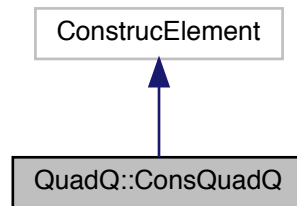
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom\_cm9pti.h

## 6.138 Référence de la classe QuadQ::ConsQuadQ

Grphe d'héritage de QuadQ::ConsQuadQ:



Grphe de collaboration de QuadQ::ConsQuadQ:



### Fonctions membres publiques

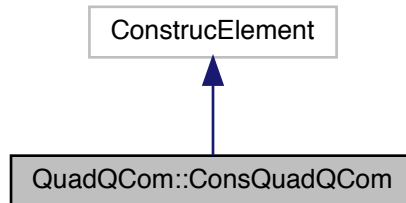
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

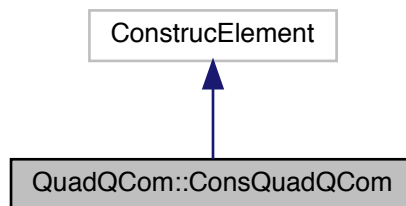
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQ.h`

## 6.139 Référence de la classe QuadQCom::ConsQuadQCom

Grappe d'héritage de QuadQCom::ConsQuadQCom:



Grappe de collaboration de QuadQCom::ConsQuadQCom:



### Fonctions membres publiques

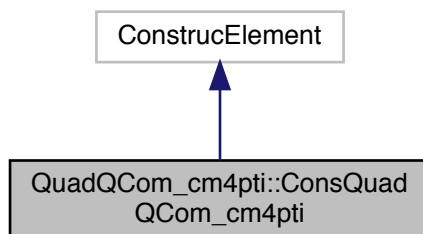
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

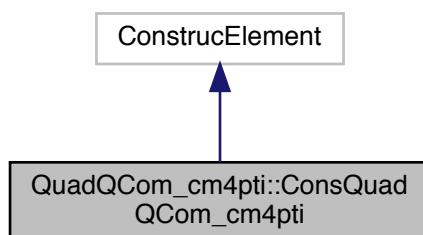
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom.h`

## 6.140 Référence de la classe QuadQCom\_cm4pti::ConsQuadQCom\_cm4pti

Graphe d'héritage de QuadQCom\_cm4pti::ConsQuadQCom\_cm4pti:



Graphe de collaboration de QuadQCom\_cm4pti::ConsQuadQCom\_cm4pti:



### Fonctions membres publiques

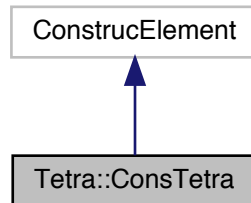
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

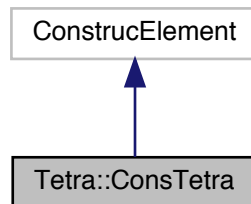
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom\_cm4pti.h

## 6.141 Référence de la classe Tetra::ConsTetra

Grphe d'héritage de Tetra::ConsTetra:



Grphe de collaboration de Tetra::ConsTetra:



### Fonctions membres publiques

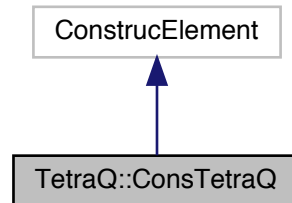
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

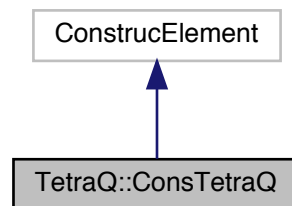
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/Tetra.h`

## 6.142 Référence de la classe TetraQ::ConsTetraQ

Grphe d'héritage de TetraQ::ConsTetraQ:



Grphe de collaboration de TetraQ::ConsTetraQ:



### Fonctions membres publiques

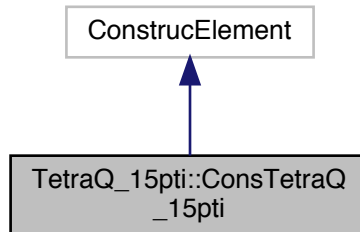
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

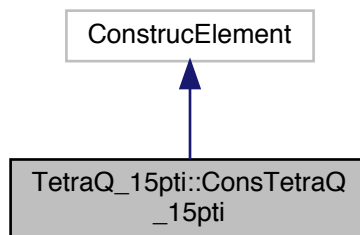
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ.h`

## 6.143 Référence de la classe TetraQ\_15pti::ConsTetraQ\_15pti

Grphe d'héritage de TetraQ\_15pti::ConsTetraQ\_15pti:



Grphe de collaboration de TetraQ\_15pti::ConsTetraQ\_15pti:



### Fonctions membres publiques

- `Element *` **NouvelElement** (`int num_maill, int num`)
- `bool` **Element\_possible** ()

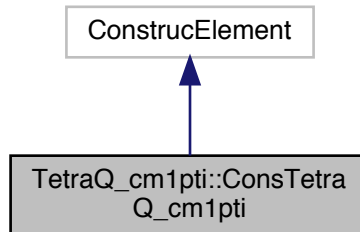
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ_cm15pti.h`

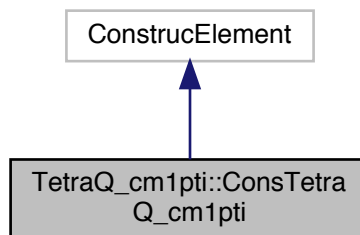


## 6.144 Référence de la classe TetraQ\_cm1pti::ConstetraQ\_cm1pti

Grphe d'héritage de TetraQ\_cm1pti::ConstetraQ\_cm1pti:



Grphe de collaboration de TetraQ\_cm1pti::ConstetraQ\_cm1pti:



### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm1pti.h

## 6.145 Référence de la classe ConstMath

### Attributs publics statiques

- static double **trespetit** = 1.E-16
- static double **pasmalpetit** = 1.E-12
- static double **petit** = 1.E-10
- static double **unpeupetit** = 1.E-7
- static double **unpeugrand** = 1.E7
- static double **grand** = 1.E10
- static double **pasmalgrand** = 1.E12
- static double **tresgrand** = 1.E16
- static double **Pi** = 3.141592653589793238463

— static double **eps\_machine** = ConstMath::d\_epsilon()

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ConstMath.h  
 — /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ConstMath.cc

## 6.146 Référence de la classe ConstPhysico

### Attributs publics statiques

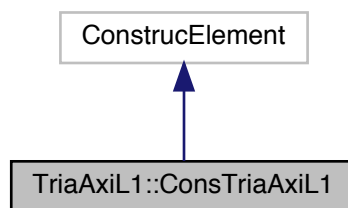
— static double **R** = 8.314472

La documentation de cette classe a été générée à partir du fichier suivant :

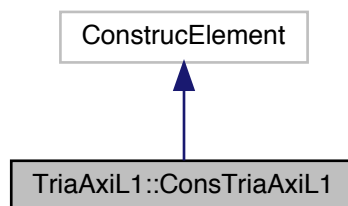
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ConstPhysico.h  
 — /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ConstPhysico.cc

## 6.147 Référence de la classe TriaAxiL1::ConsTriaAxiL1

Graphe d'héritage de TriaAxiL1::ConsTriaAxiL1:



Graphe de collaboration de TriaAxiL1::ConsTriaAxiL1:



### Fonctions membres publiques

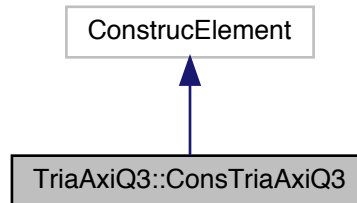
— Element \* **NouvelElement** (int num\_maill, int num)  
 — bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

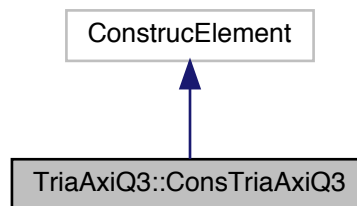
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiL1.h

## 6.148 Référence de la classe TriaAxiQ3::ConsTriaAxiQ3

Grphe d'héritage de TriaAxiQ3::ConsTriaAxiQ3:



Grphe de collaboration de TriaAxiQ3::ConsTriaAxiQ3:



### Fonctions membres publiques

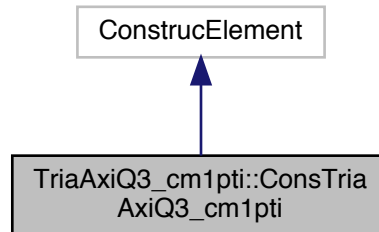
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

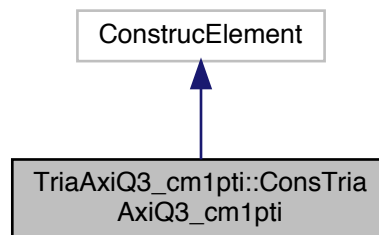
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3.h

## 6.149 Référence de la classe TriaAxiQ3\_cm1pti::ConsTriaAxiQ3\_cm1pti

Grphe d'héritage de TriaAxiQ3\_cm1pti::ConsTriaAxiQ3\_cm1pti:



Grphe de collaboration de TriaAxiQ3\_cm1pti::ConsTriaAxiQ3\_cm1pti:



### Fonctions membres publiques

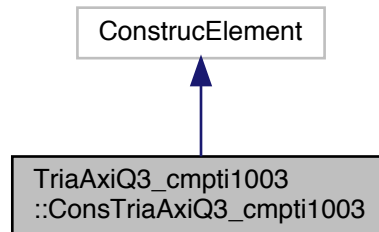
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

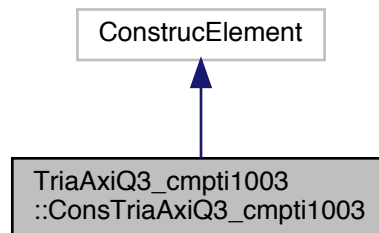
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_cm1pti.h

## 6.150 Référence de la classe TriaAxiQ3\_cmpti1003::ConsTriaAxiQ3\_cmpti1003

Graphe d'héritage de TriaAxiQ3\_cmpti1003::ConsTriaAxiQ3\_cmpti1003:



Graphe de collaboration de TriaAxiQ3\_cmpti1003::ConsTriaAxiQ3\_cmpti1003:



### Fonctions membres publiques

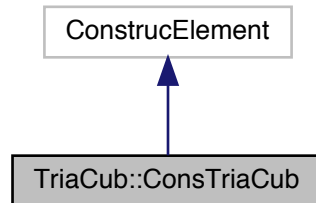
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

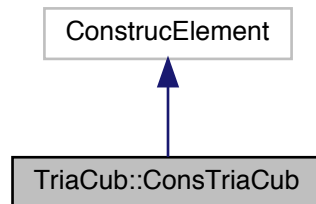
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_↔  
cmpti1003.h

## 6.151 Référence de la classe TriaCub::ConsTriaCub

Grphe d'héritage de TriaCub::ConsTriaCub:



Grphe de collaboration de TriaCub::ConsTriaCub:



### Fonctions membres publiques

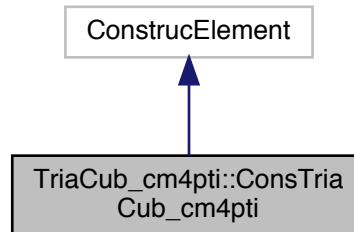
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

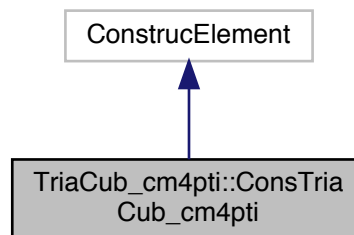
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub.h`

## 6.152 Référence de la classe TriaCub\_cm4pti::ConsTriaCub\_cm4pti

Grphe d'héritage de TriaCub\_cm4pti::ConsTriaCub\_cm4pti:



Grphe de collaboration de TriaCub\_cm4pti::ConsTriaCub\_cm4pti:



### Fonctions membres publiques

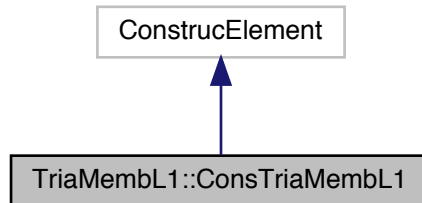
- `Element *` **NouvelElement** (`int num_maill, int num`)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

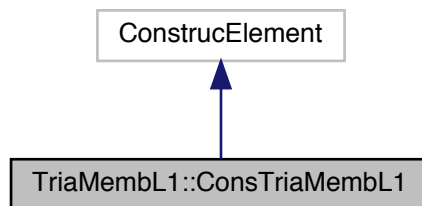
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub_cm4pti.h`

## 6.153 Référence de la classe TriaMembL1::ConsTriaMembL1

Grphe d'héritage de TriaMembL1::ConsTriaMembL1:



Grphe de collaboration de TriaMembL1::ConsTriaMembL1:



### Fonctions membres publiques

- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

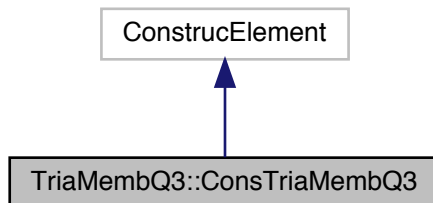
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembL1.h`

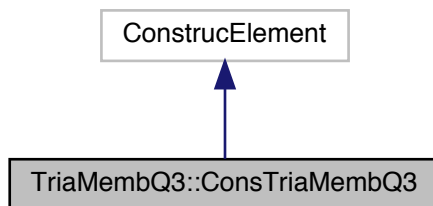


## 6.154 Référence de la classe TriaMembQ3::ConsTriaMembQ3

Grphe d'héritage de TriaMembQ3::ConsTriaMembQ3:



Grphe de collaboration de TriaMembQ3::ConsTriaMembQ3:



### Fonctions membres publiques

- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

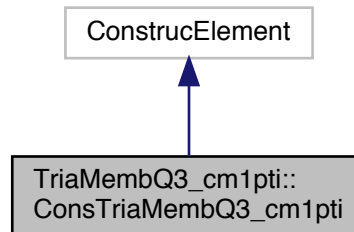
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3.h`

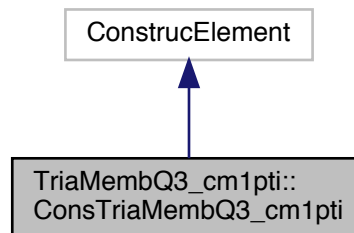
## 6.155 Référence de la classe

### TriaMembQ3\_cm1pti::ConsTriaMembQ3\_cm1pti

Graphe d'héritage de TriaMembQ3\_cm1pti::ConsTriaMembQ3\_cm1pti:



Graphe de collaboration de TriaMembQ3\_cm1pti::ConsTriaMembQ3\_cm1pti:



### Fonctions membres publiques

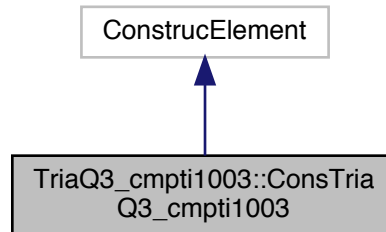
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

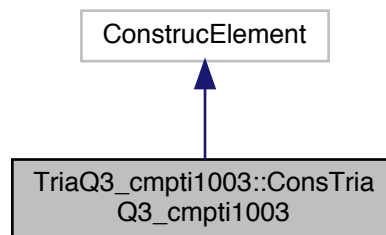
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3\_cm1pti.h

## 6.156 Référence de la classe TriaQ3\_cmpti1003::ConsTriaQ3\_cmpti1003

Grphe d'héritage de TriaQ3\_cmpti1003::ConsTriaQ3\_cmpti1003:



Grphe de collaboration de TriaQ3\_cmpti1003::ConsTriaQ3\_cmpti1003:



### Fonctions membres publiques

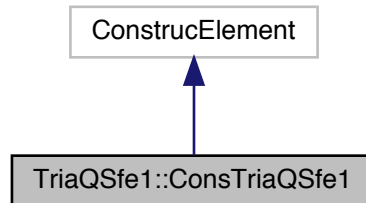
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

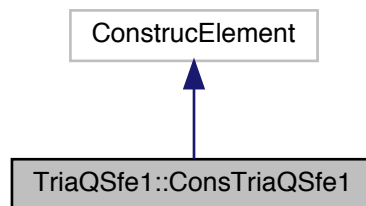
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaQ3\_cmpti1003.h

## 6.157 Référence de la classe TriaQSfe1::ConsTriaQSfe1

Grphe d'héritage de TriaQSfe1::ConsTriaQSfe1:



Grphe de collaboration de TriaQSfe1::ConsTriaQSfe1:



### Fonctions membres publiques

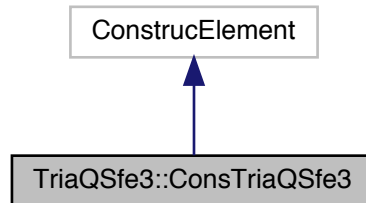
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

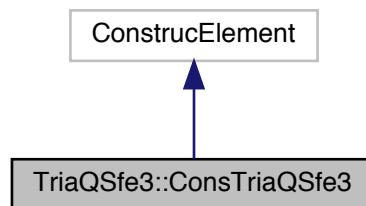
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe1.h`

## 6.158 Référence de la classe TriaQSfe3::ConsTriaQSfe3

Grphe d'héritage de TriaQSfe3::ConsTriaQSfe3:



Grphe de collaboration de TriaQSfe3::ConsTriaQSfe3:



### Fonctions membres publiques

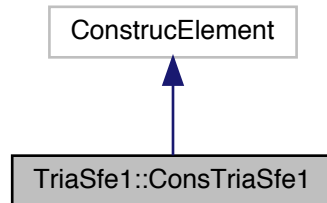
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

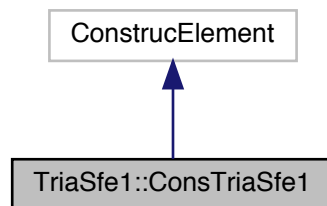
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe3.h`

## 6.159 Référence de la classe TriaSfe1::ConsTriaSfe1

Grphe d'héritage de TriaSfe1::ConsTriaSfe1:



Grphe de collaboration de TriaSfe1::ConsTriaSfe1:



### Fonctions membres publiques

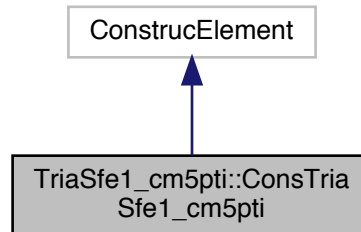
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

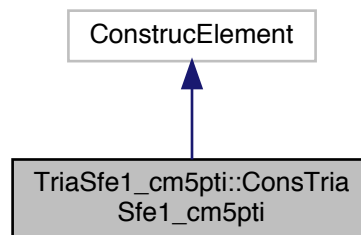
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1.h`

## 6.160 Référence de la classe TriaSfe1\_cm5pti::ConsTriaSfe1\_cm5pti

Grphe d'héritage de TriaSfe1\_cm5pti::ConsTriaSfe1\_cm5pti:



Grphe de collaboration de TriaSfe1\_cm5pti::ConsTriaSfe1\_cm5pti:



### Fonctions membres publiques

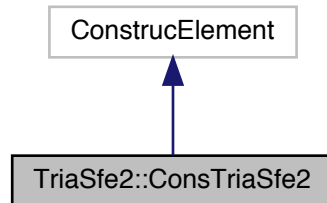
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

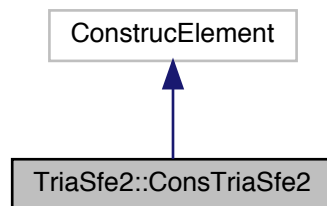
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1\_cm5pti.h

## 6.161 Référence de la classe TriaSfe2::ConsTriaSfe2

Grphe d'héritage de TriaSfe2::ConsTriaSfe2:



Grphe de collaboration de TriaSfe2::ConsTriaSfe2:



### Fonctions membres publiques

- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

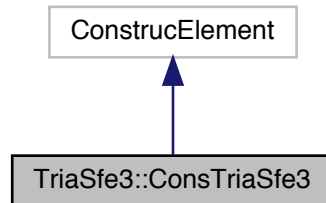
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe2.h`

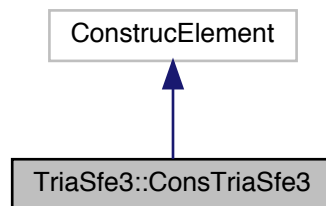


## 6.162 Référence de la classe TriaSfe3::ConsTriaSfe3

Grphe d'héritage de TriaSfe3::ConsTriaSfe3:



Grphe de collaboration de TriaSfe3::ConsTriaSfe3:



### Fonctions membres publiques

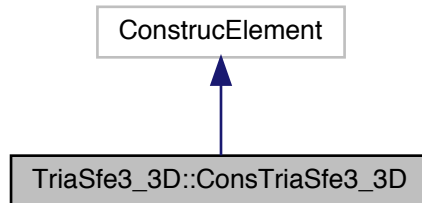
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

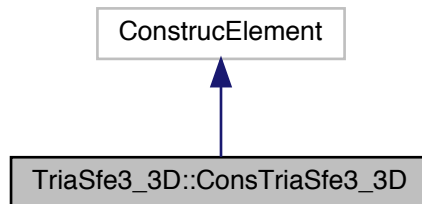
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3.h`

## 6.163 Référence de la classe TriaSfe3\_3D::ConsTriaSfe3\_3D

Grphe d'héritage de TriaSfe3\_3D::ConsTriaSfe3\_3D:



Grphe de collaboration de TriaSfe3\_3D::ConsTriaSfe3\_3D:



### Fonctions membres publiques

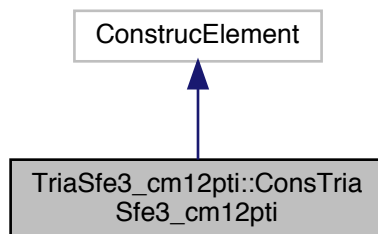
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

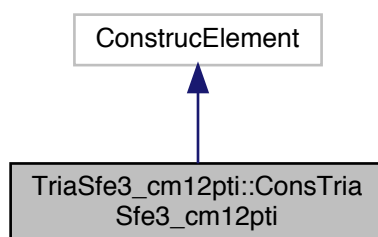
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_3D.h`

## 6.164 Référence de la classe TriaSfe3\_cm12pti::ConsTriaSfe3\_cm12pti

Graphe d'héritage de TriaSfe3\_cm12pti::ConsTriaSfe3\_cm12pti:



Graphe de collaboration de TriaSfe3\_cm12pti::ConsTriaSfe3\_cm12pti:



### Fonctions membres publiques

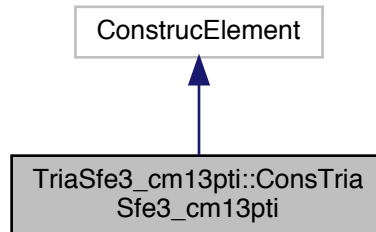
- `Element *` **NouvelElement** (`int num_maill, int num`)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

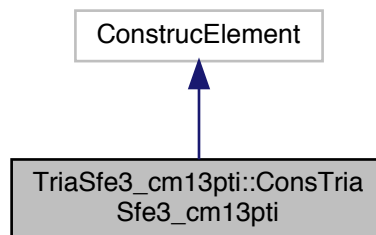
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm12pti.h`

## 6.165 Référence de la classe TriaSfe3\_cm13pti::ConsTriaSfe3\_cm13pti

Grphe d'héritage de TriaSfe3\_cm13pti::ConsTriaSfe3\_cm13pti:



Grphe de collaboration de TriaSfe3\_cm13pti::ConsTriaSfe3\_cm13pti:



### Fonctions membres publiques

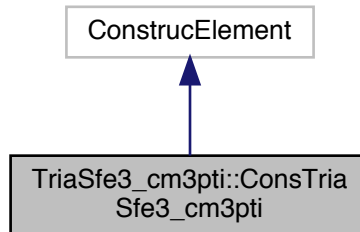
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

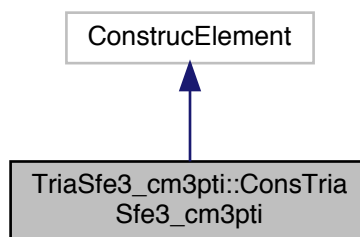
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm13pti.h

## 6.166 Référence de la classe TriaSfe3\_cm3pti::ConsTriaSfe3\_cm3pti

Grappe d'héritage de TriaSfe3\_cm3pti::ConsTriaSfe3\_cm3pti:



Grappe de collaboration de TriaSfe3\_cm3pti::ConsTriaSfe3\_cm3pti:



### Fonctions membres publiques

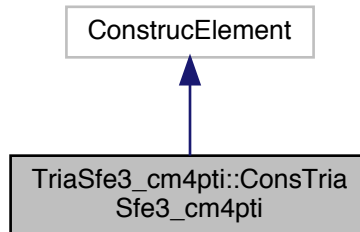
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

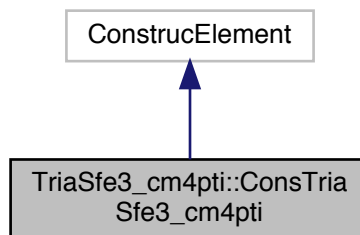
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm3pti.h

## 6.167 Référence de la classe TriaSfe3\_cm4pti::ConsTriaSfe3\_cm4pti

Grphe d'héritage de TriaSfe3\_cm4pti::ConsTriaSfe3\_cm4pti:



Grphe de collaboration de TriaSfe3\_cm4pti::ConsTriaSfe3\_cm4pti:



### Fonctions membres publiques

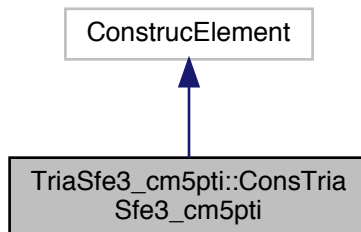
- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

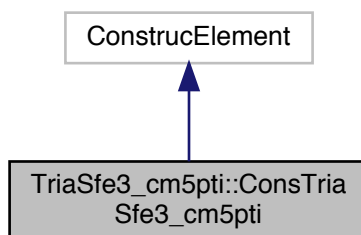
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm4pti.h

## 6.168 Référence de la classe TriaSfe3\_cm5pti::ConsTriaSfe3\_cm5pti

Grphe d'héritage de TriaSfe3\_cm5pti::ConsTriaSfe3\_cm5pti:



Grphe de collaboration de TriaSfe3\_cm5pti::ConsTriaSfe3\_cm5pti:



### Fonctions membres publiques

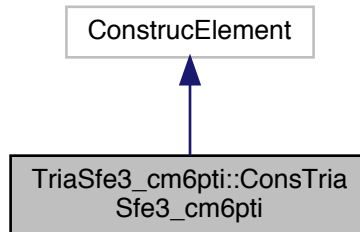
- `Element *` **NouvelElement** (`int num_maill, int num`)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

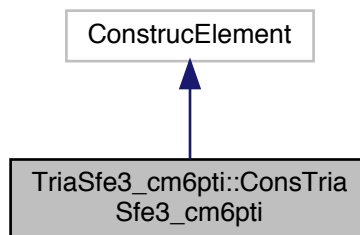
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm5pti.h`

## 6.169 Référence de la classe TriaSfe3\_cm6pti::ConsTriaSfe3\_cm6pti

Grphe d'héritage de TriaSfe3\_cm6pti::ConsTriaSfe3\_cm6pti:



Grphe de collaboration de TriaSfe3\_cm6pti::ConsTriaSfe3\_cm6pti:



### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

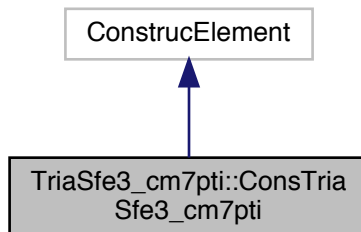
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm6pti.h

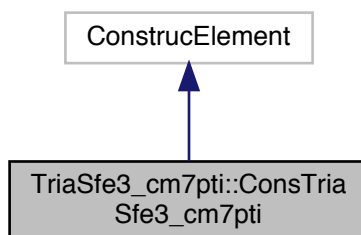


## 6.170 Référence de la classe TriaSfe3\_cm7pti::ConsTriaSfe3\_cm7pti

Grphe d'héritage de TriaSfe3\_cm7pti::ConsTriaSfe3\_cm7pti:



Grphe de collaboration de TriaSfe3\_cm7pti::ConsTriaSfe3\_cm7pti:



### Fonctions membres publiques

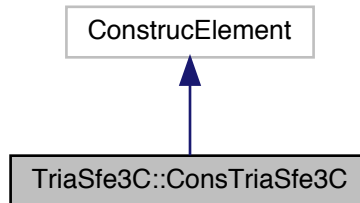
- `Element *` **NouvelElement** (`int num_maill, int num`)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

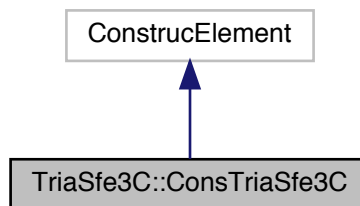
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm7pti.h`

## 6.171 Référence de la classe TriaSfe3C::ConsTriaSfe3C

Graphe d'héritage de TriaSfe3C::ConsTriaSfe3C:



Graphe de collaboration de TriaSfe3C::ConsTriaSfe3C:



### Fonctions membres publiques

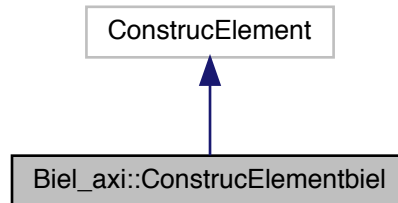
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

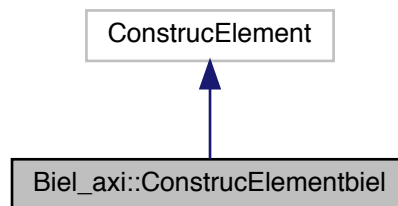
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3C.h`

## 6.172 Référence de la classe Biel\_axi::ConstrucElementbiel

Grphe d'héritage de Biel\_axi::ConstrucElementbiel:



Grphe de collaboration de Biel\_axi::ConstrucElementbiel:



### Fonctions membres publiques

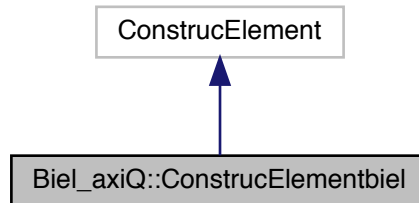
- Element \* **NouvelElement** (int nb\_mail, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

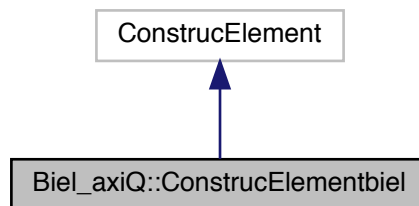
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axi.h

## 6.173 Référence de la classe Biel\_axiQ::ConstrucElementbiel

Grphe d'héritage de Biel\_axiQ::ConstrucElementbiel:



Grphe de collaboration de Biel\_axiQ::ConstrucElementbiel:



### Fonctions membres publiques

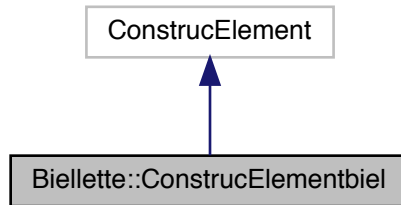
- Element \* **NouvelElement** (int nb\_mail, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

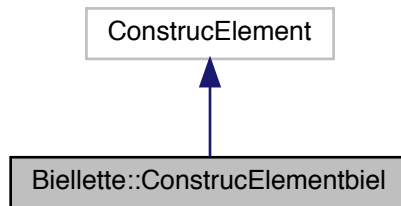
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axiQ.h

## 6.174 Référence de la classe Bielle::ConstrucElementbiel

Grphe d'héritage de Bielle::ConstrucElementbiel:



Grphe de collaboration de Bielle::ConstrucElementbiel:



### Fonctions membres publiques

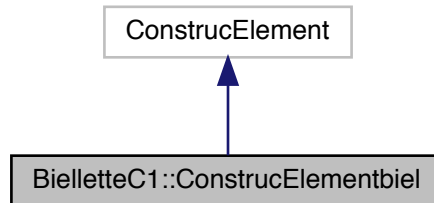
- `Element *` **NouvelElement** (int nb\_mail, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

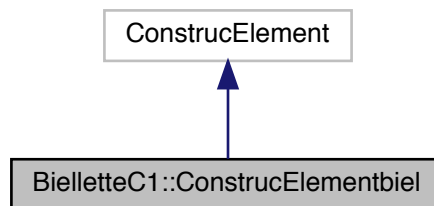
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Bielle/Bielle.h`

## 6.175 Référence de la classe BielleC1::ConstrucElementbiel

Grphe d'héritage de BielleC1::ConstrucElementbiel:



Grphe de collaboration de BielleC1::ConstrucElementbiel:



### Fonctions membres publiques

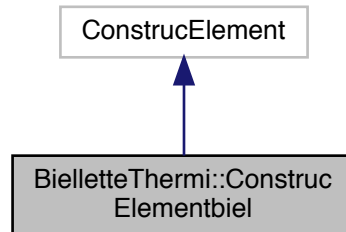
- `Element *` **NouvelElement** (int num\_mail, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

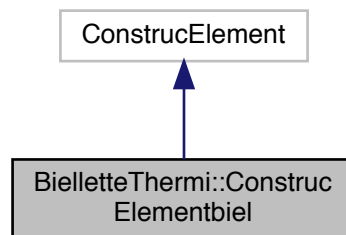
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/BielleC1/BielleC1.h`

## 6.176 Référence de la classe BielleThermi::ConstrucElementbiel

Grappe d'héritage de BielleThermi::ConstrucElementbiel:



Grappe de collaboration de BielleThermi::ConstrucElementbiel:



### Fonctions membres publiques

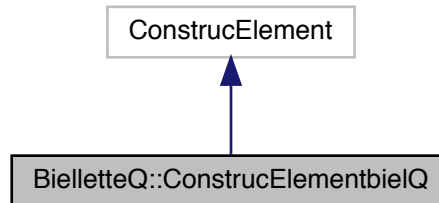
- `Element *` **NouvelElement** (int nb\_mail, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

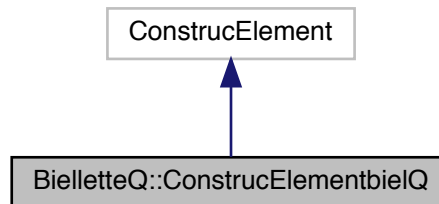
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/Bielle/BielleThermi.h`

## 6.177 Référence de la classe BielleQ::ConstrucElementbielQ

Grphe d'héritage de BielleQ::ConstrucElementbielQ:



Grphe de collaboration de BielleQ::ConstrucElementbielQ:



### Fonctions membres publiques

- Element \* **NouvelElement** (int nb\_mail, int num)
- bool **Element\_possible** ()

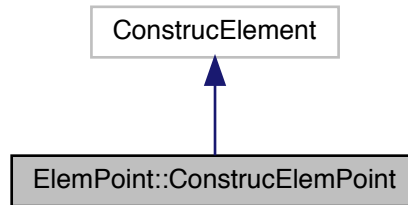
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Bielle/BielleQ.h

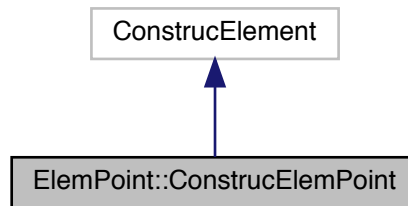


## 6.178 Référence de la classe ElemPoint::ConstrucElemPoint

Grphe d'héritage de ElemPoint::ConstrucElemPoint:



Grphe de collaboration de ElemPoint::ConstrucElemPoint:



### Fonctions membres publiques

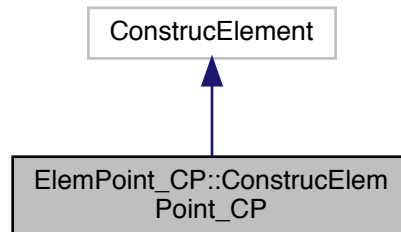
- `Element *` **NouvelElement** (int num\_maill, int num)
- `bool` **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

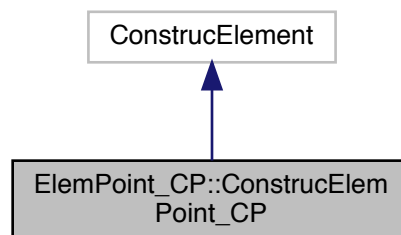
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.h`

## 6.179 Référence de la classe ElemPoint\_CP::ConstrucElemPoint\_CP

Graphe d'héritage de ElemPoint\_CP::ConstrucElemPoint\_CP:



Graphe de collaboration de ElemPoint\_CP::ConstrucElemPoint\_CP:



### Fonctions membres publiques

- Element \* **NouvelElement** (int num\_maill, int num)
- bool **Element\_possible** ()

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint\_CP.h

## 6.180 Référence de la classe Isovaleurs\_Gmsh::ConstructTabScalVecTensGmsh

### Attributs publics

- Tableau< string > **scalar\_pourView**
- Tableau< string > **vector\_pourView**
- Tableau< string > **tensor\_pourView**
- Tableau< string > **text\_pourView**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Isovaleurs\_Gmsh.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Isovaleurs\_Gmsh.cc

## 6.181 Référence de la classe `GeomSeg::Construire_Gauss_Lobatto`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/Ligne/GeomSeg.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/Ligne/GeomSeg.cc`

## 6.182 Référence de la classe `ElemGeomC0::ConteneurExtrapolation`

### Fonctions membres publiques

- `ConteneurExtrapolation` (const `ConteneurExtrapolation` &a)
- `ConteneurExtrapolation` & **operator=** (const `ConteneurExtrapolation` &a)

### Attributs publics

- `Tableau`< `Tableau`< double > > `tab`
- `Tableau`< `Tableau`< int > > `indir`

### Amis

- `istream` & **operator**>> (`istream` &, `ConteneurExtrapolation` &)
- `ostream` & **operator**<< (`ostream` &, const `ConteneurExtrapolation` &)

La documentation de cette classe a été générée à partir du fichier suivant :

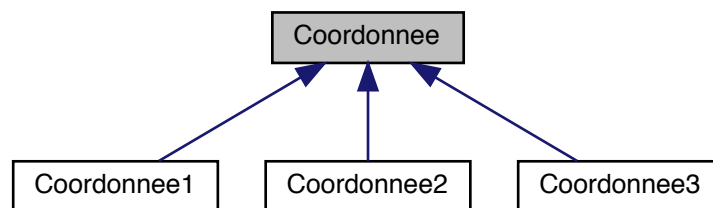
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/ElemGeomC0.h`

## 6.183 Référence de la classe `Coordonnee`

Coordonnees simples sans variances.

```
#include <Coordonnee.h>
```

Graphe d'héritage de `Coordonnee`:



### Fonctions membres publiques

- `Coordonnee` ()  
*Constructeur par défaut.*
- `Coordonnee` (int dimension)  
*Constructeur fonction de la dimension du probleme les coordonnees sont initialise a zero.*
- `Coordonnee` (double x)  
*Constructeur pour une localisation unidimensionnelle.*
- `Coordonnee` (double x, double y)  
*Constructeur pour une localisation bidimensionnelle.*
- `Coordonnee` (double x, double y, double z)

- **Coordonnee** (int dimension, double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees et d'une dimension ( l'existence de la place memoire est a la charge de l'utilisateur et ne sera pas détruite par le destructeur.)*
- **Coordonnee** (const [Coordonnee](#) &c)  
*Constructeur de copie.*
- virtual ~**Coordonnee** ()  
*DESTRUCTEUR :*
- void **Change\_place** (int dimension, double \*t)  
*fonction équivalente au constructeur: changement pour une place externe via un pointeur ( l'existence de la place memoire est a la charge de l'utilisateur et ne sera pas détruite par le destructeur.)*
- void **Change\_Coordonnee** (int dimension, double x)  
*changement rapide des coordonnées: dimension 1*
- void **Change\_Coordonnee** (int dimension, double x, double y)  
*changement rapide des coordonnées: dimension 2*
- void **Change\_Coordonnee** (int dimension, double x, double y, double z)  
*changement rapide des coordonnées: dimension 3*
- virtual int **Dimension** () const  
*Renvoie le nombre de coordonnees.*
- virtual void **Libere** ()  
*Desallocation de la place memoire allouee.*
- virtual double & **operator()** (int i)  
*Renvoie la ieme coordonnee.*
- virtual double **operator()** (int i) const  
*Renvoie une copie de la ieme coordonnee.*
- **Coordonnee & operator=** (const [Coordonnee](#) &c)  
*Surcharge de l'operateur = : realise l'affectation entre deux points.*
- **Coordonnee operator-** () const  
*Surcharge de l'operateur - : renvoie l'oppose d'un point.*
- **Coordonnee operator-** (const [Coordonnee](#) &c) const  
*Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points.*
- **Coordonnee operator+** (const [Coordonnee](#) &c) const  
*Surcharge de l'operateur + : realise l'addition des coordonnees de deux points.*
- void **operator+=** (const [Coordonnee](#) &c)  
*Surcharge de l'operateur +=.*
- void **operator-=** (const [Coordonnee](#) &c)  
*Surcharge de l'operateur -=.*
- void **operator\*=  
Coordonnee operator\*** (double val) const  
*Surcharge de l'operateur \*=.*
- **Coordonnee operator\*** (double val) const  
*Surcharge de l'operateur \* : multiplication de coordonnees par un scalaire.*
- double **operator\*** (const [Coordonnee](#) &c) const  
*Surcharge de l'operateur \* : produit scalaire entre coordonnees.*
- **Coordonnee operator/** (double val) const  
*Surcharge de l'operateur / : division de coordonnees par un scalaire.*
- void **operator/=** (double val)  
*Surcharge de l'operateur /= : division de coordonnees par un scalaire.*
- int **operator==** (const [Coordonnee](#) &c) const  
*Surcharge de l'operateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- int **operator!=** (const [Coordonnee](#) &c) const  
*Surcharge de l'operateur != Renvoie 1 si les deux positions ne sont pas identiques Renvoie 0 sinon.*
- virtual void **Affiche** () const  
*Affiche les coordonnees du point à l'écran entre accolades.*
- virtual void **Affiche** (ostream &sort) const  
*Affiche les coordonnees du point dans sort entre accolades.*
- virtual void **Affiche** (ostream &sort, int nb) const  
*Affiche les coordonnees du point dans sort sur nb digit plus un blanc et rien d'autre (pas d'accolade)*
- virtual void **Affiche\_1** (ostream &sort) const  
*Affiche les coordonnees du point dans sort et rien d'autre (pas d'accolade)*
- virtual void **Lecture** ([UtilLecture](#) &entreePrinc)  
*lecture brut des coordonnées sans la dimension*

- virtual void **Lecture** ()  
*lecture brut des coordonnées sans la dimension dans le flux par défaut*
- virtual void **Change\_dim** (int dim)  
*changement de la dimension dans le cas d'une nouvelle dimension inférieur on supprime les dernières coord dans le cas d'une dimension supérieur, on ajoute des coord initialisées a zero*
- virtual **Vecteur Vect** () const  
*création d'un Vecteur équivalent*
- virtual void **Zero** ()  
*mise a zero des coordonnées*
- virtual double **Norme** () const  
*Calcul de la norme euclidienne des composantes du point.*
- **Coordonnee & Normer** ()  
*norme le vecteur coordonnée*
- double **Max\_val\_abs** () const  
*Calcul du maximum en valeur absolu des composantes du vecteur.*
- int **Indice\_max\_val\_abs** () const  
*ramène l'indice du maxi en valeur absolu*
- double **Max\_val\_abs** (int &i) const  
*Calcul du maximum en valeur absolu des composantes du vecteur ramene egalement l'indice de tableau du maximum.*
- double **Max\_val\_abs\_signe** () const  
*Calcul du maximum en valeur absolu des composantes du vecteur mais ramène la grandeur signée (avec son signe)*
- double **Max\_val\_abs\_signe** (int &i) const  
*Calcul du maximum en valeur absolu des composantes du vecteur mais ramène la grandeur signée (avec son signe) ramene egalement l'indice de tableau du maximum.*
- void **Modif\_en\_max** (const **Coordonnee** &v)  
*modifie éventuellement les coordonnées de this pour quelles soient supérieures ou égales aux coordonnées en paramètre*
- void **Modif\_en\_min** (const **Coordonnee** &v)  
*modifie éventuellement les coordonnées de this pour quelles soient inférieures ou égales aux coordonnées en paramètre*
- void **Ajout\_meme\_valeur** (double val)  
*ajoute une même valeur à tous les coordonnées*
- double **Carre** () const  
*calcul la norme euclidienne au carré*

### Fonctions membres publiques statiques

- static void **SchemaXML\_Coordonnee** (ofstream &sort, const **Enum\_IO\_XML** enu)  
*sortie du schemaXML: en fonction de enu*

### Fonctions membres protégées

- **Coordonnee** (bool)  
*Constructeur inline qui ne fait rien utilisé par les classes dérivées.*
- void **Meme\_place** (**CoordonneeB** &vB)  
*définit des coordonnées sans variance à la même place que des coordonnées avec variances*
- void **Meme\_place** (**CoordonneeH** &vH)  
*définit des coordonnées sans variance à la même place que des coordonnées avec variances*

### Attributs protégés

- short int **dim**  
*dimension*
- bool **memoire**  
*indique s'il y a allocation ou pas*
- double \* **coord**  
*stockage*

## Amis

- class **BaseH**
- class **BaseB**
- `istream & operator>>` (`istream &`, [Coordonnee &](#))  
*surcharge de l'operator de lecture avec le type*
- `ostream & operator<<` (`ostream &`, `const Coordonnee &`)  
*surcharge de l'operator d'écriture*
- [Coordonnee](#) `operator*` (`double val`, `const Coordonnee &c`)  
*Surcharge de l'opérateur \* : multiplication entre un scalaire et des coordonnées.*

### 6.183.1 Description détaillée

Coordonnées simples sans variances.

### 6.183.2 Documentation des fonctions membres

#### 6.183.2.1 Change\_dim()

```
virtual void Coordonnee::Change_dim (
    int dim ) [virtual]
```

changement de la dimension dans le cas d'une nouvelle dimension inferieur on supprime les dernieres coord dans le cas d'une dimension superieur, on ajoute des coord initialisees a zero

Réimplémentée dans [Coordonnee1](#), [Coordonnee2](#), et [Coordonnee3](#).

#### 6.183.2.2 Dimension()

```
virtual int Coordonnee::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnées.

Réimplémentée dans [Coordonnee1](#), [Coordonnee2](#), et [Coordonnee3](#).

#### 6.183.2.3 Libere()

```
virtual void Coordonnee::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee.

Réimplémentée dans [Coordonnee1](#), [Coordonnee2](#), et [Coordonnee3](#).

#### 6.183.2.4 Norme()

```
virtual double Coordonnee::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée dans [Coordonnee1](#), [Coordonnee2](#), et [Coordonnee3](#).

#### 6.183.2.5 Vect()

```
Vecteur Coordonnee::Vect ( ) const [virtual]
```

création d'un [Vecteur](#) équivalent

Réimplémentée dans [Coordonnee1](#), [Coordonnee2](#), et [Coordonnee3](#).

### 6.183.2.6 Zero()

```
virtual void Coordonnee::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée dans [Coordonnee1](#), [Coordonnee2](#), et [Coordonnee3](#).

La documentation de cette classe a été générée à partir du fichier suivant :

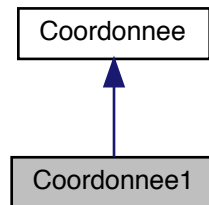
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee\_2.cc

## 6.184 Référence de la classe Coordonnee1

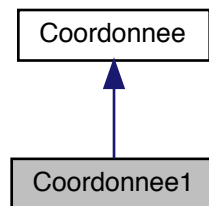
cas des coordonnées simples sans variance

```
#include <Coordonnee1.h>
```

Graphe d'héritage de Coordonnee1:



Graphe de collaboration de Coordonnee1:



### Fonctions membres publiques

- **Coordonnee1** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee1** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee1** (double x)  
*Constructeur pour une localisation monodimensionnelle.*
- **Coordonnee1** (double \*t)

- constructeur fonction d'une adresse memoire ou sont stockee les coordonnees ( l'existence de la place memoire est a la charge de l'utilisateur!!).*
- **Coordonnee1** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 1.*
  - **Coordonnee1** (const [Coordonnee1](#) &c)  
*Constructeur de copie.*
  - **Coordonnee1** (const [Coordonnee](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*
  - virtual ~**Coordonnee1** ()  
*DESTRUCTEUR :*
  - int **Dimension** () const  
*Renvoie le nombre de coordonnees.*
  - void **Libere** ()  
*Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.*
  - void **Change\_dim** (int dim)  
*changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
  - [Coordonnee1](#) & **operator=** (const [Coordonnee1](#) &c)  
*Surcharge de l'operateur = : realise l'affectation entre deux points.*
  - [Coordonnee1](#) **operator-** () const  
*Surcharge de l'operateur - : renvoie l'oppose d'un point.*
  - [Coordonnee1](#) **operator-** (const [Coordonnee1](#) &c) const  
*Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points.*
  - [Coordonnee1](#) **operator+** (const [Coordonnee1](#) &c) const  
*Surcharge de l'operateur + : realise l'addition des coordonnees de deux points.*
  - void **operator+=** (const [Coordonnee1](#) &c)  
*Surcharge de l'operateur +=.*
  - void **operator-=** (const [Coordonnee1](#) &c)  
*Surcharge de l'operateur -=.*
  - void **operator\*=  
Coordonnee1 operator\*** (double val) const  
*Surcharge de l'operateur \*=.*
  - double **operator\*** (const [Coordonnee1](#) &c) const  
*Surcharge de l'operateur \* : multiplication de coordonnees par un scalaire.*
  - [Coordonnee1](#) **operator\*** (double val) const  
*Surcharge de l'operateur \* : produit scalaire entre coordonnees.*
  - void **operator/=  
Coordonnee1 operator/** (double val) const  
*Surcharge de l'operateur / : division de coordonnees par un scalaire.*
  - int **operator==** (const [Coordonnee1](#) &c) const  
*Surcharge de l'operateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
  - [Vecteur](#) **Vect** () const  
*conversion en Vecteur*
  - void **Zero** ()  
*mise a zero des coordonnees*
  - double **Norme** () const  
*Calcul de la norme euclidienne des composantes du point.*
  - [Coordonnee1](#) & **Normer** ()  
*norme le vecteur coordonnée*
  - double **Somme** () const  
*somme de tous les composantes*

## Fonctions membres publiques statiques

- static void **SchemaXML\_Coordonnee** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **coord1** [1]



## Amis

- `Coordonnee1 operator*` (double val, const `Coordonnee1` &c)  
*Surcharge de l'opérateur \* : multiplication entre un scalaire et des coordonnées.*

## Membres hérités additionnels

### 6.184.1 Description détaillée

cas des coordonnées simples sans variance

### 6.184.2 Documentation des fonctions membres

#### 6.184.2.1 `Change_dim()`

```
void Coordonnee1::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [Coordonnee](#).

#### 6.184.2.2 `Dimension()`

```
int Coordonnee1::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnées.

Réimplémentée à partir de [Coordonnee](#).

#### 6.184.2.3 `Libere()`

```
void Coordonnee1::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.

Réimplémentée à partir de [Coordonnee](#).

#### 6.184.2.4 `Norme()`

```
double Coordonnee1::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [Coordonnee](#).

#### 6.184.2.5 `Vect()`

```
Vecteur Coordonnee1::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [Coordonnee](#).

#### 6.184.2.6 `Zero()`

```
void Coordonnee1::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [Coordonnee](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee1.h`

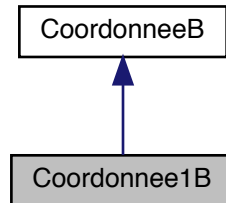
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee1\_2.cc

## 6.185 Référence de la classe Coordonnee1B

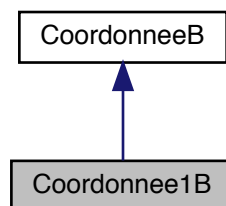
cas des coordonnées covariantes

```
#include <Coordonnee1.h>
```

Graphe d'héritage de Coordonnee1B:



Graphe de collaboration de Coordonnee1B:



### Fonctions membres publiques

- **Coordonnee1B** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee1B** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee1B** (double x)  
*Constructeur pour une localisation monodimensionnelle.*
- **Coordonnee1B** (double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees ( l'existence de la place mémoire est a la charge de l'utilisateur!!).*
- **Coordonnee1B** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 1.*
- **Coordonnee1B** (const [Coordonnee1B](#) &c)  
*Constructeur de copie.*
- **Coordonnee1B** (const [CoordonneeB](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*

- virtual `~Coordonnee1B ()`  
*DESTRUCTEUR :*
- int `Dimension () const`  
*Renvoie le nombre de coordonnees.*
- void `Libere ()`  
*Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.*
- void `Change_dim (int dim)`  
*changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
- `Coordonnee1B & operator= (const Coordonnee1B &c)`  
*Surcharge de l'opérateur = : realise l'affectation entre deux points.*
- `Coordonnee1B operator- () const`  
*Surcharge de l'opérateur - : renvoie l'opposé d'un point.*
- `Coordonnee1B operator- (const Coordonnee1B &c) const`  
*Surcharge de l'opérateur - : realise la soustraction des coordonnees de deux points.*
- `Coordonnee1B operator+ (const Coordonnee1B &c) const`  
*Surcharge de l'opérateur + : realise l'addition des coordonnees de deux points.*
- void `operator+= (const Coordonnee1B &c)`  
*Surcharge de l'opérateur +=.*
- void `operator-= (const Coordonnee1B &c)`  
*Surcharge de l'opérateur -=.*
- void `operator*-= (double val)`  
*Surcharge de l'opérateur \*-=.*
- `Coordonnee1B operator* (double val) const`  
*Surcharge de l'opérateur \* : multiplication de coordonnees par un scalaire.*
- double `operator* (const Coordonnee1H &c) const`  
*Surcharge de l'opérateur \* : produit scalaire entre coordonnees.*
- double `ScalBB (const Coordonnee1B &c) const`  
*produit scalaire entre coordonnees covariantes et covariantes*
- `Coordonnee1B operator/ (double val) const`  
*Surcharge de l'opérateur / : division de coordonnees par un scalaire.*
- void `operator/= (double val)`  
*Surcharge de l'opérateur /= : division de coordonnees par un scalaire.*
- int `operator==(const Coordonnee1B &c) const`  
*Surcharge de l'opérateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- `Vecteur Vect () const`  
*conversion en Vecteur*
- void `Zero ()`  
*mise a zero des coordonnées*
- double `Norme () const`  
*Calcul de la norme euclidienne des composantes du point.*
- `Coordonnee1B & Normer ()`  
*norme le vecteur coordonnée*
- double `Somme () const`  
*somme de tous les composantes*

## Fonctions membres publiques statiques

- static void `SchemaXML_Coordonnee (ofstream &sort, const Enum_IO_XML enu)`  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double `coord1 [1]`

## Amis

- class `Coordonnee1H`
- `Coordonnee1B operator* (double val, const Coordonnee1B &c)`  
*Surcharge de l'opérateur \* : multiplication entre un scalaire et des coordonnees.*

## Membres hérités additionnels

### 6.185.1 Description détaillée

cas des coordonnées covariantes

### 6.185.2 Documentation des fonctions membres

#### 6.185.2.1 Change\_dim()

```
void Coordonnee1B::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [CoordonneeB](#).

#### 6.185.2.2 Dimension()

```
int Coordonnee1B::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnées.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.185.2.3 Libere()

```
void Coordonnee1B::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.185.2.4 Norme()

```
double Coordonnee1B::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.185.2.5 Vect()

```
Vecteur Coordonnee1B::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [CoordonneeB](#).

#### 6.185.2.6 Zero()

```
void Coordonnee1B::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [CoordonneeB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

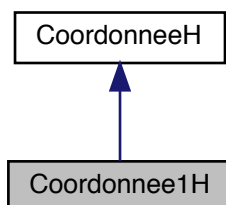
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee1B\_2.cc

## 6.186 Référence de la classe Coordonnee1H

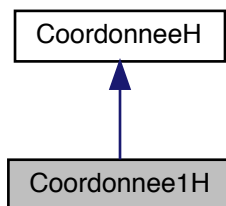
cas des coordonnées contravariantes

```
#include <Coordonnee1.h>
```

Graphe d'héritage de Coordonnee1H:



Graphe de collaboration de Coordonnee1H:



### Fonctions membres publiques

- **Coordonnee1H** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee1H** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee1H** (double x)  
*Constructeur pour une localisation monodimensionnelle.*
- **Coordonnee1H** (double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees ( l'existence de la place mémoire est a la charge de l'utilisateur !!).*
- **Coordonnee1H** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 1.*
- **Coordonnee1H** (const [Coordonnee1H](#) &c)  
*Constructeur de copie.*
- **Coordonnee1H** (const [CoordonneeH](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*
- virtual ~**Coordonnee1H** ()  
*DESTRUCTEUR :*
- int [Dimension](#) () const

- *Renvoie le nombre de coordonnees.*
- void **Libere** ()
  - Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.*
- void **Change\_dim** (int dim)
  - changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
- **Coordonnee1H & operator=** (const **Coordonnee1H** &c)
  - Surcharge de l'operateur = : realise l'affectation entre deux points.*
- **Coordonnee1H operator-** () const
  - Surcharge de l'operateur - : renvoie l'oppose d'un point.*
- **Coordonnee1H operator-** (const **Coordonnee1H** &c) const
  - Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points.*
- **Coordonnee1H operator+** (const **Coordonnee1H** &c) const
  - Surcharge de l'operateur + : realise l'addition des coordonnees de deux points.*
- void **operator+=** (const **Coordonnee1H** &c)
  - Surcharge de l'operateur +=.*
- void **operator-=** (const **Coordonnee1H** &c)
  - Surcharge de l'operateur -=.*
- void **operator\*= **(double val)****
  - Surcharge de l'operateur \*=.*
- **Coordonnee1H operator\*** (double val) const
  - Surcharge de l'operateur \* : multiplication de coordonnees par un scalaire.*
- double **operator\*** (const **Coordonnee1B** &c) const
  - Surcharge de l'operateur \* : produit scalaire entre coordonnees.*
- double **ScalHH** (const **Coordonnee1H** &c) const
  - produit scalaire entre coordonnees contravariantes et contravariantes*
- **Coordonnee1H operator/** (double val) const
  - Surcharge de l'operateur / : division de coordonnees par un scalaire.*
- void **operator/=** (double val)
  - Surcharge de l'operateur /= : division de coordonnees par un scalaire.*
- int **operator==** (const **Coordonnee1H** &c) const
  - Surcharge de l'operateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- **Vecteur Vect** () const
  - conversion en Vecteur*
- void **Zero** ()
  - mise a zero des coordonnees*
- double **Norme** () const
  - Calcul de la norme euclidienne des composantes du point.*
- **Coordonnee1H & Normer** ()
  - norme le vecteur coordonnée*
- double **Somme** () const
  - somme de tous les composantes*

## Fonctions membres publiques statiques

- static void **SchemaXML\_Coordonnee** (ofstream &sort, const **Enum\_IO\_XML** enu)
  - sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **coord1** [1]

## Amis

- class **Coordonnee1B**
- **Coordonnee1H operator\*** (double val, const **Coordonnee1H** &c)
  - Surcharge de l'operateur \* : multiplication entre un scalaire et des coordonnees.*

## Membres hérités additionnels

### 6.186.1 Description détaillée

cas des coordonnées contravariantes

### 6.186.2 Documentation des fonctions membres

#### 6.186.2.1 `Change_dim()`

```
void Coordonnee1H::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [CoordonneeH](#).

#### 6.186.2.2 `Dimension()`

```
int Coordonnee1H::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnées.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.186.2.3 `Libere()`

```
void Coordonnee1H::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.186.2.4 `Norme()`

```
double Coordonnee1H::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.186.2.5 `Vect()`

```
Vecteur Coordonnee1H::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [CoordonneeH](#).

#### 6.186.2.6 `Zero()`

```
void Coordonnee1H::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [CoordonneeH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

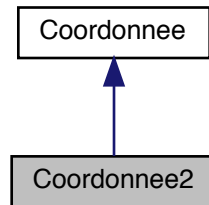
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee1.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee1H_2.cc`

## 6.187 Référence de la classe Coordonnee2

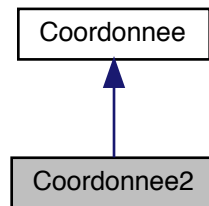
cas des coordonnées simples sans variance

```
#include <Coordonnee2.h>
```

Grappe d'héritage de Coordonnee2:



Grappe de collaboration de Coordonnee2:



### Fonctions membres publiques

- **Coordonnee2** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee2** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee2** (double x, double y)  
*Constructeur pour une localisation bidimensionnelle.*
- **Coordonnee2** (double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnées ( l'existence de la place mémoire est a la charge de l'utilisateur !!).*
- **Coordonnee2** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 2.*
- **Coordonnee2** (const [Coordonnee2](#) &c)  
*Constructeur de copie.*
- **Coordonnee2** (const [Coordonnee](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*
- virtual ~**Coordonnee2** ()  
*DESTRUCTEUR :*
- int [Dimension](#) () const



- *Renvoie le nombre de coordonnees.*
- void `Libere` ()
- *Desallocation de la place memoire allouee.*
- void `Change_dim` (int `dim`)
- *changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
- `Coordonnee2 & operator=` (const `Coordonnee2` &c)
- *Surcharge de l'opérateur = : realise l'affectation entre deux points.*
- `Coordonnee2 operator-` () const
- *Surcharge de l'opérateur - : renvoie l'opposé d'un point.*
- `Coordonnee2 operator-` (const `Coordonnee2` &c) const
- *Surcharge de l'opérateur - : realise la soustraction des coordonnees de deux points.*
- `Coordonnee2 operator+` (const `Coordonnee2` &c) const
- *Surcharge de l'opérateur + : realise l'addition des coordonnees de deux points.*
- void `operator+=` (const `Coordonnee2` &c)
- *Surcharge de l'opérateur +=.*
- void `operator-=` (const `Coordonnee2` &c)
- *Surcharge de l'opérateur -=.*
- void `operator*= (double val)`
- *Surcharge de l'opérateur \*=.*
- `Coordonnee2 operator*` (double val) const
- *Surcharge de l'opérateur \* : multiplication de coordonnees par un scalaire.*
- double `operator*` (const `Coordonnee2` &c) const
- *Surcharge de l'opérateur \* : produit scalaire entre coordonnees.*
- `Coordonnee2 operator/` (double val) const
- *Surcharge de l'opérateur / : division de coordonnees par un scalaire.*
- void `operator/=` (double val)
- *Surcharge de l'opérateur /= : division de coordonnees par un scalaire.*
- int `operator==` (const `Coordonnee2` &c) const
- *Surcharge de l'opérateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- `Vecteur Vect` () const
- *conversion en Vecteur*
- void `Zero` ()
- *mise a zero des coordonnées*
- double `Norme` () const
- *Calcul de la norme euclidienne des composantes du point.*
- `Coordonnee2 & Normer` ()
- *norme le vecteur coordonné*
- double `Somme` () const
- *somme de tous les composantes*

## Fonctions membres publiques statiques

- static void `SchemaXML_Coordonnee` (ofstream &sort, const `Enum_IO_XML` enu)
- *sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double `coord2` [2]

## Amis

- `Coordonnee2 operator*` (double val, const `Coordonnee2` &c)

## Membres hérités additionnels

### 6.187.1 Description détaillée

cas des coordonnées simples sans variance

## 6.187.2 Documentation des fonctions membres

### 6.187.2.1 Change\_dim()

```
void Coordonnee2::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [Coordonnee](#).

### 6.187.2.2 Dimension()

```
int Coordonnee2::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnees.

Réimplémentée à partir de [Coordonnee](#).

### 6.187.2.3 Libere()

```
void Coordonnee2::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee.

Réimplémentée à partir de [Coordonnee](#).

### 6.187.2.4 Norme()

```
double Coordonnee2::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [Coordonnee](#).

### 6.187.2.5 Vect()

```
Vecteur Coordonnee2::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [Coordonnee](#).

### 6.187.2.6 Zero()

```
void Coordonnee2::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [Coordonnee](#).

La documentation de cette classe a été générée à partir du fichier suivant :

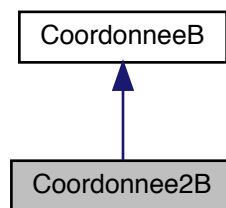
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee2\_2.cc

## 6.188 Référence de la classe Coordonnee2B

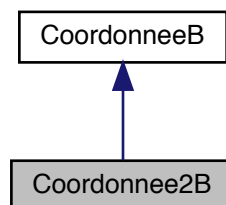
cas des coordonnées covariantes

```
#include <Coordonnee2.h>
```

Graphe d'héritage de Coordonnee2B:



Graphe de collaboration de Coordonnee2B:



## Fonctions membres publiques

- **Coordonnee2B** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee2B** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee2B** (double x, double y)  
*Constructeur pour une localisation bidimensionnelle.*
- **Coordonnee2B** (double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees ( l'existence de la place memoire est a la charge de l'utilisateur !!).*
- **Coordonnee2B** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 2.*
- **Coordonnee2B** (const [Coordonnee2B](#) &c)  
*Constructeur de copie.*
- **Coordonnee2B** (const [CoordonneeB](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*
- virtual ~**Coordonnee2B** ()  
*DESTRUCTEUR :*
- int [Dimension](#) () const  
*Renvoie le nombre de coordonnees.*
- void [Libere](#) ()  
*Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.*

- void **Change\_dim** (int dim)  
*changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
- **Coordonnee2B & operator=** (const **Coordonnee2B** &c)  
*Surcharge de l'opérateur = : realise l'affectation entre deux points.*
- **Coordonnee2B operator-** () const  
*Surcharge de l'opérateur - : renvoie l'opposé d'un point.*
- **Coordonnee2B operator-** (const **Coordonnee2B** &c) const  
*Surcharge de l'opérateur - : realise la soustraction des coordonnées de deux points.*
- **Coordonnee2B operator+** (const **Coordonnee2B** &c) const  
*Surcharge de l'opérateur + : realise l'addition des coordonnées de deux points.*
- void **operator+=** (const **Coordonnee2B** &c)  
*Surcharge de l'opérateur +=.*
- void **operator-=** (const **Coordonnee2B** &c)  
*Surcharge de l'opérateur -=.*
- void **operator\*=-** (double val)  
*Surcharge de l'opérateur \*=.*
- **Coordonnee2B operator\*** (double val) const  
*Surcharge de l'opérateur \* : multiplication de coordonnées par un scalaire.*
- double **operator\*** (const **Coordonnee2H** &c) const  
*Surcharge de l'opérateur \* : produit scalaire entre coordonnées.*
- double **ScalBB** (const **Coordonnee2B** &c) const  
*produit scalaire entre coordonnées covariantes et covariantes*
- **Coordonnee2B operator/** (double val) const  
*Surcharge de l'opérateur / : division de coordonnées par un scalaire.*
- void **operator/=** (double val)  
*Surcharge de l'opérateur /= : division de coordonnées par un scalaire.*
- int **operator==** (const **Coordonnee2B** &c) const  
*Surcharge de l'opérateur == : test d'égalité Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- **Vecteur Vect** () const  
*conversion en Vecteur*
- void **Zero** ()  
*mise a zero des coordonnées*
- double **Norme** () const  
*Calcul de la norme euclidienne des composantes du point.*
- **Coordonnee2B & Normer** ()  
*norme le vecteur coordonnée*
- double **Somme** () const  
*somme de tous les composantes*

## Fonctions membres publiques statiques

- static void **SchemaXML\_Coordonnee** (ofstream &sort, const **Enum\_IO\_XML** enu)  
*sortie du schemaXML: en fonction de enu*
- static double **Determinant2B** (const **Coordonnee2B** &v1, const **Coordonnee2B** &v2)  
*calcul du déterminant de deux vecteurs coordonnées*

## Attributs protégés

- double **coord2** [2]

## Amis

- class **Coordonnee2H**
- **Coordonnee2B operator\*** (double val, const **Coordonnee2B** &c)  
*Surcharge de l'opérateur \* : multiplication entre un scalaire et des coordonnées.*

## Membres hérités additionnels

### 6.188.1 Description détaillée

cas des coordonnées covariantes

### 6.188.2 Documentation des fonctions membres

#### 6.188.2.1 `Change_dim()`

```
void Coordonnee2B::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [CoordonneeB](#).

#### 6.188.2.2 `Dimension()`

```
int Coordonnee2B::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnées.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.188.2.3 `Libere()`

```
void Coordonnee2B::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.188.2.4 `Norme()`

```
double Coordonnee2B::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.188.2.5 `Vect()`

```
Vecteur Coordonnee2B::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [CoordonneeB](#).

#### 6.188.2.6 `Zero()`

```
void Coordonnee2B::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [CoordonneeB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

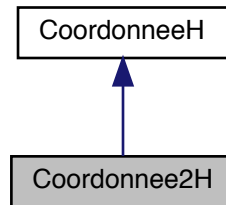
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee2.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee2B_2.cc`

## 6.189 Référence de la classe Coordonnee2H

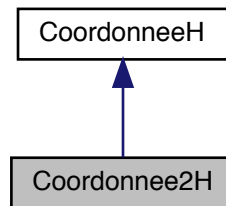
cas des coordonnées contravariantes

```
#include <Coordonnee2.h>
```

Graphe d'héritage de Coordonnee2H:



Graphe de collaboration de Coordonnee2H:



### Fonctions membres publiques

- **Coordonnee2H** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee2H** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee2H** (double x, double y)  
*Constructeur pour une localisation bidimensionnelle.*
- **Coordonnee2H** (double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees ( l'existence de la place mémoire est a la charge de l'utilisateur !!).*
- **Coordonnee2H** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 2.*
- **Coordonnee2H** (const [Coordonnee2H](#) &c)  
*Constructeur de copie.*
- **Coordonnee2H** (const [CoordonneeH](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*
- virtual ~**Coordonnee2H** ()  
*DESTRUCTEUR :*
- int [Dimension](#) () const

- *Renvoie le nombre de coordonnees.*
- void `Libere` ()
  - Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.*
- void `Change_dim` (int dim)
  - changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
- `Coordonnee2H & operator=` (const `Coordonnee2H` &c)
  - Surcharge de l'opérateur = : realise l'affectation entre deux points.*
- `Coordonnee2H operator-` () const
  - Surcharge de l'opérateur - : renvoie l'opposé d'un point.*
- `Coordonnee2H operator-` (const `Coordonnee2H` &c) const
  - Surcharge de l'opérateur - : realise la soustraction des coordonnees de deux points.*
- `Coordonnee2H operator+` (const `Coordonnee2H` &c) const
  - Surcharge de l'opérateur + : realise l'addition des coordonnees de deux points.*
- void `operator+=` (const `Coordonnee2H` &c)
  - Surcharge de l'opérateur +=.*
- void `operator-=` (const `Coordonnee2H` &c)
  - Surcharge de l'opérateur -=.*
- void `operator*=  
Surcharge de l'opérateur *=.` (double val)
  - Surcharge de l'opérateur \*=.*
- `Coordonnee2H operator*` (double val) const
  - Surcharge de l'opérateur \* : multiplication de coordonnees par un scalaire.*
- double `operator*` (const `Coordonnee2B` &c) const
  - Surcharge de l'opérateur \* : produit scalaire entre coordonnees.*
- double `ScalHH` (const `Coordonnee2H` &c) const
  - produit scalaire entre coordonnees contravariantes et contravariantes*
- `Coordonnee2H operator/` (double val) const
  - Surcharge de l'opérateur / : division de coordonnees par un scalaire.*
- void `operator/=` (double val)
  - Surcharge de l'opérateur /= : division de coordonnees par un scalaire.*
- int `operator==` (const `Coordonnee2H` &c) const
  - Surcharge de l'opérateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- `Vecteur Vect` () const
  - conversion en Vecteur*
- void `Zero` ()
  - mise a zero des coordonnées*
- double `Norme` () const
  - Calcul de la norme euclidienne des composantes du point.*
- `Coordonnee2H & Normer` ()
  - norme le vecteur coordonnée*
- double `Somme` () const
  - somme de tous les composantes*

## Fonctions membres publiques statiques

- static void `SchemaXML_Coordonnee` (ofstream &sort, const `Enum_IO_XML` enu)
  - sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double `coord2` [2]

## Amis

- class `Coordonnee2B`
- `Coordonnee2H operator*` (double val, const `Coordonnee2H` &c)

## Membres hérités additionnels

### 6.189.1 Description détaillée

cas des coordonnées contravariantes

### 6.189.2 Documentation des fonctions membres

#### 6.189.2.1 Change\_dim()

```
void Coordonnee2H::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [CoordonneeH](#).

#### 6.189.2.2 Dimension()

```
int Coordonnee2H::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnees.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.189.2.3 Libere()

```
void Coordonnee2H::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.189.2.4 Norme()

```
double Coordonnee2H::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.189.2.5 Vect()

```
Vecteur Coordonnee2H::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [CoordonneeH](#).

#### 6.189.2.6 Zero()

```
void Coordonnee2H::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [CoordonneeH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee2H\_2.cc

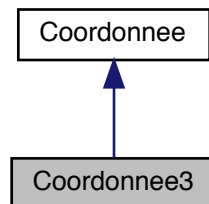


## 6.190 Référence de la classe Coordonnee3

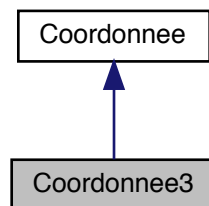
cas des coordonnées simples sans variance

```
#include <Coordonnee3.h>
```

Grphe d'héritage de Coordonnee3:



Grphe de collaboration de Coordonnee3:



### Fonctions membres publiques

- **Coordonnee3** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee3** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee3** (double x, double y, double z)  
*Constructeur pour une localisation tridimensionnelle.*
- **Coordonnee3** (double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnées ( l'existence de la place mémoire est a la charge de l'utilisateur !!).*
- **Coordonnee3** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 3.*
- **Coordonnee3** (const [Coordonnee3](#) &c)  
*Constructeur de copie.*
- **Coordonnee3** (const [Coordonnee](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*
- virtual ~**Coordonnee3** ()  
*DESTRUCTEUR :*
- int [Dimension](#) () const

- *Renvoie le nombre de coordonnees.*
- void **Libere** ()
  - Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.*
- void **Change\_dim** (int dim)
  - changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
- **Coordonnee3** & **operator=** (const **Coordonnee3** &c)
  - Surcharge de l'operateur = : realise l'affectation entre deux points.*
- **Coordonnee3** **operator-** () const
  - Surcharge de l'operateur - : renvoie l'oppose d'un point.*
- **Coordonnee3** **operator-** (const **Coordonnee3** &c) const
  - Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points.*
- **Coordonnee3** **operator+** (const **Coordonnee3** &c) const
  - Surcharge de l'operateur + : realise l'addition des coordonnees de deux points.*
- void **operator+=** (const **Coordonnee3** &c)
  - Surcharge de l'operateur +=.*
- void **operator-=** (const **Coordonnee3** &c)
  - Surcharge de l'operateur -=.*
- void **operator\*=** (double val)
  - Surcharge de l'operateur \*=.*
- **Coordonnee3** **operator\*** (double val) const
  - Surcharge de l'operateur \* : multiplication de coordonnees par un scalaire.*
- double **operator\*** (const **Coordonnee3** &c) const
  - Surcharge de l'operateur \* : produit scalaire entre coordonnees.*
- **Coordonnee3** **operator/** (double val) const
  - Surcharge de l'operateur / : division de coordonnees par un scalaire.*
- void **operator/=** (double val)
  - Surcharge de l'operateur /= : division de coordonnees par un scalaire.*
- int **operator==** (const **Coordonnee3** &c) const
  - Surcharge de l'operateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- **Vecteur** **Vect** () const
  - conversion en Vecteur*
- void **Zero** ()
  - mise a zero des coordonnées*
- double **Norme** () const
  - Calcul de la norme euclidienne des composantes du point.*
- **Coordonnee3** & **Normer** ()
  - norme le vecteur coordonnée*
- double **Somme** () const
  - somme de tous les composantes*

## Fonctions membres publiques statiques

- static void **SchemaXML\_Coordonnee** (ofstream &sort, const **Enum\_IO\_XML** enu)
  - sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **coord3** [3]

## Amis

- **Coordonnee3** **operator\*** (double val, const **Coordonnee3** &c)
  - Surcharge de l'operateur \* : multiplication entre un scalaire et des coordonnees.*

## Membres hérités additionnels

### 6.190.1 Description détaillée

cas des coordonnées simples sans variance

## 6.190.2 Documentation des fonctions membres

### 6.190.2.1 Change\_dim()

```
void Coordonnee3::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [Coordonnee](#).

### 6.190.2.2 Dimension()

```
int Coordonnee3::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnees.

Réimplémentée à partir de [Coordonnee](#).

### 6.190.2.3 Libere()

```
void Coordonnee3::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.

Réimplémentée à partir de [Coordonnee](#).

### 6.190.2.4 Norme()

```
double Coordonnee3::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [Coordonnee](#).

### 6.190.2.5 Vect()

```
Vecteur Coordonnee3::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [Coordonnee](#).

### 6.190.2.6 Zero()

```
void Coordonnee3::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [Coordonnee](#).

La documentation de cette classe a été générée à partir du fichier suivant :

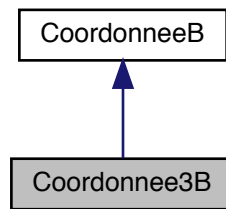
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee3\_2.cc

## 6.191 Référence de la classe Coordonnee3B

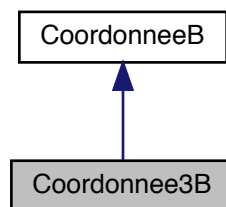
cas des coordonnées covariantes

```
#include <Coordonnee3.h>
```

Graphe d'héritage de Coordonnee3B:



Graphe de collaboration de Coordonnee3B:



## Fonctions membres publiques

- **Coordonnee3B** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee3B** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee3B** (double x, double y, double z)  
*Constructeur pour une localisation tridimensionnelle.*
- **Coordonnee3B** (double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees ( l'existence de la place memoire est a la charge de l'utilisateur !!).*
- **Coordonnee3B** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 3.*
- **Coordonnee3B** (const [Coordonnee3B](#) &c)  
*Constructeur de copie.*
- **Coordonnee3B** (const [CoordonneeB](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*
- virtual ~**Coordonnee3B** ()  
*DESTRUCTEUR :*
- int [Dimension](#) () const  
*Renvoie le nombre de coordonnees.*
- void [Libere](#) ()  
*Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.*

- void `Change_dim` (int dim)  
*changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
- `Coordonnee3B & operator=` (const `Coordonnee3B &c`)  
*Surcharge de l'opérateur = : réalise l'affectation entre deux points.*
- `Coordonnee3B operator-` () const  
*Surcharge de l'opérateur - : renvoie l'opposé d'un point.*
- `Coordonnee3B operator-` (const `Coordonnee3B &c`) const  
*Surcharge de l'opérateur - : réalise la soustraction des coordonnées de deux points.*
- `Coordonnee3B operator+` (const `Coordonnee3B &c`) const  
*Surcharge de l'opérateur + : réalise l'addition des coordonnées de deux points.*
- void `operator+=` (const `Coordonnee3B &c`)  
*Surcharge de l'opérateur +=.*
- void `operator-=` (const `Coordonnee3B &c`)  
*Surcharge de l'opérateur -=.*
- void `operator*=-` (double val)  
*Surcharge de l'opérateur \*=.*
- `Coordonnee3B operator*` (double val) const  
*Surcharge de l'opérateur \* : multiplication de coordonnées par un scalaire.*
- double `operator*` (const `Coordonnee3H &c`) const  
*Surcharge de l'opérateur \* : produit scalaire entre coordonnées.*
- double `ScalBB` (const `Coordonnee3B &c`) const  
*produit scalaire entre coordonnées covariantes et covariantes*
- `Coordonnee3B operator/` (double val) const  
*Surcharge de l'opérateur / : division de coordonnées par un scalaire.*
- void `operator/=` (double val)  
*Surcharge de l'opérateur /= : division de coordonnées par un scalaire.*
- int `operator==` (const `Coordonnee3B &c`) const  
*Surcharge de l'opérateur == : test d'égalité Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- `Vecteur Vect` () const  
*conversion en `Vecteur`*
- void `Zero` ()  
*mise à zéro des coordonnées*
- double `Norme` () const  
*Calcul de la norme euclidienne des composantes du point.*
- `Coordonnee3B & Normer` ()  
*norme le vecteur coordonnée*
- double `Somme` () const  
*somme de tous les composantes*

## Fonctions membres publiques statiques

- static void `SchemaXML_Coordonnee` (ofstream &sort, const `Enum_IO_XML` enu)  
*sortie du schemaXML: en fonction de enu*
- static double `Determinant3B` (const `Coordonnee3B &v1`, const `Coordonnee3B &v2`, const `Coordonnee3B &v3`)  
*calcul du produit mixte de trois vecteurs coordonnées*
- static `Coordonnee3H Vectoriel` (const `Coordonnee3B &v1`, const `Coordonnee3B &v2`)  
*calcul du produit vectoriel de 2 vecteurs coordonnées*

## Attributs protégés

- double `coord3` [3]

## Amis

- class `Coordonnee3H`
- `Coordonnee3B operator*` (double val, const `Coordonnee3B &c`)  
*Surcharge de l'opérateur \* : multiplication entre un scalaire et des coordonnées.*

## Membres hérités additionnels

### 6.191.1 Description détaillée

cas des coordonnées covariantes

### 6.191.2 Documentation des fonctions membres

#### 6.191.2.1 Change\_dim()

```
void Coordonnee3B::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [CoordonneeB](#).

#### 6.191.2.2 Dimension()

```
int Coordonnee3B::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnees.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.191.2.3 Libere()

```
void Coordonnee3B::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.191.2.4 Norme()

```
double Coordonnee3B::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [CoordonneeB](#).

#### 6.191.2.5 Vect()

```
Vecteur Coordonnee3B::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [CoordonneeB](#).

#### 6.191.2.6 Zero()

```
void Coordonnee3B::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [CoordonneeB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

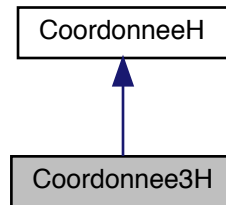
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee3B\_2.cc

## 6.192 Référence de la classe Coordonnee3H

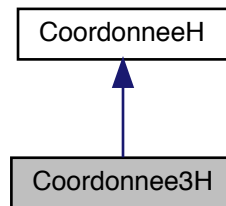
cas des coordonnées contravariantes

```
#include <Coordonnee3.h>
```

Grphe d'héritage de Coordonnee3H:



Grphe de collaboration de Coordonnee3H:



### Fonctions membres publiques

- **Coordonnee3H** ()  
*Constructeur par défaut il y a initialisation des coordonnées à zéro par défaut.*
- **Coordonnee3H** (bool test)  
*Constructeur suivant un booléen quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées ceci pour aller plus vite par rapport au constructeur par défaut.*
- **Coordonnee3H** (double x, double y, double z)  
*Constructeur pour une localisation tridimensionnelle.*
- **Coordonnee3H** (double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees ( l'existence de la place mémoire est a la charge de l'utilisateur !!).*
- **Coordonnee3H** (const [Vecteur](#) &vec)  
*Constructeur fonction d'un vecteur qui doit avoir une 3.*
- **Coordonnee3H** (const [Coordonnee3H](#) &c)  
*Constructeur de copie.*
- **Coordonnee3H** (const [CoordonneeH](#) &c)  
*Constructeur de copie pour une instance indifférenciée.*
- virtual ~**Coordonnee3H** ()  
*DESTRUCTEUR :*
- int [Dimension](#) () const

- *Renvoie le nombre de coordonnees.*
- void **Libere** ()  
*Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.*
- void **Change\_dim** (int dim)  
*changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur*
- **Coordonnee3H & operator=** (const **Coordonnee3H** &c)  
*Surcharge de l'operateur = : realise l'affectation entre deux points.*
- **Coordonnee3H operator-** () const  
*Surcharge de l'operateur - : renvoie l'oppose d'un point.*
- **Coordonnee3H operator-** (const **Coordonnee3H** &c) const  
*Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points.*
- **Coordonnee3H operator+** (const **Coordonnee3H** &c) const  
*Surcharge de l'operateur + : realise l'addition des coordonnees de deux points.*
- void **operator+=** (const **Coordonnee3H** &c)  
*Surcharge de l'operateur +=.*
- void **operator-=** (const **Coordonnee3H** &c)  
*Surcharge de l'operateur -=.*
- void **operator\*= **(double val)****  
*Surcharge de l'operateur \*=.*
- **Coordonnee3H operator\*** (double val) const  
*Surcharge de l'operateur \* : multiplication de coordonnees par un scalaire.*
- double **operator\*** (const **Coordonnee3B** &c) const  
*Surcharge de l'operateur \* : produit scalaire entre coordonnees.*
- double **ScalHH** (const **Coordonnee3H** &c) const  
*produit scalaire entre coordonnees contravariantes et contravariantes*
- **Coordonnee3H operator/** (double val) const  
*Surcharge de l'operateur / : division de coordonnees par un scalaire.*
- void **operator/=** (double val)  
*Surcharge de l'operateur /= : division de coordonnees par un scalaire.*
- int **operator==** (const **Coordonnee3H** &c) const  
*Surcharge de l'operateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- **Vecteur Vect** () const  
*conversion en Vecteur*
- void **Zero** ()  
*mise a zero des coordonnees*
- double **Norme** () const  
*Calcul de la norme euclidienne des composantes du point.*
- **Coordonnee3H & Normer** ()  
*norme le vecteur coordonnée*
- double **Somme** () const  
*somme de tous les composantes*

## Fonctions membres publiques statiques

- static void **SchemaXML\_Coordonnee** (ofstream &sort, const **Enum\_IO\_XML** enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **coord3** [3]

## Amis

- class **Coordonnee3B**
- **Coordonnee3H operator\*** (double val, const **Coordonnee3H** &c)  
*Surcharge de l'operateur \* : multiplication entre un scalaire et des coordonnees.*



## Membres hérités additionnels

### 6.192.1 Description détaillée

cas des coordonnées contravariantes

### 6.192.2 Documentation des fonctions membres

#### 6.192.2.1 `Change_dim()`

```
void Coordonnee3H::Change_dim (
    int dim ) [virtual]
```

changement de la dimension fonction définie dans la classe mère générique mais qui n'a pas de sens ici, affiche un message d'erreur

Réimplémentée à partir de [CoordonneeH](#).

#### 6.192.2.2 `Dimension()`

```
int Coordonnee3H::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnées.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.192.2.3 `Libere()`

```
void Coordonnee3H::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee fonction définie dans la classe mère générique mais qui n'a pas de sens ici.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.192.2.4 `Norme()`

```
double Coordonnee3H::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée à partir de [CoordonneeH](#).

#### 6.192.2.5 `Vect()`

```
Vecteur Coordonnee3H::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée à partir de [CoordonneeH](#).

#### 6.192.2.6 `Zero()`

```
void Coordonnee3H::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée à partir de [CoordonneeH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

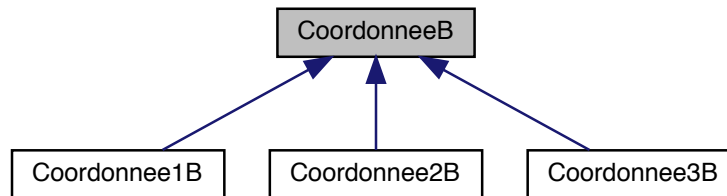
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee3.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Coordonnees/Coordonnee3H_2.cc`

## 6.193 Référence de la classe CoordonneeB

cas des coordonnées covariantes

```
#include <Coordonnee.h>
```

Graphe d'héritage de CoordonneeB:



### Fonctions membres publiques

- **CoordonneeB** ()  
*Constructeur par défaut.*
- **CoordonneeB** (int dimension)  
*Constructeur fonction de la dimension du probleme les coordonnees sont initialise a zero.*
- **CoordonneeB** (double x)  
*Constructeur pour une localisation unidimensionnelle.*
- **CoordonneeB** (double x, double y)  
*Constructeur pour une localisation bidimensionnelle.*
- **CoordonneeB** (double x, double y, double z)  
*Constructeur pour une localisation tridimensionnelle.*
- **CoordonneeB** (int dimension, double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees et d'une dimension ( l'existence de la place memoire est a la charge de l'utilisateur et ne sera pas détruite par le destructeur.*
- **CoordonneeB** (const [CoordonneeB](#) &c)  
*Constructeur de copie.*
- virtual ~**CoordonneeB** ()  
*DESTRUCTEUR :*
- void **ConstructionAPartirDe\_H** (const [CoordonneeH](#) &aH)  
*construction "explicite" à partir d'une instance de [CoordonneeB](#) intéressant si this est initialement construit par défaut (donc vide) cela permet de créer un [CoordonneeH](#) à partir d'un B, mais de manière explicite, donc activé quand on le veut (et non pas par le compilateur au gré de conversion pas toujours clair!!)*
- virtual int **Dimension** () const  
*Renvoie le nombre de coordonnees.*
- virtual void **Libere** ()  
*Desallocation de la place memoire allouee.*
- virtual double & **operator()** (int i)  
*Renvoie la ieme coordonnee.*
- virtual double **operator()** (int i) const  
*Renvoie une copie de la ieme coordonnee.*
- [CoordonneeB](#) & **operator=** (const [CoordonneeB](#) &c)  
*Surcharge de l'opérateur = : realise l'affectation entre deux points.*
- void **Change\_val** (const [Coordonnee](#) &c)  
*change les valeurs en fonction d'un point sans variance*
- [CoordonneeB](#) **operator-** () const  
*Surcharge de l'opérateur - : renvoie l'opposé d'un point.*
- [CoordonneeB](#) **operator-** (const [CoordonneeB](#) &c) const  
*Surcharge de l'opérateur - : realise la soustraction des coordonnees de deux points.*

- `CoordonneeB operator+` (const `CoordonneeB` &c) const  
*Surcharge de l'opérateur + : réalise l'addition des coordonnées de deux points.*
- void `operator+=` (const `CoordonneeB` &c)  
*Surcharge de l'opérateur +=.*
- void `operator-=` (const `CoordonneeB` &c)  
*Surcharge de l'opérateur -=.*
- void `operator*=  
CoordonneeB operator*` (double val) const  
*Surcharge de l'opérateur \*= : multiplication de coordonnées par un scalaire.*
- double `operator*` (const `CoordonneeH` &c) const  
*Surcharge de l'opérateur \* : produit scalaire entre coordonnées.*
- double `ScalBB` (const `CoordonneeB` &c) const  
*produit scalaire entre coordonnées covariantes et covariantes*
- `CoordonneeB operator/` (double val) const  
*Surcharge de l'opérateur / : division de coordonnées par un scalaire.*
- void `operator/=` (double val)  
*Surcharge de l'opérateur /= : division de coordonnées par un scalaire.*
- int `operator==` (const `CoordonneeB` &c) const  
*Surcharge de l'opérateur == : test d'égalité Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- int `operator!=` (const `CoordonneeB` &c) const  
*Surcharge de l'opérateur != Renvoie 1 si les deux positions ne sont pas identiques Renvoie 0 sinon.*
- virtual void `Affiche` () const  
*Affiche les coordonnées du point à l'écran.*
- virtual void `Affiche` (ostream &sort) const  
*Affiche les coordonnées du point dans sort.*
- virtual void `Affiche` (ostream &sort, int nb) const
- virtual void `Lecture` (`UtilLecture` &entreePrinc)  
*lecture brut des coordonnées sans la dimension*
- virtual void `Change_dim` (int dim)  
*changement de la dimension dans le cas d'une nouvelle dimension inférieur on supprime les dernières coord dans le cas d'une dimension supérieur, on ajoute des coord initialisées a zero*
- virtual `Vecteur Vect` () const  
*conversion en Vecteur*
- virtual `Coordonnee Coor` () const  
*conversion explicite en coordonnées sans variance*
- virtual const `Coordonnee Coor_const` () const  
*création explicite en coordonnées sans variance mais le vecteur est à la même place pour un coût de construction minimum, il est accessible en lecture uniquement*
- virtual `CoordonneeH Bas_haut` () const  
*conversion explicite de B en H*
- virtual void `Zero` ()  
*mise a zero des coordonnées*
- virtual double `Norme` () const  
*Calcul de la norme euclidienne des composantes du point.*
- `CoordonneeB & Normer` ()  
*norme le vecteur coordonnée*
- double `Max_val_abs` () const  
*Calcul du maximum en valeur absolu des composantes du vecteur.*
- double `Max_val_abs` (int &i) const  
*Calcul du maximum en valeur absolu des composantes du vecteur ramene également l'indice de tableau du maximum.*
- double `Max_val_abs_signe` () const
- double `Max_val_abs_signe` (int &i) const  
*Calcul du maximum en valeur absolu des composantes du vecteur mais ramène la grandeur signée (avec son signe) ramene également l'indice de tableau du maximum.*
- void `Modif_en_max` (const `CoordonneeB` &v)  
*modifie éventuellement les coordonnées de this pour quelles soient supérieures ou égales aux coordonnées en paramètre*
- void `Modif_en_min` (const `CoordonneeB` &v)  
*modifie éventuellement les coordonnées de this pour quelles soient inférieures ou égales aux coordonnées en paramètre*
- void `Ajout_meme_valeur` (double val)  
*ajoute une même valeur à tous les coordonnées*

## Fonctions membres publiques statiques

- static void **SchemaXML\_Coordonnee** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Fonctions membres protégées

- **CoordonneeB** (bool)

## Attributs protégés

- short int **dim**
- bool **memoire**
- double \* **coord**

## Amis

- class **Coordonnee**
- class **CoordonneeH**
- istream & **operator**>> (istream &, [CoordonneeB](#) &)  
*surcharge de l'operator de lecture avec le type*
- ostream & **operator**<< (ostream &, const [CoordonneeB](#) &)  
*surcharge de l'operator d'écriture*
- [CoordonneeB](#) **operator**\* (double val, [CoordonneeB](#) &c)  
*Surcharge de l'operator \* : multiplication entre un scalaire et des coordonnees.*

### 6.193.1 Description détaillée

cas des coordonnées covariantes

### 6.193.2 Documentation des fonctions membres

#### 6.193.2.1 Change\_dim()

```
virtual void CoordonneeB::Change_dim (
    int dim ) [virtual]
```

changement de la dimension dans le cas d'une nouvelle dimension inferieur on supprime les dernieres coord dans le cas d'une dimension superieur, on ajoute des coord initialisees a zero  
Réimplémentée dans [Coordonnee1B](#), [Coordonnee2B](#), et [Coordonnee3B](#).

#### 6.193.2.2 Dimension()

```
virtual int CoordonneeB::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnees.

Réimplémentée dans [Coordonnee1B](#), [Coordonnee2B](#), et [Coordonnee3B](#).

#### 6.193.2.3 Libere()

```
virtual void CoordonneeB::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee.

Réimplémentée dans [Coordonnee1B](#), [Coordonnee2B](#), et [Coordonnee3B](#).

#### 6.193.2.4 Norme()

```
virtual double CoordonneeB::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée dans [Coordonnee1B](#), [Coordonnee2B](#), et [Coordonnee3B](#).

#### 6.193.2.5 Vect()

```
Vecteur CoordonneeB::Vect ( ) const [virtual]
```

conversion en [Vecteur](#)

Réimplémentée dans [Coordonnee1B](#), [Coordonnee2B](#), et [Coordonnee3B](#).

#### 6.193.2.6 Zero()

```
virtual void CoordonneeB::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée dans [Coordonnee1B](#), [Coordonnee2B](#), et [Coordonnee3B](#).

La documentation de cette classe a été générée à partir du fichier suivant :

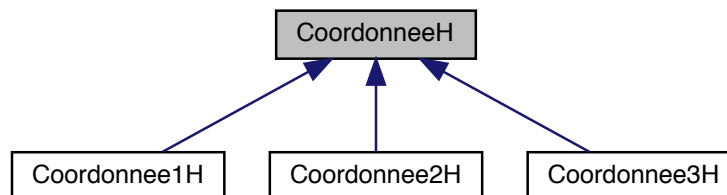
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/CoordonneeB\_2.cc

## 6.194 Référence de la classe CoordonneeH

cas des coordonnées contravariantes

```
#include <Coordonnee.h>
```

Graphe d'héritage de CoordonneeH:



### Fonctions membres publiques

- **CoordonneeH** ()  
*Constructeur par défaut.*
- **CoordonneeH** (int dimension)  
*Constructeur fonction de la dimension du probleme les coordonnees sont initialise a zero.*
- **CoordonneeH** (double x)  
*Constructeur pour une localisation unidimensionnelle.*
- **CoordonneeH** (double x, double y)  
*Constructeur pour une localisation bidimensionnelle.*
- **CoordonneeH** (double x, double y, double z)  
*Constructeur pour une localisation tridimensionnelle.*
- **CoordonneeH** (int dimension, double \*t)  
*constructeur fonction d'une adresse memoire ou sont stockee les coordonnees et d'une dimension ( l'existence de la place memoire est a la charge de l'utilisateur et ne sera pas détruite par le destructeur.*

- **CoordonneeH** (const [CoordonneeH](#) &c)  
*Constructeur de copie.*
- virtual ~**CoordonneeH** ()  
*DESTRUCTEUR :*
- void **ConstructionAPartirDe\_B** (const [CoordonneeB](#) &aB)  
*construction "explicite" à partir d'une instance de [CoordonneeB](#) intéressant si this est initialement construit par défaut (donc vide) cela permet de créer un [CoordonneeH](#) à partir d'un B, mais de manière explicite, donc activé quand on le veut (et non pas par le compilateur au gré de conversion pas toujours clair!!)*
- virtual int **Dimension** () const  
*Renvoie le nombre de coordonnées.*
- virtual void **Libere** ()  
*Desallocation de la place memoire allouee.*
- virtual double & **operator()** (int i)  
*Renvoie la ieme coordonnee.*
- virtual double **operator()** (int i) const  
*Renvoie une copie de la ieme coordonnee.*
- [CoordonneeH](#) & **operator=** (const [CoordonneeH](#) &c)
- void **Change\_val** (const [Coordonnee](#) &c)  
*change les valeurs en fonction d'un point sans variance*
- [CoordonneeH](#) **operator-** () const  
*Surcharge de l'operateur - : renvoie l'oppose d'un point.*
- [CoordonneeH](#) **operator-** (const [CoordonneeH](#) &c) const  
*Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points.*
- [CoordonneeH](#) **operator+** (const [CoordonneeH](#) &c) const  
*Surcharge de l'operateur + : realise l'addition des coordonnees de deux points.*
- void **operator+=** (const [CoordonneeH](#) &c)  
*Surcharge de l'operateur +=.*
- void **operator-=** (const [CoordonneeH](#) &c)  
*Surcharge de l'operateur -=.*
- void **operator\*=  
operator\*** (double val)  
*Surcharge de l'operateur \*=.*
- [CoordonneeH](#) **operator\*** (double val) const  
*Surcharge de l'operateur \* : multiplication de coordonnees par un scalaire.*
- double **operator\*** (const [CoordonneeB](#) &c) const  
*Surcharge de l'operateur \* : produit scalaire entre coordonnees.*
- double **ScalHH** (const [CoordonneeH](#) &c) const  
*produit scalaire entre coordonnees contravariantes et contravariantes*
- [CoordonneeH](#) **operator/** (double val) const  
*Surcharge de l'operateur / : division de coordonnees par un scalaire.*
- void **operator/=** (double val)  
*Surcharge de l'operateur /= : division de coordonnees par un scalaire.*
- int **operator==** (const [CoordonneeH](#) &c) const  
*Surcharge de l'operateur == : test d'egalite Renvoie 1 si les deux positions sont identiques Renvoie 0 sinon.*
- int **operator!=** (const [CoordonneeH](#) &c) const  
*Surcharge de l'operateur != Renvoie 1 si les deux positions ne sont pas identiques Renvoie 0 sinon.*
- virtual void **Affiche** () const  
*Affiche les coordonnees du point à l'écran.*
- virtual void **Affiche** (ostream &sort) const  
*Affiche les coordonnees du point dans sort.*
- virtual void **Affiche** (ostream &sort, int nb) const  
*Affiche les coordonnees du point dans sort sur nb digit plus un blanc et rien d'autre.*
- virtual void **Lecture** ([UtilLecture](#) &entreePrinc)  
*lecture brut des coordonnées sans la dimension*
- virtual void **Change\_dim** (int dim)  
*changement de la dimension dans le cas d'une nouvelle dimension inferieur on supprime les dernieres coord dans le cas d'une dimension superieur, on ajoute des coord initialisees a zero*
- virtual [Vecteur](#) **Vect** () const  
*création d'un [Vecteur](#) équivalent*
- virtual [Coordonnee](#) **Coor** () const  
*création de coordonnées équivalentes sans variance*

- virtual const `Coordonnee` **Coor\_const** () const  
*création explicite en coordonnées sans variance mais le vecteur est à la même place pour un coût de construction minimum, il est accessible en lecture uniquement*
- virtual `CoordonneeB` **Haut\_bas** () const  
*création explicite de H en B*
- virtual void **Zero** ()  
*mise a zero des coordonnées*
- virtual double **Norme** () const  
*Calcul de la norme euclidienne des composantes du point.*
- `CoordonneeH` & **Normer** ()  
*norme le vecteur coordonnée*
- double **Max\_val\_abs** () const  
*Calcul du maximum en valeur absolu des composantes du vecteur.*
- double **Max\_val\_abs** (int &i) const  
*Calcul du maximum en valeur absolu des composantes du vecteur ramene également l'indice de tableau du maximum.*
- double **Max\_val\_abs\_signe** () const  
*Calcul du maximum en valeur absolu des composantes du vecteur mais ramène la grandeur signée (avec son signe)*
- double **Max\_val\_abs\_signe** (int &i) const  
*Calcul du maximum en valeur absolu des composantes du vecteur mais ramène la grandeur signée (avec son signe) ramene également l'indice de tableau du maximum.*
- void **Modif\_en\_max** (const `CoordonneeH` &v)  
*modifie éventuellement les coordonnées de this pour quelles soient supérieures ou égales aux coordonnées en paramètre*
- void **Modif\_en\_min** (const `CoordonneeH` &v)  
*modifie éventuellement les coordonnées de this pour quelles soient inférieures ou égales aux coordonnées en paramètre*
- void **Ajout\_meme\_valeur** (double val)  
*ajoute une même valeur à tous les coordonnées*

### Fonctions membres publiques statiques

- static void **SchemaXML\_Coordonnee** (ofstream &sort, const `Enum_IO_XML` enu)  
*sortie du schemaXML: en fonction de enu*

### Fonctions membres protégées

- `CoordonneeH` (bool)  
*Constructeur inline qui ne fait rien.*

### Attributs protégés

- short int **dim**
- bool **memoire**
- double \* **coord**

### Amis

- class `Coordonnee`
- class `CoordonneeB`
- istream & **operator**>> (istream &, `CoordonneeH` &)  
*surcharge de l'operator de lecture avec le type*
- ostream & **operator**<< (ostream &, const `CoordonneeH` &)  
*surcharge de l'operator d'écriture*
- `CoordonneeH` **operator**\* (double val, const `CoordonneeH` &c)  
*Surcharge de l'operateur \* : multiplication entre un scalaire et des coordonnees.*

#### 6.194.1 Description détaillée

cas des coordonnées contravariantes

## 6.194.2 Documentation des fonctions membres

### 6.194.2.1 Change\_dim()

```
virtual void CoordonneeH::Change_dim (
    int dim ) [virtual]
```

changement de la dimension dans le cas d'une nouvelle dimension inferieur on supprime les dernieres coord dans le cas d'une dimension superieur, on ajoute des coord initialisees a zero  
Réimplémentée dans [Coordonnee1H](#), [Coordonnee2H](#), et [Coordonnee3H](#).

### 6.194.2.2 Dimension()

```
virtual int CoordonneeH::Dimension ( ) const [virtual]
```

Renvoie le nombre de coordonnees.

Réimplémentée dans [Coordonnee1H](#), [Coordonnee2H](#), et [Coordonnee3H](#).

### 6.194.2.3 Libere()

```
virtual void CoordonneeH::Libere ( ) [virtual]
```

Desallocation de la place memoire allouee.

Réimplémentée dans [Coordonnee1H](#), [Coordonnee2H](#), et [Coordonnee3H](#).

### 6.194.2.4 Norme()

```
virtual double CoordonneeH::Norme ( ) const [virtual]
```

Calcul de la norme euclidienne des composantes du point.

Réimplémentée dans [Coordonnee1H](#), [Coordonnee2H](#), et [Coordonnee3H](#).

### 6.194.2.5 Vect()

```
Vecteur CoordonneeH::Vect ( ) const [virtual]
```

création d'un [Vecteur](#) équivalent

Réimplémentée dans [Coordonnee1H](#), [Coordonnee2H](#), et [Coordonnee3H](#).

### 6.194.2.6 Zero()

```
virtual void CoordonneeH::Zero ( ) [virtual]
```

mise a zero des coordonnées

Réimplémentée dans [Coordonnee1H](#), [Coordonnee2H](#), et [Coordonnee3H](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/Coordonnee.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Coordonnees/CoordonneeH\_2.cc

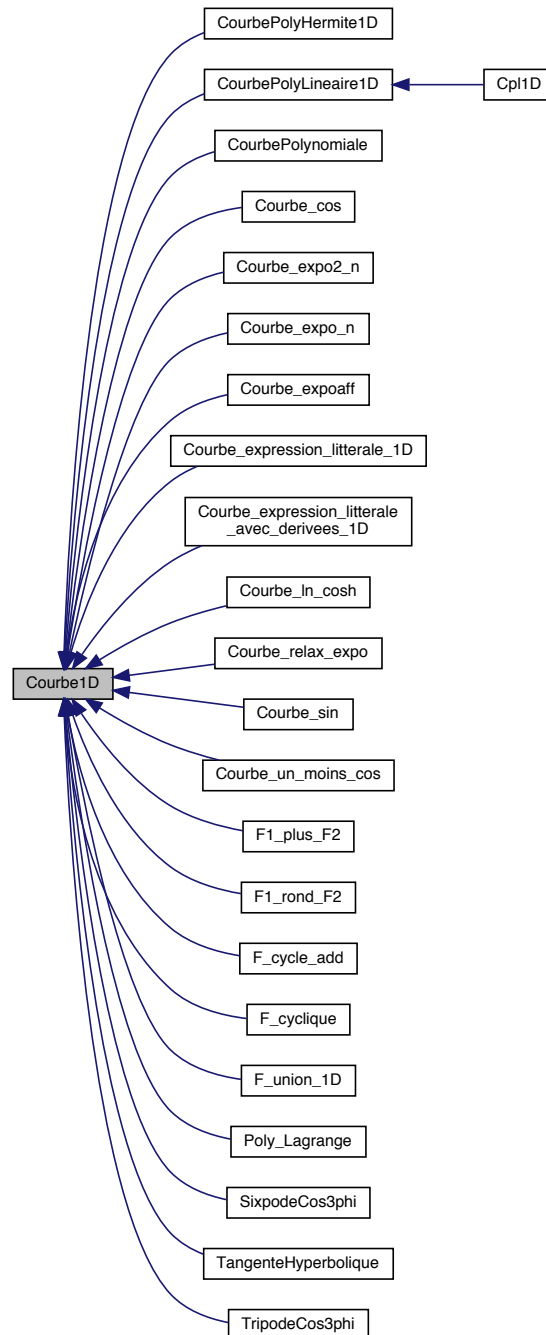
## 6.195 Référence de la classe Courbe1D

Classe virtuelle permettant le calcul d'une fonction 1D ainsi qu'éventuellement un certain nombre d'information supplémentaires telles que dérivées.

```
#include <Courbe1D.h>
```



Graphe d'héritage de Courbe1D:



## Classes

- class [Valbool](#)  
*conteneur public pour une valeur et un booléen pour le strictement inclus*
- class [ValDer](#)  
*conteneur public pour une valeur et une dérivée*
- class [ValDer2](#)  
*conteneur public pour une valeur, une dérivée première et une dérivée seconde*
- class [ValDerbool](#)

conteneur public pour une valeur et une dérivée et un booléen pour le strictement inclus

## Fonctions membres publiques

- **Courbe1D** (string nom="", [EnumCourbe1D](#) typ=AUCUNE\_COURBE1D)  
*CONSTRUCTEURS : par défaut.*
- **Courbe1D** (const [Courbe1D](#) &Co)  
*de copie*
- virtual ~**Courbe1D** ()  
*DESTRUCTEUR :*
- virtual void **Affiche** () const =0  
*affichage de la courbe*
- const string & **NomCourbe** () const  
*ramène le nom de la courbe*
- virtual bool **Complet\_courbe** () const =0  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- virtual void **LectDonnParticulieres\_courbes** (const string &nom, [UtilLecture](#) \*)=0  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- virtual bool **DependAutreCourbes** () const  
*établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non*
- virtual list< string > & **ListDependanceCourbes** (list< string > &lico) const  
*par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this*
- virtual void **Lien\_entre\_courbe** (list< [Courbe1D](#) \* > &)  
*3) établit la connection entre la demande de \*this et les courbes passées en paramètres*
- virtual void **Info\_commande\_Courbes1D** ([UtilLecture](#) &entreePrinc)=0  
*def info fichier de commande*
- virtual double **Valeur** (double x)=0  
*ramène la valeur*
- virtual [Courbe1D::ValDer](#) **Valeur\_Et\_derivee** (double x)=0  
*ramène la valeur et la dérivée en paramètre*
- virtual double **Derivee** (double x)=0  
*ramène la dérivée*
- virtual [Courbe1D::ValDer2](#) **Valeur\_Et\_der12** (double x)=0  
*ramène la valeur et les dérivées première et seconde en paramètre*
- virtual double **Der\_sec** (double x)=0  
*ramène la dérivée seconde*
- virtual [Valbool](#) **Valeur\_stricte** (double x)=0  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- virtual [ValDerbool](#) **Valeur\_Et\_derivee\_stricte** (double x)=0  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)=0  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)=0  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- virtual void **SchemaXML\_Courbes1D** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)=0  
*sortie du schemaXML: en fonction de enu*
- [EnumCourbe1D](#) **Type\_courbe** () const  
*ramène le type de la courbe*

## Fonctions membres publiques statiques

- static [Courbe1D](#) \* [New\\_Courbe1D](#) (string &nom, [EnumCourbe1D](#) typeCourbe)  
*ramène un pointeur sur la courbe correspondant au type de courbe passé en paramètre IMPORTANT : il y a création d'une courbe (utilisation d'un new)*
- static [Courbe1D](#) \* [New\\_Courbe1D](#) (const [Courbe1D](#) &Co)  
*ramène un pointeur sur une courbe copie de celle passée en paramètre IMPORTANT : il y a création d'une courbe (utilisation d'un new)*
- static list< [EnumCourbe1D](#) > [Liste\\_courbe\\_disponible](#) ()  
*ramène la liste des identificateurs de courbes actuellement disponibles*

## Fonctions membres protégées

- bool [Complet\\_var](#) () const

## Attributs protégés

- [EnumCourbe1D](#) typeCourbe
- string nom\_ref
- int permet\_affichage

### 6.195.1 Description détaillée

Classe virtuelle permettant le calcul d'une fonction 1D ainsi qu'éventuellement un certain nombre d'information supplémentaires telles que dérivées.

BUT: Classe virtuelle permettant le calcul d'une fonction 1D ainsi qu'éventuellement un certain nombre d'information supplémentaires telles que dérivées. si le nom de la courbe = "\_" il s'agit d'une courbe interne à un objet, c'est-à-dire gérée seulement par l'entité qui la contient, donc pas besoin de nom (elle n'est pas utilisée autre part). Si le nom est différent de "\_" c'est une courbe qui est gérée et référencée dans LesCourbes, donc à partir de son nom, on peut la retrouver.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.195.2 Documentation des fonctions membres

#### 6.195.2.1 Affiche()

```
virtual void Courbe1D::Affiche ( ) const [pure virtual]
```

affichage de la courbe

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_In\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

#### 6.195.2.2 Complet\_courbe()

```
virtual bool Courbe1D::Complet_courbe ( ) const [pure virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_In\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#),

[CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

### 6.195.2.3 DependAutreCourbes()

```
virtual bool Courbe1D::DependAutreCourbes ( ) const [inline], [virtual]
```

établir le lien entre la courbe et des courbes déjà existantes dont on connaît que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non

Réimplémentée dans [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), et [F\\_union\\_1D](#).

### 6.195.2.4 Der\_sec()

```
virtual double Courbe1D::Der_sec (
    double x ) [pure virtual]
```

ramène la dérivée seconde

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_ln\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

### 6.195.2.5 Derivee()

```
virtual double Courbe1D::Derivee (
    double x ) [pure virtual]
```

ramène la dérivée

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_ln\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

### 6.195.2.6 Ecriture\_base\_info()

```
virtual void Courbe1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [pure virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_ln\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

### 6.195.2.7 Info\_commande\_Courbes1D()

```
virtual void Courbe1D::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [pure virtual]
```

def info fichier de commande

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_ln\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [Cpl1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

**6.195.2.8 LectDonnParticulieres\_courbes()**

```
virtual void Courbe1D::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [pure virtual]
```

Lecture des données de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la méthode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_ln\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [Cpl1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

**6.195.2.9 Lecture\_base\_info()**

```
virtual void Courbe1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [pure virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_ln\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

**6.195.2.10 Lien\_entre\_courbe()**

```
virtual void Courbe1D::Lien_entre_courbe (
    list< Courbe1D * > & ) [inline], [virtual]
```

3) établit la connection entre la demande de \*this et les courbes passées en paramètres

Réimplémentée dans [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), et [F\\_union\\_1D](#).

**6.195.2.11 ListDependanceCourbes()**

```
list< string > & Courbe1D::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this

Réimplémentée dans [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), et [F\\_union\\_1D](#).

**6.195.2.12 SchemaXML\_Courbes1D()**

```
virtual void Courbe1D::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [pure virtual]
```

sortie du schemaXML: en fonction de enu

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_ln\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [Cpl1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

**6.195.2.13 Valeur()**

```
virtual double Courbe1D::Valeur (
    double x ) [pure virtual]
```

ramène la valeur

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_In\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

#### 6.195.2.14 Valeur\_Et\_der12()

```
virtual Courbe1D::ValDer2 Courbe1D::Valeur_Et_der12 (
    double x ) [pure virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_In\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

#### 6.195.2.15 Valeur\_Et\_derivee()

```
virtual Courbe1D::ValDer Courbe1D::Valeur_Et_derivee (
    double x ) [pure virtual]
```

ramène la valeur et la dérivée en paramètre

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_In\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

#### 6.195.2.16 Valeur\_Et\_derivee\_stricte()

```
virtual ValDerbool Courbe1D::Valeur_Et_derivee_stricte (
    double x ) [pure virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_In\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

#### 6.195.2.17 Valeur\_stricte()

```
virtual Valbool Courbe1D::Valeur_stricte (
    double x ) [pure virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémenté dans [Courbe\\_cos](#), [Courbe\\_expo2\\_n](#), [Courbe\\_expo\\_n](#), [Courbe\\_expoaff](#), [Courbe\\_expression\\_litterale\\_1D](#), [Courbe\\_expression\\_litterale\\_avec\\_derivees\\_1D](#), [Courbe\\_In\\_cosh](#), [Courbe\\_relax\\_expo](#), [Courbe\\_sin](#), [Courbe\\_un\\_moins\\_cos](#), [CourbePolyHermite1D](#), [CourbePolyLineaire1D](#), [CourbePolynomiale](#), [F1\\_plus\\_F2](#), [F1\\_rond\\_F2](#), [F\\_cycle\\_add](#), [F\\_cyclique](#), [F\\_union\\_1D](#), [Poly\\_Lagrange](#), [SixpodeCos3phi](#), [TangenteHyperbolique](#), et [TripodeCos3phi](#).

La documentation de cette classe a été générée à partir du fichier suivant :

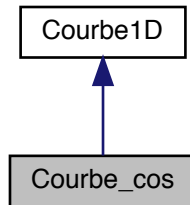
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe1D.cc

## 6.196 Référence de la classe Courbe\_cos

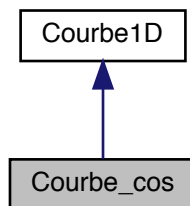
Classe permettant le calcul d'une fonction 1D de type :  $\text{ampli} \cdot \cos(\alpha \cdot x + \beta)$

```
#include <Courbe_cos.h>
```

Grphe d'héritage de Courbe\_cos:



Grphe de collaboration de Courbe\_cos:



## Fonctions membres publiques

- **Courbe\_cos** (string nom="")
- **Courbe\_cos** (const [Courbe\\_cos](#) &Co)
- **Courbe\_cos** (const [Courbe1D](#) &Co)
- void [Affiche](#) () const  
*affichage de la courbe*
- bool [Compleet\\_courbe](#) () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void [LectDonnParticulieres\\_courbes](#) (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void [Info\\_commande\\_Courbes1D](#) ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double [Valeur](#) (double x)  
*ramène la valeur*
- [Courbe1D::ValDer](#) [Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2](#) [Valeur\\_Et\\_der12](#) (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double [Der\\_sec](#) (double x)  
*ramène la dérivée seconde*

- [Courbe1D::Valbool Valeur\\_stricte](#) (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **ax**
- double **bx**
- bool **en\_degre**
- double **ampli**
- double **alph**
- double **bet**

## Membres hérités additionnels

### 6.196.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $\text{ampli} \cdot \cos(\alpha \cdot x + \beta)$

BUT: Classe permettant le calcul d'une fonction 1D de type :  $\text{ampli} \cdot \cos(\alpha \cdot x + \beta)$  avec un mini et/ou maxi éventuels ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.196.2 Documentation des fonctions membres

#### 6.196.2.1 Affiche()

```
void Courbe_cos::Affiche ( ) const [virtual]
```

affichage de la courbe

Implémente [Courbe1D](#).

#### 6.196.2.2 Complet\_courbe()

```
bool Courbe_cos::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon

Implémente [Courbe1D](#).



### 6.196.2.3 Der\_sec()

```
double Courbe_cos::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

### 6.196.2.4 Derivee()

```
double Courbe_cos::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

### 6.196.2.5 Ecriture\_base\_info()

```
void Courbe_cos::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.196.2.6 Info\_commande\_Courbes1D()

```
void Courbe_cos::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

### 6.196.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_cos::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

### 6.196.2.8 Lecture\_base\_info()

```
void Courbe_cos::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.196.2.9 SchemaXML\_Courbes1D()

```
void Courbe_cos::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

#### 6.196.2.10 Valeur()

```
double Courbe_cos::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

#### 6.196.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_cos::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

#### 6.196.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_cos::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

#### 6.196.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_cos::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

#### 6.196.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_cos::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

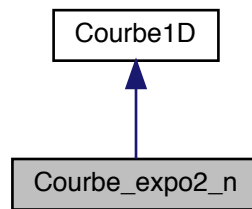
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_cos.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_cos.cc

## 6.197 Référence de la classe Courbe\_expo2\_n

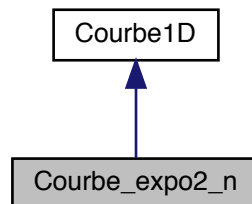
Classe permettant le calcul d'une fonction 1D de type :  $(\gamma + \alpha \cdot (x \cdot x)^\alpha)^n$

```
#include <Courbe_expo2_n.h>
```

Graphe d'héritage de Courbe\_expo2\_n:



Graphe de collaboration de Courbe\_expo2\_n:



## Fonctions membres publiques

- **Courbe\_expo2\_n** (string nom="")
- **Courbe\_expo2\_n** (const [Courbe\\_expo2\\_n](#) &Co)
- **Courbe\_expo2\_n** (const [Courbe1D](#) &Co)
- void [Affiche](#) () const  
*affichage de la courbe*
- bool [Compleet\\_courbe](#) () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void [LectDonnParticulieres\\_courbes](#) (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void [Info\\_commande\\_Courbes1D](#) ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double [Valeur](#) (double x)  
*ramène la valeur*
- [Courbe1D::ValDer](#) [Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2](#) [Valeur\\_Et\\_der12](#) (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double [Der\\_sec](#) (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool](#) [Valeur\\_stricte](#) (double x)

- ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
  - void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
  - void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
  - void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **alpha**
- double **xn**
- double **gamma**

## Membres hérités additionnels

### 6.197.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $(\text{gamma} + \text{alpha} * (x * x)^n)$

BUT: Classe permettant le calcul d'une fonction 1D 1D de type :  $(\text{gamma} + \text{alpha} * (x * x)^n)$  ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

06/04/2008

### 6.197.2 Documentation des fonctions membres

#### 6.197.2.1 Affiche()

```
void Courbe_expo2_n::Affiche ( ) const [virtual]
affichage de la courbe
Implémente Courbe1D.
```

#### 6.197.2.2 Complet\_courbe()

```
bool Courbe_expo2_n::Complet_courbe ( ) const [virtual]
vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon
Implémente Courbe1D.
```

### 6.197.2.3 Der\_sec()

```
double Courbe_expo2_n::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

### 6.197.2.4 Derivee()

```
double Courbe_expo2_n::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

### 6.197.2.5 Ecriture\_base\_info()

```
void Courbe_expo2_n::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.197.2.6 Info\_commande\_Courbes1D()

```
void Courbe_expo2_n::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

### 6.197.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_expo2_n::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

### 6.197.2.8 Lecture\_base\_info()

```
void Courbe_expo2_n::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.197.2.9 SchemaXML\_Courbes1D()

```
void Courbe_expo2_n::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

#### 6.197.2.10 Valeur()

```
double Courbe_expo2_n::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

#### 6.197.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_expo2_n::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

#### 6.197.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_expo2_n::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

#### 6.197.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_expo2_n::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

#### 6.197.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_expo2_n::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

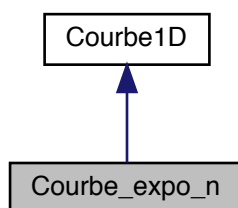
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expo2\_n.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expo2\_n.cc

## 6.198 Référence de la classe Courbe\_expo\_n

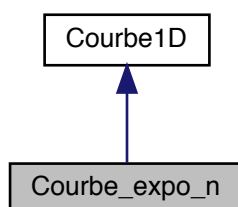
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = (\text{gamma} + \alpha * (x)^n)$

```
#include <Courbe_expo_n.h>
```

Grappe d'héritage de Courbe\_expo\_n:



Grappe de collaboration de Courbe\_expo\_n:



## Fonctions membres publiques

- **Courbe\_expo\_n** (string nom="")
- **Courbe\_expo\_n** (const [Courbe\\_expo\\_n](#) &Co)
- **Courbe\_expo\_n** (const [Courbe1D](#) &Co)
- void [Affiche](#) () const  
*affichage de la courbe*
- bool [Compleet\\_courbe](#) () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void [LectDonnParticulieres\\_courbes](#) (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void [Info\\_commande\\_Courbes1D](#) ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double [Valeur](#) (double x)  
*ramène la valeur*
- [Courbe1D::ValDer Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2 Valeur\\_Et\\_der12](#) (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double [Der\\_sec](#) (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool Valeur\\_stricte](#) (double x)

- ramène la valeur si dans le domaine strictement de définition si c'est inférieur au  $x_{\text{mini}}$ , ramène la valeur minimale possible de  $y$  si supérieur au  $x_{\text{maxi}}$ , ramène la valeur maximale possible de  $y$*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au  $x_{\text{mini}}$ , ramène la valeur minimale possible de  $y$  et  $Y'$  correspondant si supérieur au  $x_{\text{maxi}}$ , ramène la valeur maximale possible de  $y$  et  $Y'$  correspondant*
  - void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
  - void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
  - void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **alpha**
- double **xn**
- double **gamma**

## Membres hérités additionnels

### 6.198.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = (\text{gamma} + \text{alpha} * (x)^n)$

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x) = (\text{gamma} + \text{alpha} * (x)^n)$  ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.198.2 Documentation des fonctions membres

#### 6.198.2.1 Affiche()

```
void Courbe_expo_n::Affiche ( ) const [virtual]
affichage de la courbe
Implémente Courbe1D.
```

#### 6.198.2.2 Complet\_courbe()

```
bool Courbe_expo_n::Complet_courbe ( ) const [virtual]
vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon
Implémente Courbe1D.
```



### 6.198.2.3 Der\_sec()

```
double Courbe_expo_n::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

### 6.198.2.4 Derivee()

```
double Courbe_expo_n::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

### 6.198.2.5 Ecriture\_base\_info()

```
void Courbe_expo_n::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.198.2.6 Info\_commande\_Courbes1D()

```
void Courbe_expo_n::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

### 6.198.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_expo_n::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

### 6.198.2.8 Lecture\_base\_info()

```
void Courbe_expo_n::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.198.2.9 SchemaXML\_Courbes1D()

```
void Courbe_expo_n::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

#### 6.198.2.10 Valeur()

```
double Courbe_expo_n::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

#### 6.198.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_expo_n::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

#### 6.198.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_expo_n::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

#### 6.198.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_expo_n::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

#### 6.198.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_expo_n::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

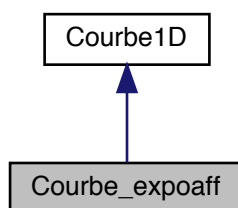
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expo\_n.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expo\_n.cc

## 6.199 Référence de la classe Courbe\_expoaff

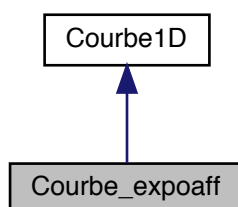
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{gamma} + \alpha * (|x|^n)$

```
#include <Courbe_expoaff.h>
```

Graphe d'héritage de Courbe\_expoaff:



Graphe de collaboration de Courbe\_expoaff:



## Fonctions membres publiques

- **Courbe\_expoaff** (string nom="")
- **Courbe\_expoaff** (const [Courbe\\_expoaff](#) &Co)
- **Courbe\_expoaff** (const [Courbe1D](#) &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Compleet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- [Courbe1D::ValDer](#) **Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2](#) **Valeur\_Et\_der12** (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double **Der\_sec** (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool](#) **Valeur\_stricte** (double x)

- ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
  - void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
  - void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
  - void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **alpha**
- double **xn**
- double **gamma**

## Membres hérités additionnels

### 6.199.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{gamma} + \text{alpha} * (|x|^n)$

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{gamma} + \text{alpha} * (|x|^n)$  ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.199.2 Documentation des fonctions membres

#### 6.199.2.1 Affiche()

```
void Courbe_expoaff::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

#### 6.199.2.2 Complet\_courbe()

```
bool Courbe_expoaff::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

### 6.199.2.3 Der\_sec()

```
double Courbe_expoaff::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

### 6.199.2.4 Derivee()

```
double Courbe_expoaff::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

### 6.199.2.5 Ecriture\_base\_info()

```
void Courbe_expoaff::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.199.2.6 Info\_commande\_Courbes1D()

```
void Courbe_expoaff::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

### 6.199.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_expoaff::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

### 6.199.2.8 Lecture\_base\_info()

```
void Courbe_expoaff::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.199.2.9 SchemaXML\_Courbes1D()

```
void Courbe_expoaff::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

#### 6.199.2.10 Valeur()

```
double Courbe_expoaff::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

#### 6.199.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_expoaff::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

#### 6.199.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_expoaff::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

#### 6.199.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_expoaff::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

#### 6.199.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_expoaff::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

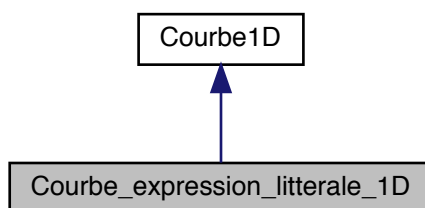
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expoaff.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expoaff.cc

## 6.200 Référence de la classe Courbe\_expression\_litterale\_1D

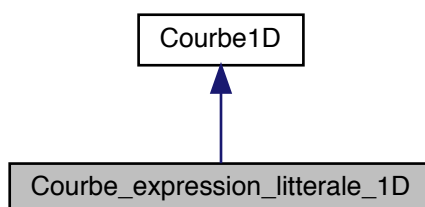
Classe permettant le calcul d'une fonction 1D de type : une expression littérale.

```
#include <Courbe_expression_litterale_1D.h>
```

Graphe d'héritage de Courbe\_expression\_litterale\_1D:



Graphe de collaboration de Courbe\_expression\_litterale\_1D:



## Fonctions membres publiques

- **Courbe\_expression\_litterale\_1D** (string nom="")
- **Courbe\_expression\_litterale\_1D** (const [Courbe\\_expression\\_litterale\\_1D](#) &Co)
- **Courbe\_expression\_litterale\_1D** (const [Courbe1D](#) &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Compleet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- [Courbe1D::ValDer](#) **Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2](#) **Valeur\_Et\_der12** (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double **Der\_sec** (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool](#) **Valeur\_stricte** (double x)

- ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
  - void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
  - void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
  - void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **ax**
- double **bx**
- string **expression\_fonction**
- mu::Parser **p**
- double **fVal**
- double **delta\_xSur\_x**

## Membres hérités additionnels

### 6.200.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type : une expression littérale.

BUT: Classe permettant le calcul d'une fonction 1D de type : une expression littérale, exploitée ensuite par le parser : muparser avec un mini et/ou maxi éventuels ainsi qu'un certain nombre d'information supplémentaires telles que dérivées. IMPORTANT: les dérivées sont calculées de manière numérique.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

07/04/2014

### 6.200.2 Documentation des fonctions membres

#### 6.200.2.1 Affiche()

```
void Courbe_expression_litterale_1D::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

#### 6.200.2.2 Complet\_courbe()

```
bool Courbe_expression_litterale_1D::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).



### 6.200.2.3 Der\_sec()

```
double Courbe_expression_litterale_1D::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

### 6.200.2.4 Derivee()

```
double Courbe_expression_litterale_1D::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

### 6.200.2.5 Ecriture\_base\_info()

```
void Courbe_expression_litterale_1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.200.2.6 Info\_commande\_Courbes1D()

```
void Courbe_expression_litterale_1D::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

### 6.200.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_expression_litterale_1D::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

### 6.200.2.8 Lecture\_base\_info()

```
void Courbe_expression_litterale_1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.200.2.9 SchemaXML\_Courbes1D()

```
void Courbe_expression_litterale_1D::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

#### 6.200.2.10 Valeur()

```
double Courbe_expression_litterale_1D::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

#### 6.200.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_expression_litterale_1D::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

#### 6.200.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_expression_litterale_1D::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

#### 6.200.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_expression_litterale_1D::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

#### 6.200.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_expression_litterale_1D::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expression\_litterale\_1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expression\_litterale\_1D.cc

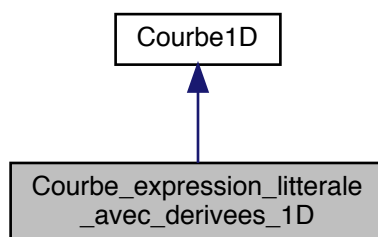
## 6.201 Référence de la classe

### **Courbe\_expression\_litterale\_avec\_derivees\_1D**

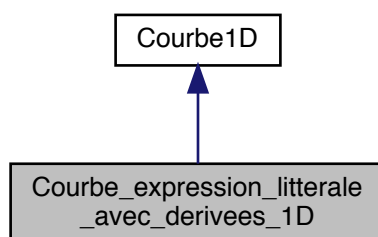
Classe permettant le calcul d'une fonction 1D et de sa dérivée de type : une expression littérale.

```
#include <Courbe_expression_litterale_avec_derivees_1D.h>
```

Graphe d'héritage de Courbe\_expression\_litterale\_avec\_derivees\_1D:



Graphe de collaboration de Courbe\_expression\_litterale\_avec\_derivees\_1D:



## Fonctions membres publiques

- `Courbe_expression_litterale_avec_derivees_1D` (string nom="")
- `Courbe_expression_litterale_avec_derivees_1D` (const `Courbe_expression_litterale_avec_derivees_1D` &Co)
- `Courbe_expression_litterale_avec_derivees_1D` (const `Courbe1D` &Co)
- void `Affiche` () const  
*affichage de la courbe*
- bool `Complet_courbe` () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void `LectDonnParticulieres_courbes` (const string &nom, `UtilLecture` \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void `Info_commande_Courbes1D` (`UtilLecture` &entreePrinc)  
*def info fichier de commande*
- double `Valeur` (double x)  
*ramène la valeur*
- `Courbe1D::ValDer Valeur_Et_derivee` (double x)  
*ramène la valeur et la dérivée en paramètre*
- double `Derivee` (double x)  
*ramène la dérivée*
- `Courbe1D::ValDer2 Valeur_Et_der12` (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*

- double [Der\\_sec](#) (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool Valeur\\_stricte](#) (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **ax**
- double **bx**
- string **expression\_fonction**
- mu::Parser **p**
- double **fVal**
- string **expression\_der\_fonct**
- mu::Parser **der\_p**
- double **fVal\_der**
- string **expression\_der2\_fonct**
- mu::Parser **der2\_p**
- double **fVal\_der2**

## Membres hérités additionnels

### 6.201.1 Description détaillée

Classe permettant le calcul d'une fonction 1D et de sa dérivée de type : une expression littérale.

BUT: Classe permettant le calcul d'une fonction 1D de type : une expression littérale, exploitée ensuite par le parser : muparser avec un mini et/ou maxi éventuels ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

Auteur

Gérard Rio

Version

1.0

Date

07/04/2014

### 6.201.2 Documentation des fonctions membres

#### 6.201.2.1 Affiche()

```
void Courbe_expression_litterale_avec_derivees_1D::Affiche ( ) const [virtual]
affichage de la courbe
Implémente Courbe1D.
```

### 6.201.2.2 Complet\_courbe()

```
bool Courbe_expression_litterale_avec_derivees_1D::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon

Implémente [Courbe1D](#).

### 6.201.2.3 Der\_sec()

```
double Courbe_expression_litterale_avec_derivees_1D::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

### 6.201.2.4 Derivee()

```
double Courbe_expression_litterale_avec_derivees_1D::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

### 6.201.2.5 Ecriture\_base\_info()

```
void Courbe_expression_litterale_avec_derivees_1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.201.2.6 Info\_commande\_Courbes1D()

```
void Courbe_expression_litterale_avec_derivees_1D::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

### 6.201.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_expression_litterale_avec_derivees_1D::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

### 6.201.2.8 Lecture\_base\_info()

```
void Courbe_expression_litterale_avec_derivees_1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

---- lecture écriture de restart ---- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.201.2.9 SchemaXML\_Courbes1D()

```
void Courbe_expression_litterale_avec_derivees_1D::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

### 6.201.2.10 Valeur()

```
double Courbe_expression_litterale_avec_derivees_1D::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

### 6.201.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_expression_litterale_avec_derivees_1D::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

### 6.201.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_expression_litterale_avec_derivees_1D::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

### 6.201.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_expression_litterale_avec_derivees_1D::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

### 6.201.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_expression_litterale_avec_derivees_1D::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

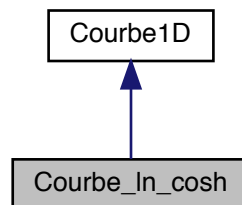
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expression\_litterale\_avec\_↔  
derivees\_1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_expression\_litterale\_avec\_↔  
derivees\_1D.cc

## 6.202 Référence de la classe Courbe\_In\_cosh

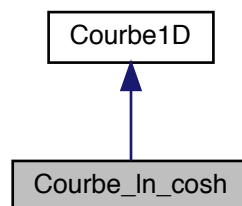
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = aI*(be + ga*x)^n * \ln (\cosh(de*(he+ee*x)))+ ke*(ge+re*x)$

```
#include <Courbe_In_cosh.h>
```

Graphe d'héritage de Courbe\_In\_cosh:



Graphe de collaboration de Courbe\_In\_cosh:



### Fonctions membres publiques

- **Courbe\_In\_cosh** (string nom="")
- **Courbe\_In\_cosh** (const [Courbe\\_In\\_cosh](#) &Co)
- **Courbe\_In\_cosh** (const [Courbe1D](#) &Co)
- void [Affiche](#) () const  
*affichage de la courbe*
- bool [Compleet\\_courbe](#) () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void [LectDonnParticulieres\\_courbes](#) (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void [Info\\_commande\\_Courbes1D](#) ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double [Valeur](#) (double x)  
*ramène la valeur*
- [Courbe1D::ValDer](#) [Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)  
*ramène la dérivée*

- `Courbe1D::ValDer2 Valeur_Et_der12` (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- `double Der_sec` (double x)  
*ramène la dérivée seconde*
- `Courbe1D::Valbool Valeur_stricte` (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- `Courbe1D::ValDerbool Valeur_Et_derivee_stricte` (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
- `void Lecture_base_info` (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- `void Ecriture_base_info` (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- `void SchemaXML_Courbes1D` (ofstream &sort, const Enum\_IO\_XML enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- `double al`
- `double be`
- `double ga`
- `double de`
- `double he`
- `double ee`
- `double ke`
- `double ge`
- `double re`
- `double se`
- `int n`
- `double ax`
- `double bx`

## Membres hérités additionnels

### 6.202.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = al*(be + ga*x)^n * \ln (\cosh(de*(he+ee*x)))+ ke*(ge+re*x)$

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x) = al*(be + ga*x)^n * \ln (\cosh(de*(he+ee*x)))+ ke*(ge+re*x)$  avec un mini et/ou maxi éventuels ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.202.2 Documentation des fonctions membres



### 6.202.2.1 Affiche()

```
void Courbe_In_cosh::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

### 6.202.2.2 Complet\_courbe()

```
bool Courbe_In_cosh::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

### 6.202.2.3 Der\_sec()

```
double Courbe_In_cosh::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

### 6.202.2.4 Derivee()

```
double Courbe_In_cosh::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

### 6.202.2.5 Ecriture\_base\_info()

```
void Courbe_In_cosh::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.202.2.6 Info\_commande\_Courbes1D()

```
void Courbe_In_cosh::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

### 6.202.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_In_cosh::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.  
Implémente [Courbe1D](#).

**6.202.2.8 Lecture\_base\_info()**

```
void Courbe_ln_cosh::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

---- lecture écriture de restart ---- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.202.2.9 SchemaXML\_Courbes1D()**

```
void Courbe_ln_cosh::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

**6.202.2.10 Valeur()**

```
double Courbe_ln_cosh::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

**6.202.2.11 Valeur\_Et\_der12()**

```
Courbe1D::ValDer2 Courbe_ln_cosh::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

**6.202.2.12 Valeur\_Et\_derivee()**

```
Courbe1D::ValDer Courbe_ln_cosh::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

**6.202.2.13 Valeur\_Et\_derivee\_stricte()**

```
Courbe1D::ValDerbool Courbe_ln_cosh::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

**6.202.2.14 Valeur\_stricte()**

```
Courbe1D::Valbool Courbe_ln_cosh::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

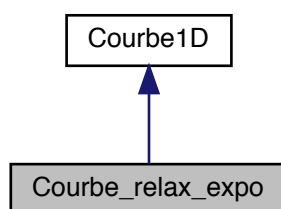
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_In\_cosh.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_In\_cosh.cc

## 6.203 Référence de la classe Courbe\_relax\_expo

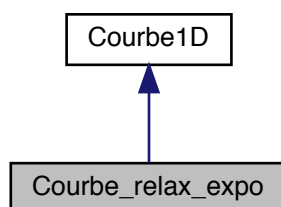
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = (a-b)*\exp(-c*x) + b$ .

```
#include <Courbe_relax_expo.h>
```

Graphe d'héritage de Courbe\_relax\_expo:



Graphe de collaboration de Courbe\_relax\_expo:



### Fonctions membres publiques

- **Courbe\_relax\_expo** (string nom="")
- **Courbe\_relax\_expo** (const [Courbe\\_relax\\_expo](#) &Co)
- **Courbe\_relax\_expo** (const [Courbe1D](#) &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Compleet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*

- [Courbe1D::ValDer Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2 Valeur\\_Et\\_der12](#) (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double [Der\\_sec](#) (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool Valeur\\_stricte](#) (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **xa**
- double **xb**
- double **xc**
- double **xmin**
- double **xmax**

## Membres hérités additionnels

### 6.203.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = (a-b)*\exp(-c*x) + b$ .

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x) = (a-b)*\exp(-c*x) + b$  avec un mini et/ou maxi éventuels ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.203.2 Documentation des fonctions membres

#### 6.203.2.1 Affiche()

```
void Courbe_relax_expo::Affiche ( ) const [virtual]
```

affichage de la courbe

Implémente [Courbe1D](#).

### 6.203.2.2 Complet\_courbe()

```
bool Courbe_relax_expo::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

### 6.203.2.3 Der\_sec()

```
double Courbe_relax_expo::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

### 6.203.2.4 Derivee()

```
double Courbe_relax_expo::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

### 6.203.2.5 Ecriture\_base\_info()

```
void Courbe_relax_expo::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.203.2.6 Info\_commande\_Courbes1D()

```
void Courbe_relax_expo::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

### 6.203.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_relax_expo::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.  
Implémente [Courbe1D](#).

### 6.203.2.8 Lecture\_base\_info()

```
void Courbe_relax_expo::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

---- lecture écriture de restart ---- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.203.2.9 SchemaXML\_Courbes1D()

```
void Courbe_relax_expo::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

### 6.203.2.10 Valeur()

```
double Courbe_relax_expo::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

### 6.203.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_relax_expo::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

### 6.203.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_relax_expo::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

### 6.203.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_relax_expo::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

### 6.203.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_relax_expo::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

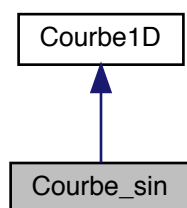
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_relax\_expo.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_relax\_expo.cc

## 6.204 Référence de la classe Courbe\_sin

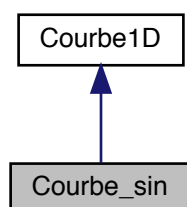
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{ampli} * \sin(\text{alpha} * x + \text{beta})$

```
#include <Courbe_sin.h>
```

Graphe d'héritage de Courbe\_sin:



Graphe de collaboration de Courbe\_sin:



## Fonctions membres publiques

- **Courbe\_sin** (string nom="")
- **Courbe\_sin** (const **Courbe\_sin** &Co)
- **Courbe\_sin** (const **Courbe1D** &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Compleet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, **UtilLecture** \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** (**UtilLecture** &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- **Courbe1D::ValDer Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*
- **Courbe1D::ValDer2 Valeur\_Et\_der12** (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double **Der\_sec** (double x)  
*ramène la dérivée seconde*
- **Courbe1D::Valbool Valeur\_stricte** (double x)

- ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
  - void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
  - void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
  - void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **ax**
- double **bx**
- bool **en\_degre**
- double **ampli**
- double **alph**
- double **bet**

## Membres hérités additionnels

### 6.204.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{ampli} \cdot \sin(\text{alpha} \cdot x + \text{beta})$

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{ampli} \cdot \sin(\text{alpha} \cdot x + \text{beta})$  avec un mini et/ou maxi éventuels ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.204.2 Documentation des fonctions membres

#### 6.204.2.1 Affiche()

```
void Courbe_sin::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

#### 6.204.2.2 Complet\_courbe()

```
bool Courbe_sin::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).



**6.204.2.3 Der\_sec()**

```
double Courbe_sin::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

**6.204.2.4 Derivee()**

```
double Courbe_sin::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

**6.204.2.5 Ecriture\_base\_info()**

```
void Courbe_sin::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.204.2.6 Info\_commande\_Courbes1D()**

```
void Courbe_sin::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

**6.204.2.7 LectDonnParticulieres\_courbes()**

```
void Courbe_sin::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

**6.204.2.8 Lecture\_base\_info()**

```
void Courbe_sin::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.204.2.9 SchemaXML\_Courbes1D()**

```
void Courbe_sin::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

#### 6.204.2.10 Valeur()

```
double Courbe_sin::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

#### 6.204.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_sin::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

#### 6.204.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_sin::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

#### 6.204.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_sin::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

#### 6.204.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_sin::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

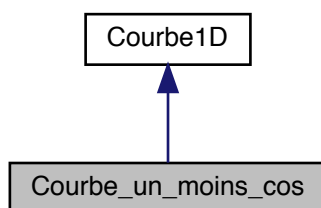
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_sin.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_sin.cc

## 6.205 Référence de la classe Courbe\_un\_moins\_cos

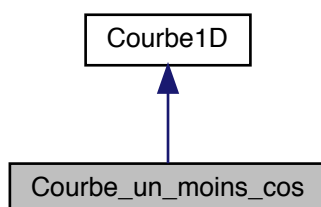
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = c*(1-\cos((x-a)\pi/(b-a)))$

```
#include <Courbe_un_moins_cos.h>
```

Graphe d'héritage de Courbe\_un\_moins\_cos:



Graphe de collaboration de Courbe\_un\_moins\_cos:



## Fonctions membres publiques

- **Courbe\_un\_moins\_cos** (string nom="")
- **Courbe\_un\_moins\_cos** (const [Courbe\\_un\\_moins\\_cos](#) &Co)
- **Courbe\_un\_moins\_cos** (const [Courbe1D](#) &Co)
- void [Affiche](#) () const  
*affichage de la courbe*
- bool [Compleet\\_courbe](#) () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void [LectDonnParticulieres\\_courbes](#) (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void [Info\\_commande\\_Courbes1D](#) ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double [Valeur](#) (double x)  
*ramène la valeur*
- [Courbe1D::ValDer Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2 Valeur\\_Et\\_der12](#) (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double [Der\\_sec](#) (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool Valeur\\_stricte](#) (double x)

- ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
  - void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
  - void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
  - void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- double **ax**
- double **bx**
- double **cx**

## Membres hérités additionnels

### 6.205.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = c*(1-\cos((x-a)Pi/(b-a)))$

BUT: Classe permettant le calcul d'une fonction 1D 1D de type :  $c*(1-\cos((x-a)Pi/(b-a)))$  entre a et b, en dessous de a, -> 0 et au dessus de b -> c ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.205.2 Documentation des fonctions membres

#### 6.205.2.1 Affiche()

```
void Courbe_un_moins_cos::Affiche ( ) const [virtual]
affichage de la courbe
Implémente Courbe1D.
```

#### 6.205.2.2 Complet\_courbe()

```
bool Courbe_un_moins_cos::Complet_courbe ( ) const [virtual]
vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon
Implémente Courbe1D.
```

### 6.205.2.3 Der\_sec()

```
double Courbe_un_moins_cos::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

### 6.205.2.4 Derivee()

```
double Courbe_un_moins_cos::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

### 6.205.2.5 Ecriture\_base\_info()

```
void Courbe_un_moins_cos::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.205.2.6 Info\_commande\_Courbes1D()

```
void Courbe_un_moins_cos::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

### 6.205.2.7 LectDonnParticulieres\_courbes()

```
void Courbe_un_moins_cos::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.  
Implémente [Courbe1D](#).

### 6.205.2.8 Lecture\_base\_info()

```
void Courbe_un_moins_cos::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.205.2.9 SchemaXML\_Courbes1D()

```
void Courbe_un_moins_cos::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

#### 6.205.2.10 Valeur()

```
double Courbe_un_moins_cos::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

#### 6.205.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 Courbe_un_moins_cos::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

#### 6.205.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer Courbe_un_moins_cos::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

#### 6.205.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool Courbe_un_moins_cos::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

#### 6.205.2.14 Valeur\_stricte()

```
Courbe1D::Valbool Courbe_un_moins_cos::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

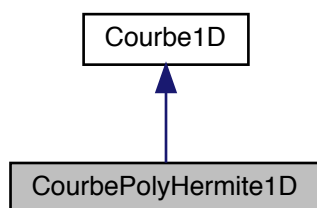
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_un\_moins\_cos.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe\_un\_moins\_cos.cc

## 6.206 Référence de la classe CourbePolyHermite1D

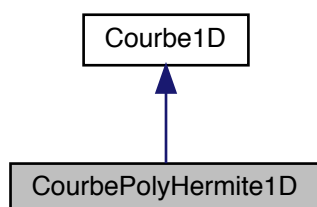
Classe permettant le calcul d'une fonction 1D polynomiale 1D par morceau, avec continuité de la dérivée via une interpolation de type hermite.

```
#include <CourbePolyHermitel1D.h>
```

Graphe d'héritage de CourbePolyHermite1D:



Graphe de collaboration de CourbePolyHermite1D:



## Fonctions membres publiques

- **CourbePolyHermite1D** (string nom="")
- **CourbePolyHermite1D** (Tableau< Coordonnee3 > &pt, string nom="")
- **CourbePolyHermite1D** (const CourbePolyHermite1D &Co)
- **CourbePolyHermite1D** (const Courbe1D &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Compleet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, UtilLecture \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** (UtilLecture &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- **Courbe1D::ValDer Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*
- **Courbe1D::ValDer2 Valeur\_Et\_der12** (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double **Der\_sec** (double x)  
*ramène la dérivée seconde*

- `Courbe1D::Valbool Valeur_stricte` (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- `Courbe1D::ValDerbool Valeur_Et_derivee_stricte` (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
- void `Change_tabPoints` (`Tableau< Coordonnee3 > &pt`)
- void `Lecture_base_info` (`ifstream &ent`, `const int cas`)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void `Ecriture_base_info` (`ofstream &sort`, `const int cas`)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- virtual void `SchemaXML_Courbes1D` (`ofstream &sort`, `const Enum_IO_XML enu`)  
*sortie du schemaXML: en fonction de enu*
- int `NombrePoint` ()
- const `Coordonnee3 & Pt_nbi` (int i)

## Fonctions membres protégées

- `CourbePolyHermite1D` (string nom, `EnumCourbe1D` typ)

## Attributs protégés

- `Tableau< Coordonnee3 > points`
- int `indice_precedant`

## Membres hérités additionnels

### 6.206.1 Description détaillée

Classe permettant le calcul d'une fonction 1D polynomiale 1D par morceau, avec continuité de la dérivée via une interpolation de type hermite.

BUT: Classe permettant le calcul d'une fonction 1D poly- nomiale 1D par morceau, avec continuité de la dérivée via une interpolation de type hermite, ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

Auteur

Gérard Rio

Version

1.0

Date

19/01/2001

### 6.206.2 Documentation des fonctions membres

#### 6.206.2.1 Affiche()

```
void CourbePolyHermite1D::Affiche ( ) const [virtual]
affichage de la courbe
Implémente Courbe1D.
```



### 6.206.2.2 Complet\_courbe()

```
bool CourbePolyHermite1D::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

### 6.206.2.3 Der\_sec()

```
double CourbePolyHermite1D::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

### 6.206.2.4 Derivee()

```
double CourbePolyHermite1D::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

### 6.206.2.5 Ecriture\_base\_info()

```
void CourbePolyHermite1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.206.2.6 Info\_commande\_Courbes1D()

```
void CourbePolyHermite1D::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

### 6.206.2.7 LectDonnParticulieres\_courbes()

```
void CourbePolyHermite1D::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.  
Implémente [Courbe1D](#).

### 6.206.2.8 Lecture\_base\_info()

```
void CourbePolyHermite1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

---- lecture écriture de restart ---- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.206.2.9 SchemaXML\_Courbes1D()

```
void CourbePolyHermitelD::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

### 6.206.2.10 Valeur()

```
double CourbePolyHermitelD::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

### 6.206.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 CourbePolyHermitelD::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

### 6.206.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer CourbePolyHermitelD::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

### 6.206.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool CourbePolyHermitelD::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

### 6.206.2.14 Valeur\_stricte()

```
Courbe1D::Valbool CourbePolyHermitelD::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

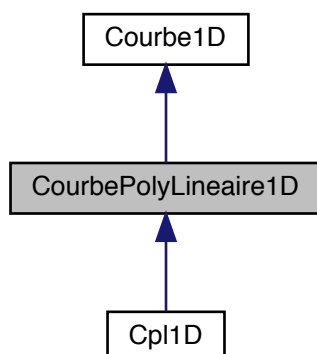
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyHermite1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyHermite1D.cc

## 6.207 Référence de la classe CourbePolyLineaire1D

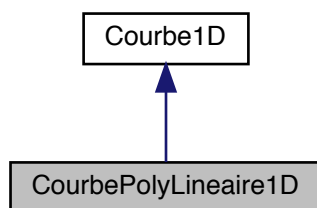
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \text{poly-linéaire}$  c-a-d linéaire par morceaux limités par 2 points.

```
#include <CourbePolyLineaire1D.h>
```

Graphe d'héritage de CourbePolyLineaire1D:



Graphe de collaboration de CourbePolyLineaire1D:



## Fonctions membres publiques

- **CourbePolyLineaire1D** (string nom="")
- **CourbePolyLineaire1D** (Tableau< Coordonnee2 > &pt, string nom="")
- **CourbePolyLineaire1D** (const CourbePolyLineaire1D &Co)
- **CourbePolyLineaire1D** (const Courbe1D &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Compleet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, UtilLecture \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** (UtilLecture &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- **Courbe1D::ValDer Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)

- *ramène la dérivée*
- `Courbe1D::ValDer2 Valeur_Et_der12` (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double `Der_sec` (double x)  
*ramène la dérivée seconde*
- `Courbe1D::Valbool Valeur_stricte` (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- `Courbe1D::ValDerbool Valeur_Et_derivee_stricte` (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void `Change_tabPoints` (`Tableau< Coordonnee2 > &pt`)
- void `Lecture_base_info` (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void `Ecriture_base_info` (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- virtual void `SchemaXML_Courbes1D` (ofstream &sort, const `Enum_IO_XML` enu)  
*sortie du schemaXML: en fonction de enu*
- int `NombrePoint` ()
- const `Coordonnee2 & Pt_nbi` (int i)

## Fonctions membres protégées

- `CourbePolyLineaire1D` (string nom, `EnumCourbe1D` typ)

## Attributs protégés

- `Tableau< Coordonnee2 > points`
- double `der_init`
- double `der_finale`
- int `indice_precedant`

## Membres hérités additionnels

### 6.207.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x)$  = poly-linéaire c-a-d linéaire par morceaux limités par 2 points.

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x)$  = poly-linéaire c-a-d linéaire par morceaux limités par 2 points ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.207.2 Documentation des fonctions membres

### 6.207.2.1 Affiche()

```
void CourbePolyLineaire1D::Affiche ( ) const [virtual]
```

affichage de la courbe

Implémente [Courbe1D](#).

### 6.207.2.2 Complet\_courbe()

```
bool CourbePolyLineaire1D::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon

Implémente [Courbe1D](#).

### 6.207.2.3 Der\_sec()

```
double CourbePolyLineaire1D::Der_sec (
```

```
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

### 6.207.2.4 Derivee()

```
double CourbePolyLineaire1D::Derivee (
```

```
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

### 6.207.2.5 Ecriture\_base\_info()

```
void CourbePolyLineaire1D::Ecriture_base_info (
```

```
    ofstream & sort,
```

```
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.207.2.6 Info\_commande\_Courbes1D()

```
void CourbePolyLineaire1D::Info_commande_Courbes1D (
```

```
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

Réimplémentée dans [Cpl1D](#).

### 6.207.2.7 LectDonnParticulieres\_courbes()

```
void CourbePolyLineaire1D::LectDonnParticulieres_courbes (
```

```
    const string & nom,
```

```
    UtilLecture * ) [virtual]
```

Lecture des données de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la méthode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

Réimplémentée dans [Cpl1D](#).

**6.207.2.8 Lecture\_base\_info()**

```
void CourbePolyLineaire1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.207.2.9 SchemaXML\_Courbes1D()**

```
void CourbePolyLineaire1D::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

Réimplémentée dans [Cpl1D](#).

**6.207.2.10 Valeur()**

```
double CourbePolyLineaire1D::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

**6.207.2.11 Valeur\_Et\_der12()**

```
Courbe1D::ValDer2 CourbePolyLineaire1D::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

**6.207.2.12 Valeur\_Et\_derivee()**

```
Courbe1D::ValDer CourbePolyLineaire1D::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

**6.207.2.13 Valeur\_Et\_derivee\_stricte()**

```
Courbe1D::ValDerbool CourbePolyLineaire1D::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

**6.207.2.14 Valeur\_stricte()**

```
Courbe1D::Valbool CourbePolyLineaire1D::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

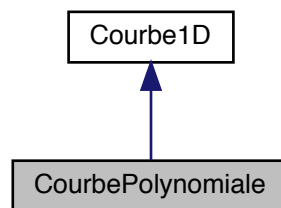
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyLineaire1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyLineaire.cc

## 6.208 Référence de la classe CourbePolynomiale

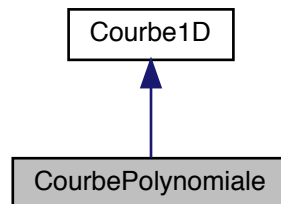
Classe permettant le calcul d'un polynôme 1D de type :  $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots$

```
#include <CourbePolynomiale.h>
```

Graphe d'héritage de CourbePolynomiale:



Graphe de collaboration de CourbePolynomiale:



### Fonctions membres publiques

- **CourbePolynomiale** (string nom="")
- **CourbePolynomiale** (const [CourbePolynomiale](#) &Co)
- **CourbePolynomiale** (const [Courbe1D](#) &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Completer\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)

- ramène la valeur*
- `Courbe1D::ValDer Valeur_Et_derivee` (double x)  
*ramène la valeur et la dérivée en paramètre*
- double `Derivee` (double x)  
*ramène la dérivée*
- `Courbe1D::ValDer2 Valeur_Et_der12` (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double `Der_sec` (double x)  
*ramène la dérivée seconde*
- `Courbe1D::Valbool Valeur_stricte` (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- `Courbe1D::ValDerbool Valeur_Et_derivee_stricte` (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void `Lecture_base_info` (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void `Ecriture_base_info` (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- void `SchemaXML_Courbes1D` (ofstream &sort, const Enum\_IO\_XML enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- `Tableau` < double > `coef`

## Membres hérités additionnels

### 6.208.1 Description détaillée

Classe permettant le calcul d'un polynôme 1D de type :  $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots$

BUT: Classe permettant le calcul d'un polynôme 1D de type :  $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots$  ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.208.2 Documentation des fonctions membres

#### 6.208.2.1 Affiche()

```
void CourbePolynomiale::Affiche ( ) const [virtual]
```

affichage de la courbe

Implémente `Courbe1D`.



### 6.208.2.2 Complet\_courbe()

```
bool CourbePolynomiale::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

### 6.208.2.3 Der\_sec()

```
double CourbePolynomiale::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

### 6.208.2.4 Derivee()

```
double CourbePolynomiale::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

### 6.208.2.5 Ecriture\_base\_info()

```
void CourbePolynomiale::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.208.2.6 Info\_commande\_Courbes1D()

```
void CourbePolynomiale::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

### 6.208.2.7 LectDonnParticulieres\_courbes()

```
void CourbePolynomiale::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.  
Implémente [Courbe1D](#).

### 6.208.2.8 Lecture\_base\_info()

```
void CourbePolynomiale::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

---- lecture écriture de restart ---- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.208.2.9 SchemaXML\_Courbes1D()

```
void CourbePolynomiale::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

### 6.208.2.10 Valeur()

```
double CourbePolynomiale::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

### 6.208.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 CourbePolynomiale::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

### 6.208.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer CourbePolynomiale::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

### 6.208.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool CourbePolynomiale::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

### 6.208.2.14 Valeur\_stricte()

```
Courbe1D::Valbool CourbePolynomiale::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

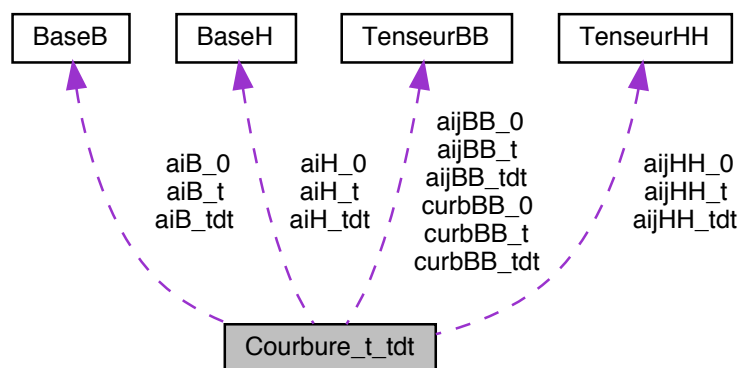
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolynomiale.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolynomiale.cc

## 6.209 Référence de la classe Courbure\_t\_tdt

CLASSE CONTENEUR : cas des grandeurs relatives à la courbure, grandeurs à 0, t et tdt les données sont publiques.

```
#include <Met_Sfel_struct_donnees.h>
```

Graphes de collaboration de Courbure\_t\_tdt:



## Fonctions membres publiques

- **Courbure\_t\_tdt** (`BaseB *aaiB_0`, `BaseH *aaiH_0`, `BaseB *aaiB_t`, `BaseH *aaiH_t`, `BaseB *aaiB_tdt`, `BaseH *aaiH_tdt`, `TenseurBB *aaijBB_0`, `TenseurHH *aaijHH_0`, `TenseurBB *aaijBB_t`, `TenseurHH *aaijHH_t`, `TenseurBB *aaijBB_tdt`, `TenseurHH *aaijHH_tdt`, `TenseurBB *gcurbBB_t`, `TenseurBB *gcurbBB_tdt`, `TenseurBB *gcurbBB_0`, `double *gjacobien`, `double *gajacobien_t`, `double *gajacobien_0`)
- **Courbure\_t\_tdt** (`const Courbure_t_tdt &ex`)
- **Courbure\_t\_tdt & operator=** (`const Courbure_t_tdt &ex`)
- **void Mise\_a\_jour\_grandeur** (`BaseB *aaiB_0`, `BaseH *aaiH_0`, `BaseB *aaiB_t`, `BaseH *aaiH_t`, `BaseB *aaiB_tdt`, `BaseH *aaiH_tdt`, `TenseurBB *aaijBB_0`, `TenseurHH *aaijHH_0`, `TenseurBB *aaijBB_t`, `TenseurHH *aaijHH_t`, `TenseurBB *aaijBB_tdt`, `TenseurHH *aaijHH_tdt`, `TenseurBB *gcurbBB_t`, `TenseurBB *gcurbBB_tdt`, `TenseurBB *gcurbBB_0`, `double *gjacobien`, `double *gajacobien_t`, `double *gajacobien_0`)
- **void Recup\_grandeur\_0\_t** (`BaseB &aaiB_0`, `BaseH &aaiH_0`, `BaseB &aaiB_t`, `BaseH &aaiH_t`, `TenseurBB &aaijBB_0`, `TenseurHH &aaijHH_0`, `TenseurBB &aaijBB_t`, `TenseurHH &aaijHH_t`, `TenseurBB &`, `double &gajacobien_0`, `double &gajacobien_t`)

## Attributs publics

- `BaseB * aiB_0`
- `BaseH * aiH_0`
- `BaseB * aiB_t`
- `BaseH * aiH_t`
- `BaseB * aiB_tdt`
- `BaseH * aiH_tdt`
- `TenseurBB * aijBB_0`
- `TenseurHH * aijHH_0`
- `TenseurBB * aijBB_t`
- `TenseurHH * aijHH_t`
- `TenseurBB * aijBB_tdt`
- `TenseurHH * aijHH_tdt`
- `TenseurBB * curbBB_t`
- `TenseurBB * curbBB_tdt`
- `TenseurBB * curbBB_0`
- `double * ajacobien_tdt`
- `double * ajacobien_t`
- `double * ajacobien_0`

### 6.209.1 Description détaillée

CLASSE CONTENEUR : cas des grandeurs relatives à la courbure, grandeurs à 0, t et tdt les données sont publiques.

La documentation de cette classe a été générée à partir du fichier suivant :

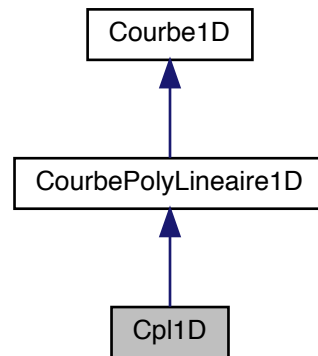
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met\_Sfe1\_struct\_↔  
donnees.h

### 6.210 Référence de la classe Cpl1D

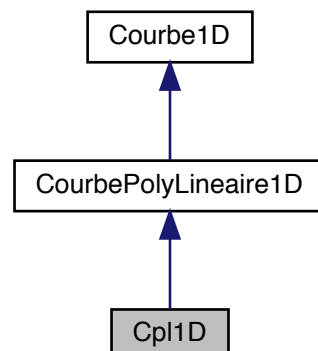
Classe permettant le calcul d'une fonction 1D de type : poly-linéaire 1D simplifiée, qui hérite de la fonction poly-linéaire.

```
#include <CourbePolyLineaire1D_simpli.h>
```

Graphe d'héritage de Cpl1D:



Graphe de collaboration de Cpl1D:



## Fonctions membres publiques

- **Cpl1D** (string nom="")
- **Cpl1D** (const [Cpl1D](#) &Co)
- **Cpl1D** (const [Courbe1D](#) &Co)
- void [LectDonnParticulieres\\_courbes](#) (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void [Info\\_commande\\_Courbes1D](#) ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Membres hérités additionnels

### 6.210.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type : poly-linéaire 1D simplifiée, qui hérite de la fonction poly-linéaire.

BUT: Cas d'une définition très simplifiée d'une fonction 1D poly-linéaire 1D qui hérite de la fonction poly-linéaire générique. Permet ainsi de minimiser la taille des infos à lire.

Auteur

Gérard Rio

Version

1.0

Date

03/04/2004

### 6.210.2 Documentation des fonctions membres

#### 6.210.2.1 Info\_commande\_Courbes1D()

```
void Cpl1D::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
def info fichier de commande
Réimplémentée à partir de CourbePolyLineaire1D.
```

#### 6.210.2.2 LectDonnParticulieres\_courbes()

```
void Cpl1D::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Réimplémentée à partir de [CourbePolyLineaire1D](#).

#### 6.210.2.3 SchemaXML\_Courbes1D()

```
void Cpl1D::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum\_IO\_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

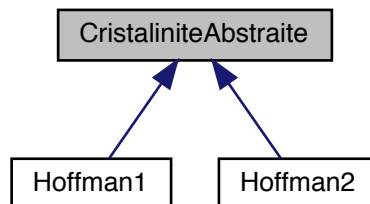
Réimplémentée à partir de [CourbePolyLineaire1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

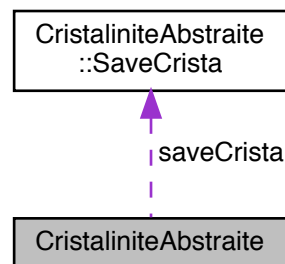
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyLineaire1D\_simpli.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/CourbePolyLineaire1D\_simpli.cc

## 6.211 Référence de la classe CristaliniteAbstraite

Graphe d'héritage de CristaliniteAbstraite:



Graphe de collaboration de CristaliniteAbstraite:



### Classes

- class [SaveCrista](#)

### Fonctions membres publiques

- **CristaliniteAbstraite** ([Enum\\_crista](#) id\_crista)
- **CristaliniteAbstraite** (char \*nom)
- **CristaliniteAbstraite** (const [CristaliniteAbstraite](#) &a)
- virtual [SaveCrista](#) \* **New\_et\_Initialise** ()
- virtual [SaveCrista](#) \* **New\_et\_Initialise** (const [SaveCrista](#) &)
- virtual void **AfficheDataSpecif** (ofstream &, [SaveCrista](#) \*) const
- virtual void **Affiche** () const =0
- virtual void **LectureDonneesLoiCrista** ([UtilLecture](#) \*entreePrinc, [LesCourbes1D](#) &, [LesFonctions\\_nD](#) &)=0
- virtual void **Info\_commande\_LoisCrista** ([UtilLecture](#) &entreePrinc)=0
- virtual double **fct\_KT** (const double &P, const double &T) const =0
- virtual double **Cristalinite** (const double &P\_t, const double &T\_t, [CristaliniteAbstraite::SaveCrista](#) \*saveTP, const double &P, const double &T, [Enum\\_dure](#) temps)=0

- virtual double **Cristalinite** ([CristaliniteAbstraite::SaveCrista](#) \*saveTP, const double &P, const double &T)=0
- virtual void **Lecture\_don\_base\_info** (ifstream &, const int, [LesCourbes1D](#) &, [LesFonctions\\_nD](#) &)=0
- virtual void **Ecriture\_don\_base\_info** (ofstream &, const int) const =0
- [Enum\\_crista](#) **Type\_Crista** () const

### Fonctions membres publiques statiques

- static [CristaliniteAbstraite](#) \* **New\_Cristalinite** ([Enum\\_crista](#) typeCrista)
- static [CristaliniteAbstraite](#) \* **New\_Cristalinite** (const [CristaliniteAbstraite](#) &Co)

### Attributs publics

- [SaveCrista](#) \* **saveCrista**

### Attributs protégés

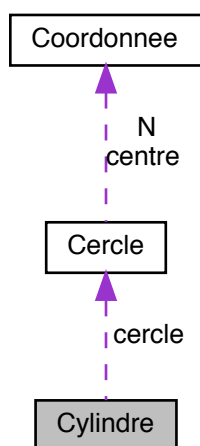
- [Enum\\_crista](#) **id\_crista**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Cristalinite↔  
Abstraite.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Cristalinite↔  
Abstraite.cc

## 6.212 Référence de la classe Cylindre

Graphe de collaboration de Cylindre:



### Fonctions membres publiques

- **Cylindre** (const [Cercle](#) &B)
- **Cylindre** (int dim)
- **Cylindre** (const [Cylindre](#) &a)
- [Cylindre](#) & **operator=** (const [Cylindre](#) &P)
- const [Cercle](#) & **CercleCylindre** () const
- void **Change\_cercle** (const [Cercle](#) &B)

- void **change\_donnees** (const [Cercle](#) &cer)
- int **Intersection** (const [Droite](#) &D, [Coordonnee](#) &M1, [Coordonnee](#) &M2)
- double **Distance\_au\_Cylindre** (const [Coordonnee](#) &M) const
- bool **Dedans** (const [Coordonnee](#) &M) const
- [Coordonnee](#) **Projete** (const [Coordonnee](#) &M, bool &projection\_ok) const

### Attributs protégés

- [Cercle](#) **cercle**

### Amis

- istream & **operator>>** (istream &, [Cylindre](#) &)
- ostream & **operator<<** (ostream &, const [Cylindre](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Cylindre.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Cylindre.cc

## 6.213 Référence de la classe Ddl

BUT: La classe [Ddl](#) permet de declarer des degres de liberte. soit c'est une variable ou soit c'est une données qui peuvent être soit active soit hors-service ensuite soit il est fixe ou soit il est bloqué 1) libre, fixe, ou 2) hors service libre ou hors service fixe ==> une variable 3) lisible libre ou fixe ou 4) hors service libre ou fixe ==> une donnée 1) c'est pour les ddl qui sont des variables que le calcul va déterminer, 2) sont les variables qui sont hors service: on ne les considère pas 3) sont des variables que l'on peut utiliser mais que le calcul ne cherche pas à déterminer: elle sont en lecture seule en quelque sorte 4) se sont des données qui ne sont pas dispon pour la consultation (par exemple hors temps)

```
#include <Ddl.h>
```

### Fonctions membres publiques

- **Ddl** (Enum\_ddl id\_nom\_ddl=NU\_DDL, double val=0.0, [Enum\\_boolddl](#) val\_fixe=HS\_LISIBLE\_LIBRE)
- **Ddl** (char \*nom, double val=0.0, [Enum\\_boolddl](#) val\_fixe=HS\_LISIBLE\_LIBRE)
- **Ddl** (const [Ddl](#) &d)
- bool **Fixe** () const
- bool **UneVariable** () const
- void **ChangeVariable\_a\_Donnee** ()
- void **ChangeDonnee\_a\_Variable** ()
- const [Enum\\_boolddl](#) **Retour\_Fixe** () const
- void **Change\_fixe** (bool val)
- void **CopieToutesLesConditions** ([Enum\\_boolddl](#) en)
- bool **Service** () const
- bool **SousSurFixe** () const
- void **Retour\_normal\_sur\_ou\_sous\_fixe** ()
- void **Met\_hors\_service** ()
- void **Met\_en\_service** ()
- void **Met\_hors\_service\_ddl** ()
- void **Met\_en\_service\_ddl** ()
- double & **Valeur** ()
- const string **Nom** () const
- Enum\_ddl **Id\_nom** () const
- void **Change\_nom** (char \*nouveau\_nom)
- void **Change\_nom** (Enum\_ddl nouveau\_id)
- void **Affiche** () const
- [Ddl](#) & **operator=** (const [Ddl](#) &d)
- bool **operator==** (const [Ddl](#) &a) const
- bool **operator!=** (const [Ddl](#) &a) const
- int **TestComplet** () const



### Attributs protégés

- Enum\_ddl **id\_nom**
- Enum\_boolddl **fixe**
- double **val\_ddl**

### Amis

- istream & **operator**>> (istream &entree, Ddl &a)
- ostream & **operator**<< (ostream &sort, const Ddl &a)

#### 6.213.1 Description détaillée

BUT: La classe [Ddl](#) permet de declarer des degres de liberte. soit c'est une variable ou soit c'est une données qui peuvent être soit active soit hors-service ensuite soit il est fixe ou soit il est bloqué 1) libre, fixe, ou 2) hors service libre ou hors service fixe ==> une variable 3) lisible libre ou fixe ou 4) hors service libre ou fixe ==> une donnée 1) c'est pour les ddl qui sont des variables que le calcul va déterminer, 2) sont les variables qui sont hors service: on ne les considère pas 3) sont des variables que l'on peut utiliser mais que le calcul ne cherche pas à déterminer: elle sont en lecture seule en quelque sorte 4) se sont des données qui ne sont pas dispon pour la consultation (par exemple hors temps)

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

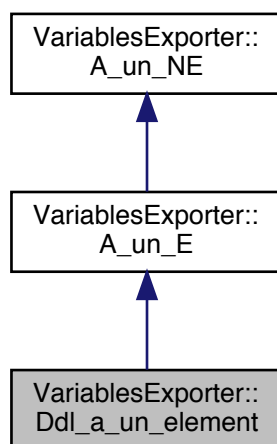
23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

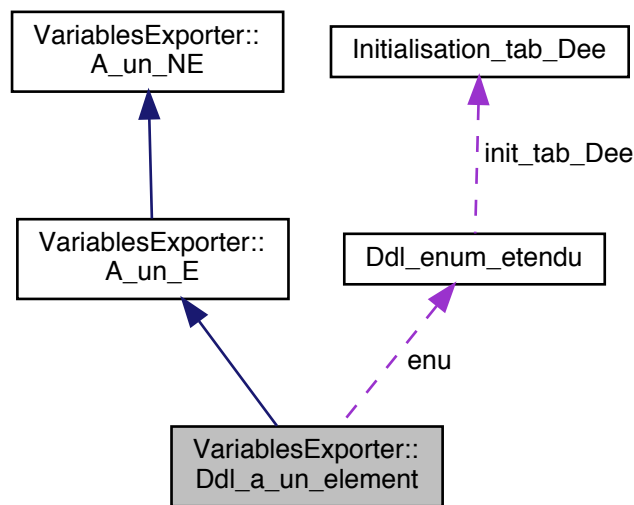
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Ddl.h

## 6.214 Référence de la classe VariablesExporter::Ddl\_a\_un\_element

Graphe d'héritage de VariablesExporter::Ddl\_a\_un\_element:



Graphe de collaboration de VariablesExporter::Ddl\_a\_un\_element:



## Fonctions membres publiques

- **Ddl\_a\_un\_element** (int absolu\_, [Ddl\\_enum\\_etendu](#) e, string ref\_noeud, string nom\_mail\_, [Enum\\_dure](#) tps, string nom\_var\_, int nbpti)
- **Ddl\_a\_un\_element** (const [Ddl\\_a\\_un\\_element](#) &a)
- [Ddl\\_a\\_un\\_element](#) & **operator=** (const [Ddl\\_a\\_un\\_element](#) &a)
- bool **operator==** (const [Ddl\\_a\\_un\\_element](#) &a) const
- bool **operator!=** (const [Ddl\\_a\\_un\\_element](#) &a) const
- bool **operator<** (const [Ddl\\_a\\_un\\_element](#) &a) const
- bool **operator<=** (const [Ddl\\_a\\_un\\_element](#) &a) const
- bool **operator>** (const [Ddl\\_a\\_un\\_element](#) &a) const
- bool **operator>=** (const [Ddl\\_a\\_un\\_element](#) &a) const
- void **Affiche** ()
- const [Ddl\\_enum\\_etendu](#) & **Enu\_const** () const
- [Ddl\\_enum\\_etendu](#) & **Enu** ()
- const [Enum\\_dure](#) & **Temps\_const** () const
- [Enum\\_dure](#) & **Temps** ()

## Attributs protégés

- [Ddl\\_enum\\_etendu](#) **enu**
- [Enum\\_dure](#) **temps**

## Amis

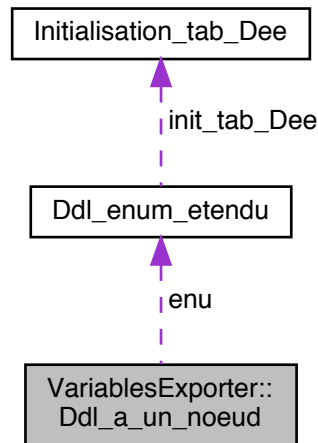
- istream & **operator>>** (istream &, [Ddl\\_a\\_un\\_element](#) &)
- ostream & **operator<<** (ostream &, const [Ddl\\_a\\_un\\_element](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

## 6.215 Référence de la classe VariablesExporter::Ddl\_a\_un\_noeud

Grphe de collaboration de VariablesExporter::Ddl\_a\_un\_noeud:



### Fonctions membres publiques

- `Ddl_a_un_noeud` (`Ddl_enum_etendu` e, string ref\_noeud, string nom\_mail\_, string nom\_var\_, `Enum_dure` tps)
- `Ddl_a_un_noeud` (const `Ddl_a_un_noeud` &a)
- `Ddl_a_un_noeud` & `operator=` (const `Ddl_a_un_noeud` &a)
- bool `operator==` (const `Ddl_a_un_noeud` &a) const
- bool `operator!=` (const `Ddl_a_un_noeud` &a) const
- bool `operator<` (const `Ddl_a_un_noeud` &a) const
- bool `operator<=` (const `Ddl_a_un_noeud` &a) const
- bool `operator>` (const `Ddl_a_un_noeud` &a) const
- bool `operator>=` (const `Ddl_a_un_noeud` &a) const
- void `Affiche` ()
- const `Ddl_enum_etendu` & `Enu_const` () const
- const string & `Ref_Num_NE_const` () const
- const string & `Nom_mail_const` () const
- const string & `Nom_var_const` () const
- const `Enum_dure` & `Temps_const` () const
- string \* `Pointeur_Nom_mail` ()
- `Ddl_enum_etendu` & `Enu` ()
- string & `Ref_Num_NE` ()
- string & `Nom_mail` ()
- string & `Nom_var` ()
- `Enum_dure` & `Temps` ()

### Attributs protégés

- `Ddl_enum_etendu` enu
- string ref\_num\_NE
- string nom\_mail
- string nom\_var
- `Enum_dure` temps

## Amis

- `istream & operator>>` (`istream &`, [Ddl\\_a\\_un\\_noeud &](#))
- `ostream & operator<<` (`ostream &`, `const Ddl\_a\_un\_noeud &`)

La documentation de cette classe a été générée à partir du fichier suivant :

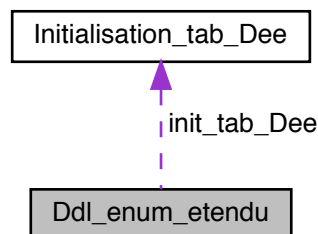
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc`

## 6.216 Référence de la classe `Ddl_enum_etendu`

la classe principale: [Ddl\\_enum\\_etendu](#)

```
#include <Ddl_enum_etendu.h>
```

Graphe de collaboration de `Ddl_enum_etendu`:



## Fonctions membres publiques

- `Ddl_enum_etendu` (`Enum_ddl en=NU_DDL`, `string no=""`)
- `Ddl_enum_etendu` (`string no`)
- `Ddl_enum_etendu` (`char *no`)
- `Ddl_enum_etendu` (`const Ddl\_enum\_etendu &a`)
- `Ddl\_enum\_etendu & operator=` (`const Ddl\_enum\_etendu &a`)
- `bool operator==` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator!=` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator>` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator>=` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator<` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator<=` (`const Ddl\_enum\_etendu &a`) `const`
- `Enum_ddl Enum` () `const`
- `string Nom` () `const`
- `string NomGenerique` () `const`
- `int Position` () `const`
- `bool Nom_vider` () `const`
- `EnumTypeGrandeur TypeDeGrandeur` () `const`
- `Ddl_enum_etendu` (`Enum_ddl en=NU_DDL`, `string no=""`)
- `Ddl_enum_etendu` (`string no`)
- `Ddl_enum_etendu` (`const Ddl\_enum\_etendu &a`)
- `Ddl\_enum\_etendu & operator=` (`const Ddl\_enum\_etendu &a`)
- `bool operator==` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator!=` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator>` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator>=` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator<` (`const Ddl\_enum\_etendu &a`) `const`
- `bool operator<=` (`const Ddl\_enum\_etendu &a`) `const`
- `Enum_ddl Enum` () `const`

- string **Nom** () const
- string **NomGenerique** () const
- string **Nom\_plein** () const
- int **Position** () const
- bool **Nom\_vide** () const
- [EnumTypeGrandeur](#) **TypeDeGrandeur** () const

## Fonctions membres publiques statiques

- static bool **VerifExistence** (string no)
- static [Ddl\\_enum\\_etendu](#) **RecupDdl\_enum\_etendu** (string to)
- static [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > **TransfoList\_io** (const [List\\_io](#)< [Enum\\_ddl](#) > &li)
- static [Tableau](#)< [Ddl\\_enum\\_etendu](#) > **TransfoTableau** (const [Tableau](#)< [Enum\\_ddl](#) > &tab)
- static bool **Existe\_dans\_la\_liste** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lis, const [Ddl\\_enum\\_etendu](#) &dd)
- static int **NBmax\_ddl\_enum\_etendue** ()
- static bool **VerifExistence** (string no)
- static [Ddl\\_enum\\_etendu](#) **RecupDdl\_enum\_etendu** (string to)
- static [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > **TransfoList\_io** (const [List\\_io](#)< [Enum\\_ddl](#) > &li)
- static [Tableau](#)< [Ddl\\_enum\\_etendu](#) > **TransfoTableau** (const [Tableau](#)< [Enum\\_ddl](#) > &tab)
- static bool **Existe\_dans\_la\_liste** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lis, const [Ddl\\_enum\\_etendu](#) &dd)
- static int **NBmax\_ddl\_enum\_etendue** ()
- static const [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & **Tab\_FN\_FT** ()
- static [Ddl\\_enum\\_etendu](#) **PremierDdlEnumEtenduFamille** (const [Ddl\\_enum\\_etendu](#) &a)
- static void **Equivalent\_en\_grandeur\_quelconque** (const list< [Ddl\\_enum\\_etendu](#) > &list\_enu\_etendu, list< [EnumTypeQuelconque](#) > &list\_enu\_quelconque, list< [Ddl\\_enum\\_etendu](#) > &list\_enu\_restant)
- static [EnumTypeQuelconque](#) **Equivalent\_en\_grandeur\_quelconque** (const [Ddl\\_enum\\_etendu](#) &enu\_↔  
etendu)

## Attributs protégés

- string **nom**
- [Enum\\_ddl](#) **enu**
- int **posi\_nom**

## Attributs protégés statiques

- static [Tableau](#)< [Ddl\\_enum\\_etendu](#) > **tab\_De**
- static [Initialisation\\_tab\\_De](#) **init\_tab\_De**
- static int **tailTab** = 0
- static [Tableau](#)< [Ddl\\_enum\\_etendu](#) > **tab\_FN\_FT**
- static map< string, int, std::less< string > > **map\_Ddl\_enum\_etendu**

## Amis

- class [Initialisation\\_tab\\_De](#)
- istream & **operator**>> (istream &ent, [Ddl\\_enum\\_etendu](#) &a)
- ostream & **operator**<< (ostream &sort, const [Ddl\\_enum\\_etendu](#) &a)
- istream & **operator**>> (istream &ent, [Ddl\\_enum\\_etendu](#) &a)
- ostream & **operator**<< (ostream &sort, const [Ddl\\_enum\\_etendu](#) &a)

### 6.216.1 Description détaillée

la classe principale: [Ddl\\_enum\\_etendu](#)

La documentation de cette classe a été générée à partir du fichier suivant :

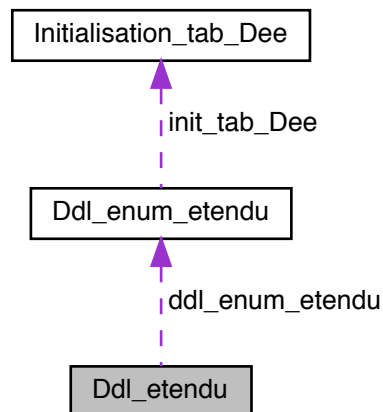
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl\_enum\_etendu-copie.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl\_enum\_etendu.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl\_enum\_etendu-copie.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl\_enum\_etendu.cc

## 6.217 Référence de la classe Ddl\_etendu

BUT: class de stockage d'un ddl étendue, associé au type [Ddl\\_enum\\_etendu](#). En fait il s'agit du pendant des types [Ddl](#) associé au type `enum_ddl` Noter que la grandeur stockée est un scalaire !!

```
#include <Ddl_etendu.h>
```

Graphe de collaboration de `Ddl_etendu`:



### Fonctions membres publiques

- `Ddl_etendu` (`const Ddl_enum_etendu &ddl`, `double vale=0.0`)
- `Ddl_etendu` (`const Ddl_etendu &d`)
- `double & Valeur` ()
- `const double & ConstValeur` () `const`
- `Ddl_enum_etendu & DdlEnumEtendu` ()
- `const Ddl_enum_etendu & Const_DdlEnumEtendu` () `const`
- `void Affiche` () `const`
- `void Affiche` (`ostream &sort`) `const`
- `void Affiche` (`ostream &sort`, `int nb`) `const`
- `Ddl_etendu & operator=` (`const Ddl_etendu &d`)
- `void operator+=` (`const Ddl_etendu &c`)
- `void operator-=` (`const Ddl_etendu &c`)
- `void operator*=` (`double val`)
- `void operator/=` (`double val`)
- `bool operator==` (`Ddl_etendu &a`) `const`
- `bool operator!=` (`Ddl_etendu &a`) `const`
- `bool Identique` (`Ddl_etendu &a`)

### Attributs protégés

- `Ddl_enum_etendu ddl_enum_etendu`
- `double val_ddl`

### Amis

- `istream & operator>>` (`istream &entree`, `Ddl_etendu &a`)
- `ostream & operator<<` (`ostream &sort`, `const Ddl_etendu &a`)

### 6.217.1 Description détaillée

BUT: class de stockage d'un ddl étendue, associé au type [Ddl\\_enum\\_etendu](#). En fait il s'agit du pendant des types [Ddl](#) associé au type enum\_ddl. Noter que la grandeur stockée est un scalaire !!

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

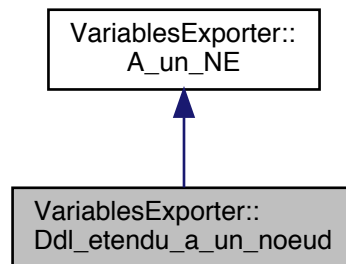
19/01/2006

La documentation de cette classe a été générée à partir du fichier suivant :

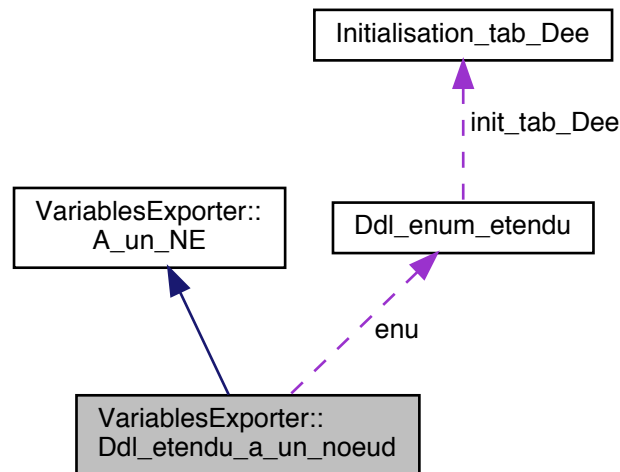
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Ddl\_etendu.h

## 6.218 Référence de la classe VariablesExporter::Ddl\_etendu\_a\_un\_noeud

Graphe d'héritage de VariablesExporter::Ddl\_etendu\_a\_un\_noeud:



Graphe de collaboration de VariablesExporter::Ddl\_etendu\_a\_un\_noeud:



## Fonctions membres publiques

- `Ddl_etendu_a_un_noeud` (`Ddl_enum_etendu e`, `string ref_noeud`, `string nom_mail_`, `string nom_var_`)
- `Ddl_etendu_a_un_noeud` (`const Ddl_etendu_a_un_noeud &a`)
- `Ddl_etendu_a_un_noeud & operator=` (`const Ddl_etendu_a_un_noeud &a`)
- `bool operator==` (`const Ddl_etendu_a_un_noeud &a`) `const`
- `bool operator!=` (`const Ddl_etendu_a_un_noeud &a`) `const`
- `bool operator<` (`const Ddl_etendu_a_un_noeud &a`) `const`
- `bool operator<=` (`const Ddl_etendu_a_un_noeud &a`) `const`
- `bool operator>` (`const Ddl_etendu_a_un_noeud &a`) `const`
- `bool operator>=` (`const Ddl_etendu_a_un_noeud &a`) `const`
- `void Affiche` ()
- `Ddl_enum_etendu Enu_const` () `const`
- `Ddl_enum_etendu & Enu` ()

## Attributs protégés

- `Ddl_enum_etendu enu`

## Amis

- `istream & operator>>` (`istream &`, `Ddl_etendu_a_un_noeud &`)
- `ostream & operator<<` (`ostream &`, `const Ddl_etendu_a_un_noeud &`)

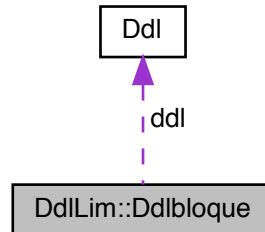
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc`



## 6.219 Référence de la classe DdlLim::Ddlbloque

Grphe de collaboration de DdlLim::Ddlbloque:



### Fonctions membres publiques

- **Ddlbloque** (const [Ddlbloque](#) &d)
- **Ddlbloque** ([Ddl](#) ddl\_)
- [Ddlbloque](#) & **operator=** (const [Ddlbloque](#) &d)
- bool **operator==** ([Ddlbloque](#) &a)
- bool **operator!=** ([Ddlbloque](#) &a)

### Attributs publics

- [Ddl](#) **ddl**
- string **co\_charge**
- string **f\_charge**
- double **echelle**

### Amis

- istream & **operator>>** (istream &entree, [Ddlbloque](#) &d)
- ostream & **operator<<** (ostream &sort, const [Ddlbloque](#) &d)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlLim.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlLim.cc

## 6.220 Référence de la classe DdlElement

BUT: Definition, gestion et stockage des ddl de l'element.

```
#include <DdlElement.h>
```

### Classes

- class [Int\\_initer](#)

### Fonctions membres publiques

- **DdlElement** (int n)
- **DdlElement** (int n, int m)
- **DdlElement** (int n, const [Tableau](#)< int > &mddl)
- **DdlElement** (int n, [DdlNoeudElement](#) d)

- **DdlElement** (const [DdlElement](#) &a)
- int **NbNoeud** () const
- void **Change\_Enum** (int i, int j, const Enum\_ddl enu)
- Enum\_ddl **operator()** (const int i, const int j) const
- const [DdlNoeudElement](#) & **operator()** (int i) const
- int **NbDdl** () const
- int **NbDdl** (int i) const
- int **NbDdl\_famille** (Enum\_ddl enu) const
- void **Change\_taille** (int n, int m)
- void **Change\_taille** (int n, [Tableau](#)< int > mddl)
- void **Change\_un\_ddlNoeudElement** (int i, const [DdlNoeudElement](#) &add)
- void **TailleZero** ()
- [DdlElement](#) & **operator=** (const [DdlElement](#) &a)
- bool **operator==** (const [DdlElement](#) &a) const
- bool **operator!=** (const [DdlElement](#) &a) const

### Fonctions membres protégées

- void **Calcul\_NbDdl** ()
- void **Calcul\_NbDdl\_typer** ()

### Attributs protégés

- [Tableau](#)< [DdlNoeudElement](#) > **te**
- int **nbddl**
- map< Enum\_ddl, [Int\\_initer](#), std::less< Enum\_ddl > > **tab\_enum\_famille**

### Amis

- istream & **operator**>> (istream &entree, [DdlElement](#) &a)
- ostream & **operator**<< (ostream &sort, const [DdlElement](#) &a)

## 6.220.1 Description détaillée

BUT: Definition, gestion et stockage des ddl de l'element.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

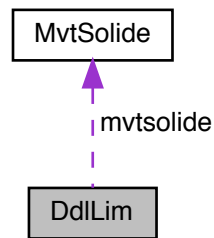
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlElement.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlElement.cc

## 6.221 Référence de la classe DdlLim

BUT: La classe [DdlLim](#) permet de declarer un ensemble de ddl coiffe par une reference utilise pour les conditions limites.

```
#include <DdlLim.h>
```

Graphe de collaboration de DdlLim:



## Classes

- class [Ddlbloque](#)

## Fonctions membres publiques

- **DdlLim** (const [DdlLim](#) &d)
- const string & **NomRef** () const
- const string \* **NomMaillage** () const
- [EnumTypeCL](#) **TypeDeCL** () const
- int **Taille** () const
- [Ddl](#) & **ElemLim** (int i)
- const [Ddl](#) & **ElemLim\_const** (int i) const
- const string & **Nom\_courbe** (int i) const
- const string & **NomF\_charge** (int i) const
- bool **Champ** () const
- bool **BlocageRelatif** () const
- bool **Mouvement\_Solide** () const
- const [MvtSolide](#) \* **Const\_MouvementSolide** () const
- [MvtSolide](#) \* **MouvementSolide** () const
- double **Echelle\_courbe** (int i) const
- int **Temps\_depend\_nD** () const
- string **Nom\_fctnD\_tmin** () const
- string **Nom\_fctnD\_tmax** () const
- void **Mise\_a\_jour\_tmin\_tmax** (double tmin, double tmax)
- void **Mise\_a\_jour\_tmin** (double tmin)
- void **Mise\_a\_jour\_tmax** (double tmax)
- bool **Temps\_actif** (const double &temps) const
- bool **Pas\_a\_prendre\_en\_compte** (const double &temps) const
- bool **Pas\_a\_prendre\_en\_compte\_dans\_intervalle** (const double &temps) const
- void **Info\_commande\_DdlLim** (ofstream &sort, bool plusieurs\_maillages)
- void **Validation** (const double &temps)
- bool **Etat\_validation** () const
- void **Lecture** ([UtilLecture](#) &entreePrinc)
- bool **MemeCibleMaisDataDifferentes** (const [DdlLim](#) &d) const
- bool **Existe\_ici\_leDdl** ([Enum\\_ddl](#) enu) const
- bool **Existe\_ici\_un\_deplacement** () const
- void **Affiche** () const
- [DdlLim](#) & **operator=** (const [DdlLim](#) &d)
- bool **operator==** (const [DdlLim](#) &a) const
- bool **operator!=** (const [DdlLim](#) &a) const

## Fonctions membres protégées

- void **VerifDimDdl** ()
- list< [Ddlbloque](#) > **Lecture\_champ** ([UtilLecture](#) &entreePrinc)
- list< [Ddlbloque](#) > **Lecture\_mvtSolide** ([UtilLecture](#) &entreePrinc)
- void **Change\_ref** (string nouveau)
- void **Change\_nom\_maillage** (const string &nouveau)

## Attributs protégés

- string \* **nom\_maillage**
- string **ref**
- [EnumTypeCL](#) **typeCL**
- [Tableau](#)< [Ddlbloque](#) > **tab**
- double **t\_min**
- double **t\_max**
- string **nom\_fnD\_t\_min**
- string **nom\_fnD\_t\_max**
- int **precedent**
- bool **champ**
- bool **blocage\_relatif**
- [MvtSolide](#) \* **mvtsolide**

## Amis

- istream & **operator**>> (istream &, [DdlLim](#) &)
- ostream & **operator**<< (ostream &, const [DdlLim](#) &)

### 6.221.1 Description détaillée

BUT: La classe [DdlLim](#) permet de declarer un ensemble de ddl coiffe par une reference utilise pour les conditions limites.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

15/04/1997

### 6.221.2 Documentation des fonctions membres

#### 6.221.2.1 Lecture\_champ()

```
list< DdlLim::Ddlbloque > DdlLim::Lecture\_champ (
    UtilLecture & entreePrinc ) [protected]
```

\_\_\_\_\_ la grande boucle

\_\_\_\_\_ fin de la grande boucle

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlLim.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlLim.cc

## 6.222 Référence de la classe DdlNoeudElement

BUT: Def des ddl lie a un noeud d'un element ( different des ddl lies aux noeuds globaux)

```
#include <DdlNoeudElement.h>
```

## Fonctions membres publiques

- **DdlNoeudElement** (int n)
- **DdlNoeudElement** (Enum\_ddl e)
- **DdlNoeudElement** (const [DdlNoeudElement](#) &a)
- [DdlNoeudElement](#) & **operator=** (const [DdlNoeudElement](#) &a)
- bool **operator==** (const [DdlNoeudElement](#) &a) const
- bool **operator!=** (const [DdlNoeudElement](#) &a) const

## Attributs publics

- [Tableau](#)< Enum\_ddl > **tb**

## Amis

- istream & **operator**>> (istream &entree, [DdlNoeudElement](#) &a)
- ostream & **operator**<< (ostream &sort, const [DdlNoeudElement](#) &a)

### 6.222.1 Description détaillée

BUT: Def des ddl lie a un noeud d'un element ( different des ddl lies aux noeuds globaux)

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

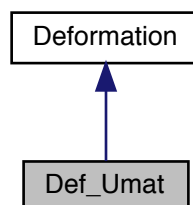
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlNoeudElement.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlNoeudElement.cc

## 6.223 Référence de la classe Def\_Umat

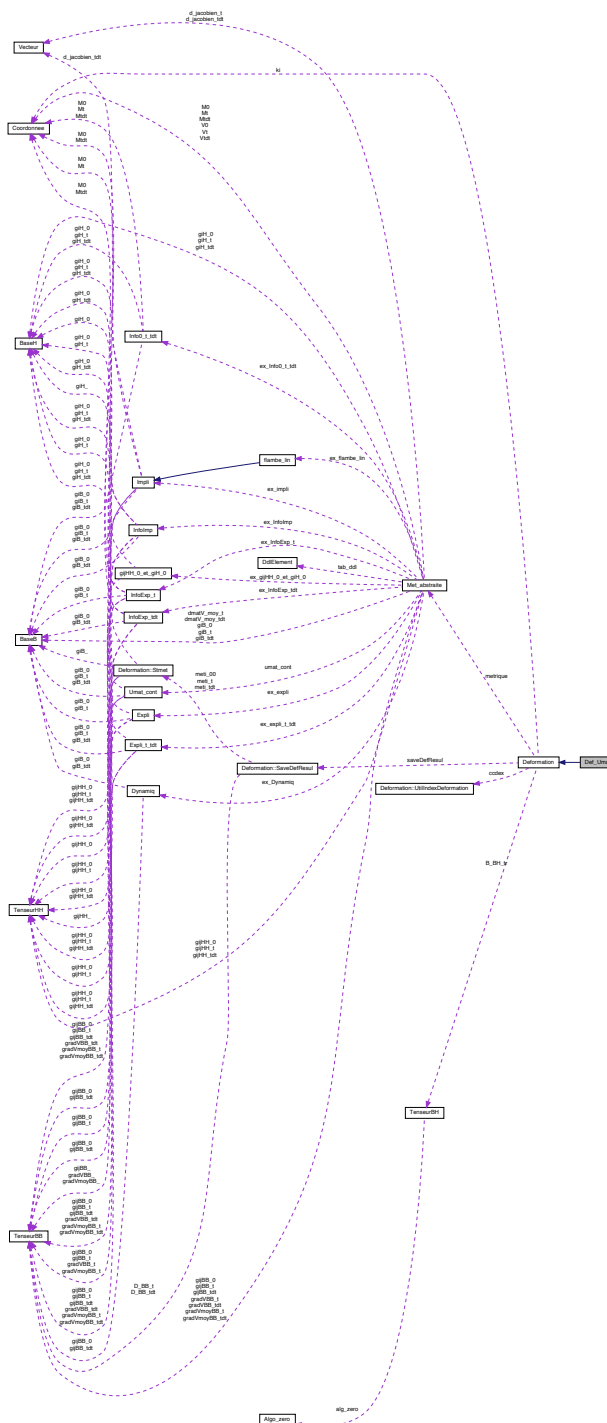
BUT: Calcul des differentes grandeurs liee a la deformation  
Dans le cas des grandeurs relatives aux procédures Umat.

```
#include <Def_Umat.h>
```

Graphe d'héritage de Def\_Umat:



Graphe de collaboration de Def\_Umat:



## Fonctions membres publiques

- **Def\_Umat** (**Met\_abstraite** &, **Tableau**< **Noeud** \* > &tabnoeud, **Tableau**< **Mat\_pleine** > const &tabDphi, **Tableau**< **Vecteur** >const &tabPhi, **Enum\_type\_deformation** type\_de\_deformation=DEFORMATION\_STANDART)
- **Def\_Umat** (const **Def\_Umat** &)
- virtual **Deformation** & operator= (const **Deformation** &def)
- virtual **Def\_Umat** \* **Nevez\_deformation** (**Tableau**< **Noeud** \* > &tabN) const
- const **Met\_ElemPoint::Umat\_cont** & **Cal\_defUmat** (bool premier\_calcul)

- void **Mise\_a\_jourUmat** (const [UmatAbaqus](#) &umat)
- void **CalDefEqui** (const [Tableau](#)< double > &def\_equi\_t, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< double > &def\_equi, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB\_tdt, const [Met\\_abstraite::Umat\\_cont](#) &ex, bool premier\_calcul)

## Membres hérités additionnels

### 6.223.1 Description détaillée

BUT: Calcul des différentes grandeurs liées à la déformation  
 Dans le cas des grandeurs relatives aux procédures Umat.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

2/02/2005

### 6.223.2 Documentation des fonctions membres

#### 6.223.2.1 Nevez\_deformation()

```
virtual Def\_Umat * Def\_Umat::Nevez\_deformation (
    Tableau< Noeud * > & tabN ) const [inline], [virtual]
```

Réimplémentée à partir de [Deformation](#).

#### 6.223.2.2 operator=()

```
virtual Deformation & Def\_Umat::operator= (
    const Deformation & def ) [inline], [virtual]
```

Réimplémentée à partir de [Deformation](#).

La documentation de cette classe a été générée à partir du fichier suivant :

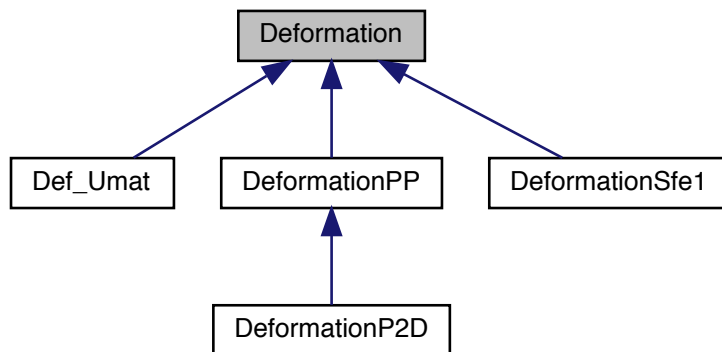
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Def\_↔  
Umat.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Def\_↔  
Umat.cc

## 6.224 Référence de la classe Deformation

BUT: Calcul des différentes grandeurs liées à la déformation La classe fonctionne comme une boîte à outil. On y choisit ce dont on a besoin. Bien faire attention à l'ordre d'appel des différentes méthodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stocke pas (ou très peu).

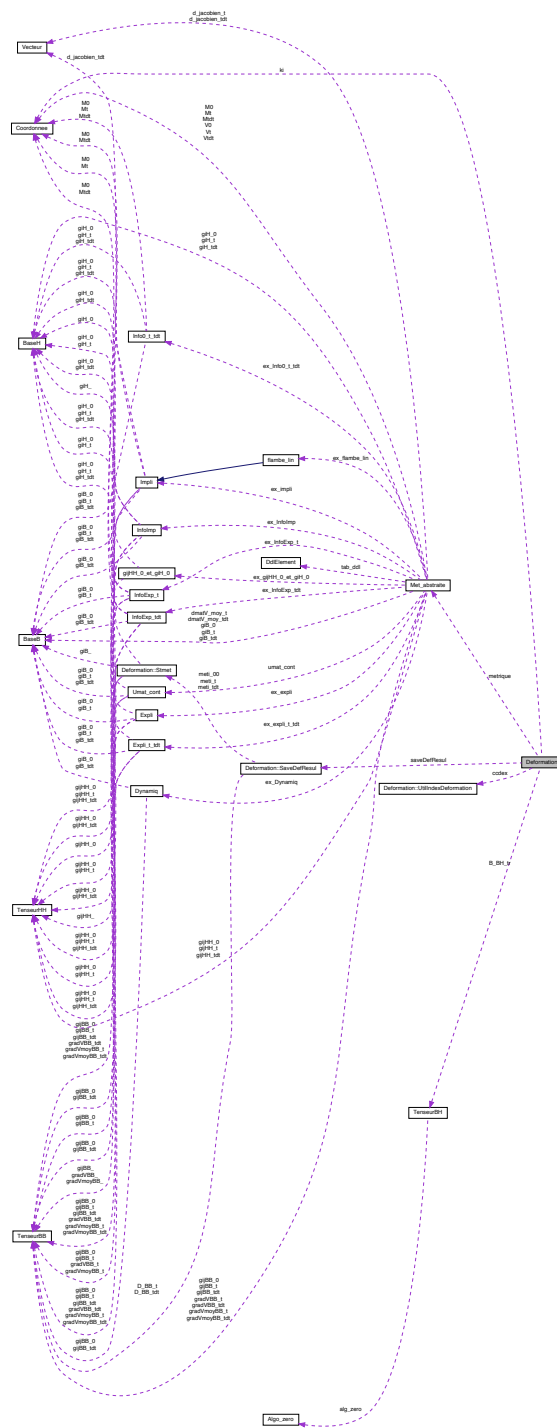
```
#include <Deformation.h>
```

Graphe d'héritage de Deformation:





Graphe de collaboration de Deformation:



### Classes

- class [SaveDefResul](#)
- class [Stmet](#)
- class [UtilIndexDeformation](#)

## Fonctions membres publiques

- **Deformation** ([Met\\_abstraite](#) &, [Tableau](#)< [Noeud](#) \* > &tabnoeud, [Tableau](#)< [Mat\\_pleine](#) > const &tabDphi, [Tableau](#)< [Vecteur](#) >const &tabPhi, [Enum\\_type\\_deformation](#) type\_de\_deformation=DEFORMATION\_←\_STANDART)
- **Deformation** (const [Deformation](#) &)
- virtual [SaveDefResul](#) \* **New\_et\_Initialise** ()
- virtual void **AfficheDataSpecif** (ofstream &sort, [SaveDefResul](#) \*a) const
- void **Mise\_a\_jour\_data\_specif** ([Deformation](#)::[SaveDefResul](#) \*don)
- virtual [Deformation](#) \* **Nevez\_deformation** ([Tableau](#)< [Noeud](#) \* > &tabN) const
- void **PointeurTableauNoeud** ([Tableau](#)< [Noeud](#) \* > &tabN)
- virtual [Deformation](#) & **operator=** (const [Deformation](#) &def)
- virtual void **Change\_type\_de\_deformation** ([Enum\\_type\\_deformation](#) type\_de\_def)
- virtual void **Affiche** () const
- virtual const [Met\\_abstraite](#)::[Expli](#) & **Cal\_explicit\_t** (const [Tableau](#)< double > &def\_equi\_t, [TenseurBB](#) &epsBB\_t, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< double > &def\_equi, [TenseurBB](#) &DepsBB, [TenseurBB](#) &DeltaEpsBB, bool premier\_calcul)
- virtual const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) & **Cal\_explicit\_tdt** (const [Tableau](#)< double > &def\_equi\_t, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< double > &def\_equi, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB\_tdt, bool premier\_calcul)
- virtual const [Met\\_abstraite](#)::[Impli](#) & **Cal\_implicit** (const [Tableau](#)< double > &def\_equi\_t, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< double > &def\_equi, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_calcul)
- virtual const [Met\\_abstraite](#)::[Expli](#) & **Cal\_explicit\_t** (bool premier\_calcul)
- virtual const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) & **Cal\_explicit\_tdt** (bool premier\_calcul)
- virtual const [Met\\_abstraite](#)::[Impli](#) & **Cal\_implicit** (bool premier\_calcul, bool avec\_var\_Xi=true)
- virtual const [Met\\_abstraite](#)::[flambe\\_lin](#) & **Cal\_flambe\_lin** (const [Tableau](#)< double > &def\_equi\_t, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< double > &def\_equi, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_calcul)
- virtual const [Met\\_abstraite](#)::[Dynamiq](#) & **Cal\_matMass** ()
- virtual double **DonneeInterpoleeScalaire** ([Enum\\_ddl](#) enu, [Enum\\_dure](#) temps) const
- virtual [Tableau](#)< double > & **DerDonneeInterpoleeScalaire** ([Tableau](#)< double > &d\_A, [Enum\\_ddl](#) enu, [Enum\\_dure](#) temps) const
- virtual [CoordonneeB](#) & **GradDonneeInterpoleeScalaire** ([CoordonneeB](#) &gradB, [Enum\\_ddl](#) enu, [Enum\\_dure](#) temps) const
- virtual [Tableau](#)< [CoordonneeB](#) > & **DerGradDonneeInterpoleeScalaire** ([Tableau](#)< [CoordonneeB](#) > &d\_←\_gradB, [Enum\\_ddl](#) enu) const
- virtual [Tableau2](#)< [CoordonneeB](#) > \* **Der2GradDonneeInterpoleeScalaire** ([Tableau2](#)< [CoordonneeB](#) > \*d2\_gradTB, [Enum\\_ddl](#) enu) const
- virtual void **DeformationThermoMecanique** (const double &temperature\_0, const [TenseurBB](#) &gijBB, const [ThermoDonnee](#) &dTP, const [TenseurBB](#) &epsBB\_totale, [TenseurBB](#) &epsBB\_therm, const double &temperature\_tdt, [TenseurBB](#) &epsBB\_meca, const double &temperature\_t, [TenseurBB](#) &DepsBB\_totale, [TenseurBB](#) &DepsBB\_therm, [TenseurBB](#) &DepsBB\_meca, bool atdt, bool avec\_repercution\_sur\_def\_meca)
- void **Cal\_var\_def\_virtuelle** (bool total, [Tableau2](#)< [TenseurBB](#) \* > &d2\_epsBB\_tdt)
- virtual const [Met\\_abstraite](#)::[Infolmp](#) **RemontImp** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite](#)::[Infolmp](#) **RemontImp** ()
- virtual const [Met\\_abstraite](#)::[InfoExp\\_t](#) **RemontExp\_t** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite](#)::[InfoExp\\_t](#) **RemontExp\_t** ()
- virtual const [Met\\_abstraite](#)::[InfoExp\\_tdt](#) **RemontExp\_tdt** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite](#)::[InfoExp\\_tdt](#) **RemontExp\_tdt** ()
- virtual const [Met\\_abstraite](#)::[Info0\\_t\\_tdt](#) **Remont0\_t\_tdt** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aat, [Mat\\_pleine](#) &Aatdt)
- virtual const [Met\\_abstraite](#)::[Info0\\_t\\_tdt](#) **Remont0\_t\_tdt** ()
- virtual const [Met\\_abstraite](#)::[Infolmp](#) **RemontImpSansCalMet** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite](#)::[InfoExp\\_t](#) **RemontExp\_tSansCalMet** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite](#)::[InfoExp\\_tdt](#) **RemontExp\_tdtSansCalMet** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite](#)::[Info0\\_t\\_tdt](#) **Remont0\_t\_tdtSansCalMet** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aat, [Mat\\_pleine](#) &Aatdt)

- virtual const Met\_abstraite::Info\_et\_metroque\_0\_t\_tdt **Remont\_et\_metroque\_0\_t\_tdtSansCalMet** (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aat, [Mat\\_pleine](#) &Aatdt)
- virtual const Met\_abstraite::Info\_et\_metroque\_0\_t\_tdt **Remont\_et\_metroque\_0\_t\_tdtSansCalMet** ()
- virtual void **Cal\_deformation** ([Enum\\_dure](#) temps, [TenseurBB](#) &epsBB)
- [Enum\\_type\\_deformation](#) **Type\_de\_deformation** () const
- [Enum\\_type\\_gradient](#) **Type\_de\_gradient\_vitesse** () const
- virtual void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &) const
- virtual void **ListeGrandeurs\_particulieres** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &) const
- int **Phi1\_Taille** () const
- virtual void **ChangeNumInteg** (int ni)
- int **NumInteg\_en\_cours** () const
- virtual void **PremierPtInteg** ()
- virtual bool **DernierPtInteg** ()
- virtual void **NevezPtInteg** ()
- virtual void **Retour\_pti\_precedant** ()
- virtual const [Coordonnee](#) & **Position\_0** ()
- virtual const [Coordonnee](#) & **Position\_t** ()
- virtual const [Coordonnee](#) & **Position\_tdt** ()
- virtual const [Tableau](#)< [Coordonnee](#) > & **Der\_Pos\_t** ()
- virtual const [Tableau](#)< [Coordonnee](#) > & **Der\_Pos\_tdt** ()
- virtual const [Coordonnee](#) & **VitesseM\_0** ()
- virtual const [Coordonnee](#) & **VitesseM\_0** (int num\_n)
- virtual const [Coordonnee](#) & **VitesseM\_0** (const [Noeud](#) \*noe)
- virtual const [Coordonnee](#) & **VitesseM\_t** ()
- virtual const [Coordonnee](#) & **VitesseM\_t** (int num\_n)
- virtual const [Coordonnee](#) & **VitesseM\_t** (const [Noeud](#) \*noe)
- virtual const [Coordonnee](#) & **VitesseM\_tdt** ()
- virtual const [Coordonnee](#) & **VitesseM\_tdt** (int num\_n)
- virtual const [Coordonnee](#) & **VitesseM\_tdt** (const [Noeud](#) \*noe)
- virtual const [Tableau](#)< [Coordonnee](#) > & **Der\_VitesseM\_t** (bool &ddl\_vitesse)
- virtual const [Tableau](#)< [Coordonnee](#) > & **Der\_VitesseM\_t** (bool &ddl\_vitesse, int num\_noeud)
- virtual const [Tableau](#)< [Coordonnee](#) > & **Der\_VitesseM\_t** (bool &ddl\_vitesse, const [Noeud](#) \*noe)
- virtual const [Tableau](#)< [Coordonnee](#) > & **Der\_VitesseM\_tdt** (bool &ddl\_vitesse)
- virtual const [Tableau](#)< [Coordonnee](#) > & **Der\_VitesseM\_tdt** (bool &ddl\_vitesse, int num\_noeud)
- virtual const [Tableau](#)< [Coordonnee](#) > & **Der\_VitesseM\_tdt** (bool &ddl\_vitesse, const [Noeud](#) \*noe)
- virtual void **BasePassage** (bool absolue, const [BaseB](#) &giB0, const [BaseB](#) &giB, const [BaseH](#) &giH0, const [BaseH](#) &giH, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aaafin)
- virtual void **BasePassage** (bool absolue, const [BaseB](#) &giB0, const [BaseB](#) &giB\_t, const [BaseB](#) &giB\_←tdt, const [BaseH](#) &giH0, const [BaseH](#) &giH\_t, const [BaseH](#) &giH\_tdt, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aa\_t, [Mat\\_pleine](#) &Aa\_tdt)
- virtual void **BasePassage** (bool absolue, const [BaseB](#) &giB0, const [BaseH](#) &giH0, [Mat\\_pleine](#) &Aa0)
- double **JacobienInitial** ()

## Fonctions membres protégées

- const Met\_abstraite::Impli & **Cal\_implicit\_Almani** (bool gradV\_instantane, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_←calcul, const Met\_abstraite::Impli &ex)
- const Met\_abstraite::Impli & **Cal\_implicit\_Logarithmique** (bool gradV\_instantane, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_calcul, const Met\_abstraite::Impli &ex)
- const Met\_abstraite::Impli & **Cal\_implicit\_def\_cumule** (bool gradV\_instantane, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_calcul, const Met\_abstraite::Impli &ex)
- const Met\_abstraite::Expli & **Cal\_explicit\_Almani** (bool gradV\_instantane, [TenseurBB](#) &epsBB\_←t, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_←calcul, const Met\_abstraite::Expli &ex)
- const Met\_abstraite::Expli & **Cal\_explicit\_Logarithmique** (bool gradV\_instantane, [TenseurBB](#) &epsBB\_t, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_calcul, const Met\_abstraite::Expli &ex)
- const Met\_abstraite::Expli & **Cal\_explicit\_def\_cumule** (bool gradV\_instantane, [TenseurBB](#) &epsBB\_←t, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_←calcul, const Met\_abstraite::Expli &ex)

- const `Met_abstraite::Expli_t_tdt & Cal_explicit_Almansi_tdt` (bool `gradV_instantane`, `TenseurBB` &epsBB↔  
\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &DepsBB, `TenseurBB` &delta\_epsBB, bool `premier`↔  
\_calcul, const `Met_abstraite::Expli_t_tdt` &ex)
- const `Met_abstraite::Expli_t_tdt & Cal_explicit_logarithmique_tdt` (bool `gradV_instantane`, `TenseurBB`  
&epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &DepsBB, `TenseurBB` &delta\_epsBB, bool  
`premier_calcul`, const `Met_abstraite::Expli_t_tdt` &ex)
- const `Met_abstraite::Expli_t_tdt & Cal_explicit_def_cumule_tdt` (bool `gradV_instantane`, `TenseurBB`  
&epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &DepsBB, `TenseurBB` &delta\_epsBB, bool  
`premier_calcul`, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Cal_Almansi_auTemps` (`Enum_dure` temps, `TenseurBB` &epsBB)
- void `Cal_logarithmique_auTemps` (`Enum_dure` temps, `TenseurBB` &epsBB)
- void `Cal_def_cumule_auTemps` (`Enum_dure` temps, `TenseurBB` &epsBB)
- void `VerifCal_implicit` (bool `gradV_instantane`, const `Met_abstraite::Implici` &ex, `TenseurBB` &DepsBB)
- void `Cal_vite_rota_objectif` (`EnumTypeViteRotat` type\_rotation, bool variation, const `TenseurBB` &grad↔  
VBB, const `Tableau`< `TenseurBB` \* > &d\_gradVBB, const `TenseurBH` &gijHH\_0, const `TenseurBB` &gij\_BB,  
`Tableau`< `TenseurBB` \* > &d\_gijBB, const `TenseurHH` &gij\_HH, `Tableau`< `TenseurHH` \* > &d\_gijHH, const  
`TenseurBB` &D\_BB, `Tableau`< `TenseurBB` \* > &d\_D\_BB, `TenseurBB` &OmegaBB, `Tableau`< `TenseurBB` \* >  
&d\_OmegaBB)
- double `F_pour_rotat_objectif` (`EnumTypeViteRotat` type\_rotation, const double &x)
- double `Fprime_pour_rotat_objectif` (`EnumTypeViteRotat` type\_rotation, const double &x)
- void `Val_et_projection_prop_tenseur` (const `TenseurBH` &B\_BH, const `Tableau`< `TenseurBH` \* > &d\_↔  
B\_BH, `Coordonnee` &ki, `Tableau`< `TenseurBH` \* > &Palpha\_BH, bool variation, `Tableau`< `Coordonnee` >  
&d\_ki, `Tableau`< `Tableau`< `TenseurBH` \* > > &d\_Palpha\_BH, int &cas\_ki)
- void `Integ_vitesse_rotation` ()
- void `DimensionnementVarLog` (int dima, bool avec\_var, int nbvar)

### Attributs protégés

- `Met_abstraite` \* `metrique`
- `Tableau`< `Noeud` \* > \* `tabnoeud`
- `Tableau`< `Mat_pleine` > const \* `tabDphi`
- `Tableau`< `Vecteur` > const \* `tabPhi`
- int `nbNoeud`
- int `numInteg`
- int `saue_numInteg`
- `Enum_type_deformation` `type_deformation`
- `Enum_type_gradient` `type_gradient_vitesse`
- `SaveDefResul` \* `saveDefResul`
- `TenseurBH` \* `B_BH_tr`
- `Tableau`< `TenseurBH` \* > \* `d_B_BH_tr`
- `Coordonnee` `ki`
- `Tableau`< `Coordonnee` > \* `d_ki`
- `Tableau`< `TenseurBH` \* > \* `Palpha_BH_tr`
- `Tableau`< `Tableau`< `TenseurBH` \* > > \* `d_Palpha_BH_tr`
- int `dimensionnement_tableaux`

### Attributs protégés statiques

- static const `UtilIndexDeformation` `ccdex`
- static list< `Tableau`< `TenseurBH` \* > > `list_var_tens_BH`
- static list< `Tableau`< `Tableau`< `TenseurBH` \* > > > `list_var_var_tens_BH`
- static list< `Tableau`< `Coordonnee` > > `list_tab_coor`
- static int `nombre_d_instance_deformation` = 0
- static int `indic_VerifCal_implicit` = 0

### 6.224.1 Description détaillée

BUT: Calcul des differentes grandeurs liee a la deformation La classe fonctionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stocke pas (ou très peu).

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

**6.224.2 Documentation des fonctions membres****6.224.2.1 Cal\_explicit\_t()**

```
const Met_abstraite::Expli & Deformation::Cal_explicit_t (
    const Tableau< double > & def_equi_t,
    TenseurBB & epsBB_t,
    Tableau< TenseurBB * > & d_epsBB,
    Tableau< double > & def_equi,
    TenseurBB & DepsBB,
    TenseurBB & DeltaEpsBB,
    bool premier_calcul ) [virtual]
```

Réimplémentée dans [DeformationPP](#).**6.224.2.2 Cal\_explicit\_tdt()**

```
const Met_abstraite::Expli_t_tdt & Deformation::Cal_explicit_tdt (
    const Tableau< double > & def_equi_t,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    Tableau< double > & def_equi,
    TenseurBB & DepsBB,
    TenseurBB & delta_epsBB_tdt,
    bool premier_calcul ) [virtual]
```

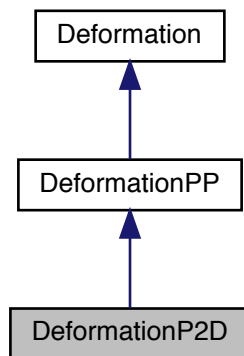
Réimplémentée dans [DeformationPP](#).

La documentation de cette classe a été générée à partir du fichier suivant :

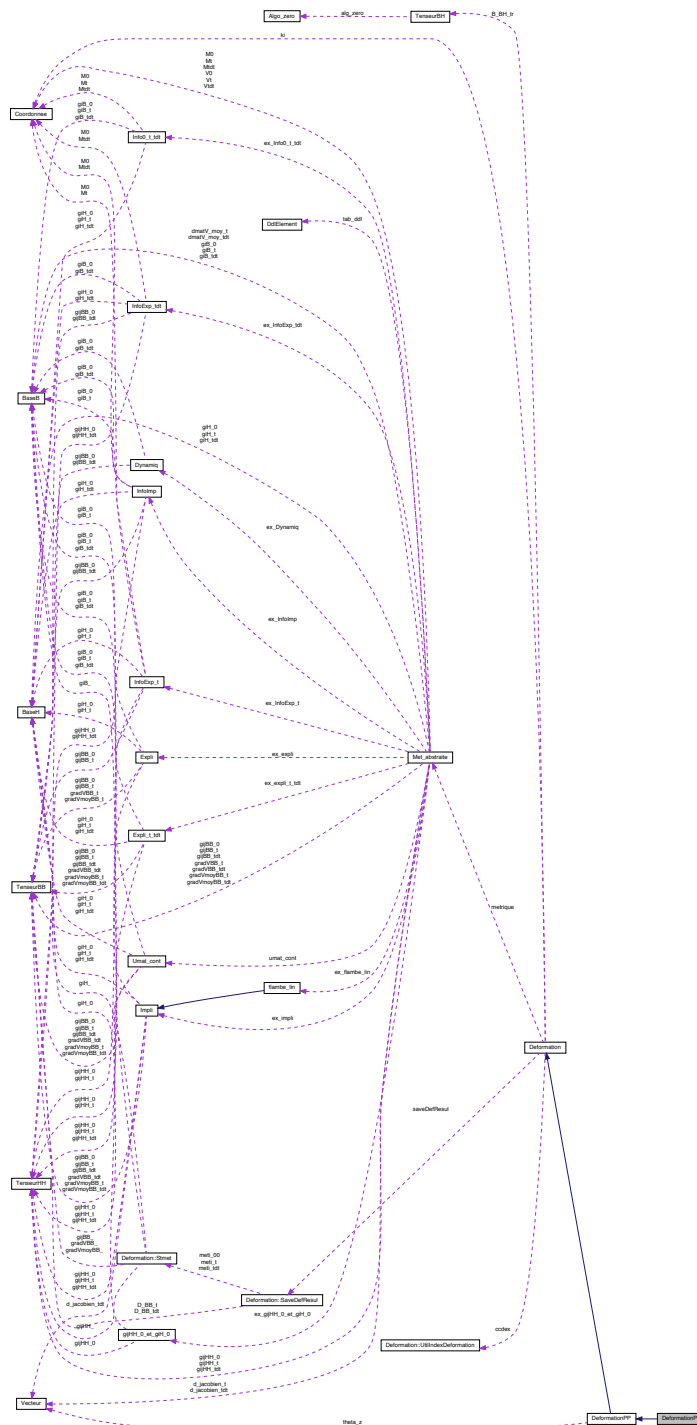
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation.↵  
h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation.↵  
cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation↵  
\_2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation↵  
\_Almansi.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation↵  
\_log.cc

## 6.225 Référence de la classe DeformationP2D

Graphe d'héritage de DeformationP2D:



Graphe de collaboration de DeformationP2D:



## Fonctions membres publiques

- **DeformationP2D** (**Met\_abstraite** &a, **Tableau**< **Noeud** \* > &tabnoeud, **Tableau**< **Mat\_pleine** > const &tabDphiH, **Tableau**< **Vecteur** > const &tabPhiH, **Tableau**< **Mat\_pleine** > const &tabDphiS, **Tableau**< **Vecteur** > const &tabPhiS, **Tableau**< **Vecteur** > const &tabD2phi)
- **DeformationP2D** (const **DeformationP2D** &)
- virtual **Deformation** \* **Nevez\_deformation** (**Tableau**< **Noeud** \* > &tabN) const
- **Deformation** & operator= (const **Deformation** &def)
- **DeformationPP** & operator= (const **DeformationPP** &def)

- [DeformationP2D](#) & **operator=** (const [DeformationP2D](#) &def)
- void [ChangeNumIntegSH](#) (int *niaxpl*, int *niepaiss*)

### Attributs protégés

- [Tableau](#)< [Vecteur](#) > const \* **tabD2phi**

### Membres hérités additionnels

#### 6.225.1 Documentation des fonctions membres

##### 6.225.1.1 [ChangeNumIntegSH\(\)](#)

```
void DeformationP2D::ChangeNumIntegSH (
    int niaxpl,
    int niepaiss ) [virtual]
```

Réimplémentée à partir de [DeformationPP](#).

##### 6.225.1.2 [Nevez\\_deformation\(\)](#)

```
virtual Deformation * DeformationP2D::Nevez_deformation (
    Tableau< Noeud * > & tabN ) const [inline], [virtual]
```

Réimplémentée à partir de [DeformationPP](#).

##### 6.225.1.3 [operator=\(\)](#) [1/2]

```
Deformation & DeformationP2D::operator= (
    const Deformation & def ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

##### 6.225.1.4 [operator=\(\)](#) [2/2]

```
DeformationPP & DeformationP2D::operator= (
    const DeformationPP & def ) [virtual]
```

Réimplémentée à partir de [DeformationPP](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/DeformationP2D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/DeformationP2D.cc

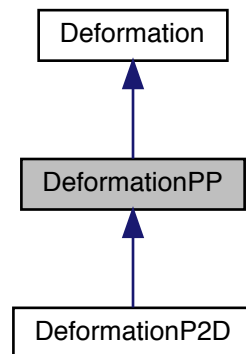
## 6.226 Référence de la classe [DeformationPP](#)

BUT: Calcul des différentes grandeurs liée à la déformation d'éléments poutres et plaques classiques. Par rapport à la classe [Deformation](#) de base, ici on considère deux directions : l'épaisseur, et l'axe ou le plan dans ces deux directions, il y a des fcts d'interpolation parti-culières. La classe fonctionne comme une boîte à outil. On y choisit ce dont on a besoin. Bien faire attention à l'ordre d'appel des différentes méthodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stock pas.

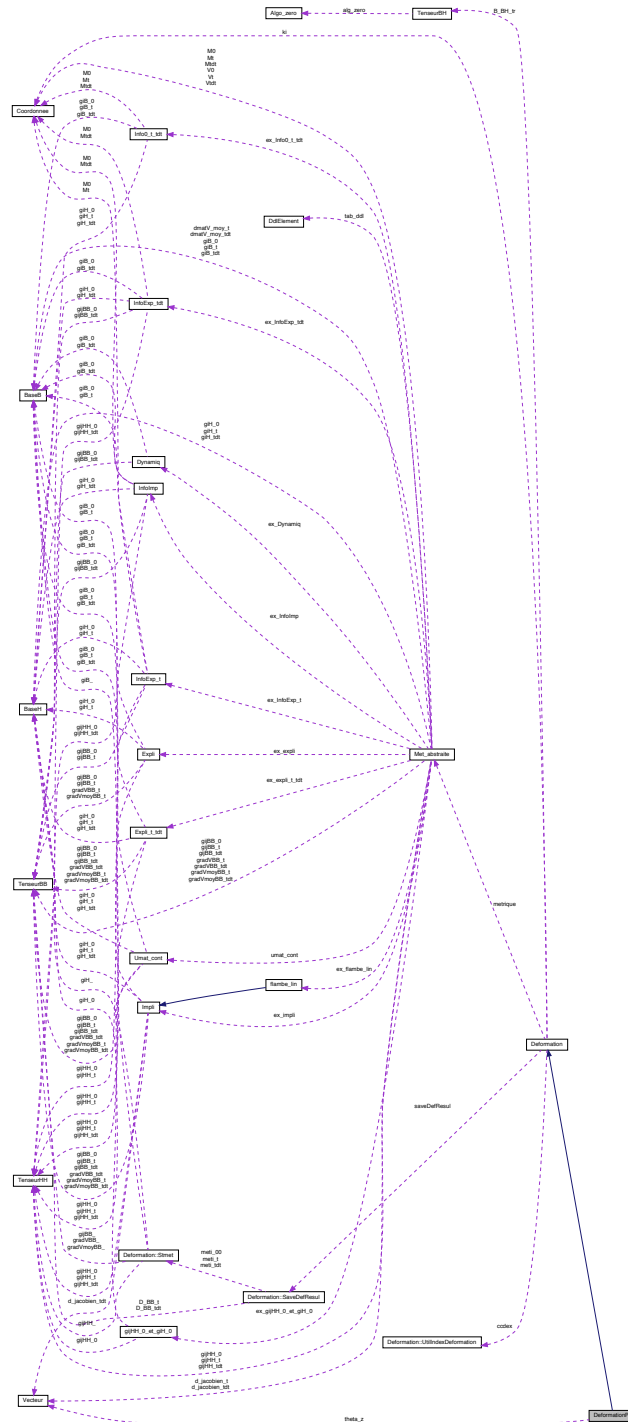
```
#include <DeformationPP.h>
```



Grphe d'héritage de DeformationPP:



Graphe de collaboration de DeformationPP:



**Fonctions membres publiques**

- **DeformationPP** (**Met\_abstraite** &, **Tableau**< **Noeud** \* > &tabnoeud, **Tableau**< **Mat\_pleine** > const &tabDphiH, **Tableau**< **Vecteur** > const &tabPhiH, **Tableau**< **Mat\_pleine** > const &tabDphiS, **Tableau**< **Vecteur** > const &tabPhiS)
- **DeformationPP** (const **DeformationPP** &)
- virtual **Deformation** \* **Nevez\_deformation** (**Tableau**< **Noeud** \* > &tabN) const
- **Deformation** & operator= (const **Deformation** &def)
- virtual **DeformationPP** & operator= (const **DeformationPP** &def)

- virtual void **ChangeNumIntegSH** (int niaxpl, int niepais)
- const Met\_abstraite::Expli & **Cal\_explicit\_t** (const Tableau< double > &def\_equi\_t, TenseurBB &epsBB\_↵  
t, Tableau< TenseurBB \* > &d\_epsBB, Tableau< double > &def\_equi, TenseurBB &DepsBB, TenseurBB  
&DeltaEpsBB, bool premier\_calcul)
- const Met\_abstraite::Expli\_t\_tdt & **Cal\_explicit\_tdt** (const Tableau< double > &def\_equi\_t, TenseurBB  
&epsBB\_tdt, Tableau< TenseurBB \* > &d\_epsBB, Tableau< double > &def\_equi, TenseurBB &DepsBB,  
TenseurBB &delta\_epsBB\_tdt, bool premier\_calcul)
- const Met\_abstraite::Impli & **Cal\_implicit** (const Tableau< double > &def\_equi\_t, TenseurBB &epsBB\_tdt,  
Tableau< TenseurBB \* > &d\_epsBB, Tableau< double > &def\_equi, Tableau2< TenseurBB \* > &d2\_↵  
epsBB\_tdt, TenseurBB &DepsBB, TenseurBB &delta\_epsBB, bool premier\_calcul)
- const Met\_abstraite::Expli & **Cal\_explicit\_t** (bool premier\_calcul)
- const Met\_abstraite::Expli\_t\_tdt & **Cal\_explicit\_tdt** (bool premier\_calcul)
- const Met\_abstraite::Impli & **Cal\_implicit** (bool premier\_calcul)
- const Met\_abstraite::InfoImp **RemontImp** (bool absolue, Mat\_pleine &Aa0, Mat\_pleine &Aafin)
- const Met\_abstraite::InfoImp **RemontImp** ()
- const Met\_abstraite::InfoExp\_t **RemontExp\_t** (bool absolue, Mat\_pleine &Aa0, Mat\_pleine &Aafin)
- const Met\_abstraite::InfoExp\_t **RemontExp\_t** ()
- const Met\_abstraite::InfoExp\_tdt **RemontExp\_tdt** (bool absolue, Mat\_pleine &Aa0, Mat\_pleine &Aafin)
- const Met\_abstraite::InfoExp\_tdt **RemontExp\_tdt** ()
- virtual void **PremierPtInteg** ()
- virtual bool **DernierPtInteg** ()
- virtual void **NevezPtInteg** ()
- void **BasePassage** (bool absolue, const BaseB &giB0, const BaseB &giB, const BaseH &giH0, const BaseH  
&giH, Mat\_pleine &Aa0, Mat\_pleine &Aafin)

### Fonctions membres protégées

- void **BasePassage** (BaseB &giB0, BaseB &giB, BaseH &giH0, BaseH &giH, Mat\_pleine &Aa0, Mat\_pleine  
&Aafin)
- void **AppliquePtInteg** ()

### Attributs protégés

- Tableau< Mat\_pleine > const \* **tabDphiH**
- Tableau< Vecteur > const \* **tabPhiH**

### Attributs protégés statiques

- static int **numInteg\_ep** = 0
- static int **nbtotalep** = 0
- static int **nbtotalaxpl** = 0
- static Vecteur **theta\_z**
- static Tableau< Mat\_pleine const \* > **taDphi**
- static Tableau< Vecteur const \* > **taPhi**

## 6.226.1 Description détaillée

BUT: Calcul des différentes grandeurs liée à la déformation d'éléments poutres et plaques classiques. Par rapport à la classe [Deformation](#) de base, ici on considère deux directions : l'épaisseur, et l'axe ou le plan dans ces deux directions, il y a des fcts d'interpolation parti-culières. La classe fonctionne comme une boîte à outil. On y choisit ce dont on a besoin. Bien faire attention à l'ordre d'appel des différentes méthodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stock pas.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

25/05/98

## 6.226.2 Documentation des fonctions membres

### 6.226.2.1 BasePassage()

```
void DeformationPP::BasePassage (
    bool absolue,
    const BaseB & giB0,
    const BaseB & giB,
    const BaseH & giH0,
    const BaseH & giH,
    Mat_pleine & Aa0,
    Mat_pleine & AaFin ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

### 6.226.2.2 Cal\_explicit\_t() [1/2]

```
const Met_abstraite::Expli & DeformationPP::Cal_explicit_t (
    bool premier_calcul ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

### 6.226.2.3 Cal\_explicit\_t() [2/2]

```
const Met_abstraite::Expli & DeformationPP::Cal_explicit_t (
    const Tableau< double > & def_equi_t,
    TenseurBB & epsBB_t,
    Tableau< TenseurBB * > & d_epsBB,
    Tableau< double > & def_equi,
    TenseurBB & DepsBB,
    TenseurBB & DeltaEpsBB,
    bool premier_calcul ) [virtual]
```

voir [Deformation](#) car ça a changé!!

Réimplémentée à partir de [Deformation](#).

### 6.226.2.4 Cal\_explicit\_tdt() [1/2]

```
const Met_abstraite::Expli_t_tdt & DeformationPP::Cal_explicit_tdt (
    bool premier_calcul ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

### 6.226.2.5 Cal\_explicit\_tdt() [2/2]

```
const Met_abstraite::Expli_t_tdt & DeformationPP::Cal_explicit_tdt (
    const Tableau< double > & def_equi_t,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    Tableau< double > & def_equi,
    TenseurBB & DepsBB,
    TenseurBB & delta_epsBB_tdt,
    bool premier_calcul ) [virtual]
```

voir [Deformation](#) car ça a changé!!

Réimplémentée à partir de [Deformation](#).

**6.226.2.6 Cal\_implicit()**

```
const Met_abstraite::Impli & DeformationPP::Cal_implicit (
    const Tableau< double > & def_equi_t,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    Tableau< double > & def_equi,
    Tableau2< TenseurBB * > & d2_epsBB_tdt,
    TenseurBB & DepsBB,
    TenseurBB & delta_epsBB,
    bool premier_calcul )
```

voir [Deformation](#) car ça a changé!!

**6.226.2.7 DernierPtInteg()**

```
bool DeformationPP::DernierPtInteg ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.8 Nevez\_deformation()**

```
virtual Deformation * DeformationPP::Nevez_deformation (
    Tableau< Noeud * > & tabN ) const [inline], [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.9 NevezPtInteg()**

```
void DeformationPP::NevezPtInteg ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.10 operator=()**

```
Deformation & DeformationPP::operator= (
    const Deformation & def ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.11 PremierPtInteg()**

```
void DeformationPP::PremierPtInteg ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.12 RemontExp\_t() [1/2]**

```
const Met_abstraite::InfoExp_t DeformationPP::RemontExp_t ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.13 RemontExp\_t() [2/2]**

```
const Met_abstraite::InfoExp_t DeformationPP::RemontExp_t (
    bool absolue,
    Mat_pleine & Aa0,
    Mat_pleine & Aa1in ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.14 RemontExp\_tdt()** [1/2]

```
const Met_abstraite::InfoExp_tdt DeformationPP::RemontExp_tdt ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.15 RemontExp\_tdt()** [2/2]

```
const Met_abstraite::InfoExp_tdt DeformationPP::RemontExp_tdt (
    bool absolue,
    Mat_pleine & Aa0,
    Mat_pleine & Aafin ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.16 RemontImp()** [1/2]

```
const Met_abstraite::InfoImp DeformationPP::RemontImp ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.226.2.17 RemontImp()** [2/2]

```
const Met_abstraite::InfoImp DeformationPP::RemontImp (
    bool absolue,
    Mat_pleine & Aa0,
    Mat_pleine & Aafin ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

La documentation de cette classe a été générée à partir du fichier suivant :

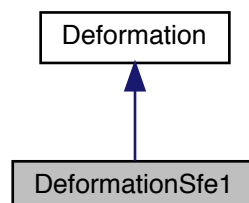
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation↔PP.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation↔PP.cc

**6.227 Référence de la classe DeformationSfe1**

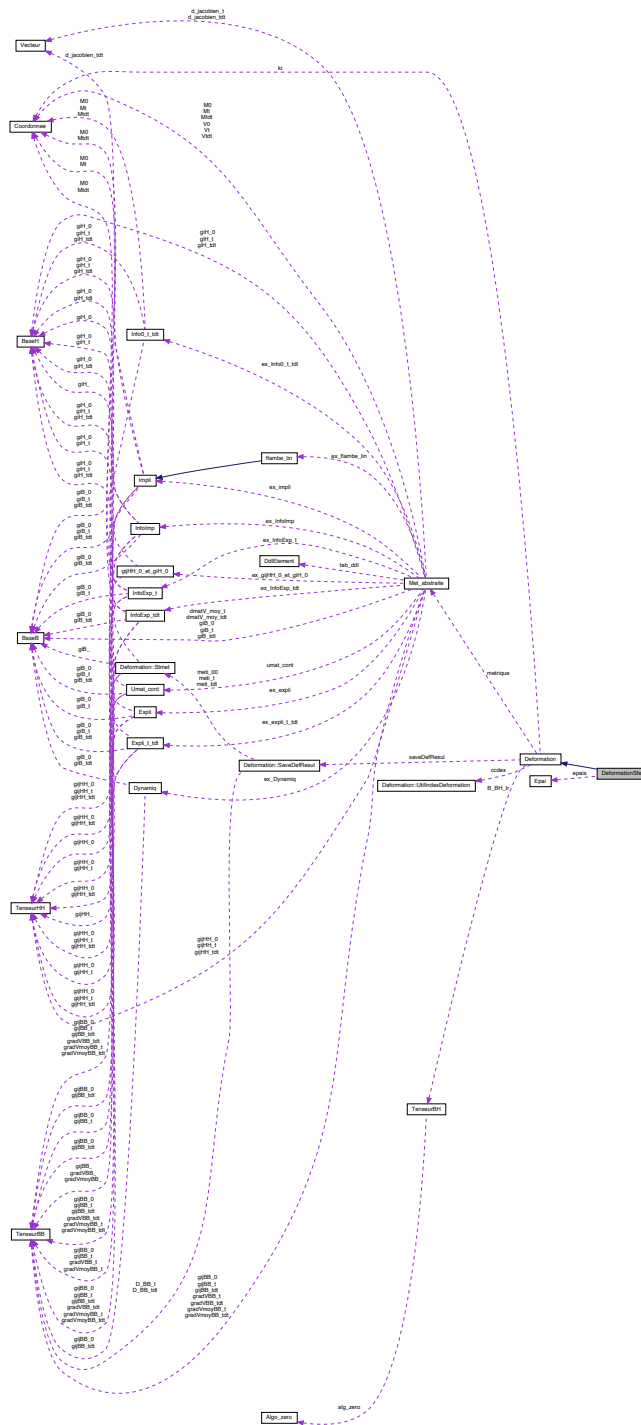
BUT: Calcul des differentes grandeurs liee a la deformation des elements sfe1 La classe fonctionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stock pas.

```
#include <DeformationSfe1.h>
```

Graphe d'héritage de DeformationSfe1:



Graphe de collaboration de DeformationSfe1:



**Classes**

- class [SaveDefResultSfe1](#)

**Fonctions membres publiques**

- **DeformationSfe1** ([Met\\_abstraite](#) &, [Tableau](#)< [Noeud](#) \* > &tabnoeud, [Tableau](#)< [Mat\\_pleine](#) > const &tabDphiH, [Tableau](#)< [Vecteur](#) > const &tabPhiH, [Tableau](#)< [Mat\\_pleine](#) > const &tabDphiS, [Tableau](#)< [Vecteur](#) > const &tabPhiS)

- **DeformationSfe1** (const [DeformationSfe1](#) &)
- virtual [SaveDefResul](#) \* [New\\_et\\_Initialise](#) ()
- virtual void [AfficheDataSpecif](#) (ofstream &sort, [SaveDefResul](#) \*a) const
- void **Mise\_a\_jour\_data\_specif** ([Deformation::SaveDefResul](#) \*don)
- virtual [Deformation](#) \* [Nevez\\_deformation](#) ([Tableau](#)< [Noeud](#) \* > &tabN) const
- [Deformation](#) & [operator=](#) (const [Deformation](#) &def)
- void **RenseigneCondLim** (const [Tableau](#)< [EnuTypeCL](#) > &arTypeCL, const [Tableau](#)< [Coordonnee3](#) > &vpla)
- virtual void [Affiche](#) () const
- virtual const [Met\\_abstraite::Expli](#) & [Cal\\_explicit\\_t](#) (const [Tableau](#)< double > &def\_equi\_t, [TenseurBB](#) &epsBB\_t, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< double > &def\_equi, [TenseurBB](#) &DepsBB, [TenseurBB](#) &DeltaEpsBB, bool premier\_calcul)
- virtual const [Met\\_abstraite::Expli\\_t\\_tdt](#) & [Cal\\_explicit\\_tdt](#) (const [Tableau](#)< double > &def\_equi\_t, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< double > &def\_equi, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB\_tdt, bool premier\_calcul)
- const [Met\\_abstraite::Impli](#) & [Cal\\_implicit](#) (const [Tableau](#)< double > &def\_equi\_t, [TenseurBB](#) &epsBB\_↔\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< double > &def\_equi, [TenseurBB](#) &DepsBB, [TenseurBB](#) &delta\_epsBB, bool premier\_calcul)
- virtual const [Met\\_abstraite::Expli](#) & [Cal\\_explicit\\_t](#) (bool premier\_calcul)
- virtual const [Met\\_abstraite::Expli\\_t\\_tdt](#) & [Cal\\_explicit\\_tdt](#) (bool premier\_calcul)
- const [Met\\_abstraite::Impli](#) & [Cal\\_implicit](#) (bool premier\_calcul)
- int **Test\_courbure\_anormale** ([Enum\\_dure](#) temps, double inf\_normale)
- virtual const [Met\\_abstraite::InfoImp](#) [RemontImp](#) (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite::InfoImp](#) [RemontImp](#) ()
- virtual const [Met\\_abstraite::InfoExp\\_t](#) [RemontExp\\_t](#) (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite::InfoExp\\_t](#) [RemontExp\\_t](#) ()
- virtual const [Met\\_abstraite::InfoExp\\_tdt](#) [RemontExp\\_tdt](#) (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual const [Met\\_abstraite::InfoExp\\_tdt](#) [RemontExp\\_tdt](#) ()
- virtual const [Met\\_abstraite::Info0\\_t\\_tdt](#) [Remont0\\_t\\_tdt](#) (bool absolue, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aat, [Mat\\_pleine](#) &Aatdt)
- virtual const [Met\\_abstraite::Info0\\_t\\_tdt](#) [Remont0\\_t\\_tdt](#) ()
- virtual const [Met\\_Sfe1::Courbure\\_t\\_tdt](#) & **RecupCourbure0\_t\_tdt** () const
- virtual void [ChangeNumInteg](#) (int ni)
- void **ChangeNumIntegSfe1** (int nisurf, int niepaiss)
- void [PremierPtInteg](#) ()
- bool [DernierPtInteg](#) ()
- void [NevezPtInteg](#) ()
- virtual void [Retour\\_pti\\_precedant](#) ()
- int **Nb\_pt\_int\_surf** () const
- int **Nb\_pt\_int\_epai** () const
- void **Change\_epaisseur** (const [Epai](#) &epaisse)
- virtual const [Coordonnee](#) & [Position\\_0](#) ()
- virtual const [Coordonnee](#) & [Position\\_t](#) ()
- virtual const [Coordonnee](#) & [Position\\_tdt](#) ()
- virtual void [BasePassage](#) (bool absolue, const [BaseB](#) &giB0, const [BaseB](#) &giB, const [BaseH](#) &giH0, const [BaseH](#) &giH, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aafin)
- virtual void [BasePassage](#) (bool absolue, const [BaseB](#) &giB0, const [BaseB](#) &giB\_t, const [BaseB](#) &giB\_↔\_tdt, const [BaseH](#) &giH0, const [BaseH](#) &giH\_t, const [BaseH](#) &giH\_tdt, [Mat\\_pleine](#) &Aa0, [Mat\\_pleine](#) &Aa\_t, [Mat\\_pleine](#) &Aa\_tdt)

### Fonctions membres protégées

- void **VerifCal\_def** (bool gradV\_instantane, const [Met\\_abstraite::Impli](#) &ex, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB\_tdt)
- void **VerifCal\_implicit** (bool gradV\_instantane, const [Met\\_abstraite::Impli](#) &ex)

### Attributs protégés

- [Tableau](#)< [Mat\\_pleine](#) > const \* **tabDphiH**
- [Tableau](#)< [Vecteur](#) > const \* **tabPhiH**
- [Epai](#) \* **epais**
- [Tableau](#)< [EnuTypeCL](#) > const \* **tabTypeCL**
- [Tableau](#)< [Coordonnee3](#) > const \* **vplan**



- int numInteg\_ep
  - int numInteg\_surf
  - int sauve\_numInteg\_ep
  - int sauve\_numInteg\_surf
- sauvegarde pour un retour au pti précédant avec la méthode Retour\_pti\_precedant();*

### Attributs protégés statiques

- static int indic\_VerifCal\_implicitSfe1 = 0

#### 6.227.1 Description détaillée

BUT: Calcul des differentes grandeurs liee a la deformation des elements sfe1 La classe fonctionne comme une boite a outil. On y choisit ce dont on a besoin. Bien faire attention a l'ordre d'appel des differentes methodes lorsque il faut suivre une chronologie. Cette classe calcul mais ne stock pas.

##### Auteur

Gérard Rio

##### Version

1.0

##### Date

23/01/97

#### 6.227.2 Documentation des fonctions membres

##### 6.227.2.1 Affiche()

```
void DeformationSfe1::Affiche ( ) const [virtual]
```

Réimplémentée à partir de [Deformation](#).

##### 6.227.2.2 AfficheDataSpecif()

```
virtual void DeformationSfe1::AfficheDataSpecif (
    ostream & sort,
    SaveDefResul * a ) const [inline], [virtual]
```

Réimplémentée à partir de [Deformation](#).

##### 6.227.2.3 BasePassage() [1/2]

```
void DeformationSfe1::BasePassage (
    bool absolute,
    const BaseB & giB0,
    const BaseB & giB,
    const BaseH & giH0,
    const BaseH & giH,
    Mat_pleine & Aa0,
    Mat_pleine & Aaafin ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.4 BasePassage() [2/2]**

```
void DeformationSfel::BasePassage (
    bool absolue,
    const BaseB & giB0,
    const BaseB & giB_t,
    const BaseB & giB_tdt,
    const BaseH & giH0,
    const BaseH & giH_t,
    const BaseH & giH_tdt,
    Mat_pleine & Aa0,
    Mat_pleine & Aa_t,
    Mat_pleine & Aa_tdt ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.5 Cal\_explicit\_t() [1/2]**

```
const Met_abstraite::Expli & DeformationSfel::Cal_explicit_t (
    bool premier_calcul ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.6 Cal\_explicit\_t() [2/2]**

```
const Met_abstraite::Expli & DeformationSfel::Cal_explicit_t (
    const Tableau< double > & def_equi_t,
    TenseurBB & epsBB_t,
    Tableau< TenseurBB * > & d_epsBB,
    Tableau< double > & def_equi,
    TenseurBB & DepsBB,
    TenseurBB & DeltaEpsBB,
    bool premier_calcul ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.7 Cal\_explicit\_tdt() [1/2]**

```
const Met_abstraite::Expli_t_tdt & DeformationSfel::Cal_explicit_tdt (
    bool premier_calcul ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.8 Cal\_explicit\_tdt() [2/2]**

```
const Met_abstraite::Expli_t_tdt & DeformationSfel::Cal_explicit_tdt (
    const Tableau< double > & def_equi_t,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    Tableau< double > & def_equi,
    TenseurBB & DepsBB,
    TenseurBB & delta_epsBB_tdt,
    bool premier_calcul ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.9 Cal\_implicit()**

```
const Met_abstraite::Impli & DeformationSfel::Cal_implicit (
    const Tableau< double > & def_equi_t,
```

```

TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
Tableau< double > & def_equi,
TenseurBB & DepsBB,
TenseurBB & delta_epsBB,
bool premier_calcul ) [virtual]

```

Réimplémentée à partir de [Deformation](#).

#### 6.227.2.10 ChangeNumInteg()

```

void DeformationSfe1::ChangeNumInteg (
    int ni ) [virtual]

```

Réimplémentée à partir de [Deformation](#).

#### 6.227.2.11 DernierPtInteg()

```

bool DeformationSfe1::DernierPtInteg ( ) [virtual]

```

Réimplémentée à partir de [Deformation](#).

#### 6.227.2.12 Nevez\_deformation()

```

virtual Deformation * DeformationSfe1::Nevez_deformation (
    Tableau< Noeud * > & tabN ) const [inline], [virtual]

```

Réimplémentée à partir de [Deformation](#).

#### 6.227.2.13 NevezPtInteg()

```

void DeformationSfe1::NevezPtInteg ( ) [virtual]

```

Réimplémentée à partir de [Deformation](#).

#### 6.227.2.14 New\_et\_Initialise()

```

virtual SaveDefResul * DeformationSfe1::New_et_Initialise ( ) [inline], [virtual]

```

Réimplémentée à partir de [Deformation](#).

#### 6.227.2.15 operator=()

```

Deformation & DeformationSfe1::operator= (
    const Deformation & def ) [inline], [virtual]

```

Réimplémentée à partir de [Deformation](#).

#### 6.227.2.16 Position\_0()

```

virtual const Coordonnee & DeformationSfe1::Position_0 ( ) [inline], [virtual]

```

Réimplémentée à partir de [Deformation](#).

#### 6.227.2.17 Position\_t()

```

virtual const Coordonnee & DeformationSfe1::Position_t ( ) [inline], [virtual]

```

Réimplémentée à partir de [Deformation](#).

**6.227.2.18 Position\_tdt()**

```
virtual const Coordonnee & DeformationSfel::Position_tdt ( ) [inline], [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.19 PremierPtInteg()**

```
void DeformationSfel::PremierPtInteg ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.20 Remont0\_t\_tdt() [1/2]**

```
const Met_abstraite::Info0_t_tdt DeformationSfel::Remont0_t_tdt ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.21 Remont0\_t\_tdt() [2/2]**

```
const Met_abstraite::Info0_t_tdt DeformationSfel::Remont0_t_tdt (
    bool absolue,
    Mat_pleine & Aa0,
    Mat_pleine & Aat,
    Mat_pleine & Aatdt ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.22 RemontExp\_t() [1/2]**

```
const Met_abstraite::InfoExp_t DeformationSfel::RemontExp_t ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.23 RemontExp\_t() [2/2]**

```
const Met_abstraite::InfoExp_t DeformationSfel::RemontExp_t (
    bool absolue,
    Mat_pleine & Aa0,
    Mat_pleine & Aa $fin$  ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.24 RemontExp\_tdt() [1/2]**

```
const Met_abstraite::InfoExp_tdt DeformationSfel::RemontExp_tdt ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.25 RemontExp\_tdt() [2/2]**

```
const Met_abstraite::InfoExp_tdt DeformationSfel::RemontExp_tdt (
    bool absolue,
    Mat_pleine & Aa0,
    Mat_pleine & Aa $fin$  ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.26 RemontImp()** [1/2]

```
const Met_abstraite::InfoImp DeformationSfe1::RemontImp ( ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.27 RemontImp()** [2/2]

```
const Met_abstraite::InfoImp DeformationSfe1::RemontImp (
    bool absolue,
    Mat_pleine & Aa0,
    Mat_pleine & Aafin ) [virtual]
```

Réimplémentée à partir de [Deformation](#).

**6.227.2.28 Retour\_pti\_precedant()**

```
void DeformationSfe1::Retour_pti_precedant ( ) [virtual]
```

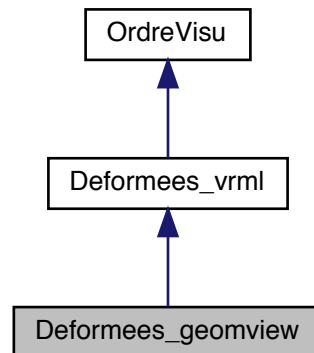
Réimplémentée à partir de [Deformation](#).

La documentation de cette classe a été générée à partir du fichier suivant :

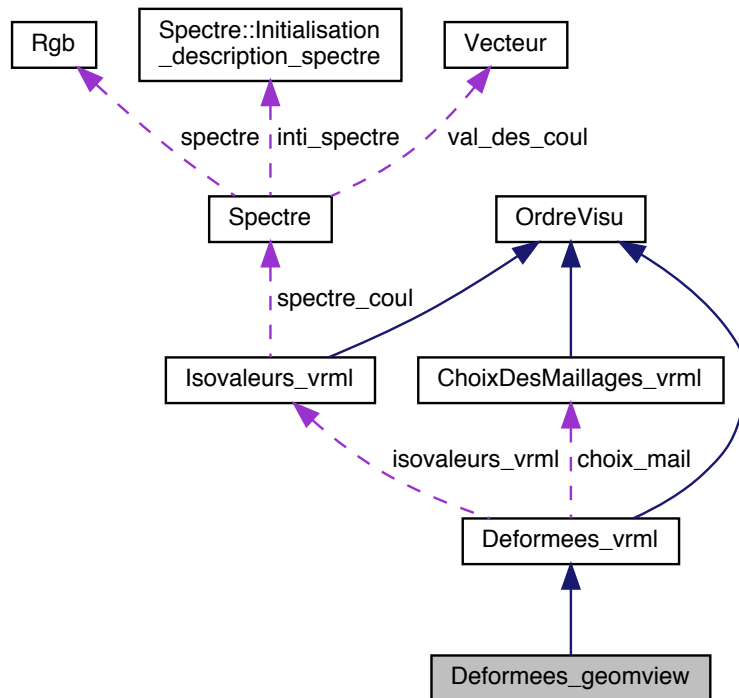
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/DeformationSfe1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/DeformationSfe1.cc

**6.228 Référence de la classe Deformees\_geomview**

Graphe d'héritage de Deformees\_geomview:



Graphe de collaboration de Deformees\_geomview:



## Fonctions membres publiques

- **Deformees\_geomview** (const string &comment\_som, const string &explication, const string &ordre)
- **Deformees\_geomview** (const [Deformees\\_geomview](#) &algo)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu](#)::EnumTypeIncre type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob)

## Membres hérités additionnels

### 6.228.1 Documentation des fonctions membres

#### 6.228.1.1 ExeOrdre()

```
void Deformees_geomview::ExeOrdre (
    ParaGlob * paraGlob,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
```

```

LesCondLim * ,
LesContacts * ,
Resultats * ,
UtilLecture & entreePrinc,
OrdreVisu::EnumTypeIncre type_incre,
int incre,
bool animation,
const map< string, const double *, std::less< string > > & listeVarGlob,
const List_io< TypeQuelconque > & listeVecGlob ) [virtual]

```

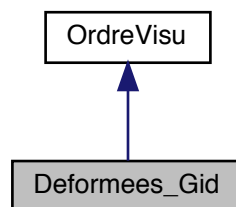
Réimplémentée à partir de [Deformees\\_vrml](#).

La documentation de cette classe a été générée à partir du fichier suivant :

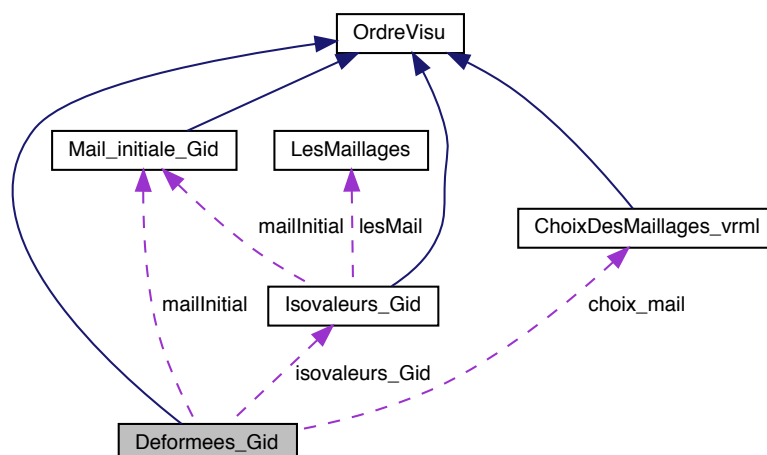
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Deformees\_geomview.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Deformees\_geomview.cc

## 6.229 Référence de la classe Deformees\_Gid

Graphe d'héritage de Deformees\_Gid:



Graphe de collaboration de Deformees\_Gid:



## Fonctions membres publiques

- `Deformees_Gid` (const string &comment\_som, const string &explication, const string &ordre)
- `Deformees_Gid` (const `Deformees_Gid` &algo)
- void `ExeOrdre` (`ParaGlob` \*, const `Tableau`< int > &tab\_mail, `LesMaillages` \*, bool unseul\_incre, `LesReferences` \*, `LesLoisDeComp` \*, `DiversStockage` \*, Charge \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*, `UtilLecture` &entreePrinc, `OrdreVisu::EnumTypeIncre` type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const `List_io`< `TypeQuelconque` > &listeVec↵Glob)
- void `ChoixOrdre` ()
- void `Jonction_isevaleur` (const `Isovaleurs_Gid` \*iso)
- void `Jonction_ChoixDesMaillages` (const `ChoixDesMaillages_vrml` \*choix\_m)
- void `Jonction_MaillagelInitiale` (const `Mail_initiale_Gid` \*maillni)
- const list< string > & `Noms_matiere` () const
- const list< string > & `Noms_couleurs` () const
- void `Lecture_parametres_OrdreVisu` (`UtilLecture` &entreePrinc)
- void `Ecriture_parametres_OrdreVisu` (`UtilLecture` &entreePrinc)

## Attributs protégés

- list< `DeuxDoubles` > `li_bornes`
- list< string > `li_nom_bornes`
- const `Mail_initiale_Gid` \* `maillinitial`
- const `Isovaleurs_Gid` \* `isovaleurs_Gid`
- list< `Deuxentiers` > `num_mail_incr`
- list< string > `noms_matiere`
- list< string > `noms_couleurs`
- const `ChoixDesMaillages_vrml` \* `choix_mail`

## Membres hérités additionnels

### 6.229.1 Documentation des fonctions membres

#### 6.229.1.1 ChoixOrdre()

```
void Deformees_Gid::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.229.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Deformees_Gid::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.229.1.3 ExeOrdre()

```
void Deformees_Gid::ExeOrdre (
    ParaGlob * paraGlob,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
```



```

Resultats * ,
UtilLecture & entreePrinc,
OrdreVisu::EnumTypeIncre type_incre,
int incre,
bool animation,
const map< string, const double *, std::less< string > > & listeVarGlob,
const List_io< TypeQuelconque > & listeVecGlob ) [virtual]

```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.229.1.4 Lecture\_parametres\_OrdreVisu()

```

void Deformees_Gid::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]

```

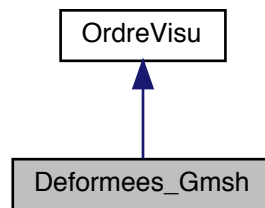
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

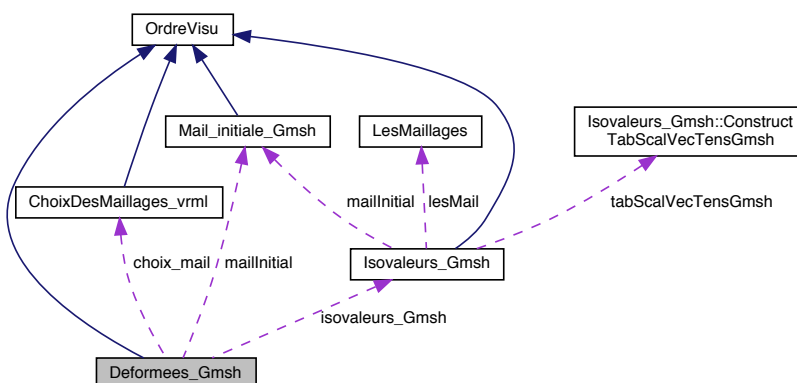
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Deformees\_Gid.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Deformees\_Gid.cc

## 6.230 Référence de la classe Deformees\_Gmsh

Graphe d'héritage de Deformees\_Gmsh:



Graphe de collaboration de Deformees\_Gmsh:



## Fonctions membres publiques

- **Deformees\_Gmsh** (const string &comment\_som, const string &explication, const string &ordre)
- **Deformees\_Gmsh** (const [Deformees\\_Gmsh](#) &algo)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, Charge \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec←Glob)
- void **ChoixOrdre** ()
- void **Jonction\_ivoaleur** (const [Isovaleurs\\_Gmsh](#) \*iso)
- void **Jonction\_ChoixDesMaillages** (const [ChoixDesMaillages\\_vrml](#) \*choix\_m)
- void **Jonction\_MaillageInitiale** ([Mail\\_initiale\\_Gmsh](#) \*maillni)
- const list< string > & **Noms\_matiere** () const
- const list< string > & **Noms\_couleurs** () const
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- const string & **Nom\_deplace** () const
- const string \* **Nom\_Vitesse** () const
- const string \* **Nom\_Acceleration** () const
- void **SortieDeformee** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail, ostream &sort, int incre)
- void **SortieVitesse** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail, ostream &sort, int incre)
- void **SortieAcceleration** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail, ostream &sort, int incre)

## Fonctions membres protégées

- void **SortieDeformee\_ancien\_format** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail, ostream &sort, int incre)

## Attributs protégés

- list< [DeuxDoubles](#) > **li\_bornes**
- list< string > **li\_nom\_bornes**
- [Mail\\_initiale\\_Gmsh](#) \* **maillinitial**
- int **avec\_vitesse**
- int **avec\_acceleration**
- string **nom\_deplace**
- string **nom\_vitesse**
- string **nom\_acceleration**
- const [Isovaleurs\\_Gmsh](#) \* **isovaleurs\_Gmsh**
- list< [Deuxentiers](#) > **num\_mail\_incr**
- list< string > **noms\_matiere**
- list< string > **noms\_couleurs**
- const [ChoixDesMaillages\\_vrml](#) \* **choix\_mail**

## Membres hérités additionnels

### 6.230.1 Documentation des fonctions membres

#### 6.230.1.1 ChoixOrdre()

```
void Deformees_Gmsh::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.230.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Deformees_Gmsh::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.230.1.3 ExeOrdre()

```
void Deformees_Gmsh::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.230.1.4 Lecture\_parametres\_OrdreVisu()

```
void Deformees_Gmsh::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

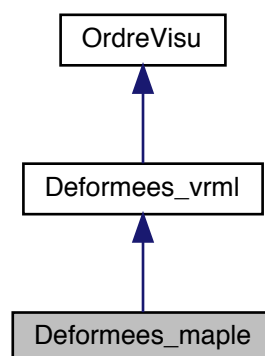
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

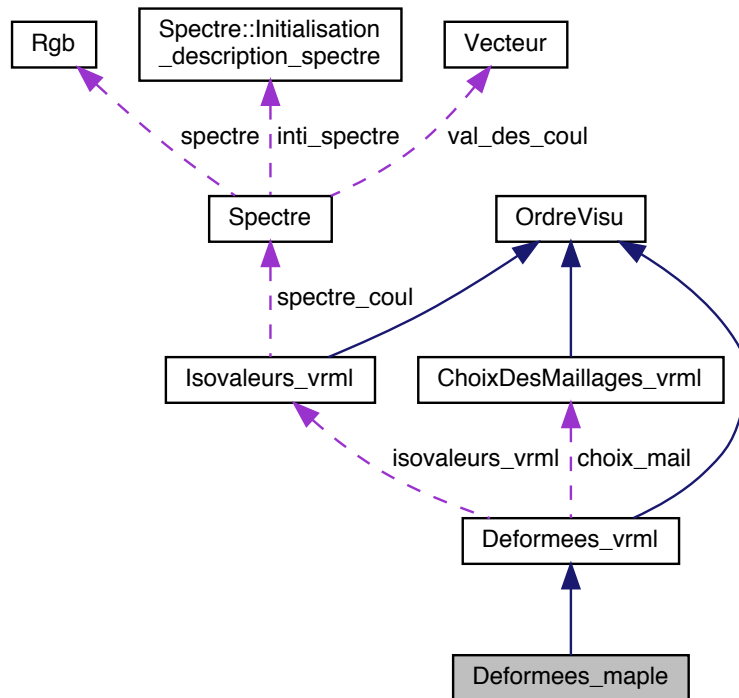
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Deformees\_Gmsh.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Deformees\_Gmsh.cc

## 6.231 Référence de la classe Deformees\_maple

Graphe d'héritage de Deformees\_maple:



Graphe de collaboration de Deformees\_maple:



## Fonctions membres publiques

- **Deformees\_maple** (const [Deformees\\_maple](#) &algo)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu](#)::EnumTypeIncre type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_je](#)< [TypeQuelconque](#) > &listeVec↔Glob)
- void **ChoixOrdre** ()

## Membres hérités additionnels

### 6.231.1 Documentation des fonctions membres

#### 6.231.1.1 ChoixOrdre()

void [Deformees\\_maple](#)::**ChoixOrdre** () [virtual]  
Réimplémentée à partir de [OrdreVisu](#).

#### 6.231.1.2 ExeOrdre()

```
void Deformees\_maple::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
```

```
LesMaillages * ,  
bool unseul_incre,  
LesReferences * ,  
LesLoisDeComp * ,  
DiversStockage * ,  
Charge * ,  
LesCondLim * ,  
LesContacts * ,  
Resultats * ,  
UtilLecture & entreePrinc,  
OrdreVisu::EnumTypeIncre type_incre,  
int incre,  
bool animation,  
const map< string, const double *, std::less< string > > & listeVarGlob,  
const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

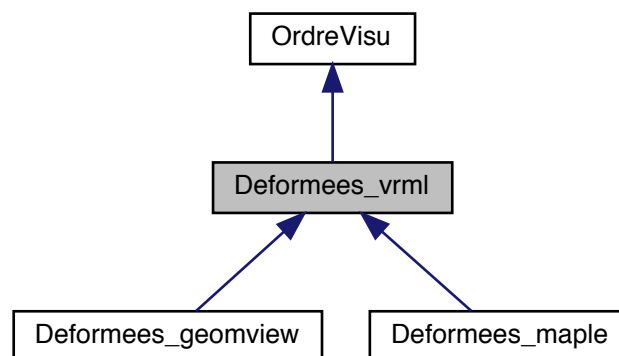
Réimplémentée à partir de [Deformees\\_vrml](#).

La documentation de cette classe a été générée à partir du fichier suivant :

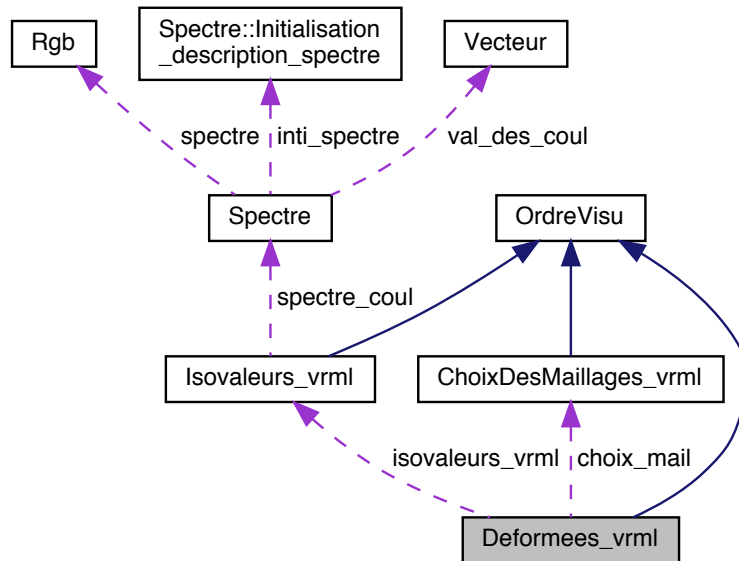
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Deformees\_maple.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Deformees\_maple.cc

## 6.232 Référence de la classe Deformees\_vrml

Grappe d'héritage de Deformees\_vrml:



Graphe de collaboration de Deformees\_vrml:



## Classes

- class [Deuxentiers](#)

## Fonctions membres publiques

- **Deformees\_vrml** (const string &comment\_som, const string &explication, const string &ordre)
- **Deformees\_vrml** (const [Deformees\\_vrml](#) &algo)
- virtual void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, Charge \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu::EnumTypeIncr](#) type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob)
- void **ChoixOrdre** ()
- void **Jonction\_isovaleur** (const [Isovaleurs\\_vrml](#) \*iso)
- void **Jonction\_ChoixDesMaillages** (const [ChoixDesMaillages\\_vrml](#) \*choix\_m)
- const list< string > & **Noms\_matiere** () const
- const list< string > & **Noms\_couleurs** () const
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

## Attributs protégés

- bool **filaire**
- bool **surface**
- bool **numero**
- double **Rcoul**
- double **Gcoul**
- double **Bcoul**
- double **Rcoulf**
- double **Gcoulf**
- double **Bcoulf**
- double **Rcouln**

- double **Gcouln**
- double **Bcouln**
- double **amplification**
- const [Isovaleurs\\_vrml](#) \* **isovaleurs\_vrml**
- list< [Tableau](#)< int > > **tab\_facette**
- list< [Tableau](#)< int > > **tab\_arrete**
- list< [Deuxentiers](#) > **num\_mail\_incr**
- list< string > **noms\_matiere**
- list< string > **noms\_couleurs**
- const [ChoixDesMaillages\\_vrml](#) \* **choix\_mail**

## Membres hérités additionnels

### 6.232.1 Documentation des fonctions membres

#### 6.232.1.1 ChoixOrdre()

```
void Deformees_vrml::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.232.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Deformees_vrml::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.232.1.3 ExeOrdre()

```
void Deformees_vrml::ExeOrdre (
    ParaGlob * paraGlob,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncr type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List\_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.232.1.4 Lecture\_parametres\_OrdreVisu()

```
void Deformees_vrml::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Deformees\_vrml.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Deformees\_vrml.cc

## 6.233 Référence de la classe Deux\_String

cas de deux String

```
#include <Basiques.h>
```

### Fonctions membres publiques

- **Deux\_String** (const string &n1, const string &n2)
- **Deux\_String** (const [Deux\\_String](#) &de)
- [Deux\\_String](#) & **operator=** (const [Deux\\_String](#) &de)
- istream & **LectXML\_Deux\_String** (istream &ent)
- ostream & **EcritXML\_Deux\_String** (ostream &sort)
- bool **operator==** (const [Deux\\_String](#) &a) const
- bool **operator!=** (const [Deux\\_String](#) &a) const
- void **SchemaXML\_Deux\_String** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const
- bool **operator>** (const [Deux\\_String](#) &a) const
- bool **operator>=** (const [Deux\\_String](#) &a) const
- bool **operator<** (const [Deux\\_String](#) &a) const
- bool **operator<=** (const [Deux\\_String](#) &a) const

### Attributs publics

- string **nom1**
- string **nom2**

### Attributs publics statiques

- static short int **impre\_schem\_XML** =0

### Amis

- istream & **operator>>** (istream &ent, [Deux\\_String](#) &de)
- ostream & **operator<<** (ostream &sort, const [Deux\\_String](#) &de)

### 6.233.1 Description détaillée

cas de deux String

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.cc

## 6.234 Référence de la classe Deux\_String\_un\_entier

cas de deux String et un entier

```
#include <Basiques.h>
```

### Fonctions membres publiques

- **Deux\_String\_un\_entier** (const string &n1, const string &n2, const int &nn)
- **Deux\_String\_un\_entier** (const [Deux\\_String\\_un\\_entier](#) &de)
- [Deux\\_String\\_un\\_entier](#) & **operator=** (const [Deux\\_String\\_un\\_entier](#) &de)
- istream & **LectXML\_Deux\_String\_un\_entier** (istream &ent)
- ostream & **EcritXML\_Deux\_String\_un\_entier** (ostream &sort)
- bool **operator==** (const [Deux\\_String\\_un\\_entier](#) &a) const
- bool **operator!=** (const [Deux\\_String\\_un\\_entier](#) &a) const
- void **SchemaXML\_Deux\_String\_un\_entier** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const
- bool **operator>** (const [Deux\\_String\\_un\\_entier](#) &a) const
- bool **operator>=** (const [Deux\\_String\\_un\\_entier](#) &a) const
- bool **operator<** (const [Deux\\_String\\_un\\_entier](#) &a) const
- bool **operator<=** (const [Deux\\_String\\_un\\_entier](#) &a) const



### Attributs publics

- string **nom1**
- string **nom2**
- int **n**

### Attributs publics statiques

- static short int **impre\_schem\_XML**

### Amis

- istream & **operator**>> (istream &ent, [Deux\\_String\\_un\\_entier](#) &de)
- ostream & **operator**<< (ostream &sort, const [Deux\\_String\\_un\\_entier](#) &de)

#### 6.234.1 Description détaillée

cas de deux String et un entier

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h

## 6.235 Référence de la classe DeuxCoordonnees

[DeuxCoordonnees](#): classe relative à 2 coordonnées.

```
#include <PlusieursCoordonnees.h>
```

### Fonctions membres publiques

- **DeuxCoordonnees** (const [Coordonnee](#) &coo1, const [Coordonnee](#) &coo2)
- **DeuxCoordonnees** (const [DeuxCoordonnees](#) &deuxcoo)
- **DeuxCoordonnees** (int dima)
- [DeuxCoordonnees](#) & **operator=** (const [DeuxCoordonnees](#) &de)
- [Coordonnee](#) & **Premier** ()
- [Coordonnee](#) & **Second** ()
- [Coordonnee](#) **Premier** () const
- [Coordonnee](#) **Second** () const

#### 6.235.1 Description détaillée

[DeuxCoordonnees](#): classe relative à 2 coordonnées.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

19/01/2001

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PlusieursCoordonnees.h

## 6.236 Référence de la classe DeuxDoubles

cas de 2 double

```
#include <Basiques.h>
```

## Fonctions membres publiques

- **DeuxDoubles** (double u, double v)
- **DeuxDoubles** (const [DeuxDoubles](#) &de)
- [DeuxDoubles](#) & **operator=** (const [DeuxDoubles](#) &de)
- istream & **LectXML\_DeuxDoubles** (istream &ent)
- ostream & **EcritXML\_DeuxDoubles** (ostream &sort)
- bool **operator==** (const [DeuxDoubles](#) &a) const
- bool **operator!=** (const [DeuxDoubles](#) &a) const
- void **SchemaXML\_DeuxDoubles** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const

## Attributs publics

- double **un**
- double **deux**

## Attributs publics statiques

- static short int **impre\_schem\_XML** =0

## Amis

- istream & **operator>>** (istream &ent, [DeuxDoubles](#) &de)
- ostream & **operator<<** (ostream &sort, const [DeuxDoubles](#) &de)

### 6.236.1 Description détaillée

cas de 2 double

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.cc

## 6.237 Référence de la classe `Deformees_vrml::Deuxentiers`

### Attributs publics

- int **mail**
- int **incr**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Deformees\_vrml.h

## 6.238 Référence de la classe `DeuxEntiers`

cas de 2 entiers

```
#include <Basiques.h>
```

### Fonctions membres publiques

- **DeuxEntiers** (int u, int v)
- **DeuxEntiers** (const [DeuxEntiers](#) &de)
- [DeuxEntiers](#) & **operator=** (const [DeuxEntiers](#) &de)
- istream & **LectXML\_DeuxEntiers** (istream &ent)
- ostream & **EcritXML\_DeuxEntiers** (ostream &sort)
- bool **operator==** (const [DeuxEntiers](#) &a) const
- bool **operator!=** (const [DeuxEntiers](#) &a) const
- bool **operator>** (const [DeuxEntiers](#) &a) const
- bool **operator>=** (const [DeuxEntiers](#) &a) const
- bool **operator<** (const [DeuxEntiers](#) &a) const
- bool **operator<=** (const [DeuxEntiers](#) &a) const
- void **SchemaXML\_DeuxEntiers** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const

### Attributs publics

- int `un`
- int `deux`

### Attributs publics statiques

- static short int `impre_schem_XML` =0

### Amis

- `istream & operator>>` (`istream &ent`, [DeuxEntiers](#) &de)
- `ostream & operator<<` (`ostream &sort`, const [DeuxEntiers](#) &de)

## 6.238.1 Description détaillée

cas de 2 entiers

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.cc](#)

## 6.239 Référence de la classe `OrdreVisu::Deuxentiers`

### Attributs publics

- int `mail`
- int `incr`

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/OrdreVisu.h](#)

## 6.240 Référence de la classe `Deuxentiers_enu`

### Attributs publics

- int `i`
- int `j`

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\\_ddl.h](#)

## 6.241 Référence de la classe `Algori::DeuxString`

### Attributs publics

- string `nomFichier`
- string `nomMaillage`

### Amis

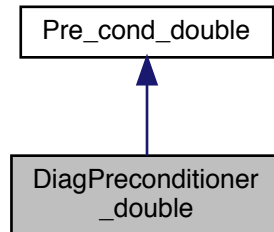
- `istream & operator>>` (`istream &`, [Algori::DeuxString](#) &)
- `ostream & operator<<` (`ostream &`, const [Algori::DeuxString](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

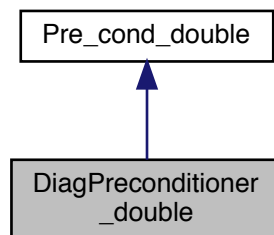
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoRef/Algori.h](#)

## 6.242 Référence de la classe DiagPreconditioner\_double

Graphe d'héritage de DiagPreconditioner\_double:



Graphe de collaboration de DiagPreconditioner\_double:



### Fonctions membres publiques

- **DiagPreconditioner\_double** (const CompCol\_Mat\_double &)
- **DiagPreconditioner\_double** (const CompRow\_Mat\_double &)
- **DiagPreconditioner\_double** (const [Mat\\_abstraite](#) &A)
- VECTOR\_double [solve](#) (const VECTOR\_double &x) const
- VECTOR\_double [trans\\_solve](#) (const VECTOR\_double &x) const

### Fonctions membres protégées

- const double & **diag** (int i) const
- double & **diag** (int i)
- int **CopyInvDiagonals** (int n, const int \*pntr, const int \*indx, const double \*sa, double \*diag)
- int **CopyInvDiagonals** (const [Mat\\_abstraite](#) &A)

#### 6.242.1 Documentation des fonctions membres

**6.242.1.1 solve()**

```
VECTOR_double DiagPreconditioner_double::solve (
    const VECTOR_double & x ) const [virtual]
```

Implémente [Pre\\_cond\\_double](#).

**6.242.1.2 trans\_solve()**

```
VECTOR_double DiagPreconditioner_double::trans_solve (
    const VECTOR_double & x ) const [virtual]
```

Implémente [Pre\\_cond\\_double](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/diagpre\_double\_GR.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/diagpre\_double\_GR.cc

**6.243 Référence de la classe DiversStockage**

Classe servant a stocker des informations intermediaires.

```
#include <DiversStockage.h>
```

**Fonctions membres publiques**

- void **Affiche** () const
- void **Affiche1** () const
- void **Affiche2** () const
- void **Lecture1** (UtilLecture &entreePrinc, LesReferences &lesRef)
- void **Lecture2** (UtilLecture &entreePrinc, LesReferences &lesRef)
- const Tableau< BlocDdlLim< BlocScal\_ou\_fctnD > > & **TabEpaiss** () const
- const Tableau< BlocDdlLim< BlocScal\_ou\_fctnD > > & **TabLargeurs** () const
- const Tableau< BlocDdlLim< BlocScal\_ou\_fctnD > > & **TabSect** () const
- const Tableau< BlocDdlLim< BlocScal > > & **TabVarSect** () const
- const Tableau< BlocDdlLim< BlocScal\_ou\_fctnD > > & **TabMasseVolu** () const
- const Tableau< BlocDdlLim< BlocScal\_ou\_fctnD > > & **TabMasseAddi** () const
- const Tableau< BlocDdlLim< BlocScal\_ou\_fctnD > > & **TabCoefDila** () const
- const Tableau< BlocDdlLim< BlocGen\_3\_1 > > & **TabGesHourglass** () const
- const Tableau< BlocDdlLim< BlocGen\_3\_0 > > & **TabIntegVol** () const
- const Tableau< BlocDdlLim< BlocGen\_3\_0 > > & **TtabIntegVol\_et\_temps** () const
- const Tableau< BlocDdlLim< BlocGen\_4\_0 > > & **TabStabMembBiel** () const
- const Tableau< BlocDdlLim< BlocGen\_6\_0 > > & **TabRepAnisotrope** () const
- void **LectureDonneesExternes** (UtilLecture &, LesReferences &, const int, const string &)
- const Tableau< BlocDdlLim< BlocGen\_3\_0 > > & **TabStatistique** () const
- const Tableau< BlocDdlLim< BlocGen\_3\_0 > > & **TabStatistique\_et\_temps** () const
- void **Info\_commande\_DiversStockage1** (UtilLecture &entreePrinc)
- void **Info\_commande\_DiversStockage2** (UtilLecture &entreePrinc)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

**6.243.1 Description détaillée**

Classe servant a stocker des informations intermediaires.

BUT:Classe servant a stocker des informations intermediaires.

Auteur

Gérard Rio

Version

1.0



## Fonctions membres publiques

- **DonnComHexa** ([ElemGeomC0](#) \*hexa1, [DdlElement](#) &tab, [DdlElement](#) &tabErr, [DdlElement](#) &tab\_Err1↵  
Sig, [Met\\_abstraite](#) &met\_gene, [Tableau](#)< [Vecteur](#) \* > &resEr, [Mat\\_pleine](#) &raidEr, [ElemGeomC0](#) \*hexae↵  
Er, [GeomQuadrangle](#) &quadS, [GeomSeg](#) &seggS, [Vecteur](#) &residu\_int, [Mat\\_pleine](#) &raideur\_int, [Tableau](#)<  
[Vecteur](#) \* > &residus\_extN, [Tableau](#)< [Mat\\_pleine](#) \* > &raideurs\_extN, [Tableau](#)< [Vecteur](#) \* > &residus\_↵  
extA, [Tableau](#)< [Mat\\_pleine](#) \* > &raideurs\_extA, [Tableau](#)< [Vecteur](#) \* > &residus\_extS, [Tableau](#)< [Mat\\_pleine](#)  
\* > &raideurs\_extS, [Mat\\_pleine](#) &mat\_masse, [ElemGeomC0](#) \*hexaeMas, int nbi, [ElemGeomC0](#) \*hexae↵  
Hourg)
- **DonnComHexa** ([DonnComHexa](#) &a)

## Attributs publics

- [ElemGeomC0](#) \* **hexaed**
- [DdlElement](#) **tab\_ddl**
- [Met\\_abstraite](#) **metrique**
- [Mat\\_pleine](#) **matGeom**
- [Mat\\_pleine](#) **matInit**
- [Tableau](#)< [TenseurBB](#) \* > **d\_epsBB**
- [Tableau](#)< [TenseurHH](#) \* > **d\_sigHH**
- [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > **d2\_epsBB**
- [GeomQuadrangle](#) **quadraS**
- [GeomSeg](#) **segS**
- [DdlElement](#) **tab\_ddlErr**
- [DdlElement](#) **tab\_Err1Sig11**
- [Tableau](#)< [Vecteur](#) \* > **resErr**
- [Mat\\_pleine](#) **raidErr**
- [ElemGeomC0](#) \* **hexaedEr**
- [Vecteur](#) **residu\_interne**
- [Mat\\_pleine](#) **raideur\_interne**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeN**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeN**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeA**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeA**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeS**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeS**
- [Mat\\_pleine](#) **matrice\_masse**
- [ElemGeomC0](#) \* **hexaedMas**
- [ElemGeomC0](#) \* **hexaedHourg**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaMemb.cc





- \* > &residus\_extS, Tableau< Mat\_pleine \* > &raideurs\_extS, Mat\_pleine &mat\_masse, ElemGeomC0
- \*pentaMas, int nbi, ElemGeomC0 \*pentaHourg)
- **DonnComPenta** (DonnComPenta &a)

### Attributs publics

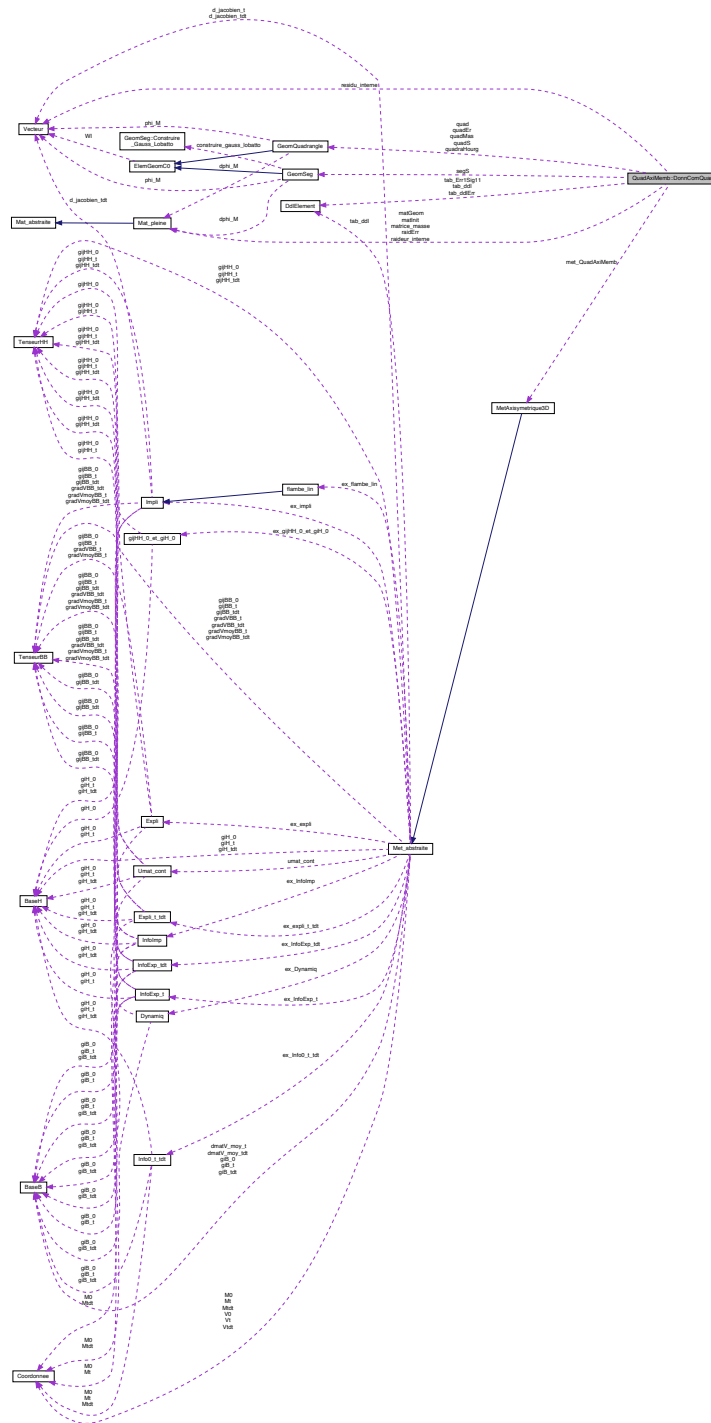
- ElemGeomC0 \* **pentaed**
- DdlElement **tab\_ddl**
- Met\_abstraite **metrique**
- Mat\_pleine **matGeom**
- Mat\_pleine **matInit**
- Tableau< TenseurBB \* > **d\_epsBB**
- Tableau< TenseurHH \* > **d\_sigHH**
- Tableau< Tableau2< TenseurBB \* > > **d2\_epsBB**
- GeomQuadrangle **quadraS**
- GeomTriangle **triaS**
- GeomSeg **segHS**
- GeomSeg **segFS**
- DdlElement **tab\_ddlErr**
- DdlElement **tab\_Err1Sig11**
- Tableau< Vecteur \* > **resErr**
- Mat\_pleine **raidErr**
- ElemGeomC0 \* **pentaedEr**
- Vecteur **residu\_interne**
- Mat\_pleine **raideur\_interne**
- Tableau< Vecteur \* > **residus\_externeN**
- Tableau< Mat\_pleine \* > **raideurs\_externeN**
- Tableau< Vecteur \* > **residus\_externeA**
- Tableau< Mat\_pleine \* > **raideurs\_externeA**
- Tableau< Vecteur \* > **residus\_externeS**
- Tableau< Mat\_pleine \* > **raideurs\_externeS**
- Mat\_pleine **matrice\_masse**
- ElemGeomC0 \* **pentaedMas**
- ElemGeomC0 \* **pentaedHourg**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaMemb.cc

### 6.246 Référence de la classe QuadAxiMemb::DonnComQuad

Graphe de collaboration de QuadAxiMemb::DonnComQuad:



#### Fonctions membres publiques

- **DonnComQuad** (*GeomQuadrangle* &tri, *DdlElement* &tab, *DdlElement* &tabErr, *DdlElement* &tab\_Err1←  
Sig, *MatAxisymetrique3D* &met\_gene, *Tableau*< *Vecteur* \* > &resEr, *Mat\_pleine* &raidEr, *GeomQuadrangle*  
&quadEr, *GeomQuadrangle* &quadS, *GeomSeg* &segmS, *Vecteur* &residu\_int, *Mat\_pleine* &raideur\_←  
int, *Tableau*< *Vecteur* \* > &residu\_extN, *Tableau*< *Mat\_pleine* \* > &raideurs\_extN, *Tableau*< *Vecteur*  
\* > &residu\_extA, *Tableau*< *Mat\_pleine* \* > &raideurs\_extA, *Tableau*< *Vecteur* \* > &residu\_extS,

- [Tableau](#)< [Mat\\_pleine](#) \* > &raideurs\_extS, [Mat\\_pleine](#) &mat\_masse, [GeomQuadrangle](#) &quadMas, int nbi, [GeomQuadrangle](#) \*quadHourg)
- **DonnComQuad** ([DonnComQuad](#) &a)

### Attributs publics

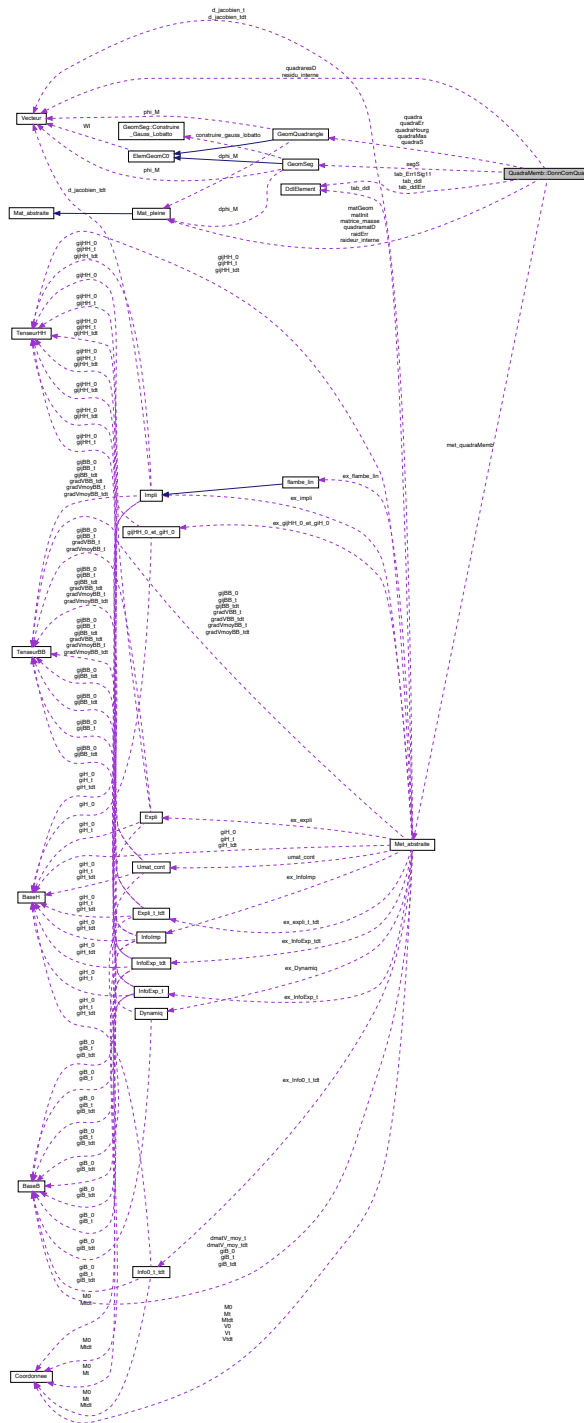
- [GeomQuadrangle](#) **quad**
- [DdlElement](#) **tab\_ddl**
- [MetAxisymetrique3D](#) **met\_QuadAxiMemb**
- [Mat\\_pleine](#) **matGeom**
- [Mat\\_pleine](#) **matInit**
- [Tableau](#)< [TenseurBB](#) \* > **d\_epsBB**
- [Tableau](#)< [TenseurHH](#) \* > **d\_sigHH**
- [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > **d2\_epsBB**
- [GeomQuadrangle](#) **quadS**
- [GeomSeg](#) **segS**
- [DdlElement](#) **tab\_ddlErr**
- [DdlElement](#) **tab\_Err1Sig11**
- [Tableau](#)< [Vecteur](#) \* > **resErr**
- [Mat\\_pleine](#) **raidErr**
- [GeomQuadrangle](#) **quadEr**
- [Vecteur](#) **residu\_interne**
- [Mat\\_pleine](#) **raideur\_interne**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeN**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeN**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeA**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeA**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeS**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeS**
- [Mat\\_pleine](#) **matrice\_masse**
- [GeomQuadrangle](#) **quadMas**
- [GeomQuadrangle](#) \* **quadraHourg**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiMemb.cc

## 6.247 Référence de la classe QuadraMemb::DonnComQuad

Grphe de collaboration de QuadraMemb::DonnComQuad:



### Fonctions membres publiques

- **DonnComQuad** (*GeomQuadrangle* &quad, *DdlElement* &tab, *DdlElement* &tabErr, *DdlElement* &tab\_←, *Err1Sig*, *Met\_abstraite* &met\_gene, *Tableau*< *Vecteur* \* > &resEr, *Mat\_pleine* &raidEr, *GeomQuadrangle* &quadEr, *GeomQuadrangle* &quadS, *GeomSeg* &segmS, *Vecteur* &residu\_int, *Mat\_pleine* &raideur\_←, int, *Tableau*< *Vecteur* \* > &residu\_extN, *Tableau*< *Mat\_pleine* \* > &raideurs\_extN, *Tableau*< *Vecteur* \* > &residu\_extA, *Tableau*< *Mat\_pleine* \* > &raideurs\_extA, *Tableau*< *Vecteur* \* > &residu\_extS,

- `Tableau< Mat_pleine * > &raideurs_extS`, `Mat_pleine &mat_masse`, `GeomQuadrangle &quadMas`, `int nbi`, `GeomQuadrangle *quadHourg`, `Mat_pleine *quadmatD`, `Vecteur *quadresD`)
- `DonnComQuad (DonnComQuad &a)`

### Attributs publics

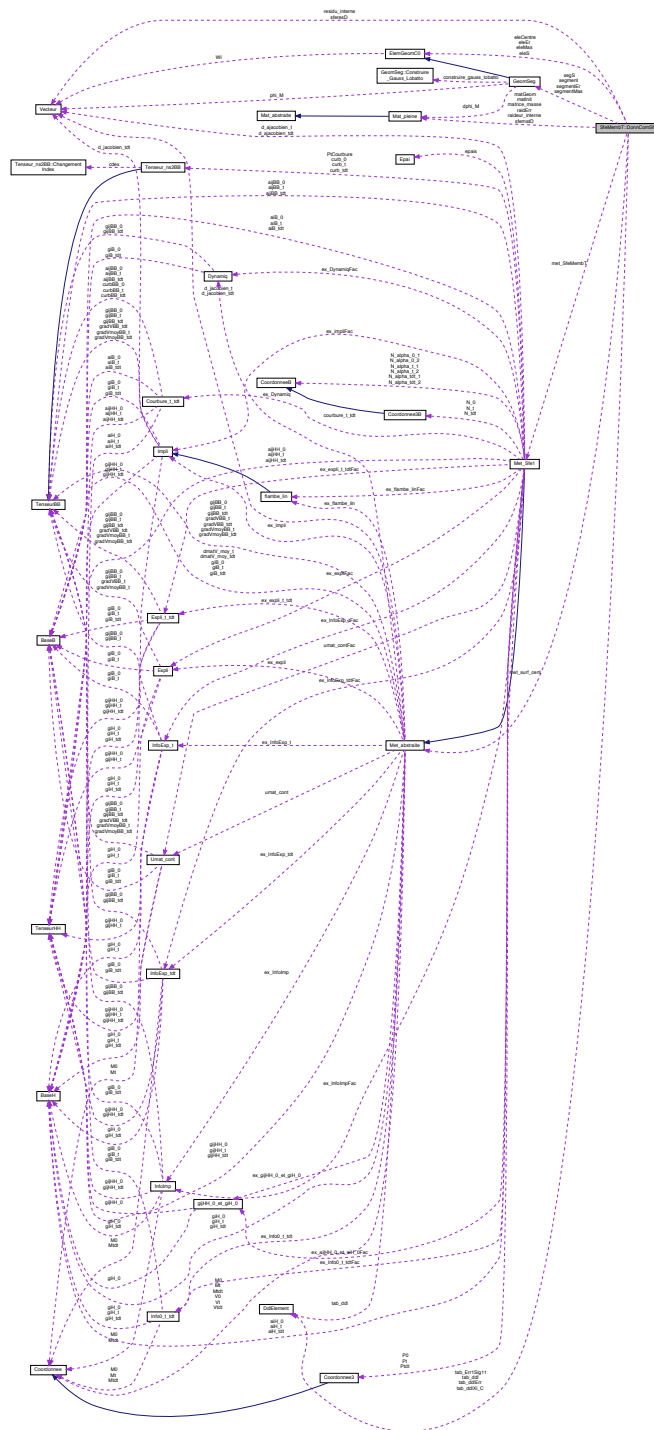
- `GeomQuadrangle quadra`
- `DdlElement tab_ddl`
- `Met_abstraite met_quadraMemb`
- `Mat_pleine matGeom`
- `Mat_pleine matInit`
- `Tableau< TenseurBB * > d_epsBB`
- `Tableau< TenseurHH * > d_sigHH`
- `Tableau< Tableau2< TenseurBB * > > d2_epsBB`
- `GeomQuadrangle quadraS`
- `GeomSeg segS`
- `DdlElement tab_ddlErr`
- `DdlElement tab_Err1Sig11`
- `Tableau< Vecteur * > resErr`
- `Mat_pleine raidErr`
- `GeomQuadrangle quadraEr`
- `Vecteur residu_interne`
- `Mat_pleine raideur_interne`
- `Tableau< Vecteur * > residus_externesN`
- `Tableau< Mat_pleine * > raideurs_externesN`
- `Tableau< Vecteur * > residus_externesA`
- `Tableau< Mat_pleine * > raideurs_externesA`
- `Tableau< Vecteur * > residus_externesS`
- `Tableau< Mat_pleine * > raideurs_externesS`
- `Mat_pleine matrice_masse`
- `GeomQuadrangle quadraMas`
- `GeomQuadrangle * quadraHourg`
- `Mat_pleine * quadramatD`
- `Vecteur * quadraresD`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.cc`

## 6.248 Référence de la classe SfeMembT::DonnComSfe

Grphe de collaboration de SfeMembT::DonnComSfe:



### Fonctions membres publiques

- **DonnComSfe** (*ElemGeomC0* \*eleCentre, const *GeomSeg* &seg, const *DdlElement* &tab, *DdlElement* &tabErr, *DdlElement* &tab\_Err1Sig, *DdlElement* &tab\_ddlXi\_C, const *Met\_Sfe1* &met\_gene, *Tableau*< *Vecteur* \* > &resEr, *Mat\_pleine* &raidEr, *ElemGeomC0* \*eleEr, *GeomSeg* &segEr, *ElemGeomC0* \*eleS, *GeomSeg* &segsS, *Vecteur* &residu\_int, *Mat\_pleine* &raideur\_int, *Tableau*< *Vecteur* \* > &residu\_extN, *Tableau*< *Mat\_pleine* \* > &raideurs\_extN, *Tableau*< *Vecteur* \* > &residu\_extA, *Tableau*< *Mat\_pleine*

- \* > &raideurs\_extA, Tableau< Vecteur \* > &residus\_extS, Tableau< Mat\_pleine \* > &raideurs\_extS, Mat\_pleine &mat\_masse, ElemGeomC0 \*eleMas, const GeomSeg &segMas, int nbi, Tableau< EnuTypeCL > const &tabTypeCL, Tableau< Coordonnee3 > const &vplan, Tableau< int > &nMetVTab\_ddl, const Met\_abstraite &met\_cent)
- **DonnComSfe** (DonnComSfe &a)

### Attributs publics

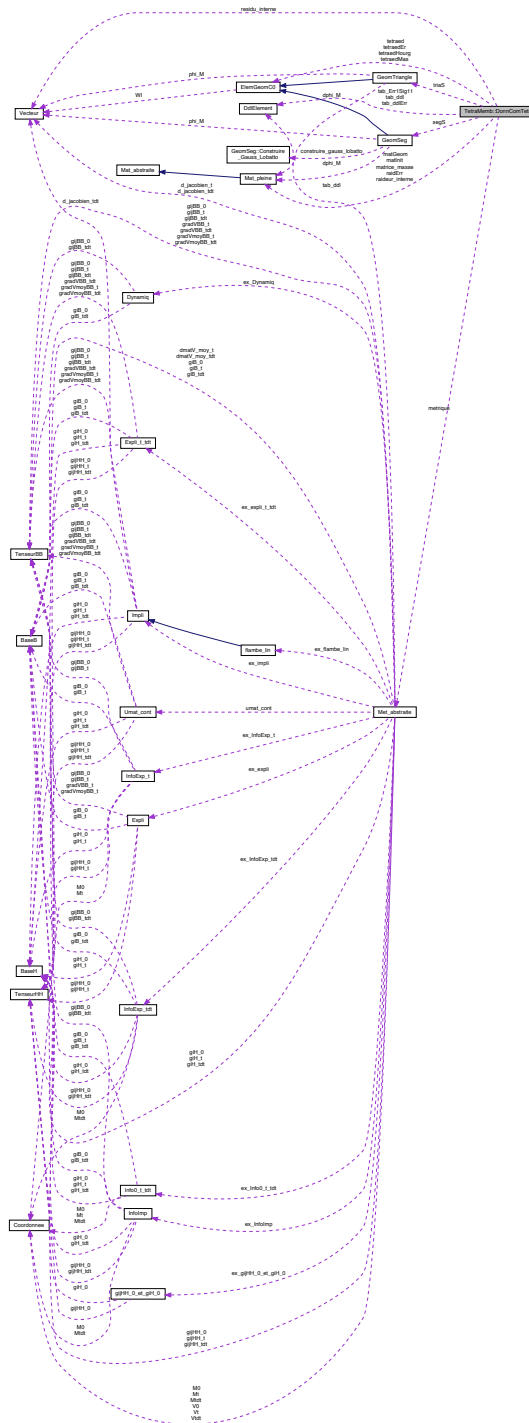
- ElemGeomC0 \* **eleCentre**
- GeomSeg **segment**
- DdlElement **tab\_ddl**
- DdlElement **tab\_ddlXi\_C**
- Tableau< int > **nMetVTab\_ddl**
- Met\_Sfe1 **met\_SfeMembT**
- Met\_abstraite **met\_surf\_cent**
- Tableau< EnuTypeCL > **tabType\_rienCL**
- Tableau< Coordonnee3 > **vplan\_rien**
- Mat\_pleine **matGeom**
- Mat\_pleine **matInit**
- Tableau< TenseurBB \* > **d\_epsBB**
- Tableau< TenseurHH \* > **d\_sigHH**
- Tableau< Tableau2< TenseurBB \* > > **d2\_epsBB**
- ElemGeomC0 \* **eleS**
- GeomSeg **segS**
- DdlElement **tab\_ddlErr**
- DdlElement **tab\_Err1Sig11**
- Tableau< Vecteur \* > **resErr**
- Mat\_pleine **raidErr**
- ElemGeomC0 \* **eleEr**
- GeomSeg **segmentEr**
- Vecteur **residu\_interne**
- Mat\_pleine **raideur\_interne**
- Tableau< Vecteur \* > **residus\_externeN**
- Tableau< Mat\_pleine \* > **raideurs\_externeN**
- Tableau< Vecteur \* > **residus\_externeA**
- Tableau< Mat\_pleine \* > **raideurs\_externeA**
- Tableau< Vecteur \* > **residus\_externeS**
- Tableau< Mat\_pleine \* > **raideurs\_externeS**
- Mat\_pleine **matrice\_masse**
- ElemGeomC0 \* **eleMas**
- GeomSeg **segmentMas**
- Mat\_pleine \* **sfematD**
- Vecteur \* **sferesD**
- Tableau< int > \* **noeud\_a\_prendre\_en\_compte**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.cc

## 6.249 Référence de la classe TetraMemb::DonnComTetra

Graphe de collaboration de TetraMemb::DonnComTetra:



### Fonctions membres publiques

- **DonnComTetra** (*ElemGeomC0* \*tetra1, *DdlElement* &tab, *DdlElement* &tabErr, *DdlElement* &tab\_Err1←  
*Sig*, *Met\_abstraite* &met\_gene, *Tableau*< *Vecteur* \* > &resEr, *Mat\_pleine* &raidEr, *ElemGeomC0* \*tetrae←  
*Er*, *GeomTriangle* triS, *GeomSeg* &seggS, *Vecteur* &residu\_int, *Mat\_pleine* &raidreur\_int, *Tableau*< *Vecteur* \* > &residus\_extN, *Tableau*< *Mat\_pleine* \* > &raideurs\_extN, *Tableau*< *Vecteur* \* > &residus\_extA, *Tableau*< *Mat\_pleine* \* > &raideurs\_extA, *Tableau*< *Vecteur* \* > &residus\_extS, *Tableau*< *Mat\_pleine* \*



- > &raideurs\_extS, [Mat\\_pleine](#) &mat\_masse, [ElemGeomC0](#) \*tetraeMas, int nbi, [ElemGeomC0](#) \*tetraeHourg)
- [DonnComTetra](#) ([DonnComTetra](#) &a)

### Attributs publics

- [ElemGeomC0](#) \* **tetraed**
- [DdlElement](#) **tab\_ddl**
- [Met\\_abstraite](#) **metrique**
- [Mat\\_pleine](#) **matGeom**
- [Mat\\_pleine](#) **matInit**
- [Tableau](#)< [TenseurBB](#) \* > **d\_epsBB**
- [Tableau](#)< [TenseurHH](#) \* > **d\_sigHH**
- [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > **d2\_epsBB**
- [GeomTriangle](#) **triaS**
- [GeomSeg](#) **segS**
- [DdlElement](#) **tab\_ddlErr**
- [DdlElement](#) **tab\_Err1Sig11**
- [Tableau](#)< [Vecteur](#) \* > **resErr**
- [Mat\\_pleine](#) **raidErr**
- [ElemGeomC0](#) \* **tetraedEr**
- [Vecteur](#) **residu\_interne**
- [Mat\\_pleine](#) **raideur\_interne**
- [Tableau](#)< [Vecteur](#) \* > **residu\_externerN**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externerN**
- [Tableau](#)< [Vecteur](#) \* > **residu\_externerA**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externerA**
- [Tableau](#)< [Vecteur](#) \* > **residu\_externerS**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externerS**
- [Mat\\_pleine](#) **matrice\_masse**
- [ElemGeomC0](#) \* **tetraedMas**
- [ElemGeomC0](#) \* **tetraedHourg**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraMemb.cc



- > &raideurs\_extS, [Mat\\_pleine](#) &mat\_masse, [GeomTriangle](#) &triMas, int nbi, [GeomTriangle](#) \*triHourg)
- **DonnComTria** ([DonnComTria](#) &a)

### Attributs publics

- [GeomTriangle](#) **tria**
- [DdlElement](#) **tab\_ddl**
- [MetAxisymetrique3D](#) **met\_TriaAxiMemb**
- [Mat\\_pleine](#) **matGeom**
- [Mat\\_pleine](#) **matInit**
- [Tableau](#)< [TenseurBB](#) \* > **d\_epsBB**
- [Tableau](#)< [TenseurHH](#) \* > **d\_sigHH**
- [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > **d2\_epsBB**
- [GeomTriangle](#) **triaS**
- [GeomSeg](#) **segS**
- [DdlElement](#) **tab\_ddlErr**
- [DdlElement](#) **tab\_Err1Sig11**
- [Tableau](#)< [Vecteur](#) \* > **resErr**
- [Mat\\_pleine](#) **raidErr**
- [GeomTriangle](#) **triaEr**
- [Vecteur](#) **residu\_interne**
- [Mat\\_pleine](#) **raideur\_interne**
- [Tableau](#)< [Vecteur](#) \* > **residu\_externerN**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externerN**
- [Tableau](#)< [Vecteur](#) \* > **residu\_externerA**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externerA**
- [Tableau](#)< [Vecteur](#) \* > **residu\_externerS**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externerS**
- [Mat\\_pleine](#) **matrice\_masse**
- [GeomTriangle](#) **triaMas**
- [GeomTriangle](#) \* **triaHourg**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxi↔Memb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxi↔Memb.cc



- \* > &raideurs\_extS, [Mat\\_pleine](#) &mat\_masse, [GeomTriangle](#) &triMas, int nbi, [GeomTriangle](#) \*triHourg, [Mat\\_pleine](#) \*trimatD, [Vecteur](#) \*triresD)
- [DonnComTria](#) ([DonnComTria](#) &a)

### Attributs publics

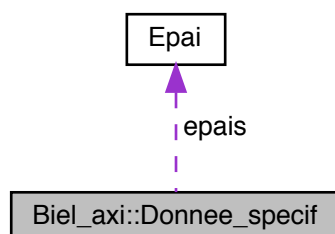
- [GeomTriangle](#) **tria**
- [DdlElement](#) **tab\_ddl**
- [Met\\_abstraite](#) **met\_triaMemb**
- [Mat\\_pleine](#) **matGeom**
- [Mat\\_pleine](#) **matInit**
- [Tableau](#)< [TenseurBB](#) \* > **d\_epsBB**
- [Tableau](#)< [TenseurHH](#) \* > **d\_sigHH**
- [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > **d2\_epsBB**
- [GeomTriangle](#) **triaS**
- [GeomSeg](#) **segS**
- [DdlElement](#) **tab\_ddlErr**
- [DdlElement](#) **tab\_Err1Sig11**
- [Tableau](#)< [Vecteur](#) \* > **resErr**
- [Mat\\_pleine](#) **raidErr**
- [GeomTriangle](#) **triaEr**
- [Vecteur](#) **residu\_interne**
- [Mat\\_pleine](#) **raideur\_interne**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeN**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeN**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeA**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeA**
- [Tableau](#)< [Vecteur](#) \* > **residus\_externeS**
- [Tableau](#)< [Mat\\_pleine](#) \* > **raideurs\_externeS**
- [Mat\\_pleine](#) **matrice\_masse**
- [GeomTriangle](#) **triaMas**
- [GeomTriangle](#) \* **triaHourg**
- [Mat\\_pleine](#) \* **triamatD**
- [Vecteur](#) \* **triresD**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.cc

## 6.252 Référence de la classe Biel\_axi::Donnee\_specif

Grphe de collaboration de Biel\_axi::Donnee\_specif:



## Fonctions membres publiques

- **Donnee\_specif** (double epai)
- **Donnee\_specif** (int casptnbi, int casptnbiEr, int casptnbiS, int casptnbiMas)
- **Donnee\_specif** (double epai0, double epai\_t, double epai\_tdt, int casptnbi, int casptnbiEr, int casptnbiS, int casptnbiMas)
- **Donnee\_specif** (const [Donnee\\_specif](#) &a)
- **Donnee\_specif** & **operator=** (const [Donnee\\_specif](#) &a)

## Attributs publics

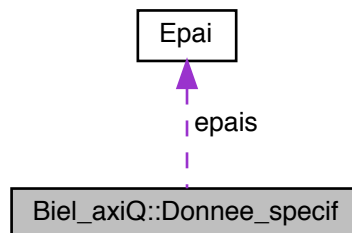
- [Epai](#) epais
- int **cas\_pti\_nbi**
- int **cas\_pti\_nbiEr**
- int **cas\_pti\_nbiS**
- int **cas\_pti\_nbiMas**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axi.h

## 6.253 Référence de la classe Biel\_axiQ::Donnee\_specif

Graphe de collaboration de Biel\_axiQ::Donnee\_specif:



## Fonctions membres publiques

- **Donnee\_specif** (double epai)
- **Donnee\_specif** (int casptnbi, int casptnbiEr, int casptnbiS, int casptnbiMas)
- **Donnee\_specif** (double epai0, double epai\_t, double epai\_tdt, int casptnbi, int casptnbiEr, int casptnbiS, int casptnbiMas)
- **Donnee\_specif** (const [Donnee\\_specif](#) &a)
- **Donnee\_specif** & **operator=** (const [Donnee\\_specif](#) &a)

## Attributs publics

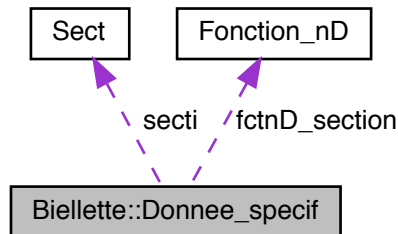
- [Epai](#) epais
- int **cas\_pti\_nbi**
- int **cas\_pti\_nbiEr**
- int **cas\_pti\_nbiS**
- int **cas\_pti\_nbiMas**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axiQ.h

## 6.254 Référence de la classe `Biellette::Donnee_specif`

Graphe de collaboration de `Biellette::Donnee_specif`:



### Fonctions membres publiques

- `Donnee_specif` (double section)
- `Donnee_specif` (double epai0, double epai\_t, double epai\_tdt, `Fonction_nD` \*fctnD, bool variation)
- `Donnee_specif` (const `Donnee_specif` &a)
- `Donnee_specif` & `operator=` (const `Donnee_specif` &a)

### Attributs publics

- `Sect` `secti`
- `Fonction_nD` \* `fctnD_section`
- bool `variation_section`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biellette.h`

## 6.255 Référence de la classe `BielletteC1::Donnee_specif`

### Fonctions membres publiques

- `Donnee_specif` (double sect)
- `Donnee_specif` (const `Donnee_specif` &a)
- `Donnee_specif` & `operator=` (const `Donnee_specif` &a)

### Attributs publics

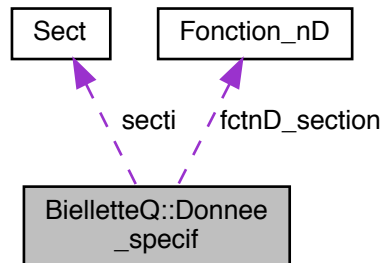
- double `section`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/BielletteC1.h`

## 6.256 Référence de la classe BielleQ::Donnee\_specif

Grphe de collaboration de BielleQ::Donnee\_specif:



### Fonctions membres publiques

- `Donnee_specif` (double section)
- `Donnee_specif` (double epai0, double epai\_t, double epai\_tdt, `Fonction_nD` \*fctnD, bool variation)
- `Donnee_specif` (const `Donnee_specif` &a)
- `Donnee_specif` & `operator=` (const `Donnee_specif` &a)

### Attributs publics

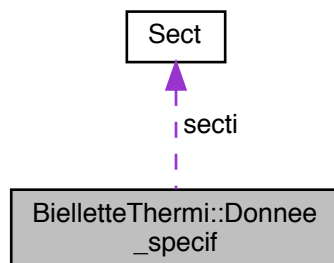
- `Sect` `secti`
- `Fonction_nD` \* `fctnD_section`
- bool `variation_section`

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/BielleQ/BielleQ.h

## 6.257 Référence de la classe BielleThermi::Donnee\_specif

Grphe de collaboration de BielleThermi::Donnee\_specif:





### Fonctions membres publiques

- **Donnee\_specif** (double section)
- **Donnee\_specif** (double epai0, double epai\_t, double epai\_tdt, bool variation)
- **Donnee\_specif** (const [Donnee\\_specif](#) &a)
- [Donnee\\_specif](#) & **operator=** (const [Donnee\\_specif](#) &a)

### Attributs publics

- [Sect secti](#)
- bool **variation\_section**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/Biellette/BielletteThermi.h

## 6.258 Référence de la classe QuadAxiMemb::Donnee\_specif

### Fonctions membres publiques

- **Donnee\_specif** (const [Donnee\\_specif](#) &)
- [Donnee\\_specif](#) & **operator=** (const [Donnee\\_specif](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiMemb.h

## 6.259 Référence de la classe QuadraMemb::Donnee\_specif

### Fonctions membres publiques

- **Donnee\_specif** (double epai, int nbi)
- **Donnee\_specif** (double epai0, double epai\_t, double epai\_tdt, int nbi)
- **Donnee\_specif** (const [Donnee\\_specif](#) &a)
- [Donnee\\_specif](#) & **operator=** (const [Donnee\\_specif](#) &a)

### Attributs publics

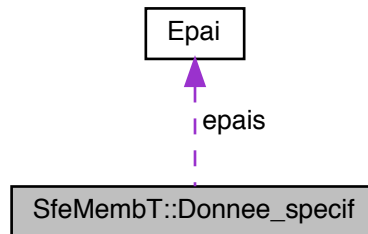
- [Tableau< Epai > epais](#)
- int **use\_epais\_moyenne**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.h

## 6.260 Référence de la classe SfeMembT::Donnee\_specif

Graphe de collaboration de SfeMembT::Donnee\_specif:



### Fonctions membres publiques

- **Donnee\_specif** (double epai0, double epai\_t, double epai\_tdt)
- **Donnee\_specif** (double epai)
- **Donnee\_specif** (const [Donnee\\_specif](#) &a)
- **Donnee\_specif** (const [Epai](#) \*epas)
- **Donnee\_specif** & **operator=** (const [Donnee\\_specif](#) &a)

### Attributs publics

- [Epai](#) \* **epais**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.h

## 6.261 Référence de la classe TriaAxiMemb::Donnee\_specif

### Fonctions membres publiques

- **Donnee\_specif** (int casptnbi, int casptnbiEr, int casptnbiS, int casptnbiMas)
- **Donnee\_specif** (const [Donnee\\_specif](#) &a)
- **Donnee\_specif** & **operator=** (const [Donnee\\_specif](#) &a)

### Attributs publics

- int **cas\_pti\_nbi**
- int **cas\_pti\_nbiEr**
- int **cas\_pti\_nbiS**
- int **cas\_pti\_nbiMas**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiMemb.h

## 6.262 Référence de la classe TriaMemb::Donnee\_specif

### Fonctions membres publiques

- **Donnee\_specif** (double epai, int nbi)

- **Donnee\_specif** (int casptnbi, int casptnbiEr, int casptnbiS, int casptnbiMas, int nbi)
- **Donnee\_specif** (double epai0, double epai\_t, double epai\_tdt, int casptnbi, int casptnbiEr, int casptnbiS, int casptnbiMas, int nbi)
- **Donnee\_specif** (const [Donnee\\_specif](#) &a)
- **Donnee\_specif** & **operator=** (const [Donnee\\_specif](#) &a)

### Attributs publics

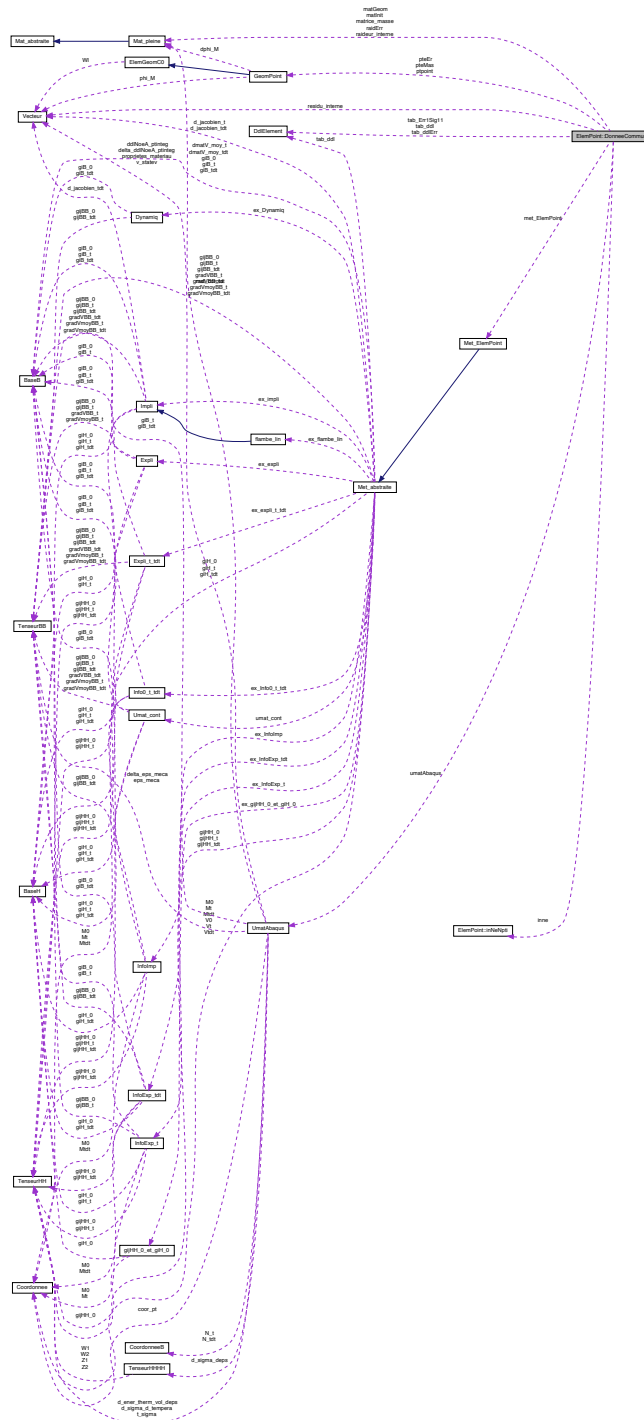
- [Tableau](#)< [Epai](#) > **epais**
- int **cas\_pti\_nbi**
- int **cas\_pti\_nbiEr**
- int **cas\_pti\_nbiS**
- int **cas\_pti\_nbiMas**
- int **use\_epais\_moyenne**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.h

## 6.263 Référence de la classe ElemPoint::DonneeCommune

Graphe de collaboration de ElemPoint::DonneeCommune:



### Fonctions membres publiques

- **DonneeCommune** (**GeomPoint** &pteg, **DdIElement** &tab, **DdIElement** &tabErr, **DdIElement** &tab\_Err1←  
Sig, **Met\_ElemPoint** &met\_point, **Tableau**< **Vecteur** \* > &resEr, **Mat\_pleine** &raidEr, **GeomPoint** &pteEr,  
**Vecteur** &residu\_int, **Mat\_pleine** &raideur\_int, **Tableau**< **Vecteur** \* > &residus\_extN, **Tableau**< **Mat\_pleine**  
\* > &raideurs\_extN, **Mat\_pleine** &mat\_masse, **GeomPoint** &pteMa, **UmatAbaqus** &umatAbaqus, int dimen-  
sion, int nbi)

- `DonneeCommune` ([DonneeCommune](#) &a)

### Attributs publics

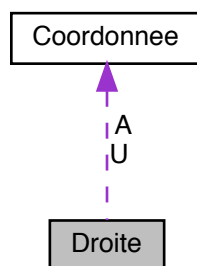
- `GeomPoint` `ptpoint`
- `DdlElement` `tab_ddl`
- `Met_ElemPoint` `met_ElemPoint`
- `Mat_pleine` `matGeom`
- `Mat_pleine` `matInit`
- `Tableau`< `TenseurBB` \* > `d_epsBB`
- `Tableau`< `TenseurHH` \* > `d_sigHH`
- `Tableau`< `Tableau2`< `TenseurBB` \* > > `d2_epsBB`
- `DdlElement` `tab_ddlErr`
- `DdlElement` `tab_Err1Sig11`
- `Tableau`< `Vecteur` \* > `resErr`
- `Mat_pleine` `raidErr`
- `GeomPoint` `pteEr`
- `Vecteur` `residu_interne`
- `Mat_pleine` `raideur_interne`
- `Tableau`< `Vecteur` \* > `residus_externeN`
- `Tableau`< `Mat_pleine` \* > `raideurs_externeN`
- `Mat_pleine` `matrice_masse`
- `GeomPoint` `pteMas`
- `UmatAbaqus` `umatAbaqus`
- `inNeNpti` `inne`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.cc`

## 6.264 Référence de la classe Droite

Graphe de collaboration de Droite:



### Fonctions membres publiques

- `Droite` (const [Coordonnee](#) &B, const [Coordonnee](#) &vec)
- `Droite` (int dim)
- `Droite` (const [Droite](#) &a)
- `Droite` & `operator=` (const [Droite](#) &P)
- const [Coordonnee](#) & `PointDroite` () const
- const [Coordonnee](#) & `VecDroite` () const
- void `Change_dim` (int dima)
- void `Change_ptref` (const [Coordonnee](#) &B)

- void **Change\_vect** (const [Coordonnee](#) &vec)
- void **change\_donnees** (const [Coordonnee](#) &B, const [Coordonnee](#) &vec)
- int **Intersection** (const [Droite](#) &D, [Coordonnee](#) &M)
- bool **Appartient\_a\_la\_droite** (const [Coordonnee](#) &B)
- [Coordonnee](#) **UneNormale** ()
- double **Distance\_a\_la\_droite** (const [Coordonnee](#) &M) const
- [Coordonnee](#) **Projete** (const [Coordonnee](#) &M) const
- bool **DuMemeCote** (const [Coordonnee](#) &M1, const [Coordonnee](#) &M2) const

### Attributs protégés

- [Coordonnee](#) **A**
- [Coordonnee](#) **U**

### Attributs protégés statiques

- static int **alea** = 0

### Amis

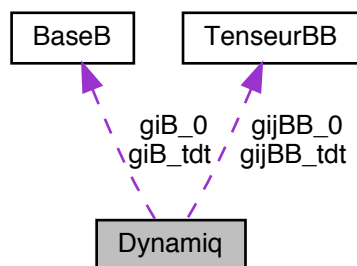
- istream & **operator**>> (istream &, [Droite](#) &)
- ostream & **operator**<< (ostream &, const [Droite](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Droite.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Droite.cc

## 6.265 Référence de la classe Dynamiq

Graphe de collaboration de Dynamiq:



### Fonctions membres publiques

- **Dynamiq** ([BaseB](#) \*ggiB\_0, [BaseB](#) \*ggiB\_tdt, [TenseurBB](#) \*ggijBB\_0, [TenseurBB](#) \*ggijBB\_tdt, double \*gjacobien\_tdt, double \*gjacobien\_0)
- **Dynamiq** (const [Dynamiq](#) &ex)
- [Dynamiq](#) & **operator=** (const [Dynamiq](#) &ex)
- void **Mise\_a\_jour\_grandeur** ([BaseB](#) \*ggiB\_0, [BaseB](#) \*ggiB\_tdt, [TenseurBB](#) \*ggijBB\_0, [TenseurBB](#) \*ggijBB\_tdt, double \*gjacobien\_tdt, double \*gjacobien\_0)

## Attributs publics

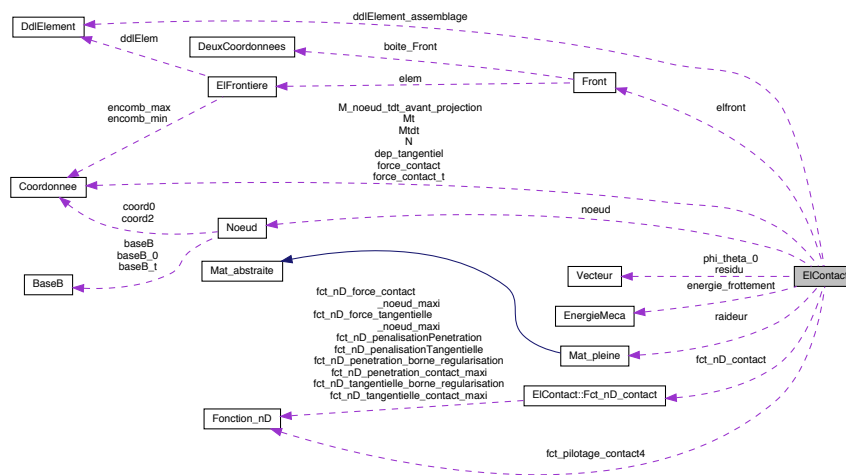
- `BaseB` \* `giB_0`
- `BaseB` \* `giB_tdt`
- `TenseurBB` \* `gijBB_0`
- `TenseurBB` \* `gijBB_tdt`
- `double` \* `jacobien_tdt`
- `double` \* `jacobien_0`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Met_↔ abstraite_struc_donnees.h`

## 6.266 Référence de la classe EIContact

Graph de collaboration de EIContact:



## Classes

- class `Fct_nD_contact`

## Fonctions membres publiques

- `EIContact` ()
- `EIContact` (`Fct_nD_contact` &fct\_contact) ----- fin particularité class `Fct_nD_contact` -----
- `EIContact` (const `Front` \*elfront, const `Noeud` \*noeud, `Fct_nD_contact` &fct\_contact\_, int collant=0)
- `EIContact` (const `EIContact` &a)
- void `Affiche` () const
- bool `Contact` (bool init=true)
- const `Coordonnee` & `Point_intersection` () const
- int & `Num_zone_contact` ()
- `Coordonnee` `Trajectoire` (int &test)
- `Coordonnee` `Intersection` (const `Coordonnee` &V, bool init)
- `Condilineaire` `ConditionLi` (int nb\_assemb)
- `Tableau`< `Noeud` \* > & `TabNoeud` ()
- const `Tableau`< `Noeud` \* > & `Const_TabNoeud` () const
- `Tableau`< `Noeud` \* > & `TabNoeud_pour_assemblage` ()
- const `DdiElement` & `TableauDdiCont` () const
- `Noeud` \* `Esclave` ()
- const `Coordonnee` & `Force_contact` () const

- double **F\_N\_MAX** () const
- double **F\_T\_MAX** () const
- const double & **Penalisation** () const
- const double & **Penalisation\_tangentielle** () const
- const **Coordonnee** & **Dep\_tangentiel** () const
- const **Coordonnee** & **Normale\_actuelle** () const
- const **Tableau**< **Coordonnee** > & **TabForce\_cont** () const
- bool **Actualisation** ()
- **Front** \* **Elfront** () const
- bool **Decol** ()
- void **Change\_force** (const **Coordonnee** &force)
- void **Change\_TabForce\_cont** (const **Tableau**< **Coordonnee** > &tab)
- int **Actif** () const
- void **Met\_Inactif** ()
- void **Met\_actif** ()
- int **Nb\_decol** () const
- int **Nb\_pene** () const
- virtual **Vecteur** \* **SM\_charge\_contact** ()
- virtual Element::ResRaid **SM\_K\_charge\_contact** ()
- const **EnergieMeca** & **EnergieFrottement** () const
- const double & **EnergiePenalisation** () const
- double **Pas\_de\_temps\_ideal** () const
- int **Cas\_solide** () const
- void **Change\_contact\_collant** (bool change)
- int **Collant** () const
- const double & **Gaptdt** () const
- const double & **Dep\_T\_tdt** () const
- void **TdtversT** ()
- void **TversTdt** ()
- void **Lec\_base\_info\_EIContact** (ifstream &ent)
- void **Ecri\_base\_info\_EIContact** (ofstream &sort)

### Fonctions membres publiques statiques

- static void **Init\_fct\_pilotage\_contact4** (**Fonction\_nD** \*fct\_pilo\_contact4)
- static int **Recup\_et\_mise\_a\_jour\_type\_contact** ()
- static void **Change\_dep\_max** (const double &deplac\_max)
- static void **Mise\_a\_jour\_niveau\_commentaire** (int niveau)

### Fonctions membres protégées

- **Coordonnee** **Calcul\_Normale** (int dim, **Plan** &pl, **Droite** &dr, int indic)
- void **Libere** ()
- void **Construction\_TabNoeud** ()
- **Tableau**< **Coordonnee** > \* **RecupInfo** (**Coordonnee** &N, **Coordonnee** &M\_impact, **Vecteur** &phii, bool avec\_var)
- void **Mise\_a\_jour\_ddlelement\_cas\_solide\_assemblage** ()
- void **RecupPlaceResidu** (int nbddl)
- void **RecupPlaceRaideur** (int nbddl)
- double **CalFactPenal** (const double &gap, double &d\_beta\_gap, int contact\_type)
- double **CalFactPenalTangentiel** (const double &gap, double &d\_beta\_gap)
- void **ChangeEnMoyGlissante** (**Coordonnee** &Noe\_atdt)
- void **Moyenne\_glissante\_penalisation** (double &penalisation, double &essai\_penalisation)
- double **Valeur\_fct\_nD** (**Fonction\_nD** \*fct\_nD) const

### Attributs protégés

- int **actif**
- **Front** \* **elfront**
- **Noeud** \* **noeud**
- int **num\_zone\_contact**
- **Tableau**< **Noeud** \* > **tabNoeud**
- **Tableau**< **Coordonnee** > **tab\_posi\_esclave**
- int **nb\_posi\_esclave\_stocker\_t**



```

— int nb_posi_esclave_stocker_tdt
— int indice_stockage_glissant_t
— int indice_stockage_glissant_tdt
— Coordonnee Mtdt
— Coordonnee Mt
— Coordonnee M_noeud_tdt_avant_projection
— Vecteur phi_theta_0
— EnergieMeca energie_frottement
— double energie_penalisation
— int cas_solide
— int cas_collant
— Vecteur * residu
— Mat_pleine * raideur
— DdlElement * ddlElement_assemblage
— Tableau< Noeud * > tabNoeud_pour_assemblage
— Coordonnee force_contact
— Coordonnee force_contact_t
— double F_N_max
— double F_T_max
— double F_N_max_t
— double F_T_max_t
— Tableau< Coordonnee > tabForce_cont
— Tableau< Coordonnee > tabForce_cont_t
— Coordonnee N
— Coordonnee dep_tangentiel
— int nb_decol_t
— int nb_decol_tdt
— double gap_t
— double gap_tdt
— double dep_T_t
— double dep_T_tdt
— double nb_pene_t
— double nb_pene_tdt
— double mult_pene_t
— double mult_pene_tdt
— double mult_tang_t
— double mult_tang_tdt
— double penalisation
— double penalisation_tangentielle
— double penalisation_t
— double penalisation_tangentielle_t
— Fct_nD_contact fct_nD_contact
— Tableau< double > val_penal
— int pt_dans_val_penal
— int type_trajectoire_t
— int type_trajectoire_tdt

```

### Attributs protégés statiques

```

— static list< DdlElement > list_Ddl_global
— static list< Vecteur > list_SM
— static list< Mat_pleine > list_raideur
— static double dep_max =1.e15
— static int niveau_commentaire =0
— static Fonction_nD * fct_pilotage_contact4 =NULL

```

## 6.266.1 Documentation des fonctions membres

### 6.266.1.1 ConditionLi()

```

Condilinaire ElContact::ConditionLi (
    int nb_assemb )

```

```

int numX = noeud->Existe(X1) -1; int pointeur = noeud->PosiAssemb(nb_assemb) + numX;// pointeur
d'assemblage

```

### 6.266.1.2 Contact()

```
bool ElContact::Contact (
    bool init = true )
{return false;};
{return false;};
```

### 6.266.1.3 Valeur\_fct\_nD()

```
double ElContact::Valeur_fct_nD (
    Fonction_nD * fct_nD ) const [protected]
```

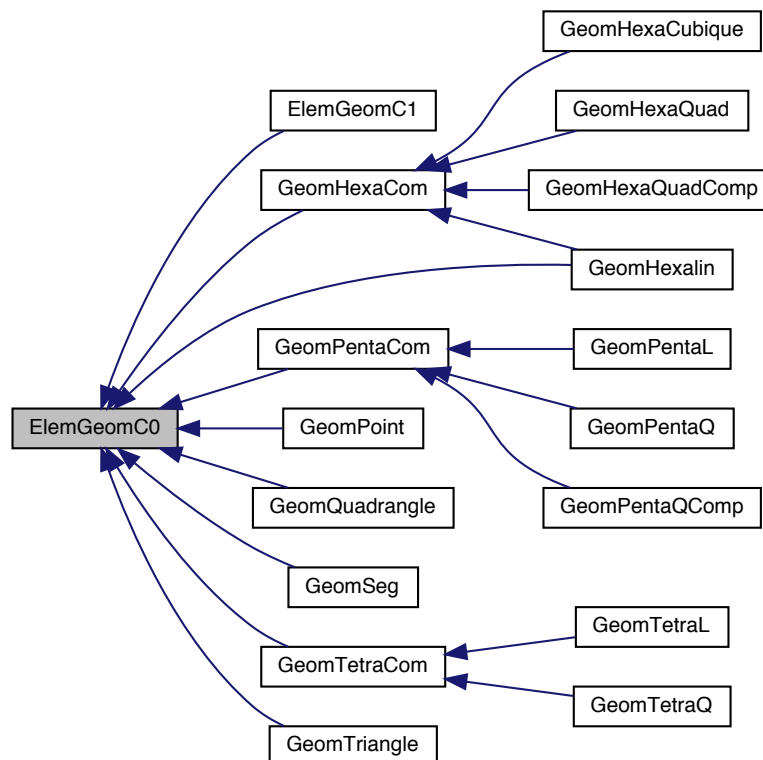
\*\*\*\* en fait les cas X1, X1\_t et X1\_t0 ne vont pas être traités ici mais vont être traités en grandeurs quelconques car il s'agit de vecteur

La documentation de cette classe a été générée à partir du fichier suivant :

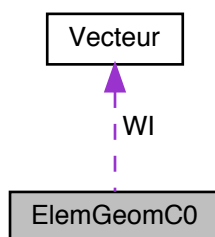
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/ElContact.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/ElContact.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/ElContact\_2.cc

## 6.267 Référence de la classe ElemGeomC0

Graphe d'héritage de ElemGeomC0:



Graphe de collaboration de ElemGeomC0:



## Classes

- class [ConteneurExtrapolation](#)

## Fonctions membres publiques

- **ElemGeomC0** (int dim, int nbi, int nbne, int nbfe, int nbse, [Enum\\_geom](#) geom, [Enum\\_interpol](#) interpol, [Enum\\_type\\_pt\\_integ](#) type\_pti=PTI\_GAUSS)
- **ElemGeomC0** (const [ElemGeomC0](#) &a)
- int **Dimension** () const
- int **Nbi** () const
- int **Nbne** () const
- int **NbFe** () const
- int **NbSe** () const
- [Coordonnee](#) const & **CoorPtInteg** (int i) const
- [Vecteur](#) const & **Phi** (int i)
- [Tableau](#)< [Vecteur](#) > const & **TaPhi** () const
- [Mat\\_pleine](#) const & **Dphi** (int i)
- [Tableau](#)< [Mat\\_pleine](#) > const & **TaDphi** ()
- double **Wi** (int i) const
- [Vecteur](#) const & **TaWi** () const
- [ElemGeomC0](#) const & **ElemFace** (int i) const
- [ElemGeomC0](#) const & **ElemSeg** (int i) const
- bool **Comple** () const
- [Tableau](#)< [Tableau](#)< int > > const & **Nonf** () const
- [Tableau](#)< [Tableau](#)< int > > const & **NonS** () const
- [Tableau](#)< [Coordonnee](#) > const & **PtelemRef** () const
- [Enum\\_geom](#) **TypeGeometrie** () const
- [Enum\\_interpol](#) **TypeInterpolation** () const
- [Enum\\_type\\_pt\\_integ](#) **TypePointIntegration** () const
- [ConteneurExtrapolation](#) const & **ExtrapolationNoeud** (int cas) const
- virtual [ElemGeomC0](#) \* **newElemGeomC0** ([ElemGeomC0](#) \*pt)=0
- [Tableau](#)< int > const & **InvConnec** () const
- [Tableau](#)< int > const & **Ind** () const
- virtual const [Vecteur](#) & **Phi** (const [Coordonnee](#) &M)=0
- virtual const [Mat\\_pleine](#) & **Dphi** (const [Coordonnee](#) &M)=0
- virtual bool **Interieur** (const [Coordonnee](#) &M)=0
- virtual [Coordonnee](#) **Maxi\_Coor\_dans\_directionGM** (const [Coordonnee](#) &M)
- const [Tableau](#)< [Tableau](#)< [Tableau](#)< int > > > & **Trian\_lin** () const
- const [Tableau](#)< [Tableau](#)< [Tableau](#)< int > > > & **Trian\_seg** () const

## Fonctions membres publiques statiques

- static bool **Bases\_naturelles\_duales** (const [Tableau](#)< [Coordonnee](#) > &tab\_M, [Tableau](#)< [Coordonnee](#) > &giB, [Tableau](#)< [Coordonnee](#) > &giH)
- static void **Coor\_phi** (const [Coordonnee](#) &O, const [Tableau](#)< [Coordonnee](#) > &giH\_, const [Coordonnee](#) &A, [Vecteur](#) &phi, [Coordonnee](#) &theta)

## Fonctions membres protégées

- bool **Bases\_naturel\_duales** (const [Tableau](#)< int > &indirec, [Tableau](#)< [Coordonnee](#) > &giB, [Tableau](#)< [Coordonnee](#) > &giH\_, int cas=1) const

## Attributs protégés

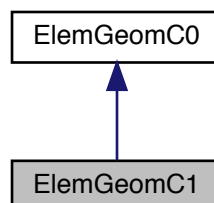
- int **dimension**
- int **NBNE**
- int **NBFE**
- int **NBSE**
- [Tableau](#)< [Tableau](#)< int > > **NONF**
- [Tableau](#)< [Tableau](#)< int > > **NONS**
- [Tableau](#)< int > **INVCONNEX**
- [Tableau](#)< int > **IND**
- [Tableau](#)< [ConteneurExtrapolation](#) > **extrapol**
- [Tableau](#)< [Coordonnee](#) > **ptelem**
- [Tableau](#)< [Coordonnee](#) > **ptInteg**
- [Tableau](#)< [Vecteur](#) > **tabPhi**
- [Tableau](#)< [Mat\\_pleine](#) > **tabDPhi**
- [Vecteur](#) **WI**
- [Tableau](#)< [ElemGeomC0](#) \* > **face**
- [Tableau](#)< [ElemGeomC0](#) \* > **seg**
- [Enum\\_geom](#) **id\_geom**
- [Enum\\_interpol](#) **id\_interpol**
- [Enum\\_type\\_pt\\_integ](#) **id\_type\_pt\_integ**
- [Tableau](#)< [Tableau](#)< [Tableau](#)< int > > > **NONFt**
- [Tableau](#)< [Tableau](#)< [Tableau](#)< int > > > **NONSs**

La documentation de cette classe a été générée à partir du fichier suivant :

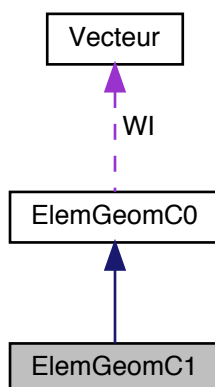
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/ElemGeomC0.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/ElemGeomC0.cc

## 6.268 Référence de la classe ElemGeomC1

Graphe d'héritage de ElemGeomC1:



Graphe de collaboration de ElemGeomC1:



### Fonctions membres publiques

- **ElemGeomC1** (int dim, int nbi, int nbne, int nbfe, int nbse, [Tableau](#)< int > nddNoeud)
- **ElemGeomC1** ([ElemGeomC1](#) &a)

### Attributs protégés

- [Tableau](#)< int > nddNoeud

### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/ElemGeomC1.h





- **ElemMeca** (int num\_maill, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")
- **ElemMeca** (int num\_maill, int num\_id, char \*nom\_interpol, char \*nom\_geom, string info="")
- **ElemMeca** (int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")
- **ElemMeca** (int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab, char \*nom\_interpol, char \*nom\_←  
geom, string info="")
- **ElemMeca** (const [ElemMeca](#) &elt)
- int **TestComplet** ()
- int **Interne\_0** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- int **Interne\_t** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- int **Interne\_tdt** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- const [EnergieMeca](#) & **EnergieTotaleElement** () const
- bool **ContrainteAbsoluePossible** ()
- virtual [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > **Les\_type\_de\_ddl\_internes** (bool absolue) const
- virtual [List\\_io](#)< [TypeQuelconque](#) > **Les\_type\_evolues\_internes** (bool absolue) const
- virtual [List\\_io](#)< [TypeQuelconque](#) > **Les\_types\_particuliers\_internes** (bool absolue) const
- virtual [List\\_io](#)< [TypeQuelconque](#) > **Les\_type\_quelconque\_de\_face** (bool absolue) const
- virtual [List\\_io](#)< [TypeQuelconque](#) > **Les\_type\_quelconque\_de\_arete** (bool absolue) const
- virtual [MatGeomInit](#) **MatricesGeometrique\_Et\_Initiale** (const [ParaAlgoControle](#) &pa)
- virtual void **Plus\_ddl\_Sigma** ()=0
- virtual void **Inactive\_ddl\_Sigma** ()=0
- virtual void **Active\_ddl\_Sigma** ()=0
- virtual void **Active\_premier\_ddl\_Sigma** ()=0
- virtual [DdlElement](#) & **Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual [DdlElement](#) **Tableau\_de\_ERREUR** () const
- virtual void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual void **Plus\_ddl\_Erreur** ()
- virtual void **Inactive\_ddl\_Erreur** ()
- virtual void **Active\_ddl\_Erreur** ()
- bool **ErreurDejaCalculee** ()
- double **Erreur** ()
- virtual void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)=0
- virtual bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)=0
- virtual double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)=0
- void **InitCalculMatriceMassePourRelaxationDynamique** (int casMass\_relax)
- void **CalculMatriceMassePourRelaxationDynamique** (const double &alph, const double &beta, const  
double &lambda, const double &gamma, const double &theta, int casMass\_relax)
- const [BaseB](#) & **Gib\_elemeca** ([Enum\\_dure](#) temps, const [Noeud](#) \*noe)
- void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, int iteg)
- void **Grandeur\_particuliere\_face** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, int face, int iteg)
- void **Grandeur\_particuliere\_arete** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, int arete, int iteg)
- [CompFrotAbstraite](#) \* **LoiDeFrottement** () const
- const [Loi\\_comp\\_abstraite](#) \* **LoiDeComportement** () const
- double **CompressibiliteMoyenne** () const
- void **TransfertAjoutAuNoeuds** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lietendu, const [Tableau](#)< [Tableau](#)<  
double > > &tab\_val, int cas)
- void **TransfertAjoutAuNoeuds** (const [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > &tab\_liQ, [List\\_io](#)<  
[TypeQuelconque](#) > &liQ\_travail, int cas)
- void **Accumul\_aux\_noeuds** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lietendu, [List\\_io](#)< [TypeQuelconque](#) >  
&liQ, int cas)
- void **ActivCalculInvariantsContraintes** ()
- void **ActivCalculInvariantsDeformation** ()
- void **ActivCalculInvariantsVitesseDeformation** ()
- bool **Modif\_orient\_elem** (int cas\_orientation)
- virtual int **CalculNormale\_noeud** ([Enum\\_dure](#) temps, const [Noeud](#) &noe, [Coordonnee](#) &coor)
- int **Interne** ([Enum\\_dure](#) temps, const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- virtual double **Epaisseurs** ([Enum\\_dure](#), const [Coordonnee](#) &)
- virtual double **EpaisseurMoyenne** ([Enum\\_dure](#))
- virtual double **Section** ([Enum\\_dure](#), const [Coordonnee](#) &)
- virtual double **SectionMoyenne** ([Enum\\_dure](#))
- virtual void **Prise\_en\_compte\_des\_consequences\_suppression\_tous\_frontieres** ()
- virtual void **Prise\_en\_compte\_des\_consequences\_suppression\_une\_frontiere** ([EIFrontiere](#) \*elemFront)



- virtual [EIFrontiere](#) \*const **Frontiere\_points** (int num, bool force)
- virtual [EIFrontiere](#) \*const **Frontiere\_lineique** (int num, bool force)
- virtual [EIFrontiere](#) \*const **Frontiere\_surfacique** (int num, bool force)
- virtual void **Initialisation\_avant\_chargement** ()
- virtual void **Mise\_a\_jour\_repere\_anisotropie** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- double **Energie\_stab\_membBiel** ()
- double **Energie\_Hourglass** () const
- double **Energie\_Bulk** () const
- double **Puissance\_Bulk** () const
- double **Force\_StabMembBiel** (int i) const
- double **Energie\_StabMembBiel** (int i) const
- [Temps\\_CPU\\_HZpp](#) **Temps\_lois\_comportement** () const
- [Temps\\_CPU\\_HZpp](#) **Temps\_métrique\_K\_SM** () const
- void **Change\_erreur\_relative** (double &nevez\_erreur\_rel)
- const double **Erreur\_sigma** () const
- const double **Erreur\_sigma\_relative** () const
- **ElemMeca** (int num\_id)
- **ElemMeca** (int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **ElemMeca** (int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt)
- **ElemMeca** (int num\_id, char \*nom\_interpol, char \*nom\_geom)
- **ElemMeca** (int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_←  
geom\_elt)
- **ElemMeca** (int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab, char \*nom\_interpol, char \*nom\_geom)
- **ElemMeca** ([ElemMeca](#) &elt)
- int **TestComplet** ()
- bool **Interne\_0** ([Coordonnee](#) &M)
- bool **Interne\_t** ([Coordonnee](#) &M)
- bool **Interne\_tdt** ([Coordonnee](#) &M)

## Fonctions membres publiques statiques

- static void **ActiveBulkViscosity** (int choix)
- static void **InactiveBulkViscosity** ()
- static void **ChangeCoefsBulkViscosity** (const [DeuxDoubles](#) &coef)

## Attributs publics

- double **E\_elem\_bulk\_t**
- double **E\_elem\_bulk\_tdt**
- double **P\_elem\_bulk**

## Fonctions membres protégées

- [Tableau](#)< [EIFrontiere](#) \* > const & **Frontiere\_elemeca** (int cas, bool force=false)
- void **Cal\_implicit** ([DdlElement](#) &tab\_ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [Tableau2](#)<  
[TenseurBB](#) \* > > d2\_epsBB, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, int nbint, const [Vecteur](#) &poids, const  
[ParaAlgoControle](#) &pa, bool cald\_Dvirtuelle)
- void **Cal\_explicit** ([DdlElement](#) &ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, int nbint, const [Vecteur](#) &poids,  
const [ParaAlgoControle](#) &pa, bool atdt=true)
- void **Cal\_matGeom\_Init** ([Mat\\_pleine](#) &matGeom, [Mat\\_pleine](#) &matInit, [DdlElement](#) &ddl, [Tableau](#)<  
[TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > d2\_epsBB, [Tableau](#)< [TenseurHH](#)  
\* > &d\_sigHH, int nbint, const [Vecteur](#) &poids, const [ParaAlgoControle](#) &pa, bool cald\_Dvirtuelle)
- void **Cal\_Mat\_masse** ([DdlElement](#) &tab\_ddl, [Enum\\_calcul\\_masse](#) type\_matrice\_masse, int nbint, const  
[Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids)
- void **Cal\_implicitap** ([DdlElement](#) &tab\_ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [Tableau2](#)<  
[TenseurBB](#) \* > > d2\_epsBB, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, int nbint1, [Vecteur](#) &poids1, int nbint2,  
const [Vecteur](#) &poids2, const [ParaAlgoControle](#) &pa)
- void **Cal\_explicitap** ([DdlElement](#) &ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, int nbint, const [Vecteur](#) &poids,  
bool atdt=true)
- void **Cal\_matGeom\_Initap** ([Mat\\_pleine](#) &matGeom, [Mat\\_pleine](#) &matInit, [DdlElement](#) &ddl, [Tableau](#)<  
[TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > d2\_epsBB, [Tableau](#)< [TenseurHH](#) \*  
> &d\_sigHH, int nbint, const [Vecteur](#) &poids)

- [Vecteur](#) & [SM\\_charge\\_surf\\_E](#) ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element::ResRaid](#) [SMR\\_charge\\_surf\\_I](#) ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & [SM\\_charge\\_pres\\_E](#) ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, double pression, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element::ResRaid](#) [SMR\\_charge\\_pres\\_I](#) ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, double pression, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & [SM\\_charge\\_line\\_E](#) ([DdlElement](#) &ddls, int nArete, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element::ResRaid](#) [SMR\\_charge\\_line\\_I](#) ([DdlElement](#) &ddlA, int nArete, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & [SM\\_charge\\_line\\_Suiv\\_E](#) ([DdlElement](#) &ddls, int nArete, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element::ResRaid](#) [SMR\\_charge\\_line\\_Suiv\\_I](#) ([DdlElement](#) &ddlA, int nArete, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & [SM\\_charge\\_surf\\_Suiv\\_E](#) ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element::ResRaid](#) [SMR\\_charge\\_surf\\_Suiv\\_I](#) ([DdlElement](#) &ddlA, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & [SM\\_charge\\_vol\\_E](#) ([DdlElement](#) &ddls, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume←\_finale\_, bool atdt=true)
- void [SMR\\_charge\\_vol\\_I](#) ([DdlElement](#) &ddls, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume←\_finale\_)
- [Vecteur](#) & [SM\\_charge\\_hydro\\_E](#) ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, const [Coordonnee](#) &M\_liquide, bool sans\_limitation, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element::ResRaid](#) [SMR\\_charge\\_hydro\\_I](#) ([DdlElement](#) &ddlA, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, const [Coordonnee](#) &M\_liquide, bool sans\_limitation, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & [SM\\_charge\\_hydrodyn\\_E](#) (const double &poidvol, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, [Courbe1D](#) \*frot\_fluid, const [Vecteur](#) &poids, [Courbe1D](#) \*coef\_aero\_n, int numfront, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &, bool atdt=true)
- [Element::ResRaid](#) [SM\\_charge\\_hydrodyn\\_I](#) (const double &poidvol, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, [Courbe1D](#) \*frot\_fluid, const [Vecteur](#) &poids, [DdlElement](#) &ddls, [Courbe1D](#) \*coef\_aero\_n, int numfront, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- virtual [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int num, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)=0
- virtual [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int num, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)=0
- void [Init\\_hourglass\\_comp](#) (const [ElemGeomC0](#) &elgeHour, const string &str\_precision, [LoiAbstraiteGeneral](#) \*loiHourglass, const [BlocGen](#) &bloc)
- void [Cal\\_implicit\\_hourglass](#) ()
- void [Cal\\_explicit\\_hourglass](#) (bool atdt)
- void [Mise\\_a\\_jour\\_A\\_calculer\\_force\\_stab](#) ()
- void [Cal\\_implicit\\_StabMembBiel](#) (int iteg, const [Met\\_abstraite::Impli](#) &met, const int &nbint, const double &poid\_volume, [Tableau](#)< int > \*noeud\_a\_prendre\_en\_compte)
- void [Cal\\_explicit\\_StabMembBiel](#) (int iteg, const [Met\\_abstraite::Expli\\_t\\_tdt](#) met, const int &nbint, const double &poids\_volume, [Tableau](#)< int > \*noeud\_a\_prendre\_en\_compte)
- void [SigmaAuNoeud\\_ResRaid](#) (const int nbne, const [Tableau](#)< [Vecteur](#) > &taphi, const [Vecteur](#) &poids, [Tableau](#)< [Vecteur](#) \* > &resErr, [Mat\\_pleine](#) &raidErr, const [Tableau](#)< [Vecteur](#) > &taphiEr, const [Vecteur](#) &poidsEr)
- void [Cal\\_ErrElem](#) (int type, double &errElemRelative, double &numérateur, double &denominateur, const int nbne, const [Tableau](#)< [Vecteur](#) > &taphi, const [Vecteur](#) &poids, const [Tableau](#)< [Vecteur](#) > &taphiEr, const [Vecteur](#) &poidsEr)

- void **Cal\_ErrAuxNoeuds** (const int nbne, const [Tableau](#)< [Vecteur](#) > &taphi, const [Vecteur](#) &poids, [Tableau](#)< [Vecteur](#) \* > &resErr)
- void **VarDualSort** (ofstream &sort, [Tableau](#)< string > &nom, int nbint, int cas)
- void **AffDefCont** (ofstream &sort, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [TenseurBB](#) &eps0BB, [TenseurBB](#) &epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &DeltaEpsBB, [TenseurHH](#) &sigHH, [TenseurHB](#) &epsHB, [TenseurHB](#) &sigHB, [Coordonnee](#) &valPropreEps, [Coordonnee](#) &valPropreDeps, [Coordonnee](#) &valPropreSig, double Mises, [TenseurBB](#) &epsAlmBB, [TenseurBB](#) &epslogBB, int indic)
- void **AffDefCont1D** (ofstream &sort, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [TenseurBB](#) &eps0BB, [TenseurBB](#) &epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &DeltaEpsBB, [TenseurHH](#) &sigHH, [TenseurHB](#) &epsHB, [TenseurHB](#) &sigHB, [Coordonnee](#) &valPropreEps, [Coordonnee](#) &valPropreDeps, [Coordonnee](#) &valPropreSig, double Mises, [TenseurBB](#) &epsAlmBB, [TenseurBB](#) &epslogBB, int indic)
- void **AffDefCont2D** (ofstream &sort, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [TenseurBB](#) &eps0BB, [TenseurBB](#) &epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &DeltaEpsBB, [TenseurHH](#) &sigHH, [TenseurHB](#) &epsHB, [TenseurHB](#) &sigHB, [Coordonnee](#) &valPropreEps, [Coordonnee](#) &valPropreDeps, [Coordonnee](#) &valPropreSig, double Mises, [TenseurBB](#) &epsAlmBB, [TenseurBB](#) &epslogBB, int indic)
- void **AffDefCont3D** (ofstream &sort, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [TenseurBB](#) &eps0BB, [TenseurBB](#) &epsBB, [TenseurBB](#) &DepsBB, [TenseurBB](#) &DeltaEpsBB, [TenseurHH](#) &sigHH, [TenseurHB](#) &epsHB, [TenseurHB](#) &sigHB, [Coordonnee](#) &valPropreEps, [Coordonnee](#) &valPropreDeps, [Coordonnee](#) &valPropreSig, double Mises, [TenseurBB](#) &epsAlmBB, [TenseurBB](#) &epslogBB, int indic)
- void **Ad\_ddl\_Sigma** (const [DdlElement](#) &tab\_ddlErr)
- void **Inact\_ddl\_primaire** ([DdlElement](#) &tab\_ddl)
- void **Act\_ddl\_primaire** ([DdlElement](#) &tab\_ddl)
- void **Inact\_ddl\_Sigma** ([DdlElement](#) &tab\_ddlErr)
- void **Act\_ddl\_Sigma** ([DdlElement](#) &tab\_ddlErr)
- void **Act\_premier\_ddl\_Sigma** ()
- void **LectureDesContraintes** (bool cas, [UtilLecture](#) \*entreePrinc, [Tableau](#)< [TenseurHH](#) \* > &tabSigHH)
- void **ContraintesEnAbsolues** (bool cas, [Tableau](#)< [TenseurHH](#) \* > &tabSigHH, [Tableau](#)< [Vecteur](#) > &tab←  
Sig)
- void **Lecture\_bas\_inf** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_bas\_inf** (ofstream &sort, const int cas)
- double **Interne\_Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps, int nb\_noeud=0)
- [Coordonnee](#) **CoordPtInt** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Coordonnee](#) **CoordPtIntFace** (int face, [Enum\\_dure](#) temps, int iteg, bool &erreur)
- [Coordonnee](#) **CoordPtIntArete** (int arete, [Enum\\_dure](#) temps, int iteg, bool &erreur)
- [Element](#) \* **Complete\_ElemMeca** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- int **PtLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Tableau](#)< double > **Valeur\_multi** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg, int cas)
- void **Valeurs\_Tensorielles** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg, int cas)
- [Tableau](#)< double > **Valeur\_multi\_interpoler\_ou\_calculer** (bool absolue, [Enum\\_dure](#) temps, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, [Deformation](#) &defor, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite](#)←  
::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite::Expli](#) \*ex\_expli)
- void **Valeurs\_Tensorielles\_interpoler\_ou\_calculer** (bool absolue, [Enum\\_dure](#) temps, [List\\_io](#)< [TypeQuelconque](#) > &enu, [Deformation](#) &defor, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite](#)←  
::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite::Expli](#) \*ex\_expli)
- virtual [ElemGeomC0](#) & **ElementGeometrie** ([Enum\\_ddl](#) ddl) const
- virtual int **NbGrandeurGene** ([Enum\\_ddl](#) enu) const
- virtual int **Dim\_sig\_eps** () const =0
- bool **Bulk\_visco\_actif** ()
- double & **C\_traceBulk** ()
- double & **C\_carreBulk** ()
- **DeuxDoubles ModifContrainteBulk** ([Enum\\_dure](#) temps, const [TenseurHH](#) &gijHH\_tdt, [TenseurHH](#) &sigHH, const [TenseurBB](#) &DepsBB\_tdt, [TenseurHH](#) &sigbulkHH)
- void **TdtversT** ()
- void **TversTdt** ()
- void **Calcul\_Integ\_vol\_et\_temps** (int cas, const double &poid\_jacobien, int iteg)
- void **Init\_Integ\_vol\_et\_temps** ()
- void **Cal\_implicit** ([DdlElement](#) &tab\_ddl, [Tableau](#)< [TenseurBB](#) \* > &tabEpsBB, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [TenseurHH](#) \* > &tabSigHH, int nbint, [Vecteur](#) &poids)
- void **Cal\_explicit** ([DdlElement](#) &ddl, [Tableau](#)< [TenseurBB](#) \* > &tabEpsBB, [Tableau](#)< [TenseurBB](#) \* > &d←  
\_epsBB, [Tableau](#)< [TenseurHH](#) \* > &tabSigHH, int nbint, [Vecteur](#) &poids)

- void **VarDualSort** (ofstream &sort, [Tableau](#)< string > &nom, [Tableau](#)< [TenseurHH](#) \* > &tabSigHH, [Tableau](#)< [TenseurBB](#) \* > &tabEpsBB, int nbint, int cas)
- void **AffDefCont** (ofstream &sort, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [TenseurBB](#) &eps0BB, [TenseurBB](#) &epsBB, [TenseurHH](#) &sigHH, [TenseurHB](#) &epsHB, [TenseurHB](#) &sigHB, [Vecteur](#) &valPropreEps, [Vecteur](#) &valPropreSig, int indic)
- void **AffDefCont1D** (ofstream &sort, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [TenseurBB](#) &eps0BB, [TenseurBB](#) &epsBB, [TenseurHH](#) &sigHH, [TenseurHB](#) &epsHB, [TenseurHB](#) &sigHB, [Vecteur](#) &valPropreEps, [Vecteur](#) &valPropreSig, int indic)
- void **AffDefCont2D** (ofstream &sort, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [TenseurBB](#) &eps0BB, [TenseurBB](#) &epsBB, [TenseurHH](#) &sigHH, [TenseurHB](#) &epsHB, [TenseurHB](#) &sigHB, [Vecteur](#) &valPropreEps, [Vecteur](#) &valPropreSig, int indic)
- void **AffDefCont3D** (ofstream &sort, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [TenseurBB](#) &eps0BB, [TenseurBB](#) &epsBB, [TenseurHH](#) &sigHH, [TenseurHB](#) &epsHB, [TenseurHB](#) &sigHB, [Vecteur](#) &valPropreEps, [Vecteur](#) &valPropreSig, int indic)
- bool **Interne** ([Coordonnee](#) &M)

### Attributs protégés

- [StabMembBiel](#) \* **pt\_StabMembBiel**
- double **maxi\_F\_t**
- [Mat\\_pleine](#) \* **matD**
- [Vecteur](#) \* **resD**
- [LesPtIntegMecaInterne](#) \* **lesPtIntegMecaInterne**
- [LesChargeExtSurElement](#) \* **lesChargeExtSurEle**
- [Loi\\_comp\\_abstraite](#) \* **loiComp**
- [CompThermoPhysiqueAbstraite](#) \* **loiTP**
- [CompFrotAbstraite](#) \* **loiFrot**
- int **dilatation**
- [Met\\_abstraite](#) \* **met**
- [Deformation](#) \* **def**
- [Deformation](#) \* **defEr**
- [Tableau](#)< [Deformation](#) \* > **defSurf**
- [Tableau](#)< [Deformation](#) \* > **defArete**
- [Deformation](#) \* **defMas**
- double \* **sigErreur**
- double \* **sigErreur\_relative**
- [Tableau](#)< [Loi\\_comp\\_abstraite::SaveResul](#) \* > **tabSaveDon**
- [Tableau](#)< [CompThermoPhysiqueAbstraite::SaveResul](#) \* > **tabSaveTP**
- [Tableau](#)< [Deformation::SaveDefResul](#) \* > **tabSaveDefDon**
- [Tableau](#)< [EnergieMeca](#) > **tab\_energ**
- [Tableau](#)< [EnergieMeca](#) > **tab\_energ\_t**
- [EnergieMeca](#) **energie\_totale\_t**
- [EnergieMeca](#) **energie\_totale**
- bool **premier\_calcul\_meca\_impli\_expli**
- double **masse\_volumique**
- [Enum\\_StabHourglass](#) **type\_stabHourglass**
- double **E\_Hourglass**

### Attributs protégés statiques

- static [Coordonnee](#) &(Met\_abstraite::\* **PointM**)([Tableau](#)< [Noeud](#) \* > &tab\_noeud, [Vecteur](#) &Phi)

## 6.269.1 Documentation des fonctions membres

### 6.269.1.1 ErreurElement()

```
void ElemMeca::ErreurElement (
    int type,
    double & errElemRelative,
```

```
double & numerateur,
double & denominateur ) [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée dans [Biel\\_axi](#), [Biel\\_axiQ](#), [Biellette](#), [BielletteC1](#), [BielletteQ](#), [ElemPoint](#), [HexaMemb](#), [PentaMemb](#), [QuadAxiMemb](#), [QuadraMemb](#), [SfeMembT](#), [TetraMemb](#), [TriaAxiMemb](#), et [TriaMemb](#).

### 6.269.1.2 Init\_hourglass\_comp()

```
void ElemMeca::Init_hourglass_comp (
const ElemGeomC0 & elgeHour,
const string & str_precision,
LoiAbstraiteGeneral * loiHourglass,
const BlocGen & bloc ) [protected]
```

!!! en fait pour l'instant on n'utilise pas elgehour, on considère que le nombre de pti complet est celui de l'élément sans info annexe !!!

### 6.269.1.3 Tableau\_de\_Sig1()

```
DdlElement & ElemMeca::Tableau_de_Sig1 ( ) const [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

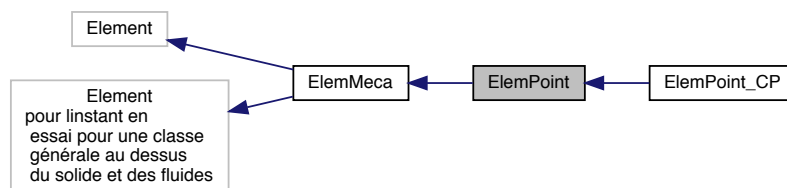
Réimplémentée dans [Biel\\_axi](#), [Biel\\_axiQ](#), [Biellette](#), [BielletteC1](#), [BielletteQ](#), [ElemPoint](#), [HexaMemb](#), [PentaMemb](#), [QuadAxiMemb](#), [QuadraMemb](#), [SfeMembT](#), [TetraMemb](#), [TriaAxiMemb](#), et [TriaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

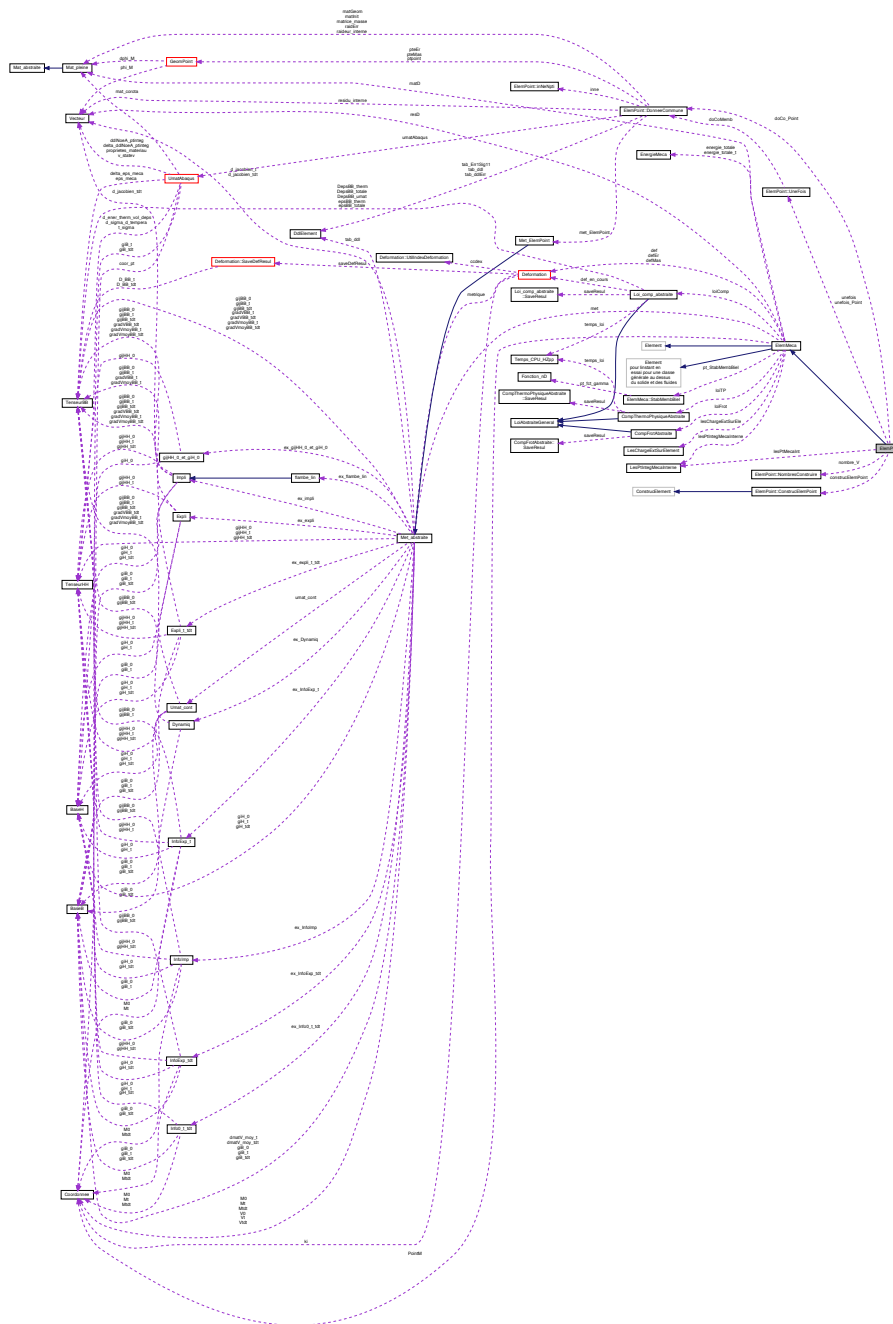
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermiGene.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca3.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca4.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca5.cc

## 6.270 Référence de la classe ElemPoint

Graphe d'héritage de ElemPoint:



Graphe de collaboration de ElemPoint:



## Classes

- class [ConstrucElemPoint](#)
- class [DonneeCommune](#)
- class [inNeNpti](#)
- class [NombresConstruire](#)
- class [UneFois](#)

## Fonctions membres publiques

- **ElemPoint** (int dimension=-1)
- **ElemPoint** (int num\_mail, int num\_id, int dimension=-1)
- **ElemPoint** (const [ElemPoint](#) &elem)



- Element \* **Nevez\_copie** () const
- ElemPoint & **operator=** (ElemPoint &biel)
- void **LectureDonneesParticulieres** (UtilLecture \*, Tableau< Noeud \* > \*)
- Element::ResRaid **Calcul\_implicit** (const ParaAlgoControle &pa)
- Vecteur \* **CalculResidu\_t** (const ParaAlgoControle &pa)
- Vecteur \* **CalculResidu\_tdt** (const ParaAlgoControle &pa)
- Mat\_pleine \* **CalculMatriceMasse** (Enum\_calcul\_masse id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** (Enum\_dure)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- const DdlElement & **TableauDdl** () const
- void **Libere** ()
- void **DefLoi** (LoiAbstraiteGeneral \*NouvelleLoi)
- int **TestComplet** ()
- Element \* **Complete** (BlocGen &bloc, LesFonctions\_nD \*lesFonctionsnD)
- Element \* **Complet\_Hourglass** (LoiAbstraiteGeneral \*NouvelleLoi, const BlocGen &bloc)
- ElemGeomC0 & **ElementGeometrique** () const
- const ElemGeomC0 & **ElementGeometrique\_const** () const
- Coordonnee & **Point\_physique** (const Coordonnee &c\_int, Coordonnee &co, Enum\_dure temps)
- void **Point\_physique** (const Coordonnee &c\_int, Tableau< Coordonnee > &t\_co)
- void **AfficheVarDual** (ofstream &sort, Tableau< string > &nom)
- void **Info\_com\_Element** (UtilLecture \*entreePrinc, string &ordre, Tableau< Noeud \* > \*tabMaillageNoeud)
- int **PointLePlusPres** (Enum\_dure, Enum\_ddl, const Coordonnee &)
- Coordonnee **CoordPtInteg** (Enum\_dure temps, Enum\_ddl enu, int iteg, bool &erreur)
- Tableau< double > **Valeur\_a\_diff\_temps** (bool absolue, Enum\_dure enu\_t, const List\_io< Ddl\_enum\_etendu > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, Enum\_dure enu\_t, List\_io< TypeQuelconque > &enu, int iteg)
- bool **SurfExiste** (int) const
- bool **ArreteExiste** (int) const
- virtual void **Associer\_noeud** (Noeud \*noeu)
- void **Ecriture\_Abaqus** (bool utilisation\_umat\_interne)
- void **CalculUmatAbaqus** (ParaAlgoControle &pa)
- virtual void **InitialisationUmatAbaqus** ()
- int **NbIteration** () const
- void **Lecture\_base\_info** (ifstream &ent, const Tableau< Noeud \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- ElemMeca::MatGeomInit **MatricesGeometrique\_Et\_Initiale** (const ParaAlgoControle &pa)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** (UtilLecture \*entreePrinc)
- bool **ContraintesAbsolues** (Tableau< Vecteur > &tabSig)
- DdlElement & **Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- Tableau< ElFrontiere \* > const & **Frontiere** (bool force=false)
- void **ConstTabDdl** ()

## Fonctions membres publiques statiques

- static const ElemPoint::inNeNpti & **Lecture\_Abaqus** (bool utilisation\_umat\_interne)
- static const ElemPoint::inNeNpti & **IncreElemPtint\_encours** ()

## Fonctions membres protégées

- int `Dim_sig_eps` () const
- virtual `ElFrontiere` \* `new_frontiere_lin` (int, `Tableau`< `Noeud` \* > &tab, `DdlElement` &ddelem)
- virtual `ElFrontiere` \* `new_frontiere_surf` (int, `Tableau`< `Noeud` \* > &tab, `DdlElement` &ddelem)
- void `Init` (int dim\_tenseur)
- `DonneeCommune` \* `Def_DonneeCommune` (int dim\_tenseur)
- void `Destruction` ()
- void `ChangeNombrePtinteg` (int nevez\_nbi, int dim\_tens)
- `Vecteur` \* `CalculResidu` (bool atdt, const `ParaAlgoControle` &pa)
- `ElemPoint` (`ElemPoint::UneFois` &unefois, `Enum_geom` nouveau\_id, int dimension=-1)
- `ElemPoint` (`ElemPoint::UneFois` &unefois, `Enum_geom` nouveau\_id, int num\_mail, int num\_id, int dimension=-1)
- void `InitialisationUmatAbaqus_interne` (int dimension=-1)

## Attributs protégés

- `LesPtIntegMecalInterne` `lesPtMecalInt`
- int `nbi`
- int `nb_appelsCalculUmat`
- `UneFois` \* `unefois`

## Attributs protégés statiques

- static `DonneeCommune` \* `doCo_Point` = NULL
- static `UneFois` `unefois_Point`
- static `NombresConstruire` `nombre_V`
- static `ConstrucElemPoint` `construcElemPoint`

## Membres hérités additionnels

### 6.270.1 Documentation des fonctions membres

#### 6.270.1.1 Active\_ddl\_Sigma()

void `ElemPoint::Active_ddl_Sigma` ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

#### 6.270.1.2 Active\_premier\_ddl\_Sigma()

void `ElemPoint::Active_premier_ddl_Sigma` ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

#### 6.270.1.3 ContraintesAbsolues()

bool `ElemPoint::ContraintesAbsolues` (   
     `Tableau`< `Vecteur` > & `tabSig` ) [virtual]  
 Implémente [ElemMeca](#).

#### 6.270.1.4 Dim\_sig\_eps()

int `ElemPoint::Dim_sig_eps` ( ) const [inline], [protected], [virtual]  
 Implémente [ElemMeca](#).



### 6.270.1.5 ErreurElement()

```
void ElemPoint::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en Réimplémentée à partir de [ElemMeca](#).

### 6.270.1.6 Inactive\_ddl\_Sigma()

```
void ElemPoint::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.270.1.7 LectureContraintes()

```
void ElemPoint::LectureContraintes (
    UtilLecture * entreePrinc ) [virtual]
```

Implémente [ElemMeca](#).

### 6.270.1.8 Long\_arrete\_mini\_sur\_c()

```
double ElemPoint::Long_arrete_mini_sur_c (
    Enum_dure ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.270.1.9 MatricesGeometrique\_Et\_Initiale()

```
ElemMeca::MatGeomInit ElemPoint::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.270.1.10 new\_frontiere\_lin()

```
virtual ElFrontiere * ElemPoint::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.270.1.11 new\_frontiere\_surf()

```
virtual ElFrontiere * ElemPoint::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.270.1.12 Plus\_ddl\_Sigma()

```
void ElemPoint::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.270.1.13 Tableau\_de\_Sig1()

`DdlElement` & `ElemPoint::Tableau_de_Sig1 ( ) const [inline], [virtual]`

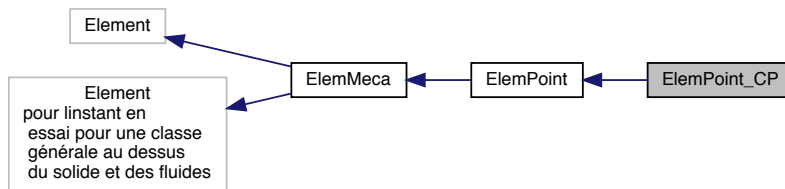
!!!!!!! pour l'instant en virtuelle il faudra après en Réimplémentée à partir de `ElemMeca`.

La documentation de cette classe a été générée à partir du fichier suivant :

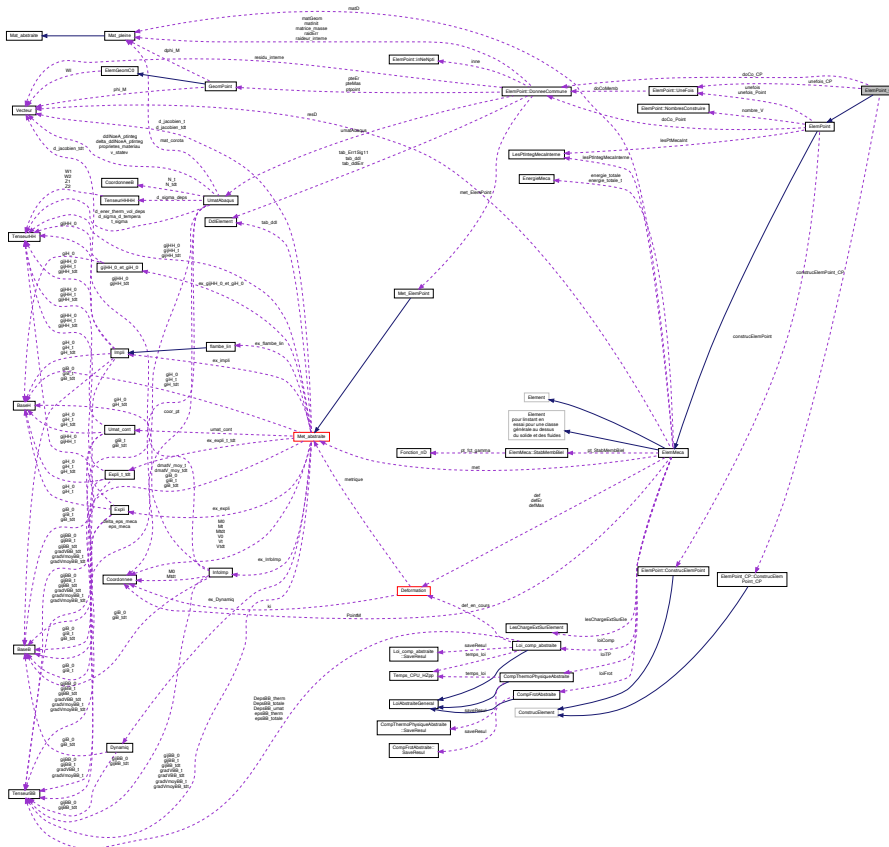
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.cc

## 6.271 Référence de la classe ElemPoint\_CP

Graphe d'héritage de `ElemPoint_CP`:



Graphe de collaboration de `ElemPoint_CP`:



## Classes

— class [ConstrucElemPoint\\_CP](#)

## Fonctions membres publiques

— [ElemPoint\\_CP](#) (int num\_mail, int num\_id)  
 — [ElemPoint\\_CP](#) (const [ElemPoint\\_CP](#) &elem)  
 — Element \* [Nevez\\_copie](#) () const  
 — [ElemPoint\\_CP](#) & [operator=](#) ([ElemPoint\\_CP](#) &biel)  
 — virtual void [InitialisationUmatAbaqus](#) ()

## Fonctions membres publiques statiques

— static const [ElemPoint\\_CP::inNeNpti](#) & [Lecture\\_Abaqus](#) (bool utilisation\_umat\_interne)  
 — static const [ElemPoint\\_CP::inNeNpti](#) & [InceElemPtint\\_encours](#) ()

## Fonctions membres protégées

— int [Dim\\_sig\\_eps](#) () const

## Attributs protégés statiques

— static [ElemPoint::DonneeCommune](#) \* [doCo\\_CP](#) = NULL  
 — static [ElemPoint::UneFois](#) [unefois\\_CP](#)  
 — static [ConstrucElemPoint\\_CP](#) [construcElemPoint\\_CP](#)

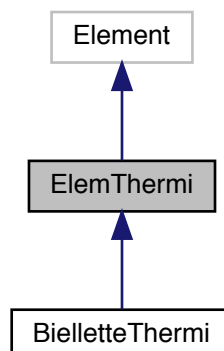
## Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

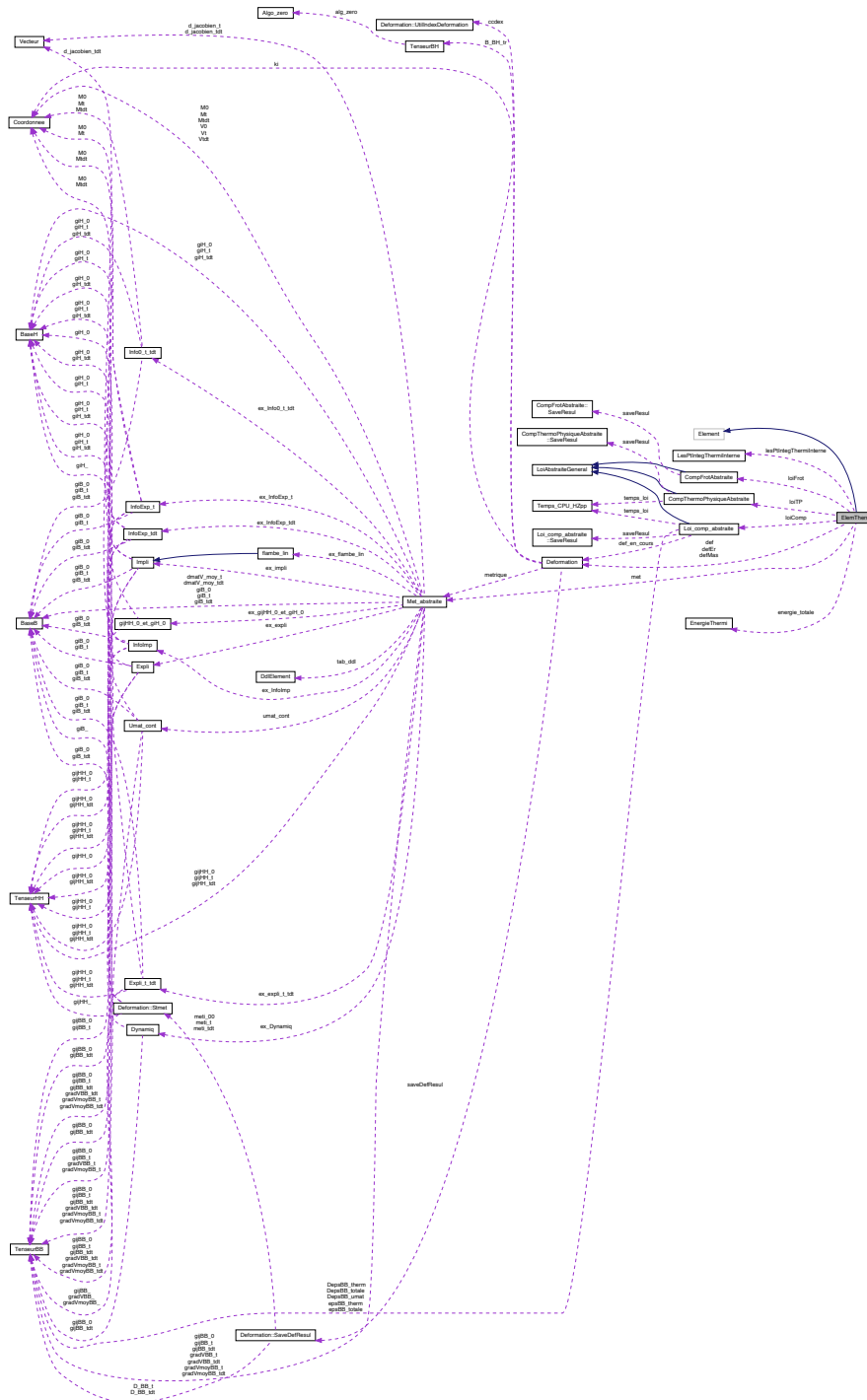
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint\_CP.h  
 — /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint\_CP.cc

## 6.272 Référence de la classe ElemThermi

Graphe d'héritage de ElemThermi:



Graphe de collaboration de ElemThermi:



**Classes**

- class [MatGeomInit](#)

**Fonctions membres publiques**

- **ElemThermi** (int num\_maill, int num\_id)
- **ElemThermi** (int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)

- **ElemThermi** (int num\_maill, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")
- **ElemThermi** (int num\_maill, int num\_id, char \*nom\_interpol, char \*nom\_geom, string info="")
- **ElemThermi** (int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")
- **ElemThermi** (int num\_maill, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab, char \*nom\_interpol, char \*nom\_←\_geom, string info="")
- **ElemThermi** (const [ElemThermi](#) &elt)
- int **TestComple** ()
- int **Interne\_0** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- int **Interne\_t** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- int **Interne\_tdt** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- const [EnergieThermi](#) & [EnergieTotaleElement](#) () const
- bool **FluxAbsoluePossible** ()
- virtual [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > **Les\_type\_de\_ddl\_internes** (bool absolue) const
- virtual [List\\_io](#)< [TypeQuelconque](#) > **Les\_type\_evolues\_internes** (bool absolue) const
- virtual [List\\_io](#)< [TypeQuelconque](#) > **Les\_types\_particuliers\_internes** (bool absolue) const
- virtual [List\\_io](#)< [TypeQuelconque](#) > **Les\_type\_quelconque\_de\_face** (bool absolue) const
- virtual [List\\_io](#)< [TypeQuelconque](#) > **Les\_type\_quelconque\_de\_arete** (bool absolue) const
- virtual [MatGeomInit](#) [MatricesGeometrique\\_Et\\_Initiale](#) (const [ParaAlgoControle](#) &pa)
- virtual void **Plus\_ddl\_Flux** ()=0
- virtual void **Inactive\_ddl\_Flux** ()=0
- virtual void **Active\_ddl\_Flux** ()=0
- virtual void **Active\_premier\_ddl\_Flux** ()=0
- virtual [DdlElement](#) & [Tableau\\_de\\_Flux1](#) () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual [DdlElement](#) [Tableau\\_de\\_ERREUR](#) () const
- virtual void [ErreurElement](#) (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual void **Plus\_ddl\_Erreur** ()
- virtual void **Inactive\_ddl\_Erreur** ()
- virtual void **Active\_ddl\_Erreur** ()
- bool **ErreurDejaCalculee** ()
- double **Erreur** ()
- virtual void **LectureFlux** ([UtilLecture](#) \*entreePrinc)=0
- virtual bool **FluxAbsolues** ([Tableau](#)< [Vecteur](#) > &tabFlux)=0
- virtual double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)=0
- void **InitCalculMatriceMassePourRelaxationDynamique** (int casMass\_relax)
- void **CalculMatriceMassePourRelaxationDynamique** (const double &alph, const double &beta, const double &lambda, const double &gamma, const double &theta, int casMass\_relax)
- const [BaseB](#) & [Gib\\_elemeca](#) ([Enum\\_dure](#) temps, const [Noeud](#) \*noe)
- void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, int iteg)
- void **Grandeur\_particuliere\_face** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, int face, int iteg)
- void **Grandeur\_particuliere\_arete** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, int arete, int iteg)
- [CompFrotAbstraite](#) \* [LoiDeFrottement](#) () const
- void **TransfertAjoutAuNoeuds** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lietendu, const [Tableau](#)< [Tableau](#)< double > > &tab\_val, int cas)
- void **TransfertAjoutAuNoeuds** (const [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > &tab\_liQ, [List\\_io](#)< [TypeQuelconque](#) > &liQ\_travail, int cas)
- void **Accumul\_aux\_noeuds** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lietendu, [List\\_io](#)< [TypeQuelconque](#) > &liQ, int cas)
- bool **Modif\_orient\_elem** (int cas\_orientation)
- virtual int **CalculNormale\_noeud** ([Enum\\_dure](#) temps, const [Noeud](#) &noe, [Coordonnee](#) &coor)
- int **Interne** ([Enum\\_dure](#) temps, const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- virtual double **Epaisseurs** ([Enum\\_dure](#), const [Coordonnee](#) &)
- virtual double **EpaisseurMoyenne** ([Enum\\_dure](#))
- virtual double **Section** ([Enum\\_dure](#), const [Coordonnee](#) &)
- virtual double **SectionMoyenne** ([Enum\\_dure](#))
- virtual void **Prise\_en\_compte\_des\_consequences\_suppression\_tous\_frontieres** ()
- virtual void **Prise\_en\_compte\_des\_consequences\_suppression\_une\_frontiere** ([EIFrontiere](#) \*elemFront)
- virtual [EIFrontiere](#) \*const **Frontiere\_points** (int num, bool force)
- virtual [EIFrontiere](#) \*const **Frontiere\_lineique** (int num, bool force)
- virtual [EIFrontiere](#) \*const **Frontiere\_surfacique** (int num, bool force)
- virtual void **Initialisation\_avant\_chargement** ()
- double **Energie\_Hourglass** ()
- double **Energie\_Bulk** () const
- double **Puissance\_Bulk** () const

## Fonctions membres publiques statiques

- static void **ActiveBulkViscosity** (int choix)
- static void **InactiveBulkViscosity** ()
- static void **ChangeCoefsBulkViscosity** (const [DeuxDoubles](#) &coef)

## Attributs publics

- double **E\_elem\_bulk\_t**
- double **E\_elem\_bulk\_tdt**
- double **P\_elem\_bulk**

## Fonctions membres protégées

- [Tableau](#)< [ElFrontiere](#) \* > const & **Frontiere\_elethermi** (int cas, bool force=false)
- void **Cal\_implicit** ([DdlElement](#) &tab\_ddl, [Tableau](#)< [CoordonneeB](#) > &d\_gradTB, [Tableau](#)< [Tableau2](#)< [CoordonneeB](#) > \* > d2\_gradTB, [Tableau](#)< [CoordonneeH](#) > &d\_fluxH, int nbint, const [Vecteur](#) &poids, const [ParaAlgoControle](#) &pa, bool cald\_DGradTvirtuelle)
- void **Cal\_explicit** ([DdlElement](#) &ddl, [Tableau](#)< [CoordonneeB](#) > &d\_gradTB, int nbint, const [Vecteur](#) &poids, const [ParaAlgoControle](#) &pa, bool atdt=true)
- void **Cal\_matGeom\_Init** ([Mat\\_pleine](#) &matGeom, [Mat\\_pleine](#) &matInit, [DdlElement](#) &ddl, [Tableau](#)< [CoordonneeB](#) > &d\_gradTB, [Tableau](#)< [Tableau2](#)< [CoordonneeB](#) > \* > d2\_gradTB, [Tableau](#)< [CoordonneeH](#) > &d\_fluxH, int nbint, const [Vecteur](#) &poids, const [ParaAlgoControle](#) &pa, bool cald\_↵Dvirtuelle)
- void **Cal\_Mat\_masse** ([DdlElement](#) &tab\_ddl, [Enum\\_calcul\\_masse](#) type\_matrice\_masse, int nbint, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids)
- void **Cal\_implicitap** ([DdlElement](#) &tab\_ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [Tableau2](#)< [CoordonneeB](#) > \* > d2\_gradTB, [Tableau](#)< [CoordonneeH](#) > &d\_fluxH, int nbint1, [Vecteur](#) &poids1, int nbint2, const [Vecteur](#) &poids2, const [ParaAlgoControle](#) &pa)
- void **Cal\_explicitap** ([DdlElement](#) &ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, int nbint, const [Vecteur](#) &poids, bool atdt=true)
- void **Cal\_matGeom\_Initap** ([Mat\\_pleine](#) &matGeom, [Mat\\_pleine](#) &matInit, [DdlElement](#) &ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [Tableau2](#)< [CoordonneeB](#) > \* > d2\_gradTB, [Tableau](#)< [CoordonneeH](#) > &d\_fluxH, int nbint, const [Vecteur](#) &poids)
- [Vecteur](#) & **SM\_charge\_surf\_E** ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element](#)::ResRaid **SMR\_charge\_surf\_I** ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & **SM\_charge\_pres\_E** ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, double pression, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element](#)::ResRaid **SMR\_charge\_pres\_I** ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, double pression, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & **SM\_charge\_line\_E** ([DdlElement](#) &ddls, int nArete, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element](#)::ResRaid **SMR\_charge\_line\_I** ([DdlElement](#) &ddlA, int nArete, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & **SM\_charge\_line\_Suiv\_E** ([DdlElement](#) &ddls, int nArete, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element](#)::ResRaid **SMR\_charge\_line\_Suiv\_I** ([DdlElement](#) &ddlA, int nArete, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & **SM\_charge\_surf\_Suiv\_E** ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa, bool atdt=true)
- [Element](#)::ResRaid **SMR\_charge\_surf\_Suiv\_I** ([DdlElement](#) &ddlA, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & **SM\_charge\_vol\_E** ([DdlElement](#) &ddls, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa, bool atdt=true)
- void **SMR\_charge\_vol\_I** ([DdlElement](#) &ddls, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &force, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) & **SM\_charge\_hydro\_E** ([DdlElement](#) &ddls, int nSurf, const [Tableau](#)< [Vecteur](#) > &taphi, int nbne, const [Vecteur](#) &poids, const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool atdt=true)

- Element::ResRaid **SMR\_charge\_hydro\_I** (DdlElement &ddlA, int nSurf, const Tableau< Vecteur > &taphi, int nbne, const Vecteur &poids, const Coordonnee &dir\_normal\_liquide, const double &poidvol, const Coordonnee &M\_liquide, const ParaAlgoControle &pa)
- Vecteur &**SM\_charge\_hydrodyn\_E** (const double &poidvol, const Tableau< Vecteur > &taphi, int nbne, Courbe1D \*frot\_fluid, const Vecteur &poids, Courbe1D \*coef\_aero\_n, int numfront, const double &coef\_mul, Courbe1D \*coef\_aero\_t, const ParaAlgoControle &, bool atdt=true)
- Element::ResRaid **SM\_charge\_hydrodyn\_I** (const double &poidvol, const Tableau< Vecteur > &taphi, int nbne, Courbe1D \*frot\_fluid, const Vecteur &poids, DdlElement &ddls, Courbe1D \*coef\_aero\_n, int numfront, const double &coef\_mul, Courbe1D \*coef\_aero\_t, const ParaAlgoControle &pa)
- virtual EIFrontiere \* **new\_frontiere\_lin** (int num, Tableau< Noeud \* > &tab, DdlElement &ddelem)=0
- virtual EIFrontiere \* **new\_frontiere\_surf** (int num, Tableau< Noeud \* > &tab, DdlElement &ddelem)=0
- void **Init\_hourglass\_comp** (const ElemGeomC0 &elgeHour, const string &str\_precision, LoiAbstraiteGeneral \*loiHourglass, const BlocGen &bloc)
- void **Cal\_implicit\_hourglass** ()
- void **Cal\_explicit\_hourglass** (bool atdt)
- void **FluxAuNoeud\_ResRaid** (const int nbne, const Tableau< Vecteur > &taphi, const Vecteur &poids, Tableau< Vecteur \* > &resErr, Mat\_pleine &raidErr, const Tableau< Vecteur > &taphiEr, const Vecteur &poidsEr)
- void **Cal\_ErrElem** (int type, double &errElemRelative, double &numérateur, double &denominateur, const int nbne, const Tableau< Vecteur > &taphi, const Vecteur &poids, const Tableau< Vecteur > &taphiEr, const Vecteur &poidsEr)
- void **Cal\_ErrAuxNoeuds** (const int nbne, const Tableau< Vecteur > &taphi, const Vecteur &poids, Tableau< Vecteur \* > &resErr)
- void **VarDualSort** (ofstream &sort, Tableau< string > &nom, int nbint, int cas)
- void **AffDefCont** (ofstream &sort, CompThermoPhysiqueAbstraite::SaveResul \*saveDon, CoordonneeB &gradTB, Coordonnee &gradT, double &norme\_gradT, CoordonneeB &DgradTB, Coordonnee &DgradT, double &norme\_dGradT, CoordonneeH &fluxDH, Coordonnee &fluxD, double &norme\_flux, double &temperature, int indic)
- void **AffDefCont1D** (ofstream &sort, CompThermoPhysiqueAbstraite::SaveResul \*saveDon, CoordonneeB &gradTB, Coordonnee &gradT, double &norme\_gradT, CoordonneeB &DgradTB, Coordonnee &DgradT, double &norme\_dGradT, CoordonneeH &fluxDH, Coordonnee &fluxD, double &norme\_flux, double &temperature, int indic)
- void **AffDefCont2D** (ofstream &sort, CompThermoPhysiqueAbstraite::SaveResul \*saveDon, CoordonneeB &gradTB, Coordonnee &gradT, double &norme\_gradT, CoordonneeB &DgradTB, Coordonnee &DgradT, double &norme\_dGradT, CoordonneeH &fluxDH, Coordonnee &fluxD, double &norme\_flux, double &temperature, int indic)
- void **AffDefCont3D** (ofstream &sort, CompThermoPhysiqueAbstraite::SaveResul \*saveDon, CoordonneeB &gradTB, Coordonnee &gradT, double &norme\_gradT, CoordonneeB &DgradTB, Coordonnee &DgradT, double &norme\_dGradT, CoordonneeH &fluxDH, Coordonnee &fluxD, double &norme\_flux, double &temperature, int indic)
- void **Ad\_ddl\_Flux** (const DdlElement &tab\_ddlErr)
- void **Inact\_ddl\_primaire** (DdlElement &tab\_ddl)
- void **Act\_ddl\_primaire** (DdlElement &tab\_ddl)
- void **Inact\_ddl\_Flux** (DdlElement &tab\_ddlErr)
- void **Act\_ddl\_Flux** (DdlElement &tab\_ddlErr)
- void **Act\_premier\_ddl\_Flux** ()
- void **LectureDesFlux** (bool cas, UtilLecture \*entreePrinc, Tableau< CoordonneeH \* > &tabfluxH)
- void **FluxEnAbsolues** (bool cas, Tableau< CoordonneeH \* > &tabfluxH, Tableau< Vecteur > &tabflux)
- void **Lecture\_bas\_inf** (ifstream &ent, const Tableau< Noeud \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_bas\_inf** (ofstream &sort, const int cas)
- double **Interne\_Long\_arrete\_mini\_sur\_c** (Enum\_dure temps, int nb\_noeud=0)
- Coordonnee **CoordPtInt** (Enum\_dure temps, Enum\_ddl enu, int iteg, bool &erreur)
- Coordonnee **CoordPtIntFace** (int face, Enum\_dure temps, int iteg, bool &erreur)
- Coordonnee **CoordPtIntArete** (int arete, Enum\_dure temps, int iteg, bool &erreur)
- Element \* **Complete\_ElemThermi** (BlocGen &bloc, LesFonctions\_nD \*lesFonctionsnD)
- int **PtLePlusPres** (Enum\_dure temps, Enum\_ddl enu, const Coordonnee &M)
- Tableau< double > **Valeur\_multi** (bool absolue, Enum\_dure enu\_t, const List\_io< Ddl\_enum\_etendu > &enu, int iteg, int cas)
- void **Valeurs\_Tensorielles** (bool absolue, Enum\_dure enu\_t, List\_io< TypeQuelconque > &enu, int iteg, int cas)
- virtual ElemGeomC0 &**ElementGeometrie** (Enum\_ddl ddl) const
- virtual int **NbGrandeurGene** (Enum\_ddl enu) const

- virtual int **Dim\_flux\_gradT** () const =0
- bool **Bulk\_visco\_actif** ()
- double & **C\_traceBulk** ()
- double & **C\_carreBulk** ()
- void **TdtversT\_** ()
- void **TversTdt\_** ()

### Attributs protégés

- [LesPtIntegThermiInterne](#) \* **lesPtIntegThermiInterne**
- [Loi\\_comp\\_abstraite](#) \* **loiComp**
- [CompThermoPhysiqueAbstraite](#) \* **loiTP**
- [CompFrotAbstraite](#) \* **loiFrot**
- bool **dilatation**
- [Met\\_abstraite](#) \* **met**
- [Deformation](#) \* **def**
- [Deformation](#) \* **defEr**
- [Tableau](#)< [Deformation](#) \* > **defSurf**
- [Tableau](#)< [Deformation](#) \* > **defArete**
- [Deformation](#) \* **defMas**
- double \* **fluxErreur**
- [Tableau](#)< [Loi\\_comp\\_abstraite::SaveResul](#) \* > **tabSaveDon**
- [Tableau](#)< [CompThermoPhysiqueAbstraite::SaveResul](#) \* > **tabSaveTP**
- [Tableau](#)< [Deformation::SaveDefResul](#) \* > **tabSaveDefDon**
- [Tableau](#)< [EnergieThermi](#) > **tab\_energ**
- [Tableau](#)< [EnergieThermi](#) > **tab\_energ\_t**
- [EnergieThermi](#) **energie\_totale**
- bool **premier\_calcul\_thermi\_impli\_expli**
- double **masse\_volumique**
- [Enum\\_StabHourglass](#) **type\_stabHourglass**
- double **E\_Hourglass**

## 6.272.1 Documentation des fonctions membres

### 6.272.1.1 ErreurElement()

```
void ElemThermi::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en  
Réimplémentée dans [BielletteThermi](#).

### 6.272.1.2 Init\_hourglass\_comp()

```
void ElemThermi::Init_hourglass_comp (
    const ElemGeomC0 & elgeHour,
    const string & str_precision,
    LoiAbstraiteGeneral * loiHourglass,
    const BlocGen & bloc ) [protected]
```

!!! en fait pour l'instant on n'utilise pas elgehour, on considère que le nombre de pti complet est celui de l'élément sans info annexe !!!



### 6.272.1.3 Tableau\_de\_Flux1()

```
DdlElement & ElemThermi::Tableau_de_Flux1 ( ) const [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

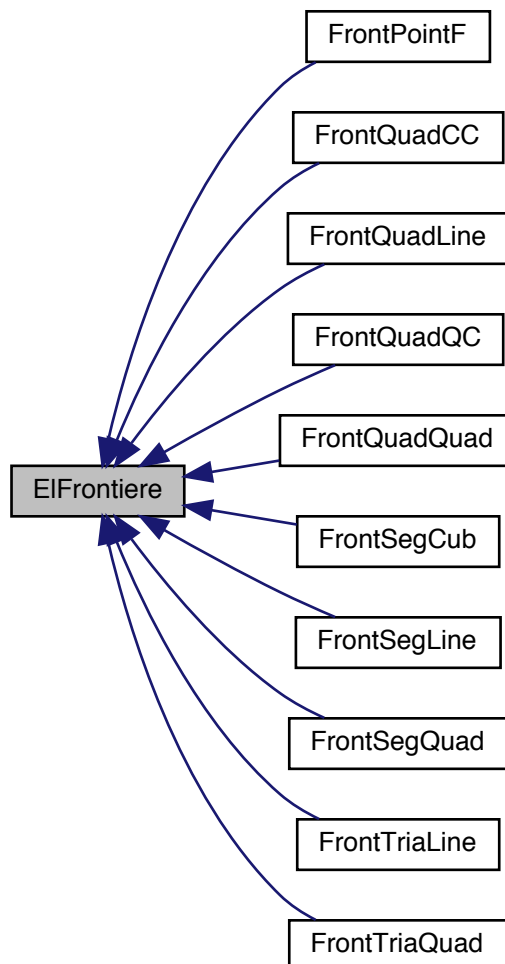
Réimplémentée dans [BielletteThermi](#).

La documentation de cette classe a été générée à partir du fichier suivant :

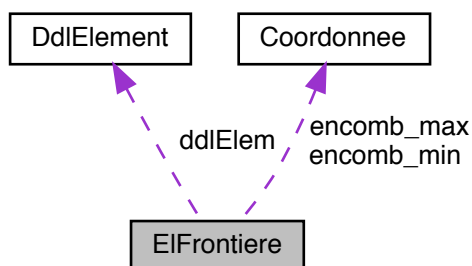
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermi.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermi.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermi2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermi3.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermi4.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermi5.cc

## 6.273 Référence de la classe EIFrontiere

Graphe d'héritage de EIFrontiere:



Graphe de collaboration de EIFrontiere:



## Fonctions membres publiques

- **EIFrontiere** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **EIFrontiere** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem, int nbnoeud)
- **EIFrontiere** (const [EIFrontiere](#) &a)
- virtual [EIFrontiere](#) & **operator=** (const [EIFrontiere](#) &a)
- virtual string **TypeFrontiere** () const =0
- [Enum\\_type\\_geom](#) **Type\_geom\_front** () const
- virtual [ElemGeomC0](#) const & **ElementGeometrique** () const =0
- virtual bool **operator==** (const [EIFrontiere](#) &a) const
- bool **operator!=** (const [EIFrontiere](#) &a) const
- const [Tableau](#)< [Noeud](#) \* > & **TabNoeud\_const** () const
- [Tableau](#)< [Noeud](#) \* > & **TabNoeud** ()
- [DdlElement](#) & **DdlElem** ()
- const [DdlElement](#) & **DdlElem\_const** () const
- [Tableau](#)< [EIFrontiere](#) \* > & **TabFront** ()
- const [Tableau](#)< [EIFrontiere](#) \* > & **TabFront\_const** () const
- virtual [EIFrontiere](#) \* **NevezElemFront** () const =0
- virtual [EIFrontiere](#) \* **NevezElemFront** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const =0
- virtual [EIFrontiere](#) \* **Oppose** () const
- virtual [Coordonnee](#) **Ref** ()=0
- virtual void **TangentRef** ([Droite](#) &dr, [Plan](#) &pl, int &indic)=0
- virtual void **Tangent** (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)=0
- virtual void **AutreTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic)=0
- virtual bool **InSurf** (const double &eps) const =0
- virtual [Tableau](#)< [Coordonnee](#) > \* **DernierTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)=0
- virtual void **Affiche** ([Enum\\_dure](#) temp=TEMPS\_tdt) const =0
- virtual bool **BonCote\_t** (const [Coordonnee](#) &a, double &r) const =0
- virtual bool **BonCote\_tdt** (const [Coordonnee](#) &a, double &r) const =0
- virtual const [Vecteur](#) & **Phi** ()=0
- virtual [Tableau](#)< [EIFrontiere](#) \* > & **Frontiere** ()=0
- virtual void **EffaceFrontiere** ()
- virtual [Met\\_abstraite](#) \* **Metrique** ()=0
- double **SurfaceApprox** ()
- virtual double **LongueurApprox** ()
- double **MaxDiagonale\_tdt** ()
- const [Coordonnee](#) & **Encom\_mini** ()
- const [Coordonnee](#) & **Encom\_maxi** ()
- void **AjourBoiteEncombement** ()
- bool **Projection\_normale** (const [Coordonnee](#) &M, [Coordonnee](#) &P)
- void **Lecture\_base\_info{EIFrontiere}** (ifstream &ent)
- void **Ecriture\_base\_info{EIFrontiere}** (ofstream &sort)
- virtual void **Lecture\_base\_info{EIFrontiere\_pour\_projection}** (ifstream &ent)=0
- virtual void **Ecriture\_base\_info{EIFrontiere\_pour\_projection}** (ofstream &sort)=0

## Fonctions membres protégées

- bool `In_boite_emcombement` (const [Coordonnee](#) &M) const

## Attributs protégés

- [Tableau](#)< [Noeud](#) \* > `tabNoeud`
- [DdlElement](#) `ddlElem`
- [Tableau](#)< [ElFrontiere](#) \* > `tabfront`
- [Coordonnee](#) `encomb_min`
- [Coordonnee](#) `encomb_max`

## Attributs protégés statiques

- static unsigned int `nrand` = 0

## 6.273.1 Documentation des fonctions membres

### 6.273.1.1 BonCote\_t()

```
virtual bool ElFrontiere::BonCote_t (
    const Coordonnee & a,
    double & r ) const [pure virtual]
```

Implémenté dans [FrontPointF](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/ElFrontiere.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/ElFrontiere.cc`

## 6.274 Référence de la classe EnergieMeca

### Fonctions membres publiques

- [EnergieMeca](#) (const double &e\_elastique, const double &d\_plastique, const double &d\_visqueuse)
- [EnergieMeca](#) (const [EnergieMeca](#) &a)
- void `Affiche` () const
- [EnergieMeca](#) `operator+` (const [EnergieMeca](#) &a) const
- [EnergieMeca](#) `operator-` (const [EnergieMeca](#) &a) const
- void `operator+=` (const [EnergieMeca](#) &a)
- void `operator-=` (const [EnergieMeca](#) &a)
- void `operator*=` (const double &x)
- void `operator/=` (const double &x)
- [EnergieMeca](#) `operator*` (const double &x) const
- [EnergieMeca](#) `operator/` (const double &x) const
- [EnergieMeca](#) & `operator=` (const [EnergieMeca](#) &a)
- void `Inita` (const double &valeur)
- void `Initialisation_differenciee` (const [EnergieMeca](#) &energlob)
- void `Ajout_differenciee` (const [EnergieMeca](#) &ener, const [EnergieMeca](#) &ener\_t, const double &coef)
- const double & `EnergieElastique` () const
- const double & `DissipationPlastique` () const
- const double & `DissipationVisqueuse` () const
- void `ChangeEnergieElastique` (double en)
- void `ChangeDissipationPlastique` (const double &en)
- void `ChangeDissipationVisqueuse` (const double &en)

### Amis

- `istream & operator>>` (istream &, [EnergieMeca](#) &)
- `ostream & operator<<` (ostream &, const [EnergieMeca](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Energies_meca/EnergieMeca.h`

## 6.275 Référence de la classe EnergieThermi

### Fonctions membres publiques

- **EnergieThermi** (const double &e\_elastique, const double &d\_plastique, const double &d\_visqueuse)
- **EnergieThermi** (const [EnergieThermi](#) &a)
- void **Affiche** () const
- [EnergieThermi](#) **operator+** (const [EnergieThermi](#) &a) const
- [EnergieThermi](#) **operator-** (const [EnergieThermi](#) &a) const
- void **operator+=** (const [EnergieThermi](#) &a)
- void **operator-=** (const [EnergieThermi](#) &a)
- void **operator\*=** (const double &x)
- void **operator/=** (const double &x)
- [EnergieThermi](#) **operator\*** (const double &x) const
- [EnergieThermi](#) **operator/** (const double &x) const
- [EnergieThermi](#) & **operator=** (const [EnergieThermi](#) &a)
- void **Inita** (const double &valeur)
- const double & **EnergieElastique** () const
- const double & **DissipationPlastique** () const
- const double & **DissipationVisqueuse** () const
- void **ChangeEnergieElastique** (double en)
- void **ChangeDissipationPlastique** (const double &en)
- void **ChangeDissipationVisqueuse** (const double &en)

### Amis

- istream & **operator>>** (istream &, [EnergieThermi](#) &)
- ostream & **operator<<** (ostream &, const [EnergieThermi](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Energies\_thermique/EnergieThermi.h

## 6.276 Référence de la classe Entier\_et\_Double

cas d' 1 entier et 1 double

```
#include <Basiques.h>
```

### Fonctions membres publiques

- **Entier\_et\_Double** (int u, double v)
- **Entier\_et\_Double** (const [Entier\\_et\\_Double](#) &de)
- [Entier\\_et\\_Double](#) & **operator=** (const [Entier\\_et\\_Double](#) &de)
- istream & **LectXML\_Entier\_et\_Double** (istream &ent)
- ostream & **EcritXML\_Entier\_et\_Double** (ostream &sort)
- bool **operator==** (const [Entier\\_et\\_Double](#) &a) const
- bool **operator!=** (const [Entier\\_et\\_Double](#) &a) const
- void **SchemaXML\_Entier\_et\_Double** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const
- bool **operator>** (const [Entier\\_et\\_Double](#) &a) const
- bool **operator>=** (const [Entier\\_et\\_Double](#) &a) const
- bool **operator<** (const [Entier\\_et\\_Double](#) &a) const
- bool **operator<=** (const [Entier\\_et\\_Double](#) &a) const

### Attributs publics

- double x
- int n

### Attributs publics statiques

- static short int **impre\_schem\_XML** =0

## Amis

- `istream & operator>>` (`istream &ent`, [Entier\\_et\\_Double](#) &de)
- `ostream & operator<<` (`ostream &sort`, const [Entier\\_et\\_Double](#) &de)

### 6.276.1 Description détaillée

cas d' 1 entier et 1 double

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.cc`

## 6.277 Référence de la classe Epai

Conteneur très basique pour les épaisseurs.

```
#include <Epai.h>
```

### Fonctions membres publiques

- **Epai** (const double ep0, const double ept, const double eptdt)
- **Epai** (const [Epai](#) &a)
- void **Change\_tout** (const double &val)
- [Epai](#) & **operator=** (const [Epai](#) &a)

### Attributs publics

- double **epaisseur0**
- double **epaisseur\_t**
- double **epaisseur\_tdt**

## Amis

- `istream & operator>>` (`istream &ent`, [Epai](#) &de)
- `ostream & operator<<` (`ostream &sort`, const [Epai](#) &de)

### 6.277.1 Description détaillée

Conteneur très basique pour les épaisseurs.

Auteur

Gérard Rio

Version

1.0

Date

08/07/2008

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Epai.h`

## 6.278 Référence de la classe Err\_inconnue\_ElemMeca

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ExceptionsElemMeca.h`

## 6.279 Référence de la classe ErrCalculFct\_nD

gestion d'exception pour des erreurs d'appel de fonction nD

```
#include <Fonction_nD.h>
```

### 6.279.1 Description détaillée

gestion d'exception pour des erreurs d'appel de fonction nD

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_nD (conflict on 2019-05-08).h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_nD.h

## 6.280 Référence de la classe ErrJacobienNegatif\_ElemMeca

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ExceptionsElemMeca.h

## 6.281 Référence de la classe ErrJacobienNegatif\_ElemThermi

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ExceptionsElemThermi.h

## 6.282 Référence de la classe ErrMathUtil2

cas d'une erreur survenue à cause d'une non convergence

```
#include <MathUtil2.h>
```

### Fonctions membres publiques

- **ErrMathUtil2** ()  
*constructeur par défaut (pas d'erreur)*
- **ErrMathUtil2** (int ca)  
*constructeur à partir d'un indice d'erreur ca donné*

### Attributs publics

- int **cas**  
*contient l'erreur*

### 6.282.1 Description détaillée

cas d'une erreur survenue à cause d'une non convergence

=0 cas courant, pas d'information particulière

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/MathUtil2.h

## 6.283 Référence de la classe ErrNonConvergence\_loiDeComportement

cas d'une erreur survenue à cause d'une non convergence pour la résolution d'une loi de comportement incrémentale

```
#include <ExceptionsLoiComp.h>
```

### Fonctions membres publiques

- **ErrNonConvergence\_loiDeComportement** (int ca)

## Attributs publics

- int **cas**

### 6.283.1 Description détaillée

cas d'une erreur survenue à cause d'une non convergence pour la résolution d'une loi de comportement incrémentale

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/ExceptionsLoiComp.h

## 6.284 Référence de la classe ErrNonConvergence\_Newton

pour la gestion d'exception pour non convergence

```
#include <Algo_zero.h>
```

## Fonctions membres publiques

- **ErrNonConvergence\_Newton** (int ca)

## Attributs publics

- int **cas**

### 6.284.1 Description détaillée

pour la gestion d'exception pour non convergence

BUT: pour la gestion d'exception pour non convergence

Auteur

Gérard Rio

Version

1.0

Date

11/10/2003

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Algo\_zero.h

## 6.285 Référence de la classe UtilLecture::ErrNouvelEnreg

## Fonctions membres publiques

- **ErrNouvelEnreg** (int entrees)

## Attributs publics

- int **lecture**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.h

## 6.286 Référence de la classe UtilLecture::ErrNouvelEnregCVisu

## Fonctions membres publiques

- **ErrNouvelEnregCVisu** (int entrees)

### Attributs publics

- `int lecture`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.h`

## 6.287 Référence de la classe `UtilLecture::ErrNouvelleDonnee`

### Fonctions membres publiques

- `ErrNouvelleDonnee` (`int entrees`)

### Attributs publics

- `int lecture`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.h`

## 6.288 Référence de la classe `UtilLecture::ErrNouvelleDonneeCVisu`

### Fonctions membres publiques

- `ErrNouvelleDonneeCVisu` (`int entrees`)

### Attributs publics

- `int lecture`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.h`

## 6.289 Référence de la classe `ErrResolve_system_lineaire`

### Fonctions membres publiques

- `ErrResolve_system_lineaire` (`int ca`)

### Attributs publics

- `int cas`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/ExceptionsMatrices.h`

## 6.290 Référence de la classe `ErrSortie`

gestion d'exception pour `Sortie`

```
#include <Sortie.h>
```

### 6.290.1 Description détaillée

gestion d'exception pour `Sortie`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Sortie.h`



## 6.291 Référence de la classe ErrSortieFinale

gestion d'exception pour Sortie finale

```
#include <Sortie.h>
```

### 6.291.1 Description détaillée

gestion d'exception pour Sortie finale

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Sortie.h

## 6.292 Référence de la classe ErrVarJacobienMini\_ElemMeca

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ExceptionsElemMeca.h

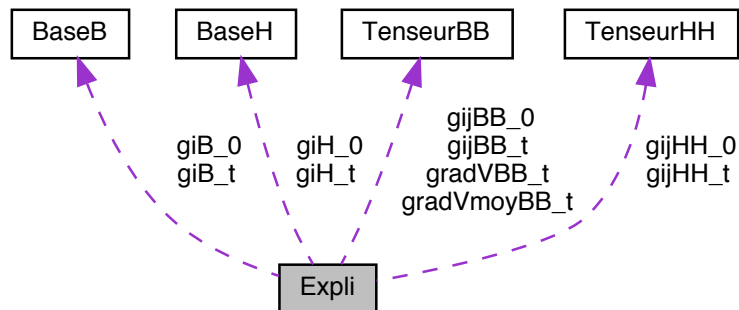
## 6.293 Référence de la classe ErrVarJacobienMini\_ElemThermi

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ExceptionsElemThermi.h

## 6.294 Référence de la classe Expli

Graphe de collaboration de Expli:



### Fonctions membres publiques

- **Expli** (BaseB \*ggiB\_0, BaseH \*ggiH\_0, BaseB \*ggiB\_t, BaseH \*ggiH\_t, TenseurBB \*ggijBB\_0, TenseurHH \*ggijHH\_0, TenseurBB \*ggijBB\_t, TenseurHH \*ggijHH\_t, TenseurBB \*ggradVmoyBB\_t, TenseurBB \*ggradVBB\_t, Tableau< TenseurBB \* > \*gd\_gijBB\_t, double \*gjacobien\_t, double \*gjacobien\_0)
- **Expli** (const Expli &ex)
- **Expli & operator=** (const Expli &ex)
- **Expli\_t tdt T dans\_tdt** () const
- void **Mise\_a\_jour\_grandeur** (BaseB \*ggiB\_0, BaseH \*ggiH\_0, BaseB \*ggiB\_t, BaseH \*ggiH\_t, TenseurBB \*ggijBB\_0, TenseurHH \*ggijHH\_0, TenseurBB \*ggijBB\_t, TenseurHH \*ggijHH\_t, TenseurBB \*ggradVmoyBB\_t, TenseurBB \*ggradVBB\_t, Tableau< TenseurBB \* > \*gd\_gijBB\_t, double \*gjacobien\_t, double \*gjacobien\_0)
- void **Recup\_grandeur\_0** (BaseB &ggiB\_0, BaseH &ggiH\_0, TenseurBB &ggijBB\_0, TenseurHH &ggijHH\_0, double &gjacobien\_0)

## Attributs publics

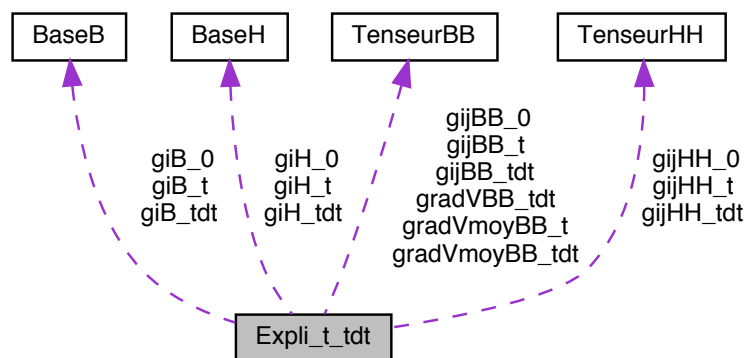
- BaseB \* **giB\_0**
- BaseH \* **giH\_0**
- BaseB \* **giB\_t**
- BaseH \* **giH\_t**
- TenseurBB \* **gijBB\_0**
- TenseurHH \* **gijHH\_0**
- TenseurBB \* **gijBB\_t**
- TenseurHH \* **gijHH\_t**
- TenseurBB \* **gradVmoyBB\_t**
- TenseurBB \* **gradVBB\_t**
- Tableau< TenseurBB \* > \* **d\_gijBB\_t**
- double \* **jacobien\_t**
- double \* **jacobien\_0**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔  
abstraite\_struc\_donnees.h

## 6.295 Référence de la classe Expli\_t\_tdt

Grappe de collaboration de Expli\_t\_tdt:



## Fonctions membres publiques

- **Expli\_t\_tdt** (BaseB \*ggiB\_0, BaseH \*ggiH\_0, BaseB \*ggiB\_t, BaseH \*ggiH\_t, BaseB \*ggiB\_tdt, BaseH \*ggiH\_tdt, TenseurBB \*ggijBB\_0, TenseurHH \*ggijHH\_0, TenseurBB \*ggijBB\_t, TenseurHH \*ggijHH\_↔  
t, TenseurBB \*ggijBB\_tdt, TenseurHH \*ggijHH\_tdt, TenseurBB \*ggradVmoyBB\_t, TenseurBB \*ggradVmoy↔  
BB\_tdt, TenseurBB \*ggradVBB\_tdt, Tableau< TenseurBB \* > \*gd\_gijBB\_tdt, double \*gjacobien\_tdt, double \*gjacobien\_t, double \*gjacobien\_0)
- **Expli\_t\_tdt** (const Expli\_t\_tdt &ex)
- **Expli\_t\_tdt & operator=** (const Expli\_t\_tdt &ex)
- **Expli\_Tdt\_dans\_t** () const
- void **Mise\_a\_jour\_grandeur** (BaseB \*ggiB\_0, BaseH \*ggiH\_0, BaseB \*ggiB\_t, BaseH \*ggiH\_t, BaseB \*ggiB\_tdt, BaseH \*ggiH\_tdt, TenseurBB \*ggijBB\_0, TenseurHH \*ggijHH\_0, TenseurBB \*ggijBB\_↔  
t, TenseurHH \*ggijHH\_t, TenseurBB \*ggijBB\_tdt, TenseurHH \*ggijHH\_tdt, TenseurBB \*ggradVmoyBB\_t, TenseurBB \*ggradVmoyBB\_tdt, TenseurBB \*ggradVBB\_tdt, Tableau< TenseurBB \* > \*gd\_gijBB\_tdt, double \*gjacobien\_tdt, double \*gjacobien\_t, double \*gjacobien\_0)
- void **Recup\_grandeur\_0\_t** (BaseB &ggiB\_0, BaseH &ggiH\_0, BaseB &ggiB\_t, BaseH &ggiH\_t, TenseurBB &ggijBB\_0, TenseurHH &ggijHH\_0, TenseurBB &ggijBB\_t, TenseurHH &ggijHH\_t, TenseurBB &, double &gjacobien\_t, double &gjacobien\_0)

- void **Passage\_de\_Ordre2\_vers\_Ordre3** (const [Expli\\_t\\_tdt](#) &ex, bool plusZero, int type\_recopie)
- void **Passage\_de\_Ordre3\_vers\_Ordre2** (const [Expli\\_t\\_tdt](#) &ex, int type\_recopie)
- void **Passage\_de\_Ordre1\_vers\_Ordre3** (const [Expli\\_t\\_tdt](#) &ex, bool plusZero, int type\_recopie)
- void **Passage\_de\_Ordre3\_vers\_Ordre1** (const [Expli\\_t\\_tdt](#) &ex, int type\_recopie)
- [Umat\\_cont](#) & **Recup\_Umat\_cont** ([Umat\\_cont](#) &ex) const

### Attributs publics

- [BaseB](#) \* **giB\_0**
- [BaseH](#) \* **giH\_0**
- [BaseB](#) \* **giB\_t**
- [BaseH](#) \* **giH\_t**
- [BaseB](#) \* **giB\_tdt**
- [BaseH](#) \* **giH\_tdt**
- [TenseurBB](#) \* **gijBB\_0**
- [TenseurHH](#) \* **gijHH\_0**
- [TenseurBB](#) \* **gijBB\_t**
- [TenseurHH](#) \* **gijHH\_t**
- [TenseurBB](#) \* **gijBB\_tdt**
- [TenseurHH](#) \* **gijHH\_tdt**
- [TenseurBB](#) \* **gradVmoyBB\_t**
- [TenseurBB](#) \* **gradVmoyBB\_tdt**
- [TenseurBB](#) \* **gradVBB\_tdt**
- [Tableau](#)< [TenseurBB](#) \* > \* **d\_gijBB\_tdt**
- double \* **jacobien\_tdt**
- double \* **jacobien\_t**
- double \* **jacobien\_0**

La documentation de cette classe a été générée à partir du fichier suivant :

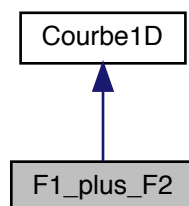
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔  
abstraite\_struc\_donnees.h

## 6.296 Référence de la classe F1\_plus\_F2

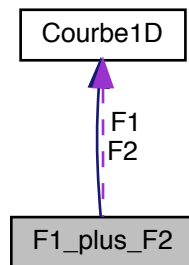
Classe permettant le calcul d'une fonction 1D de type :  $F(x) = (F1 + F2)(x) = F1(x) + F2(x)$

```
#include <F1_plus_F2.h>
```

Graphe d'héritage de F1\_plus\_F2:



Graphe de collaboration de F1\_plus\_F2:



## Fonctions membres publiques

- **F1\_plus\_F2** (string nom="")
- **F1\_plus\_F2** (Courbe1D \*FF1, Courbe1D \*FF2, string nom="")
- **F1\_plus\_F2** (const F1\_plus\_F2 &Co)
- **F1\_plus\_F2** (const Courbe1D &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Complet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, UtilLecture \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **DefCourbesMembres** (Courbe1D \*FF1, Courbe1D \*FF2)
- bool **DependAutreCourbes** () const  
*établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non*
- list< string > & **ListDependanceCourbes** (list< string > &lico) const  
*par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this*
- void **Lien\_entre\_courbe** (list< Courbe1D \* > &liptco)  
*3) établit la connection entre la demande de \*this et les courbes passées en paramètres*
- void **Info\_commande\_Courbes1D** (UtilLecture &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- Courbe1D::ValDer **Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*
- Courbe1D::ValDer2 **Valeur\_Et\_der12** (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double **Der\_sec** (double x)  
*ramène la dérivée seconde*
- Courbe1D::Valbool **Valeur\_stricte** (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- Courbe1D::ValDerbool **Valeur\_Et\_derivee\_stricte** (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void **Lecture\_base\_info** (ifstream &ent, const int cas)

- *lecture écriture de restart* — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
sortie du schemaXML: en fonction de enu

### Attributs protégés

- [Courbe1D](#) \* F1
- [Courbe1D](#) \* F2
- double **ponder1**
- double **ponder2**
- string **nom\_courbe1**
- string **nom\_courbe2**

### Membres hérités additionnels

#### 6.296.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $F(x) = (F1 + F2)(x) = F1(x) + F2(x)$

BUT: Classe permettant le calcul d'une fonction 1D somme de type :  $F(x) = (F1 + F2)(x) = F1(x) + F2(x)$  ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

##### Auteur

Gérard Rio

##### Version

1.0

##### Date

19/01/2001

#### 6.296.2 Documentation des fonctions membres

##### 6.296.2.1 Affiche()

```
void F1_plus_F2::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

##### 6.296.2.2 Complet\_courbe()

```
bool F1_plus_F2::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

##### 6.296.2.3 DependAutreCourbes()

```
bool F1_plus_F2::DependAutreCourbes ( ) const [virtual]
```

établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non  
Réimplémentée à partir de [Courbe1D](#).

**6.296.2.4 Der\_sec()**

```
double F1_plus_F2::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

**6.296.2.5 Derivee()**

```
double F1_plus_F2::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

**6.296.2.6 Ecriture\_base\_info()**

```
void F1_plus_F2::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

**6.296.2.7 Info\_commande\_Courbes1D()**

```
void F1_plus_F2::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

**6.296.2.8 LectDonnParticulieres\_courbes()**

```
void F1_plus_F2::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.  
Implémente [Courbe1D](#).

**6.296.2.9 Lecture\_base\_info()**

```
void F1_plus_F2::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

**6.296.2.10 Lien\_entre\_courbe()**

```
void F1_plus_F2::Lien_entre_courbe (
    list< Courbe1D * > & ) [virtual]
```

3) établit la connection entre la demande de \*this et les courbes passées en paramètres  
Réimplémentée à partir de [Courbe1D](#).

### 6.296.2.11 ListDependanceCourbes()

```
list< string > & F1_plus_F2::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this  
Réimplémentée à partir de [Courbe1D](#).

### 6.296.2.12 SchemaXML\_Courbes1D()

```
void F1_plus_F2::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu  
Implémente [Courbe1D](#).

### 6.296.2.13 Valeur()

```
double F1_plus_F2::Valeur (
    double x ) [virtual]
```

ramène la valeur  
Implémente [Courbe1D](#).

### 6.296.2.14 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 F1_plus_F2::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre  
Implémente [Courbe1D](#).

### 6.296.2.15 Valeur\_Et\_derivee()

```
Courbe1D::ValDer F1_plus_F2::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre  
Implémente [Courbe1D](#).

### 6.296.2.16 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool F1_plus_F2::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant  
Implémente [Courbe1D](#).

### 6.296.2.17 Valeur\_stricte()

```
Courbe1D::Valbool F1_plus_F2::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

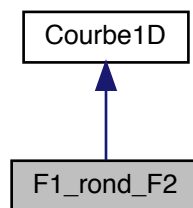
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F1\_plus\_F2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F1\_plus\_F2.cc

## 6.297 Référence de la classe F1\_rond\_F2

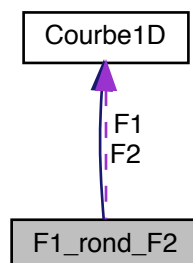
Classe permettant le calcul d'une fonction 1D composé :  $F(x) = F1(F2(x)) = F1 \circ F2(x)$

```
#include <F1_rond_F2.h>
```

Grphe d'héritage de F1\_rond\_F2:



Grphe de collaboration de F1\_rond\_F2:



### Fonctions membres publiques

- **F1\_rond\_F2** (string nom="")
- **F1\_rond\_F2** ([Courbe1D](#) \*FF1, [Courbe1D](#) \*FF2, string nom="")
- **F1\_rond\_F2** (const [F1\\_rond\\_F2](#) &Co)
- **F1\_rond\_F2** (const [Courbe1D](#) &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Compleet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*



- void **DefCourbesMembres** (Courbe1D \*FF1, Courbe1D \*FF2)
- bool **DependAutreCourbes** () const  
*établir le lien entre la courbe et des courbes déjà existantes dont on connaît que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non*
- list< string > & **ListDependanceCourbes** (list< string > &lico) const  
*par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this*
- void **Lien\_entre\_courbe** (list< Courbe1D \* > &liptco)  
*3) établit la connection entre la demande de \*this et les courbes passées en paramètres*
- void **Info\_commande\_Courbes1D** (UtilLecture &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- Courbe1D::ValDer **Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*
- Courbe1D::ValDer2 **Valeur\_Et\_der12** (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double **Der\_sec** (double x)  
*ramène la dérivée seconde*
- Courbe1D::Valbool **Valeur\_stricte** (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- Courbe1D::ValDerbool **Valeur\_Et\_derivee\_stricte** (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void **Lecture\_base\_info** (ifstream &ent, const int cas)  
*--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- void **SchemaXML\_Courbes1D** (ofstream &sort, const Enum\_IO\_XML enu)  
*sortie du schemaXML: en fonction de enu*

### Attributs protégés

- Courbe1D \* **F1**
- Courbe1D \* **F2**
- string **nom\_courbe1**
- string **nom\_courbe2**

### Membres hérités additionnels

#### 6.297.1 Description détaillée

Classe permettant le calcul d'une fonction 1D composé :  $F(x) = F1(F2(x)) = F1 \circ F2(x)$

BUT: Classe permettant le calcul d'une fonction composé 1D:  $F(x) = F1(F2(x)) = F1 \circ F2(x)$  ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

## 6.297.2 Documentation des fonctions membres

### 6.297.2.1 Affiche()

```
void F1_rond_F2::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

### 6.297.2.2 Complet\_courbe()

```
bool F1_rond_F2::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

### 6.297.2.3 DependAutreCourbes()

```
bool F1_rond_F2::DependAutreCourbes ( ) const [virtual]
```

établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non  
Réimplémentée à partir de [Courbe1D](#).

### 6.297.2.4 Der\_sec()

```
double F1_rond_F2::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

### 6.297.2.5 Derivee()

```
double F1_rond_F2::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

### 6.297.2.6 Ecriture\_base\_info()

```
void F1_rond_F2::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.297.2.7 Info\_commande\_Courbes1D()

```
void F1_rond_F2::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

**6.297.2.8 LectDonnParticulieres\_courbes()**

```
void F1_rond_F2::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

**6.297.2.9 Lecture\_base\_info()**

```
void F1_rond_F2::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.297.2.10 Lien\_entre\_courbe()**

```
void F1_rond_F2::Lien_entre_courbe (
    list< Courbe1D * > & ) [virtual]
```

3) établit la connection entre la demande de \*this et les courbes passées en paramètres

Réimplémentée à partir de [Courbe1D](#).

**6.297.2.11 ListDependanceCourbes()**

```
list< string > & F1_rond_F2::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this

Réimplémentée à partir de [Courbe1D](#).

**6.297.2.12 SchemaXML\_Courbes1D()**

```
void F1_rond_F2::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

**6.297.2.13 Valeur()**

```
double F1_rond_F2::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

**6.297.2.14 Valeur\_Et\_der12()**

```
Courbe1D::ValDer2 F1_rond_F2::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

**6.297.2.15 Valeur\_Et\_derivee()**

```
Courbe1D::ValDer F1_rond_F2::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

**6.297.2.16 Valeur\_Et\_derivee\_stricte()**

```
Courbe1D::ValDerbool F1_rond_F2::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

**6.297.2.17 Valeur\_stricte()**

```
Courbe1D::Valbool F1_rond_F2::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

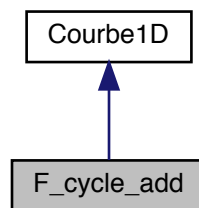
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F1\_rond\_F2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F1\_rond\_F2.cc

**6.298 Référence de la classe F\_cycle\_add**

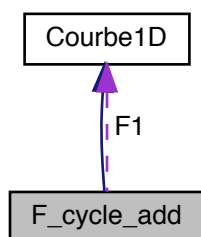
Classe permettant le calcul d'une fonction cyclique avec une amplification additive à chaque cycle.

```
#include <F_cycle_add.h>
```

Grphe d'héritage de F\_cycle\_add:



Graphe de collaboration de F\_cycle\_add:



## Fonctions membres publiques

- **F\_cycle\_add** (string nom="")
- **F\_cycle\_add** (Courbe1D \*FF1, string nom="")
- **F\_cycle\_add** (const F\_cycle\_add &Co)
- **F\_cycle\_add** (const Courbe1D &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Complet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, UtilLecture \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **DefCourbesMembres** (Courbe1D \*FF1)
- bool **DependAutreCourbes** () const  
*établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non*
- list< string > & **ListDependanceCourbes** (list< string > &lico) const  
*par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this*
- void **Lien\_entre\_courbe** (list< Courbe1D \* > &liptco)  
*3) établit la connection entre la demande de \*this et les courbes passées en paramètres*
- void **Info\_commande\_Courbes1D** (UtilLecture &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- Courbe1D::ValDer **Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*
- Courbe1D::ValDer2 **Valeur\_Et\_der12** (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double **Der\_sec** (double x)  
*ramène la dérivée seconde*
- Courbe1D::Valbool **Valeur\_stricte** (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- Courbe1D::ValDerbool **Valeur\_Et\_derivee\_stricte** (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void **Lecture\_base\_info** (ifstream &ent, const int cas)

- *lecture écriture de restart* — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
sortie du schemaXML: en fonction de enu

## Attributs protégés

- [Courbe1D](#) \* F1
- string **nom\_courbe1**
- double **ampli**
- double **longcycl**
- double **decalx**
- double **decaly**

## Membres hérités additionnels

### 6.298.1 Description détaillée

Classe permettant le calcul d'une fonction cyclique avec une amplification additive à chaque cycle.

BUT: Classe permettant le calcul d'une fonction cyclique c'est-à-dire qui tous les delta x donnée, retrouve la même forme. De plus, on ajoute un facteur d'amplification "ADDITIF" en fonction du nombre de cycles. Enfin, à chaque début de cycle, la fonction est translatée de la valeur qu'elle avait à la fin du cycle précédent.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.298.2 Documentation des fonctions membres

#### 6.298.2.1 Affiche()

```
void F_cycle_add::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

#### 6.298.2.2 Complet\_courbe()

```
bool F_cycle_add::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

#### 6.298.2.3 DependAutreCourbes()

```
bool F_cycle_add::DependAutreCourbes ( ) const [virtual]
```

établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non  
Réimplémentée à partir de [Courbe1D](#).

#### 6.298.2.4 Der\_sec()

```
double F_cycle_add::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

#### 6.298.2.5 Derivee()

```
double F_cycle_add::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

#### 6.298.2.6 Ecriture\_base\_info()

```
void F_cycle_add::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

#### 6.298.2.7 Info\_commande\_Courbes1D()

```
void F_cycle_add::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

#### 6.298.2.8 LectDonnParticulieres\_courbes()

```
void F_cycle_add::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

#### 6.298.2.9 Lecture\_base\_info()

```
void F_cycle_add::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

#### 6.298.2.10 Lien\_entre\_courbe()

```
void F_cycle_add::Lien_entre_courbe (
    list< Courbe1D * > & ) [virtual]
```

3) établit la connection entre la demande de \*this et les courbes passées en paramètres

Réimplémentée à partir de [Courbe1D](#).

**6.298.2.11 ListDependanceCourbes()**

```
list< string > & F_cycle_add::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this  
Réimplémentée à partir de [Courbe1D](#).

**6.298.2.12 SchemaXML\_Courbes1D()**

```
void F_cycle_add::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

**6.298.2.13 Valeur()**

```
double F_cycle_add::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

**6.298.2.14 Valeur\_Et\_der12()**

```
CourbelD::ValDer2 F_cycle_add::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

**6.298.2.15 Valeur\_Et\_derivee()**

```
CourbelD::ValDer F_cycle_add::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

**6.298.2.16 Valeur\_Et\_derivee\_stricte()**

```
CourbelD::ValDerbool F_cycle_add::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

**6.298.2.17 Valeur\_stricte()**

```
CourbelD::Valbool F_cycle_add::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y



Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

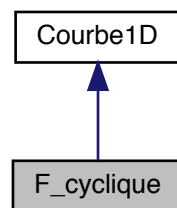
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_cycle\_add.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_cycle\_add.cc

## 6.299 Référence de la classe F\_cyclique

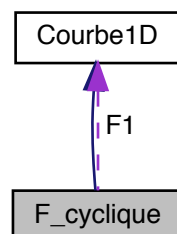
Classe permettant le calcul d'une fonction cyclique avec une amplification multiplicative à chaque cycle.

```
#include <F_cyclique.h>
```

Graphe d'héritage de F\_cyclique:



Graphe de collaboration de F\_cyclique:



### Fonctions membres publiques

- **F\_cyclique** (string nom="")
- **F\_cyclique** (Courbe1D \*FF1, string nom="")
- **F\_cyclique** (const F\_cyclique &Co)
- **F\_cyclique** (const Courbe1D &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Complet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, UtilLecture \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **DefCourbesMembres** (Courbe1D \*FF1)

- bool [DependAutreCourbes](#) () const  
*établir le lien entre la courbe et des courbes déjà existantes dont on connaît que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non*
- list< string > & [ListDependanceCourbes](#) (list< string > &lico) const  
*par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this*
- void [Lien\\_entre\\_courbe](#) (list< [Courbe1D](#) \* > &liptco)  
*3) établit la connection entre la demande de \*this et les courbes passées en paramètres*
- void [Info\\_commande\\_Courbes1D](#) ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double [Valeur](#) (double x)  
*ramène la valeur*
- [Courbe1D::ValDer](#) [Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2](#) [Valeur\\_Et\\_der12](#) (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double [Der\\_sec](#) (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool](#) [Valeur\\_stricte](#) (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- [Courbe1D::ValDerbool](#) [Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)  
*--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*

## Attributs protégés

- [Courbe1D](#) \* **F1**
- string **nom\_courbe1**
- double **ampli**
- double **longcycl**
- double **decalx**
- double **decaly**

## Membres hérités additionnels

### 6.299.1 Description détaillée

Classe permettant le calcul d'une fonction cyclique avec une amplification multiplicative à chaque cycle.

BUT: Classe permettant le calcul d'une fonction cyclique c'est-à-dire qui tous les delta x donnée, retrouve la même forme. De plus, on ajoute un facteur d'amplification "MULTIPLICATIF" en fonction du nombre de cycles. Enfin, à chaque début de cycle, la fonction est translatée de la valeur qu'elle avait à la fin du cycle précédent.

#### Auteur

Gérard Rio

**Version**

1.0

**Date**

19/01/2001

**6.299.2 Documentation des fonctions membres****6.299.2.1 Affiche()**

```
void F_cyclique::Affiche ( ) const [virtual]
```

affichage de la courbe

Implémente [Courbe1D](#).**6.299.2.2 Complet\_courbe()**

```
bool F_cyclique::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon

Implémente [Courbe1D](#).**6.299.2.3 DependAutreCourbes()**

```
bool F_cyclique::DependAutreCourbes ( ) const [virtual]
```

établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non

Réimplémentée à partir de [Courbe1D](#).**6.299.2.4 Der\_sec()**

```
double F_cyclique::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).**6.299.2.5 Derivee()**

```
double F_cyclique::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).**6.299.2.6 Ecriture\_base\_info()**

```
void F_cyclique::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.299.2.7 Info\_commande\_Courbes1D()**

```
void F_cyclique::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

**6.299.2.8 LectDonnParticulieres\_courbes()**

```
void F_cyclique::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

**6.299.2.9 Lecture\_base\_info()**

```
void F_cyclique::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.299.2.10 Lien\_entre\_courbe()**

```
void F_cyclique::Lien_entre_courbe (
    list< Courbe1D * > & ) [virtual]
```

3) établit la connection entre la demande de \*this et les courbes passées en paramètres

Réimplémentée à partir de [Courbe1D](#).

**6.299.2.11 ListDependanceCourbes()**

```
list< string > & F_cyclique::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this

Réimplémentée à partir de [Courbe1D](#).

**6.299.2.12 SchemaXML\_Courbes1D()**

```
void F_cyclique::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

**6.299.2.13 Valeur()**

```
double F_cyclique::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

**6.299.2.14 Valeur\_Et\_der12()**

```
CourbelD::ValDer2 F_cyclique::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre  
Implémente [Courbe1D](#).

**6.299.2.15 Valeur\_Et\_derivee()**

```
CourbelD::ValDer F_cyclique::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre  
Implémente [Courbe1D](#).

**6.299.2.16 Valeur\_Et\_derivee\_stricte()**

```
CourbelD::ValDerbool F_cyclique::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant  
Implémente [Courbe1D](#).

**6.299.2.17 Valeur\_stricte()**

```
CourbelD::Valbool F_cyclique::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y  
Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

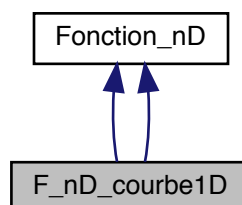
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_cyclique.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_cyclique.cc

**6.300 Référence de la classe F\_nD\_courbe1D**

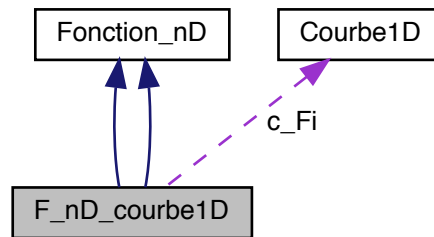
Classe permettant d'utiliser une fonction courbe à l'intérieur d'une fonction nD .

```
#include <F_nD_courbe1D.h>
```

Graphe d'héritage de F\_nD\_courbe1D:



Graphe de collaboration de F\_nD\_courbe1D:



## Fonctions membres publiques

- **F\_nD\_courbe1D** (string nom="")
- **F\_nD\_courbe1D** (const **F\_nD\_courbe1D** &Co)
- **F\_nD\_courbe1D** (const **Fonction\_nD** &Coo)
- **Fonction\_nD** & **operator=** (const **Fonction\_nD** &elt)
- void **Affiche** (int niveau=0) const
- bool **Complet\_Fonction** () const
- void **LectDonnParticulieres\_Fonction\_nD** (const string &nom, **UtilLecture** \*)
- virtual void **Mise\_a\_jour\_variables\_globales** ()
- void **Info\_commande\_Fonctions\_nD** (**UtilLecture** &entreePrinc)
- virtual **Tableau**< double > & **Valeur\_FnD** (**Tableau**< **Ddl\_etendu** > \*t\_enu, **Tableau**< **TypeQuelconque** > \*tqi)
- virtual **Tableau**< double > & **Valeur\_pour\_variables\_globales** ()
- int **NbVariable** () const
- int **NbComposante** () const
- bool **DependAutreFoncCourbes** () const
- list< string > & **ListDependanceCourbes** (list< string > &lico) const
- void **Lien\_entre\_fonc\_courbe** (list< **Fonction\_nD** \* > &liptfunc, list< **Courbe1D** \* > &liptco)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **SchemaXML\_Fonctions\_nD** (ofstream &sort, const **Enum\_IO\_XML** enu)
- **F\_nD\_courbe1D** (string nom="")
- **F\_nD\_courbe1D** (const **F\_nD\_courbe1D** &Co)
- **F\_nD\_courbe1D** (const **Fonction\_nD** &Coo)
- **Fonction\_nD** & **operator=** (const **Fonction\_nD** &elt)
- void **Affiche** (int niveau=0) const
- bool **Complet\_Fonction** (bool affichage=true) const
- void **LectDonnParticulieres\_Fonction\_nD** (const string &nom, **UtilLecture** \*)
- virtual void **Mise\_a\_jour\_variables\_globales** ()
- void **Info\_commande\_Fonctions\_nD** (**UtilLecture** &entreePrinc)
- int **NbComposante** () const
- bool **DependAutreFoncCourbes** () const
- list< string > & **ListDependanceCourbes** (list< string > &lico) const
- void **Lien\_entre\_fonc\_courbe** (list< **Fonction\_nD** \* > &liptfunc, list< **Courbe1D** \* > &liptco)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **SchemaXML\_Fonctions\_nD** (ofstream &sort, const **Enum\_IO\_XML** enu)

## Fonctions membres protégées

- void **DefFoncCourbeMembre** (**Courbe1D** \*cc\_Fi)
- virtual **Tableau**< double > & **Valeur\_FnD\_interne** (**Tableau**< double > \*val\_ddl\_enum, **Tableau**< **Coordonnee** > \*coor\_ddl\_enum, **Tableau**< **TenseurBB** \* > \*tens\_ddl\_enum, bool variables\_locales)
- void **DefFoncCourbeMembre** (**Courbe1D** \*cc\_Fi)
- virtual **Tableau**< double > & **Valeur\_FnD\_interne** (**Tableau**< double > \*val\_ddl\_enum)
- virtual **Tableau**< double > & **Valeur\_pour\_variables\_globales\_interne** ()

## Attributs protégés

- [Courbe1D](#) \* **c\_Fi**
- string **nom\_courbe1**
- [Tableau](#)< double > **tab\_ret**

## Membres hérités additionnels

### 6.300.1 Description détaillée

Classe permettant d'utiliser une fonction courbe à l'intérieur d'une fonction nD .

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

01/06/2016

### 6.300.2 Documentation des fonctions membres

#### 6.300.2.1 Affiche() [1/2]

```
void F_nD_courbe1D::Affiche (
    int niveau = 0 ) const [virtual]
```

Implémente [Fonction\\_nD](#).

#### 6.300.2.2 Affiche() [2/2]

```
void F_nD_courbe1D::Affiche (
    int niveau = 0 ) const [virtual]
```

Implémente [Fonction\\_nD](#).

#### 6.300.2.3 Complet\_Fonction() [1/2]

```
bool F_nD_courbe1D::Complet_Fonction ( ) const [virtual]
```

Implémente [Fonction\\_nD](#).

#### 6.300.2.4 Complet\_Fonction() [2/2]

```
bool F_nD_courbe1D::Complet_Fonction (
    bool affichage = true ) const [virtual]
```

Implémente [Fonction\\_nD](#).

#### 6.300.2.5 DependAutreFoncCourbes() [1/2]

```
bool F_nD_courbe1D::DependAutreFoncCourbes ( ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

### 6.300.2.6 DependAutreFoncCourbes() [2/2]

```
bool F_nD_courbelD::DependAutreFoncCourbes ( ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

### 6.300.2.7 Ecriture\_base\_info() [1/2]

```
void F_nD_courbelD::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.300.2.8 Ecriture\_base\_info() [2/2]

```
void F_nD_courbelD::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.300.2.9 Info\_commande\_Fonctions\_nD() [1/2]

```
void F_nD_courbelD::Info_commande_Fonctions_nD (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.300.2.10 Info\_commande\_Fonctions\_nD() [2/2]

```
void F_nD_courbelD::Info_commande_Fonctions_nD (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.300.2.11 LectDonnParticulieres\_Fonction\_nD() [1/2]

```
void F_nD_courbelD::LectDonnParticulieres_Fonction_nD (
    const string & nom,
    UtilLecture * entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.300.2.12 LectDonnParticulieres\_Fonction\_nD() [2/2]

```
void F_nD_courbelD::LectDonnParticulieres_Fonction_nD (
    const string & nom,
    UtilLecture * ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.300.2.13 Lecture\_base\_info() [1/2]

```
void F_nD_courbelD::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).



**6.300.2.14 Lecture\_base\_info()** [2/2]

```
void F_nD_courbe1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.300.2.15 Lien\_entre\_fonc\_courbe()** [1/2]

```
void F_nD_courbe1D::Lien_entre_fonc_courbe (
    list< Fonction_nD * > & liptfunc,
    list< Courbe1D * > & liptco ) [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.300.2.16 Lien\_entre\_fonc\_courbe()** [2/2]

```
void F_nD_courbe1D::Lien_entre_fonc_courbe (
    list< Fonction_nD * > & liptfunc,
    list< Courbe1D * > & liptco ) [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.300.2.17 ListDependanceCourbes()** [1/2]

```
list< string > & F_nD_courbe1D::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.300.2.18 ListDependanceCourbes()** [2/2]

```
list< string > & F_nD_courbe1D::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.300.2.19 Mise\_a\_jour\_variables\_globales()** [1/2]

```
void F_nD_courbe1D::Mise_a_jour_variables_globales ( ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.300.2.20 Mise\_a\_jour\_variables\_globales()** [2/2]

```
virtual void F_nD_courbe1D::Mise_a_jour_variables_globales ( ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.300.2.21 NbComposante()** [1/2]

```
int F_nD_courbe1D::NbComposante ( ) const [inline], [virtual]
```

Implémente [Fonction\\_nD](#).

**6.300.2.22 NbComposante()** [2/2]

```
int F_nD_courbe1D::NbComposante ( ) const [inline], [virtual]
```

Implémente [Fonction\\_nD](#).

**6.300.2.23 NbVariable()**

`int F_nD_courbelD::NbVariable ( ) const [inline], [virtual]`  
 Réimplémentée à partir de [Fonction\\_nD](#).

**6.300.2.24 operator=() [1/2]**

`Fonction_nD & F_nD_courbelD::operator= (`  
     `const Fonction_nD & elt ) [virtual]`  
 Implémente [Fonction\\_nD](#).

**6.300.2.25 operator=() [2/2]**

`Fonction_nD & F_nD_courbelD::operator= (`  
     `const Fonction_nD & elt ) [virtual]`  
 Implémente [Fonction\\_nD](#).

**6.300.2.26 SchemaXML\_Fonctions\_nD() [1/2]**

`void F_nD_courbelD::SchemaXML_Fonctions_nD (`  
     `ofstream & sort,`  
     `const Enum_IO_XML enu ) [virtual]`  
 Implémente [Fonction\\_nD](#).

**6.300.2.27 SchemaXML\_Fonctions\_nD() [2/2]**

`void F_nD_courbelD::SchemaXML_Fonctions_nD (`  
     `ofstream & sort,`  
     `const Enum_IO_XML enu ) [virtual]`  
 Implémente [Fonction\\_nD](#).

**6.300.2.28 Valeur\_FnD\_interne() [1/2]**

`Tableau< double > & F_nD_courbelD::Valeur_FnD_interne (`  
     `Tableau< double > * val_ddl_enum ) [protected], [virtual]`  
 Implémente [Fonction\\_nD](#).

**6.300.2.29 Valeur\_FnD\_interne() [2/2]**

`Tableau< double > & F_nD_courbelD::Valeur_FnD_interne (`  
     `Tableau< double > * val_ddl_enum,`  
     `Tableau< Coordonnee > * coor_ddl_enum,`  
     `Tableau< TenseurBB * > * tens_ddl_enum,`  
     `bool variables_locales ) [protected], [virtual]`  
 Implémente [Fonction\\_nD](#).

**6.300.2.30 Valeur\_pour\_variables\_globales()**

`Tableau< double > & F_nD_courbelD::Valeur_pour_variables_globales ( ) [virtual]`  
 Implémente [Fonction\\_nD](#).

### 6.300.2.31 Valeur\_pour\_variables\_globales\_interne()

Tableau < double > & F\_nD\_courbe1D::Valeur\_pour\_variables\_globales\_interne ( ) [protected], [virtual]

Implémente [Fonction\\_nD](#).

La documentation de cette classe a été générée à partir du fichier suivant :

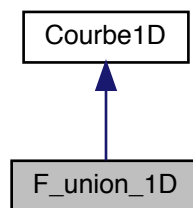
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_nD\_courbe1D (copy: conflict on 2017-11-21).h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_nD\_courbe1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_nD\_courbe1D (copy: conflict on 2017-11-21).cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_nD\_courbe1D.cc

## 6.301 Référence de la classe F\_union\_1D

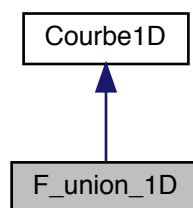
Classe permettant le calcul d'une fonction égale à l'union d'une suite de fonctions définies sur des segments contigus de manière à couvrir un segment global.

```
#include <F_union_1D.h>
```

Graphe d'héritage de F\_union\_1D:



Graphe de collaboration de F\_union\_1D:



### Fonctions membres publiques

- `F_union_1D` (string nom="")
- `F_union_1D` (const [F\\_union\\_1D](#) &Co)
- `F_union_1D` (const [Courbe1D](#) &Co)
- void [Affiche](#) () const

- *affichage de la courbe*
- bool `Complet_courbe` () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void `LectDonnParticulieres_courbes` (const string &nom, `UtilLecture` \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- bool `DependAutreCourbes` () const  
*établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non*
- list< string > & `ListDependanceCourbes` (list< string > &lico) const  
*par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this*
- void `Lien_entre_courbe` (list< `Courbe1D` \* > &liptco)  
*3) établit la connection entre la demande de \*this et les courbes passées en paramètres*
- void `Info_commande_Courbes1D` (`UtilLecture` &entreePrinc)  
*def info fichier de commande*
- double `Valeur` (double x)  
*ramène la valeur*
- `Courbe1D::ValDer Valeur_Et_derivee` (double x)  
*ramène la valeur et la dérivée en paramètre*
- double `Derivee` (double x)  
*ramène la dérivée*
- `Courbe1D::ValDer2 Valeur_Et_der12` (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double `Der_sec` (double x)  
*ramène la dérivée seconde*
- `Courbe1D::Valbool Valeur_stricte` (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- `Courbe1D::ValDerbool Valeur_Et_derivee_stricte` (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void `Lecture_base_info` (ifstream &ent, const int cas)  
*— lecture écriture de restart — cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void `Ecriture_base_info` (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- void `SchemaXML_Courbes1D` (ofstream &sort, const `Enum_IO_XML` enu)  
*sortie du schemaXML: en fonction de enu*

## Fonctions membres protégées

- void `DefCourbesMembres` (`Tableau`< `Courbe1D` \* > &FFi)

## Attributs protégés

- `Tableau`< `Courbe1D` \* > `Fi`
- `Tableau`< double > `Xi`
- `Tableau`< string > `nom_courbei`
- int `indice_precedant`

## Membres hérités additionnels

### 6.301.1 Description détaillée

Classe permettant le calcul d'une fonction égale à l'union d'une suite de fonctions définies sur des segments contigus de manière à couvrir un segment global.

BUT: Classe permettant le calcul d'une fonction égale à l'union d'une suite de fonctions définies sur des segments contigus de manière à couvrir un segment global.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

14/10/2004

**6.301.2 Documentation des fonctions membres****6.301.2.1 Affiche()**

```
void F_union_1D::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

**6.301.2.2 Complet\_courbe()**

```
bool F_union_1D::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

**6.301.2.3 DependAutreCourbes()**

```
bool F_union_1D::DependAutreCourbes ( ) const [virtual]
```

établir le lien entre la courbe et des courbes déjà existantes dont on connait que le nom permet ainsi de compléter la courbe 1) renseigne si la courbe dépend d'autre courbe ou non  
Réimplémentée à partir de [Courbe1D](#).

**6.301.2.4 Der\_sec()**

```
double F_union_1D::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

**6.301.2.5 Derivee()**

```
double F_union_1D::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

**6.301.2.6 Ecriture\_base\_info()**

```
void F_union_1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

**6.301.2.7 Info\_commande\_Courbes1D()**

```
void F_union_1D::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

**6.301.2.8 LectDonnParticulieres\_courbes()**

```
void F_union_1D::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

**6.301.2.9 Lecture\_base\_info()**

```
void F_union_1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.301.2.10 Lien\_entre\_courbe()**

```
void F_union_1D::Lien_entre_courbe (
    list< Courbe1D * > & ) [virtual]
```

3) établit la connection entre la demande de \*this et les courbes passées en paramètres

Réimplémentée à partir de [Courbe1D](#).

**6.301.2.11 ListDependanceCourbes()**

```
list< string > & F_union_1D::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

par défaut non 2) retourne une liste de nom correspondant aux noms de courbes dont dépend \*this

Réimplémentée à partir de [Courbe1D](#).

**6.301.2.12 SchemaXML\_Courbes1D()**

```
void F_union_1D::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

**6.301.2.13 Valeur()**

```
double F_union_1D::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

**6.301.2.14 Valeur\_Et\_der12()**

```
Courbe1D::ValDer2 F_union_1D::Valeur_Et_der12 (  
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre  
Implémente [Courbe1D](#).

**6.301.2.15 Valeur\_Et\_derivee()**

```
Courbe1D::ValDer F_union_1D::Valeur_Et_derivee (  
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre  
Implémente [Courbe1D](#).

**6.301.2.16 Valeur\_Et\_derivee\_stricte()**

```
Courbe1D::ValDerbool F_union_1D::Valeur_Et_derivee_stricte (  
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant  
Implémente [Courbe1D](#).

**6.301.2.17 Valeur\_stricte()**

```
Courbe1D::Valbool F_union_1D::Valeur_stricte (  
    double x ) [virtual]
```

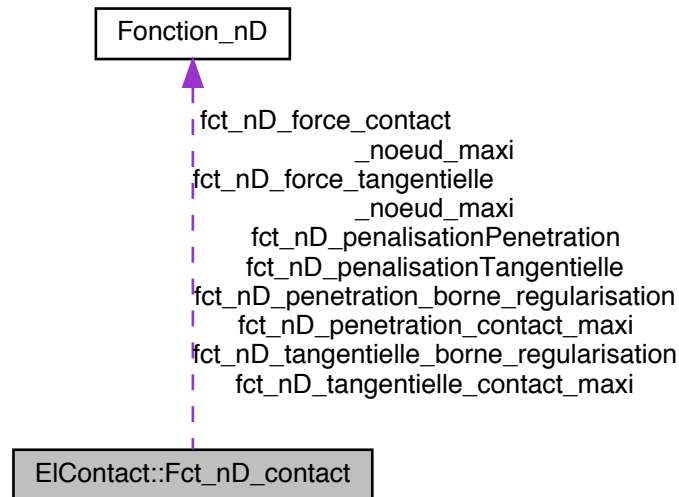
ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y  
Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_union\_1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/F\_union\_1D.cc

## 6.302 Référence de la classe EContact::Fct\_nD\_contact

Graphe de collaboration de EContact::Fct\_nD\_contact:



### Fonctions membres publiques

- **Fct\_nD\_contact** ()  
     ----- particularité class *Fct\_nD\_contact* -----
- **Fct\_nD\_contact** (const [Fct\\_nD\\_contact](#) &a)
- **Fct\_nD\_contact** & **operator=** (const [Fct\\_nD\\_contact](#) &a)

### Attributs publics

- [Fonction\\_nD](#) \* **fct\_nD\_penalisationPenetration**
- [Fonction\\_nD](#) \* **fct\_nD\_penetration\_contact\_maxi**
- [Fonction\\_nD](#) \* **fct\_nD\_penetration\_borne\_regularisation**
- [Fonction\\_nD](#) \* **fct\_nD\_force\_contact\_noeud\_maxi**
- [Fonction\\_nD](#) \* **fct\_nD\_penalisationTangentielle**
- [Fonction\\_nD](#) \* **fct\_nD\_tangentielle\_contact\_maxi**
- [Fonction\\_nD](#) \* **fct\_nD\_tangentielle\_borne\_regularisation**
- [Fonction\\_nD](#) \* **fct\_nD\_force\_tangentielle\_noeud\_maxi**

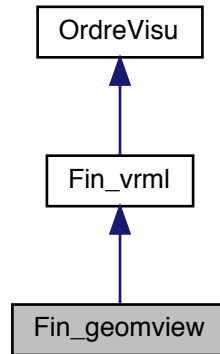
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/EContact.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/EContact.cc

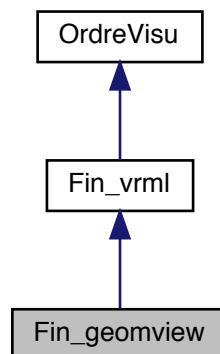


## 6.303 Référence de la classe Fin\_geomview

Graphe d'héritage de Fin\_geomview:



Graphe de collaboration de Fin\_geomview:



### Fonctions membres publiques

- **Fin\_geomview** (const [Fin\\_geomview](#) &algo)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#)↔::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

## Membres hérités additionnels

### 6.303.1 Documentation des fonctions membres

#### 6.303.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Fin_geomview::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.303.1.2 ExeOrdre()

```
void Fin_geomview::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
    const map< string, const double *, std::less< string > > & ,
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.303.1.3 Lecture\_parametres\_OrdreVisu()

```
void Fin_geomview::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

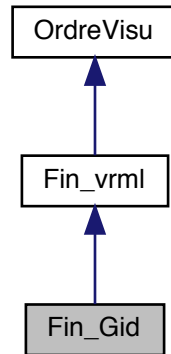
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

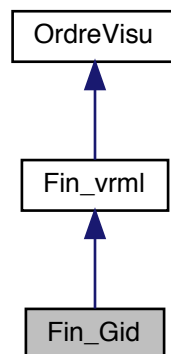
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Fin\_geomview.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Fin\_geomview.cc

## 6.304 Référence de la classe Fin\_Gid

Graphe d'héritage de Fin\_Gid:



Graphe de collaboration de Fin\_Gid:



### Fonctions membres publiques

- **Fin\_Gid** (const [Fin\\_Gid](#) &algo)
- void [ExeOrdre](#) ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#)↔::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)

## Membres hérités additionnels

### 6.304.1 Documentation des fonctions membres

#### 6.304.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Fin_Gid::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.304.1.2 ExeOrdre()

```
void Fin_Gid::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
    const map< string, const double *, std::less< string > > & ,
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.304.1.3 Lecture\_parametres\_OrdreVisu()

```
void Fin_Gid::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

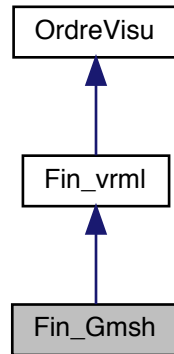
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

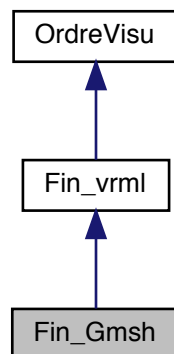
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Fin\_Gid.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Fin\_Gid.cc

## 6.305 Référence de la classe Fin\_Gmsh

Graphe d'héritage de Fin\_Gmsh:



Graphe de collaboration de Fin\_Gmsh:



### Fonctions membres publiques

- **Fin\_Gmsh** (const [Fin\\_Gmsh](#) &algo)
- void [ExeOrdre](#) ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#)↔::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)

## Membres hérités additionnels

### 6.305.1 Documentation des fonctions membres

#### 6.305.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Fin_Gmsh::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.305.1.2 ExeOrdre()

```
void Fin_Gmsh::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
    const map< string, const double *, std::less< string > > & ,
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.305.1.3 Lecture\_parametres\_OrdreVisu()

```
void Fin_Gmsh::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

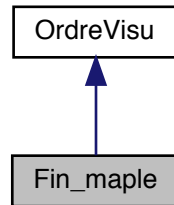
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

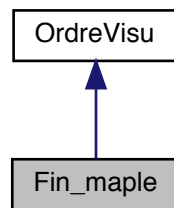
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Fin\_Gmsh.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Fin\_Gmsh.cc

## 6.306 Référence de la classe Fin\_maple

Graphe d'héritage de Fin\_maple:



Graphe de collaboration de Fin\_maple:



### Fonctions membres publiques

- **Fin\_maple** (const [Fin\\_maple](#) &algo)
- void [ExeOrdre](#) ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#) ← ::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)

### Membres hérités additionnels

#### 6.306.1 Documentation des fonctions membres

##### 6.306.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Fin_maple::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.306.1.2 ExeOrdre()

```
void Fin_maple::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
    const map< string, const double *, std::less< string > > & ,
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.306.1.3 Lecture\_parametres\_OrdreVisu()

```
void Fin_maple::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

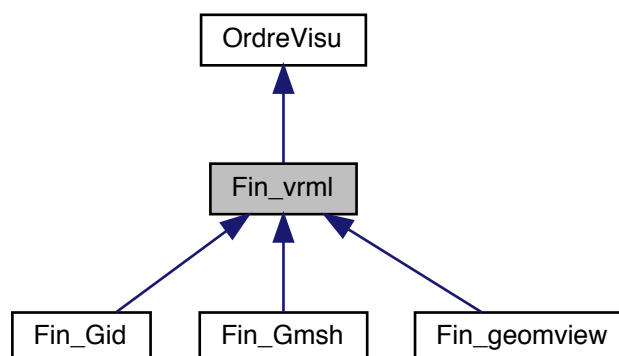
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Fin\_maple.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Fin\_maple.cc

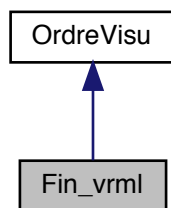
## 6.307 Référence de la classe Fin\_vrml

Grappe d'héritage de Fin\_vrml:





Graphe de collaboration de Fin\_vrml:



## Fonctions membres publiques

- `Fin_vrml` (const `Fin_vrml` &algo)
- void `ExeOrdre` (`ParaGlob` \*, const `Tableau`< int > &, `LesMaillages` \*, bool, `LesReferences` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*, `UtilLecture` &, `OrdreVisu`←  
::EnumTypeIncre, int, bool, const map< string, const double \* , std::less< string > > &, const `List_io`<  
`TypeQuelconque` > &listeVecGlob)
- void `Lecture_parametres_OrdreVisu` (`UtilLecture` &entreePrinc)
- void `Ecriture_parametres_OrdreVisu` (`UtilLecture` &entreePrinc)

## Membres hérités additionnels

### 6.307.1 Documentation des fonctions membres

#### 6.307.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Fin_vrml::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.307.1.2 ExeOrdre()

```
void Fin_vrml::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
```

```
const map< string, const double *, std::less< string > > & ,
const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.307.1.3 Lecture\_parametres\_OrdreVisu()

```
void Fin_vrml::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

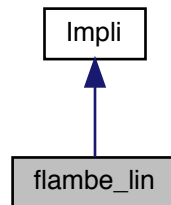
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

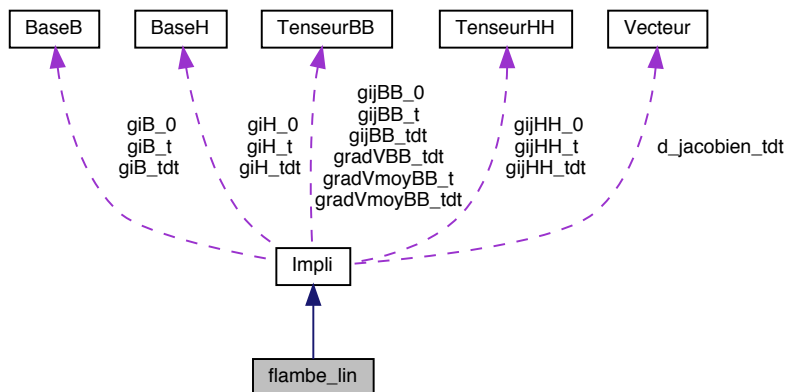
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Fin\_vrml.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Fin\_vrml.cc

## 6.308 Référence de la classe flambe\_lin

Grappe d'héritage de flambe\_lin:



Grappe de collaboration de flambe\_lin:



### Fonctions membres publiques

- **flambe\_lin** (**BaseB** \*ggiB\_0, **BaseH** \*ggiH\_0, **BaseB** \*ggiB\_t, **BaseH** \*ggiH\_t, **BaseB** \*ggiB\_tdt, **Tableau**<**BaseB**> \*d\_ggiB\_tdt, **BaseH** \*ggiH\_tdt, **Tableau**<**BaseH**> \*d\_ggiH\_tdt, **TenseurBB** \*ggijBB\_↔

```

0, TenseurHH *ggijHH_0, TenseurBB *ggijBB_t, TenseurHH *ggijHH_t, TenseurBB *ggijBB_tdt, TenseurHH
*ggijHH_tdt, TenseurBB *ggradVmoyBB_t, TenseurBB *ggradVmoyBB_tdt, TenseurBB *ggradVBB_tdt,
Tableau< TenseurBB * > *gd_gijBB_tdt, Tableau2< TenseurBB * > *gd2_gijBB_tdt, Tableau< TenseurHH
* > *gd_gijHH_tdt, double *gjacobien, double *gjacobien_t, double *gjacobien_0, Vecteur *gd_jacobien_tdt,
Tableau< TenseurBB * > *gd_gradVmoyBB_t, Tableau< TenseurBB * > *gd_gradVmoyBB_tdt, Tableau<
TenseurBB * > *gd_gradVBB_t, Tableau< TenseurBB * > *gd_gradVBB_tdt)
— flambe_lin (const flambe_lin &ex)
— flambe_lin & operator= (const flambe_lin &ex)
— void Mise_a_jour_grandeur (BaseB *ggiB_0, BaseH *ggiH_0, BaseB *ggiB_t, BaseH *ggiH_t, BaseB
*ggiB_tdt, Tableau< BaseB > *d_ggiB_tdt, BaseH *ggiH_tdt, Tableau< BaseH > *d_ggiH_tdt, TenseurBB
*ggijBB_0, TenseurHH *ggijHH_0, TenseurBB *ggijBB_t, TenseurHH *ggijHH_t, TenseurBB *ggijBB_tdt,
TenseurHH *ggijHH_tdt, TenseurBB *ggradVmoyBB_t, TenseurBB *ggradVmoyBB_tdt, TenseurBB *ggradV
VBB_tdt, Tableau< TenseurBB * > *gd_gijBB_tdt, Tableau2< TenseurBB * > *gd2_gijBB_tdt, Tableau<
TenseurHH * > *gd_gijHH_tdt, double *gjacobien, double *gjacobien_t, double *gjacobien_0, Vecteur *gd_
_jacobien_tdt, Tableau< TenseurBB * > *gd_gradVmoyBB_t, Tableau< TenseurBB * > *gd_gradVmoy_
BB_tdt, Tableau< TenseurBB * > *gd_gradVBB_t, Tableau< TenseurBB * > *gd_gradVBB_tdt)

```

### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

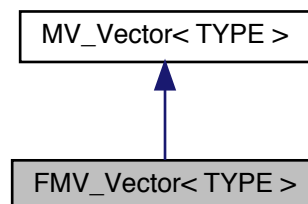
```

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Met_
abstraite_struc_donnees.h

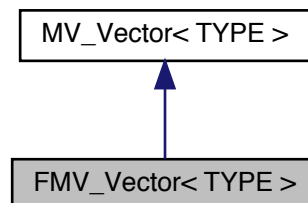
```

## 6.309 Référence du modèle de la classe FMV\_Vector< TYPE >

Grappe d'héritage de FMV\_Vector< TYPE > :



Grappe de collaboration de FMV\_Vector< TYPE > :



## Fonctions membres publiques

- **FMV\_Vector** (int n)
- **FMV\_Vector**< TYPE > & **operator=** (const **FMV\_Vector**< TYPE > &m)
- **FMV\_Vector**< TYPE > & **operator=** (const TYPE &m)

## Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

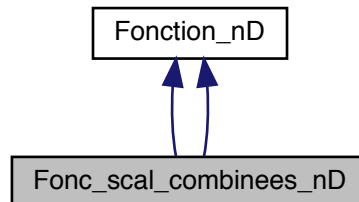
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices\_externes/MV++/mvvtp\_GR.h

## 6.310 Référence de la classe **Fonc\_scal\_combinees\_nD**

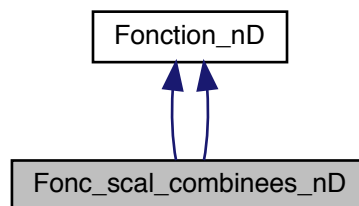
Classe permettant le calcul d'une fonction scalaire nD correspondant à une combinaison d'autres fonctions scalaire nD.

```
#include <Fonc_scal_combinees_nD.h>
```

Graphe d'héritage de **Fonc\_scal\_combinees\_nD**:



Graphe de collaboration de **Fonc\_scal\_combinees\_nD**:



## Fonctions membres publiques

- **Fonc\_scal\_combinees\_nD** (string nom="")
- **Fonc\_scal\_combinees\_nD** (const **Tableau**< string > &var, string nom="")
- **Fonc\_scal\_combinees\_nD** (const **Fonc\_scal\_combinees\_nD** &Co)
- **Fonc\_scal\_combinees\_nD** (const **Fonction\_nD** &Coo)
- **Fonction\_nD** & **operator=** (const **Fonction\_nD** &elt)
- void **Affiche** (int niveau=0) const

- `bool Complet_Fonction () const`
- `int NbComposante () const`
- `void LectDonnParticulieres_Fonction_nD (const string &nom, UtilLecture *)`
- `virtual void Mise_a_jour_variables_globales ()`
- `bool DependAutreFoncCourbes () const`
- `list< string > & ListDependanceCourbes (list< string > &lico) const`
- `list< string > & ListDependanceFonctions (list< string > &lico) const`
- `void Lien_entre_fonc_courbe (list< Fonction_nD * > &liptfonc, list< Courbe1D * > &liptco)`
- `void Info_commande_Fonctions_nD (UtilLecture &entreePrinc)`
- `Tableau< double > & Valeur_pour_variables_globales ()`
- `void Lecture_base_info (ifstream &ent, const int cas)`
- `void Ecriture_base_info (ofstream &sort, const int cas)`
- `void SchemaXML_Fonctions_nD (ofstream &sort, const Enum_IO_XML enu)`
- `Fonc_scalcombinees_nD (string nom="")`
- `Fonc_scalcombinees_nD (const Tableau< string > &var, string nom="")`
- `Fonc_scalcombinees_nD (const Fonc_scalcombinees_nD &Co)`
- `Fonc_scalcombinees_nD (const Fonction_nD &Coo)`
- `Fonction_nD & operator= (const Fonction_nD &elt)`
- `void Affiche (int niveau=0) const`
- `bool Complet_Fonction (bool affichage=true) const`
- `int NbComposante () const`
- `void LectDonnParticulieres_Fonction_nD (const string &nom, UtilLecture *)`
- `virtual void Mise_a_jour_variables_globales ()`
- `bool DependAutreFoncCourbes () const`
- `list< string > & ListDependanceCourbes (list< string > &lico) const`
- `list< string > & ListDependanceFonctions (list< string > &lico) const`
- `void Lien_entre_fonc_courbe (list< Fonction_nD * > &liptfonc, list< Courbe1D * > &liptco)`
- `void Info_commande_Fonctions_nD (UtilLecture &entreePrinc)`
- `void Lecture_base_info (ifstream &ent, const int cas)`
- `void Ecriture_base_info (ofstream &sort, const int cas)`
- `void SchemaXML_Fonctions_nD (ofstream &sort, const Enum_IO_XML enu)`

### Fonctions membres protégées

- `void DefFonctionsMembres (Tableau< Fonction_nD * > &FFi)`
- `void Init_fonction_analytique ()`
- `virtual Tableau< double > & Valeur_FnD_interne (Tableau< double > *xi, Tableau< Coordonnee > *tab←_coor, Tableau< TenseurBB * > *tab_tensBB, bool variables_locales)`
- `void DefFonctionsMembres (Tableau< Fonction_nD * > &FFi)`
- `void Mise_a_jour_variables_nD ()`
- `void Init_fonction_analytique ()`
- `virtual Tableau< double > & Valeur_FnD_interne (Tableau< double > *xi)`
- `virtual Tableau< double > & Valeur_pour_variables_globales_interne ()`

### Attributs protégés

- `string expression_fonction`
- `mu::Parser p`
- `Tableau< double > tab_fVal_int`
- `Tableau< string > nom_variables_int`
- `Tableau< double > tab_fVal`
- `Tableau< double > tab_ret`
- `Tableau< Fonction_nD * > Fi`
- `Tableau< string > nom_fonctioni`
- `Tableau< string > ident_interne`
- `Tableau< Tableau< double > > tab_fVal_Fi`
- `Tableau< Tableau< double > > tab_ret_Fi`
- `int indice_precedant`
- `Tableau< Enum_GrandeurGlobale > enu_variables_globale_int`
- `Tableau< string > nom_variables_globales_int`

## Membres hérités additionnels

### 6.310.1 Description détaillée

Classe permettant le calcul d'une fonction scalaire nD correspondant à une combinaison d'autres fonctions scalaire nD.

BUT: Classe permettant le calcul d'une fonction scalaire nD correspondant à une combinaison d'autres fonctions scalaire nD. La combinaison est définie de manière littérale. On utilise pour cela la bibliothèque MuParser

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

01/06/2016

### 6.310.2 Documentation des fonctions membres

#### 6.310.2.1 Affiche() [1/2]

```
void Fonc_scal_combinees_nD::Affiche (
    int niveau = 0 ) const [virtual]
```

Implémente [Fonction\\_nD](#).

#### 6.310.2.2 Affiche() [2/2]

```
void Fonc_scal_combinees_nD::Affiche (
    int niveau = 0 ) const [virtual]
```

Implémente [Fonction\\_nD](#).

#### 6.310.2.3 Complet\_Fonction() [1/2]

```
bool Fonc_scal_combinees_nD::Complet_Fonction ( ) const [virtual]
```

Implémente [Fonction\\_nD](#).

#### 6.310.2.4 Complet\_Fonction() [2/2]

```
bool Fonc_scal_combinees_nD::Complet_Fonction (
    bool affichage = true ) const [virtual]
```

Implémente [Fonction\\_nD](#).

#### 6.310.2.5 DependAutreFoncCourbes() [1/2]

```
bool Fonc_scal_combinees_nD::DependAutreFoncCourbes ( ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

#### 6.310.2.6 DependAutreFoncCourbes() [2/2]

```
bool Fonc_scal_combinees_nD::DependAutreFoncCourbes ( ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.310.2.7** `Ecriture_base_info()` [1/2]

```
void Fonc_scal_combinees_nD::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.8** `Ecriture_base_info()` [2/2]

```
void Fonc_scal_combinees_nD::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.9** `Info_commande_Fonctions_nD()` [1/2]

```
void Fonc_scal_combinees_nD::Info_commande_Fonctions_nD (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.10** `Info_commande_Fonctions_nD()` [2/2]

```
void Fonc_scal_combinees_nD::Info_commande_Fonctions_nD (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.11** `LectDonnParticulieres_Fonction_nD()` [1/2]

```
void Fonc_scal_combinees_nD::LectDonnParticulieres_Fonction_nD (
    const string & nom,
    UtilLecture * entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.12** `LectDonnParticulieres_Fonction_nD()` [2/2]

```
void Fonc_scal_combinees_nD::LectDonnParticulieres_Fonction_nD (
    const string & nom,
    UtilLecture * ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.13** `Lecture_base_info()` [1/2]

```
void Fonc_scal_combinees_nD::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.14** `Lecture_base_info()` [2/2]

```
void Fonc_scal_combinees_nD::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.15 Lien\_entre\_fonc\_courbe()** [1/2]

```
void Fonc_scal_combinees_nD::Lien_entre_fonc_courbe (
    list< Fonction_nD * > & liptfunc,
    list< Courbe1D * > & liptco ) [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.310.2.16 Lien\_entre\_fonc\_courbe()** [2/2]

```
void Fonc_scal_combinees_nD::Lien_entre_fonc_courbe (
    list< Fonction_nD * > & liptfunc,
    list< Courbe1D * > & liptco ) [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.310.2.17 ListDependanceCourbes()** [1/2]

```
list< string > & Fonc_scal_combinees_nD::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.310.2.18 ListDependanceCourbes()** [2/2]

```
list< string > & Fonc_scal_combinees_nD::ListDependanceCourbes (
    list< string > & lico ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.310.2.19 ListDependanceFonctions()** [1/2]

```
list< string > & Fonc_scal_combinees_nD::ListDependanceFonctions (
    list< string > & lico ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.310.2.20 ListDependanceFonctions()** [2/2]

```
list< string > & Fonc_scal_combinees_nD::ListDependanceFonctions (
    list< string > & lico ) const [virtual]
```

Réimplémentée à partir de [Fonction\\_nD](#).

**6.310.2.21 Mise\_a\_jour\_variables\_globales()** [1/2]

```
void Fonc_scal_combinees_nD::Mise_a_jour_variables_globales ( ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.310.2.22 Mise\_a\_jour\_variables\_globales()** [2/2]

```
virtual void Fonc_scal_combinees_nD::Mise_a_jour_variables_globales ( ) [virtual]
```

Implémente [Fonction\\_nD](#).



**6.310.2.23 NbComposante()** [1/2]

```
int Fonc_scal_combinees_nD::NbComposante ( ) const [inline], [virtual]
Implémente Fonction\_nD.
```

**6.310.2.24 NbComposante()** [2/2]

```
int Fonc_scal_combinees_nD::NbComposante ( ) const [inline], [virtual]
Implémente Fonction\_nD.
```

**6.310.2.25 operator=()** [1/2]

```
Fonction\_nD & Fonc_scal_combinees_nD::operator= (
    const Fonction\_nD & elt ) [virtual]
Implémente Fonction\_nD.
```

**6.310.2.26 operator=()** [2/2]

```
Fonction\_nD & Fonc_scal_combinees_nD::operator= (
    const Fonction\_nD & elt ) [virtual]
Implémente Fonction\_nD.
```

**6.310.2.27 SchemaXML\_Fonctions\_nD()** [1/2]

```
void Fonc_scal_combinees_nD::SchemaXML_Fonctions_nD (
    ofstream & sort,
    const Enum\_IO\_XML enu ) [virtual]
Implémente Fonction\_nD.
```

**6.310.2.28 SchemaXML\_Fonctions\_nD()** [2/2]

```
void Fonc_scal_combinees_nD::SchemaXML_Fonctions_nD (
    ofstream & sort,
    const Enum\_IO\_XML enu ) [virtual]
Implémente Fonction\_nD.
```

**6.310.2.29 Valeur\_FnD\_interne()** [1/2]

```
Tableau< double > & Fonc_scal_combinees_nD::Valeur_FnD_interne (
    Tableau< double > * xi ) [protected], [virtual]
Implémente Fonction\_nD.
```

**6.310.2.30 Valeur\_FnD\_interne()** [2/2]

```
Tableau< double > & Fonc_scal_combinees_nD::Valeur_FnD_interne (
    Tableau< double > * xi,
    Tableau< Coordonnee > * tab_coor,
    Tableau< TenseurBB * > * tab_tensBB,
    bool variables_locales ) [protected], [virtual]
Implémente Fonction\_nD.
```

### 6.310.2.31 Valeur\_pour\_variables\_globales()

`Tableau< double > & Fonc_scalcombinees_nD::Valeur_pour_variables_globales ( ) [virtual]`  
 Implémente [Fonction\\_nD](#).

### 6.310.2.32 Valeur\_pour\_variables\_globales\_interne()

`Tableau< double > & Fonc_scalcombinees_nD::Valeur_pour_variables_globales_interne ( ) [protected], [virtual]`

Implémente [Fonction\\_nD](#).

La documentation de cette classe a été générée à partir du fichier suivant :

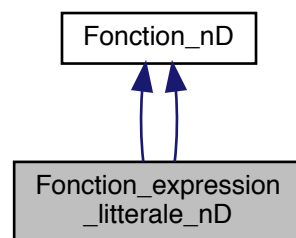
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonc\_scalcombinees\_nD (copy↔  
: conflict on 2017-11-21).h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonc\_scalcombinees\_nD.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonc\_scalcombinees\_nD (copy↔  
: conflict on 2017-11-21).cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonc\_scalcombinees\_nD.cc

## 6.311 Référence de la classe `Fonction_expression_litterale_nD`

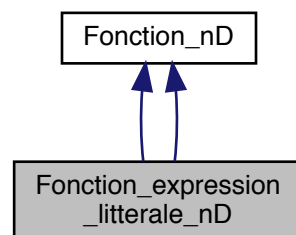
Classe permettant le calcul d'une fonction nD de type : une expression littérale.

```
#include <Fonction_expression_litterale_nD.h>
```

Graphe d'héritage de `Fonction_expression_litterale_nD`:



Graphe de collaboration de `Fonction_expression_litterale_nD`:



## Fonctions membres publiques

- `Fonction_expression_litterale_nD` (string nom="")
- `Fonction_expression_litterale_nD` (const `Fonction_expression_litterale_nD` &Co)
- `Fonction_expression_litterale_nD` (const `Fonction_nD` &Co)
- `Fonction_nD & operator=` (const `Fonction_nD` &elt)
- void `Affiche` (int niveau=0) const
- bool `Complet_Fonction` () const
- void `LectDonnParticulieres_Fonction_nD` (const string &nom, `UtilLecture` \*)
- virtual void `Mise_a_jour_variables_globales` ()
- void `Info_commande_Fonctions_nD` (`UtilLecture` &entreePrinc)
- virtual `Tableau`< double > & `Valeur_pour_variables_globales` ()
- int `NbVariable` () const
- int `NbComposante` () const
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- virtual void `SchemaXML_Fonctions_nD` (ofstream &sort, const `Enum_IO_XML` enu)
- `Fonction_expression_litterale_nD` (string nom="")
- `Fonction_expression_litterale_nD` (string nom\_ref, `Tableau`< string > &nom\_variables, `Tableau`< `Enum_GrandeurGlobale` > &enu\_variables\_globale, `Tableau`< string > &nom\_variables\_globales, string &expression\_fonction)
- `Fonction_expression_litterale_nD` (const `Fonction_expression_litterale_nD` &Co)
- `Fonction_expression_litterale_nD` (const `Fonction_nD` &Co)
- `Fonction_nD & operator=` (const `Fonction_nD` &elt)
- void `Affiche` (int niveau=0) const
- bool `Complet_Fonction` (bool affichage=true) const
- void `LectDonnParticulieres_Fonction_nD` (const string &nom, `UtilLecture` \*)
- virtual void `Mise_a_jour_variables_globales` ()
- void `Info_commande_Fonctions_nD` (`UtilLecture` &entreePrinc)
- int `NbComposante` () const
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- virtual void `SchemaXML_Fonctions_nD` (ofstream &sort, const `Enum_IO_XML` enu)

## Fonctions membres protégées

- virtual `Tableau`< double > & `Valeur_FnD_interne` (`Tableau`< double > \*xi, `Tableau`< `Coordonnee` > \*tab←\_coor, `Tableau`< `TenseurBB` \* > \*tab\_tensBB, bool variables\_locales)
- virtual `Tableau`< double > & `Valeur_FnD_interne` (`Tableau`< double > \*xi)
- virtual `Tableau`< double > & `Valeur_pour_variables_globales_interne` ()
- void `Init_fonction_analytique` ()

## Attributs protégés

- string `expression_fonction`
- `mu`::Parser `p`
- `Tableau`< double > `tab_fVal`
- `Tableau`< double > `tab_ret`

## Membres hérités additionnels

### 6.311.1 Description détaillée

Classe permettant le calcul d'une fonction nD de type : une expression littérale.

BUT: Classe permettant le calcul d'une fonction nD de type : une expression littérale, exploitée ensuite par le parser : `muparser`

Auteur

Gérard Rio

Version

1.0

Date

01/06/2016

## 6.311.2 Documentation des fonctions membres

### 6.311.2.1 Affiche() [1/2]

```
void Fonction_expression_litterale_nD::Affiche (
    int niveau = 0 ) const [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.311.2.2 Affiche() [2/2]

```
void Fonction_expression_litterale_nD::Affiche (
    int niveau = 0 ) const [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.311.2.3 Complet\_Fonction() [1/2]

```
bool Fonction_expression_litterale_nD::Complet_Fonction ( ) const [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.311.2.4 Complet\_Fonction() [2/2]

```
bool Fonction_expression_litterale_nD::Complet_Fonction (
    bool affichage = true ) const [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.311.2.5 Ecriture\_base\_info() [1/2]

```
void Fonction_expression_litterale_nD::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.311.2.6 Ecriture\_base\_info() [2/2]

```
void Fonction_expression_litterale_nD::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.311.2.7 Info\_commande\_Fonctions\_nD() [1/2]

```
void Fonction_expression_litterale_nD::Info_commande_Fonctions_nD (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.8 `Info_commande_Fonctions_nD()` [2/2]**

```
void Fonction_expression_litterale_nD::Info_commande_Fonctions_nD (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.9 `LectDonnParticulieres_Fonction_nD()` [1/2]**

```
void Fonction_expression_litterale_nD::LectDonnParticulieres_Fonction_nD (
    const string & nom,
    UtilLecture * entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.10 `LectDonnParticulieres_Fonction_nD()` [2/2]**

```
void Fonction_expression_litterale_nD::LectDonnParticulieres_Fonction_nD (
    const string & nom,
    UtilLecture * ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.11 `Lecture_base_info()` [1/2]**

```
void Fonction_expression_litterale_nD::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.12 `Lecture_base_info()` [2/2]**

```
void Fonction_expression_litterale_nD::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.13 `Mise_a_jour_variables_globales()` [1/2]**

```
void Fonction_expression_litterale_nD::Mise_a_jour_variables_globales ( ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.14 `Mise_a_jour_variables_globales()` [2/2]**

```
virtual void Fonction_expression_litterale_nD::Mise_a_jour_variables_globales ( ) [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.15 `NbComposante()` [1/2]**

```
int Fonction_expression_litterale_nD::NbComposante ( ) const [inline], [virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.16 NbComposante()** [2/2]

```
int Fonction_expression_litterale_nD::NbComposante ( ) const [inline], [virtual]
Implémente Fonction\_nD.
```

**6.311.2.17 NbVariable()**

```
int Fonction_expression_litterale_nD::NbVariable ( ) const [inline], [virtual]
Réimplémentée à partir de Fonction\_nD.
```

**6.311.2.18 operator=()** [1/2]

```
Fonction\_nD & Fonction_expression_litterale_nD::operator= (
    const Fonction\_nD & elt ) [virtual]
Implémente Fonction\_nD.
```

**6.311.2.19 operator=()** [2/2]

```
Fonction\_nD & Fonction_expression_litterale_nD::operator= (
    const Fonction\_nD & elt ) [virtual]
Implémente Fonction\_nD.
```

**6.311.2.20 SchemaXML\_Fonctions\_nD()** [1/2]

```
void Fonction_expression_litterale_nD::SchemaXML_Fonctions_nD (
    ofstream & sort,
    const Enum\_IO\_XML enu ) [virtual]
Implémente Fonction\_nD.
```

**6.311.2.21 SchemaXML\_Fonctions\_nD()** [2/2]

```
virtual void Fonction_expression_litterale_nD::SchemaXML_Fonctions_nD (
    ofstream & sort,
    const Enum\_IO\_XML enu ) [virtual]
Implémente Fonction\_nD.
```

**6.311.2.22 Valeur\_FnD\_interne()** [1/2]

```
Tableau< double > & Fonction_expression_litterale_nD::Valeur_FnD_interne (
    Tableau< double > * xi ) [protected], [virtual]
Implémente Fonction\_nD.
```

**6.311.2.23 Valeur\_FnD\_interne()** [2/2]

```
Tableau< double > & Fonction_expression_litterale_nD::Valeur_FnD_interne (
    Tableau< double > * xi,
    Tableau< Coordonnee > * tab_coor,
    Tableau< TenseurBB * > * tab_tensBB,
    bool variables_locales ) [protected], [virtual]
Implémente Fonction\_nD.
```

**6.311.2.24 Valeur\_pour\_variables\_globales()**

```
Tableau< double > & Fonction_expression_litterale_nD::Valeur_pour_variables_globales ( )
[virtual]
```

Implémente [Fonction\\_nD](#).

**6.311.2.25 Valeur\_pour\_variables\_globales\_interne()**

```
Tableau< double > & Fonction_expression_litterale_nD::Valeur_pour_variables_globales_interne (
) [protected], [virtual]
```

Implémente [Fonction\\_nD](#).

La documentation de cette classe a été générée à partir du fichier suivant :

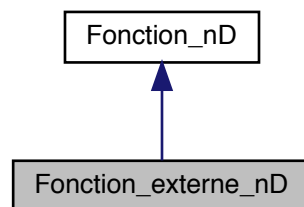
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_expression\_litterale\_nD (copy: conflict on 2017-11-21).h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_expression\_litterale\_nD.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_expression\_litterale\_nD (copy: conflict on 2017-11-21).cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_expression\_litterale\_nD.cc

**6.312 Référence de la classe Fonction\_externe\_nD**

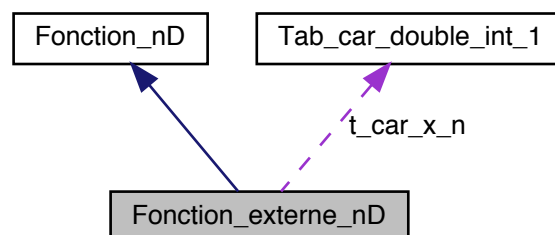
Classe permettant le calcul d'une fonction nD externe, définie par l'utilisateur, via 2 pipes nommés.

```
#include <Fonction_externe_nD.h>
```

Graphe d'héritage de Fonction\_externe\_nD:



Graphe de collaboration de Fonction\_externe\_nD:



## Fonctions membres publiques

- `Fonction_externer_nD` (string nom="")
- `Fonction_externer_nD` (string nom\_ref, `Tableau`< string > &nom\_variables, `Tableau`< `Enum_GrandeurGlobale` > &enu\_variables\_globale, `Tableau`< string > &nom\_variables\_globales, int nb\_double\_ret)
- `Fonction_externer_nD` (const `Fonction_externer_nD` &Co)
- `Fonction_externer_nD` (const `Fonction_nD` &Co)
- `Fonction_nD` & `operator=` (const `Fonction_nD` &elt)
- void `Affiche` (int niveau=0) const
- bool `Completer_Fonction` (bool affichage=true) const
- void `LectDonnParticulieres_Fonction_nD` (const string &nom, `UtilLecture` \*)
- virtual void `Mise_a_jour_variables_globales` ()
- void `Info_commande_Fonctions_nD` (`UtilLecture` &entreePrinc)
- int `NbComposante` () const
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- virtual void `SchemaXML_Fonctions_nD` (ofstream &sort, const `Enum_IO_XML` enu)

## Fonctions membres protégées

- virtual `Tableau`< double > & `Valeur_FnD_interne` (`Tableau`< double > \*xi)
- virtual `Tableau`< double > & `Valeur_pour_variables_globales_interne` ()
- void `InitialisationPipesNommes` ()
- void `LectureResultatsPipe` ()
- void `EcritureDonneesPipe` ()
- void `Change_nom_pipe_envoi` (const string &env)
- void `Change_nom_pipe_reception` (const string &recep)

## Attributs protégés

- string `envoi`
- string `reception`
- int `passage`
- `Tab_car_double_int_1 t_car_x_n`
- `Tableau`< double > `tab_fVal`
- `Tableau`< double > `tab_ret`

## Membres hérités additionnels

### 6.312.1 Description détaillée

Classe permettant le calcul d'une fonction nD externe, définie par l'utilisateur, via 2 pipes nommés.

Auteur

Gérard Rio

Version

1.0

Date

01/06/2016

### 6.312.2 Documentation des fonctions membres

#### 6.312.2.1 Affiche()

```
void Fonction_externer_nD::Affiche (
    int niveau = 0 ) const [virtual]
```

Implémente `Fonction_nD`.



### 6.312.2.2 `Complet_Fonction()`

```
bool Fonction_externe_nD::Complet_Fonction (
    bool affichage = true ) const [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.3 `Ecriture_base_info()`

```
void Fonction_externe_nD::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.4 `Info_commande_Fonctions_nD()`

```
void Fonction_externe_nD::Info_commande_Fonctions_nD (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.5 `LectDonnParticulieres_Fonction_nD()`

```
void Fonction_externe_nD::LectDonnParticulieres_Fonction_nD (
    const string & nom,
    UtilLecture * entreePrinc ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.6 `Lecture_base_info()`

```
void Fonction_externe_nD::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.7 `Mise_a_jour_variables_globales()`

```
void Fonction_externe_nD::Mise_a_jour_variables_globales ( ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.8 `NbComposante()`

```
int Fonction_externe_nD::NbComposante ( ) const [inline], [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.9 `operator=()`

```
Fonction_nD & Fonction_externe_nD::operator= (
    const Fonction_nD & elt ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.10 SchemaXML\_Fonctions\_nD()

```
void Fonction_externe_nD::SchemaXML_Fonctions_nD (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.11 Valeur\_FnD\_interne()

```
Tableau< double > & Fonction_externe_nD::Valeur_FnD_interne (
    Tableau< double > * xi ) [protected], [virtual]
```

Implémente [Fonction\\_nD](#).

### 6.312.2.12 Valeur\_pour\_variables\_globales\_interne()

```
Tableau< double > & Fonction_externe_nD::Valeur_pour_variables_globales_interne ( ) [protected],
[virtual]
```

Implémente [Fonction\\_nD](#).

La documentation de cette classe a été générée à partir du fichier suivant :

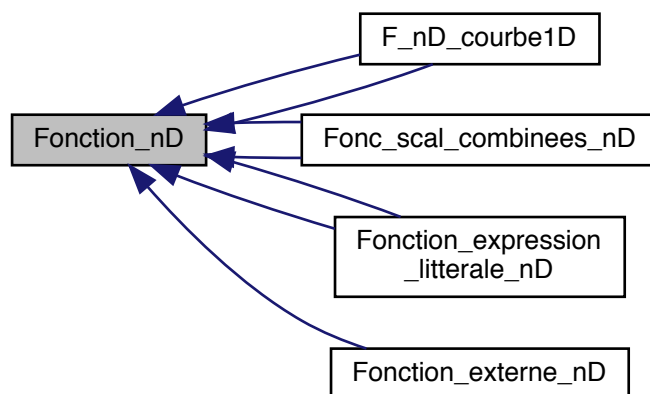
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_externe\_nD.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_externe\_nD.cc

## 6.313 Référence de la classe Fonction\_nD

Classe virtuelle d'interface permettant le calcul d'une fonction nD ainsi qu'éventuellement un certain nombre d'informations supplémentaires telles que dérivées.

```
#include <Fonction_nD.h>
```

Graphe d'héritage de [Fonction\\_nD](#):



### Fonctions membres publiques

- [Fonction\\_nD](#) (string nom="", [EnumFonction\\_nD](#) typ=AUCUNE\_FONCTION\_nD)
- [Fonction\\_nD](#) (const [Tableau](#)< string > &var, string nom="", [EnumFonction\\_nD](#) typ=AUCUNE\_↔ FONCTION\_nD)
- [Fonction\\_nD](#) (const [Fonction\\_nD](#) &Co)
- virtual [Fonction\\_nD](#) & **operator=** (const [Fonction\\_nD](#) &elt)=0

- virtual void **Affiche** (int niveau=0) const =0
- const string & **NomFonction** () const
- virtual bool **Complet\_Fonction** (bool affichage=true) const =0
- virtual int **NbVariable** () const
- virtual int **NbVariable\_locale** () const
- virtual int **NbVariable\_globale** () const
- const [Tableau](#)< string > & **Nom\_variables** () const
- const [Tableau](#)< [Enum\\_GrandeurGlobale](#) > & **Enu\_variables\_globales** () const
- const [Tableau](#)< string > & **Nom\_variables\_globales** () const
- const [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & **Tab\_enu\_etendu** () const
- const [Tableau](#)< [EnumTypeQuelconque](#) > & **Tab\_enu\_quelconque** () const
- bool **Equivalence\_nom\_enu\_etendu\_et\_enu\_quelconque** () const
- const [Tableau](#)< int > & **lindex\_enu\_etendu** () const
- const [Tableau](#)< int > & **lindex\_enu\_quelconque** () const
- const [Tableau](#)< int > & **Type\_des\_variables\_locales** () const
- const [Tableau](#)< int > & **Index\_dans\_tableau** () const
- const [Tableau](#)< int > & **Tailles\_tab** () const
- virtual int **NbComposante** () const =0
- virtual void **LectDonnParticulieres\_Fonction\_nD** (const string &nom, [UtilLecture](#) \*)=0
- virtual void **Mise\_a\_jour\_variables\_globales** ()=0
- virtual bool **DependAutreFoncCourbes** () const
- virtual list< string > & **ListDependanceCourbes** (list< string > &lico) const
- virtual list< string > & **ListDependanceFonctions** (list< string > &lico) const
- virtual void **Lien\_entre\_fonc\_courbe** (list< [Fonction\\_nD](#) \* > &liptfnc, list< [Courbe1D](#) \* > &liptco)
- virtual void **Info\_commande\_Fonctions\_nD** ([UtilLecture](#) &entreePrinc)=0
- [Tableau](#)< double > & **Valeur\_FnD\_tab\_scalaire** ([Tableau](#)< double > \*val\_double)
- [Tableau](#)< double > & **Val\_FnD\_Evoluee** ([Tableau](#)< double > \*val\_ddl\_enum, [Tableau](#)< [Coordonnee](#) > \*coor\_ddl\_enum, [Tableau](#)< [TenseurBB](#) \* > \*tens\_ddl\_enum, [Tableau](#)< [TypeQuelconque](#) > \*tqi=NULL, [Tableau](#)< int > \*t\_num\_ordre=NULL)
- [Tableau](#)< double > & **Val\_ddl\_enum** ()
- [Tableau](#)< [Coordonnee](#) > & **Coor\_ddl\_enum** ()
- [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & **Premier\_famille\_Coor** ()
- [Tableau](#)< [TenseurBB](#) \* > & **Tens\_ddl\_enum** ()
- [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & **Premier\_famille\_tenseur** ()
- [List\\_io](#)< [TypeQuelconque](#) > & **Li\_equi\_Quel\_evolue** ()
- [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > & **Li\_enu\_etendu\_scalaire** ()
- [Tableau](#)< double > & **Valeur\_FnD\_Evoluee** ([Tableau](#)< double > \*val\_ddl\_enum, [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > \*li\_evolue\_scalaire, [List\\_io](#)< [TypeQuelconque](#) > \*li\_evoluee\_non\_scalaire, [Tableau](#)< [TypeQuelconque](#) > \*tqi, [Tableau](#)< int > \*t\_num\_ordre)
- [Tableau](#)< double > & **Valeur\_FnD** ([Tableau](#)< [Ddl\\_etendu](#) > \*t\_enu, [Tableau](#)< [TypeQuelconque](#) > \*tqi, [Tableau](#)< int > \*t\_num\_ordre)
- int **Depend\_M** () const
- [Tableau](#)< double > & **Valeur\_pour\_variables\_globales** ()
- virtual [Tableau](#)< double > & **Valeur\_FnD\_interne** ([Tableau](#)< double > \*val\_ddl\_enum)=0
- virtual [Tableau](#)< double > & **Valeur\_pour\_variables\_globales\_interne** ()=0
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)=0
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)=0
- virtual void **SchemaXML\_Fonctions\_nD** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)=0
- [EnumFonction\\_nD](#) **Type\_Fonction** () const
- **Fonction\_nD** (string nom="", [EnumFonction\\_nD](#) typ=AUCUNE\_FONCTION\_nD)
- **Fonction\_nD** (const [Tableau](#)< string > &var, string nom="", [EnumFonction\\_nD](#) typ=AUCUNE\_↔  
FONCTION\_nD)
- **Fonction\_nD** (const [Fonction\\_nD](#) &Co)
- virtual [Fonction\\_nD](#) & **operator=** (const [Fonction\\_nD](#) &elt)=0
- virtual void **Affiche** (int niveau=0) const =0
- const string & **NomFonction** () const
- virtual bool **Complet\_Fonction** () const =0
- virtual int **NbVariable** () const
- const [Tableau](#)< string > & **Nom\_variables** () const
- const [Tableau](#)< [Enum\\_GrandeurGlobale](#) > & **Enu\_variables\_globales** () const
- const [Tableau](#)< string > & **Nom\_variables\_globales** () const
- const [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & **Tab\_enu\_etendu** () const
- const [Tableau](#)< [EnumTypeQuelconque](#) > & **Tab\_enu\_quelconque** () const
- bool **Equivalence\_nom\_enu\_etendu\_et\_enu\_quelconque** () const

- const [Tableau](#)< int > & [lindex\\_enu\\_etendu](#) () const
- const [Tableau](#)< int > & [lindex\\_enu\\_quelconque](#) () const
- const [Tableau](#)< int > & [Type\\_des\\_variables\\_locales](#) () const
- const [Tableau](#)< int > & [Index\\_dans\\_tableau](#) () const
- const [Tableau](#)< int > & [Tailles\\_tab](#) () const
- virtual int [NbComposante](#) () const =0
- virtual void [LectDonnParticulieres\\_Fonction\\_nD](#) (const string &nom, [UtilLecture](#) \*)=0
- virtual void [Mise\\_a\\_jour\\_variables\\_globales](#) ()=0
- virtual bool [DependAutreFoncCourbes](#) () const
- virtual list< string > & [ListDependanceCourbes](#) (list< string > &lico) const
- virtual list< string > & [ListDependanceFonctions](#) (list< string > &lico) const
- virtual void [Lien\\_entre\\_fonc\\_courbe](#) (list< [Fonction\\_nD](#) \* > &liptfonc, list< [Courbe1D](#) \* > &liptco)
- virtual void [Info\\_commande\\_Fonctions\\_nD](#) ([UtilLecture](#) &entreePrinc)=0
- virtual [Tableau](#)< double > & [Val\\_FnD\\_Evoluee](#) ([Tableau](#)< double > \*val\_ddl\_enum, [Tableau](#)< [Coordonnee](#) > \*coor\_ddl\_enum, [Tableau](#)< [TenseurBB](#) \* > \*tens\_ddl\_enum, [Tableau](#)< [TypeQuelconque](#) > \*tqi=NULL, [Tableau](#)< int > \*t\_num\_ordre=NULL)
- [Tableau](#)< double > & [Val\\_ddl\\_enum](#) ()
- [Tableau](#)< [Coordonnee](#) > & [Coor\\_ddl\\_enum](#) ()
- [Tableau](#)< [TenseurBB](#) \* > & [Tens\\_ddl\\_enum](#) ()
- virtual [Tableau](#)< double > & [Valeur\\_FnD](#) ()
- virtual [Tableau](#)< double > & [Valeur\\_FnD](#) ([Tableau](#)< [Ddl\\_etendu](#) > \*t\_enu, [Tableau](#)< [TypeQuelconque](#) > \*tqi, [Tableau](#)< int > \*t\_num\_ordre)
- int [Depend\\_M](#) () const
- virtual [Tableau](#)< double > & [Valeur\\_pour\\_variables\\_globales](#) ()=0
- virtual void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)=0
- virtual void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)=0
- virtual void [SchemaXML\\_Fonctions\\_nD](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)=0
- [EnumFonction\\_nD](#) [Type\\_Fonction](#) () const
- [Fonction\\_nD](#) (string nom="", [EnumFonction\\_nD](#) typ=AUCUNE\_FONCTION\_nD)
- [Fonction\\_nD](#) (const [Tableau](#)< string > &var, string nom="", [EnumFonction\\_nD](#) typ=AUCUNE\_←  
FONCTION\_nD)
- [Fonction\\_nD](#) (string nom\_ref\_, [Tableau](#)< string > &nom\_variables\_non\_globales, [Tableau](#)< [Enum\\_GrandeurGlobale](#) > &enu\_variables\_globale\_, [Tableau](#)< string > &nom\_variables\_globales\_, [EnumFonction\\_nD](#) typ)
- [Fonction\\_nD](#) (const [Fonction\\_nD](#) &Co)
- virtual [Fonction\\_nD](#) & [operator=](#) (const [Fonction\\_nD](#) &elt)=0
- virtual void [Affiche](#) (int niveau=0) const =0
- const string & [NomFonction](#) () const
- virtual bool [Compleet\\_Fonction](#) (bool affichage=true) const =0
- virtual int [NbVariable](#) () const
- virtual int [NbVariable\\_locale](#) () const
- virtual int [NbVariable\\_globale](#) () const
- const [Tableau](#)< string > & [Nom\\_variables](#) () const
- const [Tableau](#)< [Enum\\_GrandeurGlobale](#) > & [Enu\\_variables\\_globales](#) () const
- const [Tableau](#)< string > & [Nom\\_variables\\_globales](#) () const
- const [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & [Tab\\_enu\\_etendu](#) () const
- const [Tableau](#)< [EnumTypeQuelconque](#) > & [Tab\\_enu\\_quelconque](#) () const
- bool [Equivalence\\_nom\\_enu\\_etendu\\_et\\_enu\\_quelconque](#) () const
- const [Tableau](#)< int > & [lindex\\_enu\\_etendu](#) () const
- const [Tableau](#)< int > & [lindex\\_enu\\_quelconque](#) () const
- const [Tableau](#)< int > & [Type\\_des\\_variables\\_locales](#) () const
- const [Tableau](#)< int > & [Index\\_dans\\_tableau](#) () const
- const [Tableau](#)< int > & [Tailles\\_tab](#) () const
- virtual int [NbComposante](#) () const =0
- virtual void [LectDonnParticulieres\\_Fonction\\_nD](#) (const string &nom, [UtilLecture](#) \*)=0
- virtual void [Mise\\_a\\_jour\\_variables\\_globales](#) ()=0
- virtual bool [DependAutreFoncCourbes](#) () const
- virtual list< string > & [ListDependanceCourbes](#) (list< string > &lico) const
- virtual list< string > & [ListDependanceFonctions](#) (list< string > &lico) const
- virtual void [Lien\\_entre\\_fonc\\_courbe](#) (list< [Fonction\\_nD](#) \* > &liptfonc, list< [Courbe1D](#) \* > &liptco)
- virtual void [Info\\_commande\\_Fonctions\\_nD](#) ([UtilLecture](#) &entreePrinc)=0
- [Tableau](#)< double > & [Valeur\\_FnD\\_tab\\_scalaire](#) ([Tableau](#)< double > \*val\_double)
- [Tableau](#)< double > & [Val\\_FnD\\_Evoluee](#) ([Tableau](#)< double > \*val\_ddl\_enum, [Tableau](#)< [Coordonnee](#) > \*coor\_ddl\_enum, [Tableau](#)< [TenseurBB](#) \* > \*tens\_ddl\_enum, [Tableau](#)< const [TypeQuelconque](#) \* > \*tqi=NULL, [Tableau](#)< int > \*t\_num\_ordre=NULL)

- [Tableau](#)< double > & [Val\\_ddl\\_enum](#) ()
- [Tableau](#)< [Coordonnee](#) > & [Coor\\_ddl\\_enum](#) ()
- [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & [Premier\\_famille\\_Coor](#) ()
- [Tableau](#)< [TenseurBB](#) \* > & [Tens\\_ddl\\_enum](#) ()
- int [Absolue](#) () const
- [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & [Premier\\_famille\\_tenseur](#) ()
- [List\\_io](#)< [TypeQuelconque](#) > & [Li\\_equi\\_Quel\\_evolue](#) ()
- [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > & [Li\\_enu\\_etendu\\_scalaire](#) ()
- [Tableau](#)< double > & [Valeur\\_FnD\\_Evoluee](#) ([Tableau](#)< double > \*val\_ddl\_enum, [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > \*li\_evolue\_scalaire, [List\\_io](#)< [TypeQuelconque](#) > \*li\_evoluee\_non\_scalaire, [Tableau](#)< const [TypeQuelconque](#) \* > \*tqi, [Tableau](#)< int > \*t\_num\_ordre)
- [Tableau](#)< double > & [Valeur\\_FnD](#) ([Tableau](#)< [Ddl\\_etendu](#) > \*t\_enu, [Tableau](#)< const [TypeQuelconque](#) \* > \*tqi, [Tableau](#)< int > \*t\_num\_ordre)
- int [Depend\\_M](#) () const
- int [Depend\\_Mt](#) () const
- int [Depend\\_M0](#) () const
- [Tableau](#)< double > & [Valeur\\_pour\\_variables\\_globales](#) ()
- virtual [Tableau](#)< double > & [Valeur\\_FnD\\_interne](#) ([Tableau](#)< double > \*val\_ddl\_enum)=0
- virtual [Tableau](#)< double > & [Valeur\\_pour\\_variables\\_globales\\_interne](#) ()=0
- virtual void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)=0
- virtual void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)=0
- virtual void [SchemaXML\\_Fonctions\\_nD](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)=0
- [EnumFonction\\_nD](#) [Type\\_Fonction](#) () const

### Fonctions membres publiques statiques

- static [Fonction\\_nD](#) \* [New\\_Fonction\\_nD](#) (string &nom, [EnumFonction\\_nD](#) typeFonction)
- static [Fonction\\_nD](#) \* [New\\_Fonction\\_nD](#) (const [Fonction\\_nD](#) &Co)
- static list< [EnumFonction\\_nD](#) > [Liste\\_Fonction\\_disponible](#) ()
- static [Fonction\\_nD](#) \* [New\\_Fonction\\_nD](#) (string &nom, [EnumFonction\\_nD](#) typeFonction)
- static [Fonction\\_nD](#) \* [New\\_Fonction\\_nD](#) (const [Fonction\\_nD](#) &Co)
- static list< [EnumFonction\\_nD](#) > [Liste\\_Fonction\\_disponible](#) ()
- static [Fonction\\_nD](#) \* [New\\_Fonction\\_nD](#) (string &nom, [EnumFonction\\_nD](#) typeFonction)
- static [Fonction\\_nD](#) \* [New\\_Fonction\\_nD](#) (const [Fonction\\_nD](#) &Co)
- static list< [EnumFonction\\_nD](#) > [Liste\\_Fonction\\_disponible](#) ()

### Fonctions membres protégées

- bool [Completer\\_var](#) () const
- void [Definition\\_depend\\_M](#) ()
- void [Definition\\_depend\\_temps](#) ()
- void [Construction\\_index\\_conteneurs\\_evoluees](#) ()
- [Tableau](#)< double > & [Vers\\_tab\\_double](#) ([Tableau](#)< double > &di, const [Tableau](#)< double > \*val\_ddl\_← enum, const [Tableau](#)< [Coordonnee](#) > \*coor\_ddl\_enum, const [Tableau](#)< [TenseurBB](#) \* > \*tens\_ddl\_enum, const [Tableau](#)< [TypeQuelconque](#) > \*tqi=NULL, const [Tableau](#)< int > \*t\_num\_ordre=NULL)
- [Tableau](#)< double > & [Vers\\_tab\\_double](#) ([Tableau](#)< double > &di, const [Tableau](#)< double > \*val\_ddl\_enum, [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > \*li\_evolue\_scalaire, [List\\_io](#)< [TypeQuelconque](#) > \*li\_evoluee\_non\_scalaire, const [Tableau](#)< [TypeQuelconque](#) > \*tqi=NULL, const [Tableau](#)< int > \*t\_num\_ordre=NULL)
- bool [Mise\\_a\\_jour\\_variables\\_globales\\_interne](#) ()
- double [Val\\_avec\\_nbArgVariable](#) (double x,...)
- [Fonction\\_nD](#) & [Transfert\\_info](#) (const [Fonction\\_nD](#) &elt)
- void [Affiche\\_interne](#) (int niveau) const
- void [Lect\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecrit\\_base\\_info](#) (ofstream &sort, const int cas)
- void [SchemXML\\_Fonctions\\_nD](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)
- bool [Completer\\_var](#) () const
- void [Definition\\_depend\\_M](#) ()
- void [Definition\\_depend\\_temps](#) ()
- void [Construction\\_enu\\_etendu\\_et\\_quelconque](#) ()
- void [Construction\\_index\\_conteneurs\\_evoluees](#) ()
- bool [Mise\\_a\\_jour\\_variables\\_globales\\_interne](#) ()
- virtual [Tableau](#)< double > & [Valeur\\_FnD\\_interne](#) ([Tableau](#)< double > \*val\_ddl\_enum, [Tableau](#)< [Coordonnee](#) > \*coor\_ddl\_enum, [Tableau](#)< [TenseurBB](#) \* > \*tens\_ddl\_enum, bool variables\_locales)=0

- double **Val\_avec\_nbArgVariable** (double x,...)
- **Fonction\_nD** & **Transfert\_info** (const **Fonction\_nD** &elt)
- void **Affiche\_interne** (int niveau) const
- bool **Complet\_var** () const
- void **Definition\_depend\_M** ()
- void **Definition\_depend\_temps** ()
- void **Construction\_index\_conteneurs\_evoluees** ()
- **Tableau**< double > & **Vers\_tab\_double** (**Tableau**< double > &di, const **Tableau**< double > \*val\_ddl\_enum, const **Tableau**< **Coordonnee** > \*coor\_ddl\_enum, const **Tableau**< **TenseurBB** \* > \*tens\_ddl\_enum, const **Tableau**< const **TypeQuelconque** \* > \*tqi=NULL, const **Tableau**< int > \*t\_num\_ordre=NULL)
- **Tableau**< double > & **Vers\_tab\_double** (**Tableau**< double > &di, const **Tableau**< double > \*val\_ddl\_enum, **List\_io**< **Ddl\_enum\_etendu** > \*li\_evolue\_scalaire, **List\_io**< **TypeQuelconque** > \*li\_evoluee\_non\_scalaire, const **Tableau**< const **TypeQuelconque** \* > \*tqi=NULL, const **Tableau**< int > \*t\_num\_ordre=NULL)
- bool **Mise\_a\_jour\_variables\_globales\_interne** ()
- double **Val\_avec\_nbArgVariable** (double x,...)
- **Fonction\_nD** & **Transfert\_info** (const **Fonction\_nD** &elt)
- void **Affiche\_interne** (int niveau) const
- void **Lect\_base\_info** (ifstream &ent, const int cas)
- void **Ecrit\_base\_info** (ofstream &sort, const int cas)
- void **SchemXML\_Fonctions\_nD** (ofstream &sort, const **Enum\_IO\_XML** enu)
- void **Lecture\_variables** (string &nom\_lu, **UtilLecture** \*entreePrinc)
- bool **Est\_relatif\_a\_lecture\_variable** (string &nom\_lu)
- void **Affichage\_variables** () const
- void **Recup\_Grandeurs\_globales** ()

### Attributs protégés

- **EnumFonction\_nD** typeFonction
- string **nom\_ref**
- **Tableau**< string > **nom\_variables**
- **Tableau**< double > **t\_inter\_double**
- **Tableau**< **Enum\_GrandeurGlobale** > **enu\_variables\_globale**
- **Tableau**< string > **nom\_variables\_globales**
- **Tableau**< **Ddl\_enum\_etendu** > **tab\_enu\_etendu**
- **Tableau**< **EnumTypeQuelconque** > **tab\_enu\_quelconque**
- **Tableau**< int > **type\_des\_variables**
- **Tableau**< double > **val\_ddl\_enum**
- **List\_io**< **Ddl\_enum\_etendu** > **li\_enu\_etendu\_scalaire**
- **Tableau**< **Ddl\_enum\_etendu** > **premier\_famille\_Coord**
- **Tableau**< **Coordonnee** > **coor\_ddl\_enum**
- **Tableau**< int > **num\_dans\_coor**
- **Tableau**< **Ddl\_enum\_etendu** > **premier\_famille\_tenseur**
- **Tableau**< **TenseurBB** \* > **tens\_ddl\_enum**
- **Tableau**< **Deuxentiers\_enu** > **ind\_tens**
- **List\_io**< **TypeQuelconque** > **li\_equi\_Quel\_evolue**
- **Tableau**< **List\_io**< **TypeQuelconque** > ::iterator > **tab\_equi\_Coor**
- **Tableau**< **List\_io**< **TypeQuelconque** > ::iterator > **tab\_equi\_tens**
- **Tableau**< int > **posi\_ddl\_enum**
- **Tableau**< int > **index\_enu\_etendu**
- **Tableau**< int > **index\_enu\_quelconque**
- **Tableau**< int > **tailles\_tab**
- bool **equivalence\_nom\_enu\_etendu\_et\_enu\_quelconque**
- **Tableau**< double > **x\_x\_i**
- int **depend\_M**
- bool **depend\_temps**
- int **permet\_affichage**
- **Tableau**< **Tableau**< int > > **indice**
- int **taille\_retour**
- int **absolue**
- **Tableau**< double > **x\_glob**
- int **depend\_Mt**
- int **depend\_M0**

### 6.313.1 Description détaillée

Classe virtuelle d'interface permettant le calcul d'une fonction nD ainsi qu'éventuellement un certain nombre d'informations supplémentaires telles que dérivées.

BUT: Classe virtuelle d'interface permettant le calcul d'une fonction nD ainsi qu'éventuellement un certain nombre d'informations supplémentaires telles que dérivées. si le nom de la Fonction = "\_" il s'agit d'une Fonction interne à un objet, c'est-à-dire gérée seulement par l'entité qui la contient, donc pas besoin de nom (elle n'est pas utilisée autre part). Si le nom est différent de "\_" c'est une Fonction qui est gérée et référencée dans LesFonctions, donc à partir de son nom, on peut la retrouver.

Auteur

Gérard Rio

Version

1.0

Date

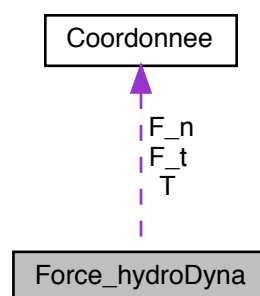
01/06/2016

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_nD (conflict on 2019-05-08).h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_nD (copy: conflict on 2017-11-21).h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_nD.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_nD (conflict on 2019-05-08).cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_nD (copy: conflict on 2017-11-21).cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_nD.cc

## 6.314 Référence de la classe Force\_hydroDyna

Graphe de collaboration de Force\_hydroDyna:



### Fonctions membres publiques

- `Force_hydroDyna` (const `Force_hydroDyna &a`)
- `Force_hydroDyna & operator=` (const `Force_hydroDyna &a`)
- `void Zero` ()

### Attributs publics

- `Coordonnee F_n`
- `Coordonnee F_t`
- `Coordonnee T`

## Amis

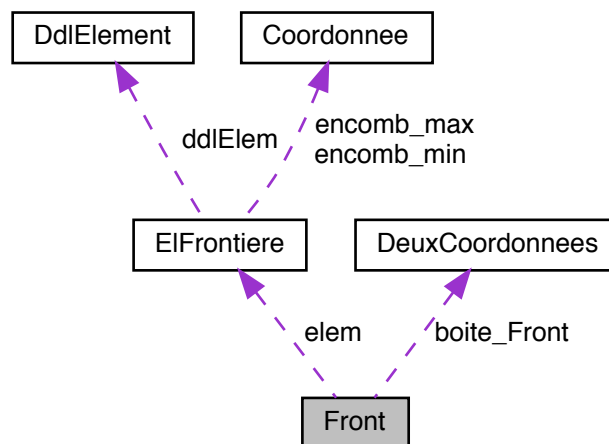
- istream & **operator**>> (istream &ent, [Force\\_hydroDyna](#) &a)
- ostream & **operator**<< (ostream &sort, const [Force\\_hydroDyna](#) &a)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/LesChargeExtSurElement.h

## 6.315 Référence de la classe Front

Graphe de collaboration de Front:



## Classes

- class [Signature\\_Front](#)

## Fonctions membres publiques

- **Front** (const [EIFrontiere](#) &el, Element \*pt, int num\_front)
- **Front** (const [Front](#) &a)
- void **Affiche** () const
- [Front](#) & **operator=** (const [Front](#) &a)
- bool **operator==** (const [Front](#) &a) const
- bool **MemeOrigine** (const [Front](#) &a) const
- bool **operator!=** (const [Front](#) &a) const
- void **DefMitoyen** (Tableau< [Front](#) \* > &tabmitoyen)
- const Tableau< [Front](#) \* > \* **TabMitoyen** () const
- const [Coordonnee](#) & **Encom\_mini\_FR** ()
- const [Coordonnee](#) & **Encom\_maxi\_FR** ()
- bool **In\_boite\_emcombement\_front** (const [Coordonnee](#) &M) const
- bool **In\_boite\_emcombement\_front** (const [Coordonnee](#) &M, double extra) const
- void **Boite\_emcombement\_frontiere** (Enum\_dure temps, double dep\_max=0.)
- [Signature\\_Front](#) **Lecture\_base\_info\_Signature\_Front** (ifstream &ent)
- void **Lecture\_base\_info\_front** (ifstream &ent)
- void **Ecriture\_base\_info\_front** (ofstream &sort)
- void **Change\_elem\_frontiere** (const [EIFrontiere](#) &el, int num\_front)
- void **Change\_PtEI** (Element \*ptei)
- [EIFrontiere](#) \* **Eleme** () const



- const `EIFrontiere * Eleme_const ()` const
- int `NumMail ()`
- `Element * PtEI ()` const
- int `Num_frontiere ()` const
- bool `BonCote_t (const Coordonnee &a, double &r)` const  
*!! il n'y a pas de vérification que l'élément frontière existe !!*
- bool `BonCote_tdt (const Coordonnee &a, double &r)` const

### Attributs protégés

- `EIFrontiere * elem`
- `Element * ptEI`
- int `num_frontiere`
- `DeuxCoordonnees` `boite_Front`
- `Tableau< Front * > * tabmitoyen`

### Attributs protégés statiques

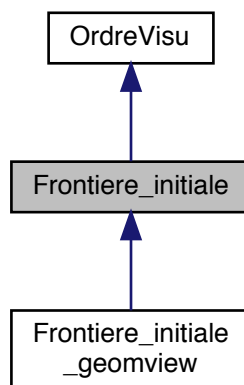
- static double `prop_mini`

La documentation de cette classe a été générée à partir du fichier suivant :

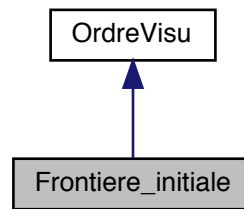
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Front.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Front.cc`

## 6.316 Référence de la classe `Frontiere_initiale`

Graphe d'héritage de `Frontiere_initiale`:



Graphe de collaboration de `Frontiere_initiale`:



## Fonctions membres publiques

- `Frontiere_initiale` (const `Frontiere_initiale` &algo)
- void `ExeOrdre` (`ParaGlob` \*, const `Tableau`< int > &tab\_mail, `LesMaillages` \*, bool unseul\_incre, `LesReferences` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*, `UtilLecture` &entreePrinc, `OrdreVisu::EnumTypeIncre` type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const `List_io`< `TypeQuelconque` > &listeVec←Glob)
- void `ChoixOrdre` ()

## Attributs protégés

- bool `filaire`
- bool `surface`
- bool `numero`
- double `Rcoull`
- double `Gcoull`
- double `Bcoull`
- double `Rcoulf`
- double `Gcoulf`
- double `Bcoulf`
- double `Rcouln`
- double `Gcouln`
- double `Bcouln`
- double `taille_numero`

## Membres hérités additionnels

### 6.316.1 Documentation des fonctions membres

#### 6.316.1.1 ChoixOrdre()

void `Frontiere_initiale::ChoixOrdre` ( ) [virtual]  
Réimplémentée à partir de `OrdreVisu`.

#### 6.316.1.2 ExeOrdre()

```
void Frontiere_initiale::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
```

```
LesMaillages * ,  
bool unseul_incre,  
LesReferences * ,  
LesLoisDeComp * ,  
DiversStockage * ,  
Charge * ,  
LesCondLim * ,  
LesContacts * ,  
Resultats * ,  
UtilLecture & entreePrinc,  
OrdreVisu::EnumTypeIncre type_incre,  
int incre,  
bool animation,  
const map< string, const double *, std::less< string > > & listeVarGlob,  
const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

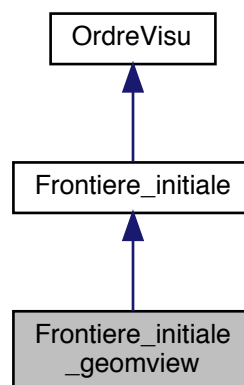
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

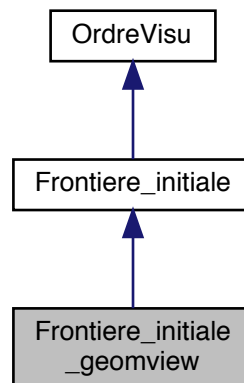
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Commun\_visu/Frontiere\_initiale.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Commun\_visu/Frontiere\_initiale.cc

## 6.317 Référence de la classe `Frontiere_initiale_geomview`

Graphe d'héritage de `Frontiere_initiale_geomview`:



Graphe de collaboration de `Frontiere_initiale_geomview`:



## Fonctions membres publiques

- `Frontiere_initiale_geomview` (const `Frontiere_initiale_geomview` &algo)
- void `ExeOrdre` (`ParaGlob` \*, const `Tableau`< int > &tab\_mail, `LesMaillages` \*, bool unseul\_incre, `LesReferences` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*, `UtilLecture` &entreePrinc, `OrdreVisu::EnumTypeIncre` type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const `List_io`< `TypeQuelconque` > &listeVec↵  
Glob)
- void `Lecture_parametres_OrdreVisu` (`UtilLecture` &entreePrinc)
- void `Ecriture_parametres_OrdreVisu` (`UtilLecture` &entreePrinc)

## Membres hérités additionnels

### 6.317.1 Documentation des fonctions membres

#### 6.317.1.1 `Ecriture_parametres_OrdreVisu()`

```
void Frontiere_initiale_geomview::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de `OrdreVisu`.

#### 6.317.1.2 `ExeOrdre()`

```
void Frontiere_initiale_geomview::ExeOrdre (
    ParaGlob * paraGlob,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
```

```
Resultats * ,  
UtilLecture & entreePrinc,  
OrdreVisu::EnumTypeIncre type_incre,  
int incre,  
bool animation,  
const map< string, const double *, std::less< string > > & listeVarGlob,  
const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [Frontiere\\_initiale](#).

### 6.317.1.3 Lecture\_parametres\_OrdreVisu()

```
void Frontiere_initiale_geomview::Lecture_parametres_OrdreVisu (  
    UtilLecture & entreePrinc ) [virtual]
```

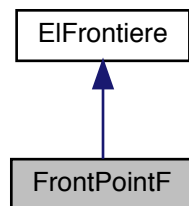
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

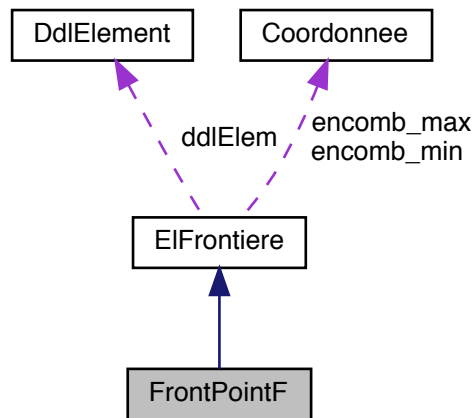
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Frontiere\_initiale\_geomview.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Frontiere\_initiale\_geomview.cc

## 6.318 Référence de la classe FrontPointF

Graphe d'héritage de FrontPointF:



Grphe de collaboration de FrontPointF:



## Fonctions membres publiques

- **FrontPointF** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem, [Coordonnee](#) \*T=NULL)
- **FrontPointF** (const [FrontPointF](#) &a)
- **EIFrontiere** & **operator=** (const [EIFrontiere](#) &a)
- string **TypeFrontiere** () const
- **ElemGeomC0** const & **ElementGeometrique** () const
- **EIFrontiere** \* **NevezElemFront** () const
- **EIFrontiere** \* **NevezElemFront** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- **Coordonnee** **Ref** ()
- void **TangentRef** ([Droite](#) &, [Plan](#) &, int &indic)
- void **Tangent** (const [Coordonnee](#) &, [Coordonnee](#) &M1, [Droite](#) &, [Plan](#) &, int &indic)
- void **AutreTangent** ([Droite](#) &, [Plan](#) &, int &indic)
- bool **InSurf** (const double &) const
- void **Affiche** ([Enum\\_dure](#) temp) const
- [Tableau](#)< [Coordonnee](#) > \* **DernierTangent** ([Droite](#) &, [Plan](#) &, int &indic, bool)
- bool **BonCote\_t** (const [Coordonnee](#) &, double &r) const
  - >>> sans objet ici ramene par default false
- bool **BonCote\_tdt** (const [Coordonnee](#) &, double &r) const
- const [Vecteur](#) & **Phi** ()
- [Tableau](#)< [EIFrontiere](#) \* > & **Frontiere** ()
- [Met\\_abstraite](#) \* **Metrique** ()
- void **Def\_Normale** (const [Coordonnee](#) &V)
- void **Lecture\_base\_info\_EIFrontiere\_pour\_projection** (ifstream &ent)
- void **Ecriture\_base\_info\_EIFrontiere\_pour\_projection** (ofstream &sort)

## Membres hérités additionnels

### 6.318.1 Documentation des fonctions membres

#### 6.318.1.1 Affiche()

```
void FrontPointF::Affiche (
    Enum\_dure temp ) const [inline], [virtual]
```

Implémente [EIFrontiere](#).

**6.318.1.2 AutreTangent()**

```
void FrontPointF::AutreTangent (
    Droite & ,
    Plan & ,
    int & indic ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.3 BonCote\_t()**

```
bool FrontPointF::BonCote_t (
    const Coordonnee & ,
    double & r ) const [inline], [virtual]
```

----->>> sans objet ici ramene par default false

Implémente [ElFrontiere](#).

**6.318.1.4 BonCote\_tdt()**

```
bool FrontPointF::BonCote_tdt (
    const Coordonnee & ,
    double & r ) const [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.5 DernierTangent()**

```
Tableau< Coordonnee > * FrontPointF::DernierTangent (
    Droite & ,
    Plan & ,
    int & indic,
    bool ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.6 Ecriture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontPointF::Ecriture_base_info_ElFrontiere_pour_projection (
    ofstream & sort ) [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.7 ElementGeometrique()**

```
ElemGeomC0 const & FrontPointF::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.8 Frontiere()**

```
Tableau< ElFrontiere * > & FrontPointF::Frontiere ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.9 InSurf()**

```
bool FrontPointF::InSurf (
    const double & ) const [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontPointF::Lecture_base_info_ElFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.11 Metrique()**

```
Met_abstraite * FrontPointF::Metrrique ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.12 NevezElemFront() [1/2]**

```
ElFrontiere * FrontPointF::NevezElemFront ( ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.13 NevezElemFront() [2/2]**

```
ElFrontiere * FrontPointF::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.14 operator=()**

```
ElFrontiere & FrontPointF::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.318.1.15 Phi()**

```
const Vecteur & FrontPointF::Phi ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.16 Ref()**

```
Coordonnee FrontPointF::Ref ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.318.1.17 Tangent()**

```
void FrontPointF::Tangent (
    const Coordonnee & ,
    Coordonnee & M1,
    Droite & ,
    Plan & ,
    int & indic ) [inline], [virtual]
```

Implémente [ElFrontiere](#).



**6.318.1.18 TangentRef()**

```
void FrontPointF::TangentRef (
    Droite & ,
    Plan & ,
    int & indic ) [inline], [virtual]
```

Implémente [EIFrontiere](#).

**6.318.1.19 TypeFrontiere()**

```
string FrontPointF::TypeFrontiere ( ) const [virtual]
```

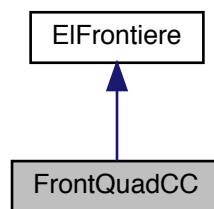
Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

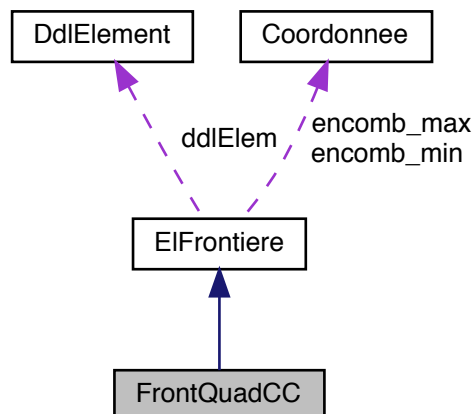
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Point/FrontPointF.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Point/FrontPointF.cc

**6.319 Référence de la classe FrontQuadCC**

Grappe d'héritage de FrontQuadCC:



Grappe de collaboration de FrontQuadCC:



## Fonctions membres publiques

- `FrontQuadCC` (const `Tableau< Noeud * >` &tab, const `DdlElement` &ddlElem)
- `FrontQuadCC` (const `FrontQuadCC` &a)
- `EIFrontiere` & `operator=` (const `EIFrontiere` &a)
- `string TypeFrontiere` () const
- `ElemGeomC0` const & `ElementGeometrique` () const
- `EIFrontiere * NevezElemFront` () const
- `EIFrontiere * NevezElemFront` (const `Tableau< Noeud * >` &tab, const `DdlElement` &ddlElem) const
- `Coordonnee Ref` ()
- void `TangentRef` (`Droite` &dr, `Plan` &pl, int &indic)
- void `Tangent` (const `Coordonnee` &M, `Coordonnee` &M1, `Droite` &dr, `Plan` &pl, int &indic)
- void `AutreTangent` (`Droite` &dr, `Plan` &pl, int &indic)
- bool `InSurf` (const double &eps) const
- `Tableau< Coordonnee > * DernierTangent` (`Droite` &dr, `Plan` &pl, int &indic, bool avec\_var=false)
- bool `BonCote_t` (const `Coordonnee` &a, double &r) const
- bool `BonCote_tdt` (const `Coordonnee` &a, double &r) const
- const `Vecteur` & `Phi` ()
- void `Affiche` (`Enum_dure` temp=TEMPS\_tdt) const
- `Tableau< EIFrontiere * >` & `Frontiere` ()
- `Met_abstraite` \* `Metrique` ()
- void `Lecture_base_info{EIFrontiere_pour_projection}` (ifstream &ent)
- void `Ecriture_base_info{EIFrontiere_pour_projection}` (ofstream &sort)

## Membres hérités additionnels

### 6.319.1 Documentation des fonctions membres

#### 6.319.1.1 Affiche()

```
void FrontQuadCC::Affiche (
    Enum_dure temp = TEMPS_tdt ) const [virtual]
Implémente EIFrontiere.
```

#### 6.319.1.2 AutreTangent()

```
void FrontQuadCC::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
Implémente EIFrontiere.
```

#### 6.319.1.3 BonCote\_t()

```
bool FrontQuadCC::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
Implémente EIFrontiere.
```

#### 6.319.1.4 BonCote\_tdt()

```
bool FrontQuadCC::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
Implémente EIFrontiere.
```

**6.319.1.5 DernierTangent()**

```
Tableau< Coordonnee > * FrontQuadCC::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [EIFrontiere](#).

**6.319.1.6 Ecriture\_base\_info EIFrontiere\_pour\_projection()**

```
void FrontQuadCC::Ecriture_base_info EIFrontiere_pour_projection (
    ofstream & sort ) [virtual]
```

Implémente [EIFrontiere](#).

**6.319.1.7 ElementGeometrique()**

```
ElemGeomC0 const & FrontQuadCC::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [EIFrontiere](#).

**6.319.1.8 Frontiere()**

```
Tableau< EIFrontiere * > & FrontQuadCC::Frontiere ( ) [virtual]
```

Implémente [EIFrontiere](#).

**6.319.1.9 InSurf()**

```
bool FrontQuadCC::InSurf (
    const double & eps ) const [virtual]
```

Implémente [EIFrontiere](#).

**6.319.1.10 Lecture\_base\_info EIFrontiere\_pour\_projection()**

```
void FrontQuadCC::Lecture_base_info EIFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [EIFrontiere](#).

**6.319.1.11 Metrique()**

```
Met_abstraite * FrontQuadCC::Metrrique ( ) [inline], [virtual]
```

Implémente [EIFrontiere](#).

**6.319.1.12 NevezElemFront() [1/2]**

```
EIFrontiere * FrontQuadCC::NevezElemFront ( ) const [virtual]
```

Implémente [EIFrontiere](#).

**6.319.1.13 NevezElemFront() [2/2]**

```
EIFrontiere * FrontQuadCC::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

#### 6.319.1.14 operator=()

```
ElFrontiere & FrontQuadCC::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

#### 6.319.1.15 Phi()

```
const Vecteur & FrontQuadCC::Phi ( ) [virtual]
```

Implémente [ElFrontiere](#).

#### 6.319.1.16 Ref()

```
Coordonnee FrontQuadCC::Ref ( ) [virtual]
```

Implémente [ElFrontiere](#).

#### 6.319.1.17 Tangent()

```
void FrontQuadCC::Tangent (
    const Coordonnee & M,
    Coordonnee & ML,
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

#### 6.319.1.18 TangentRef()

```
void FrontQuadCC::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

#### 6.319.1.19 TypeFrontiere()

```
string FrontQuadCC::TypeFrontiere ( ) const [virtual]
```

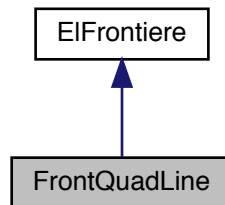
Implémente [ElFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

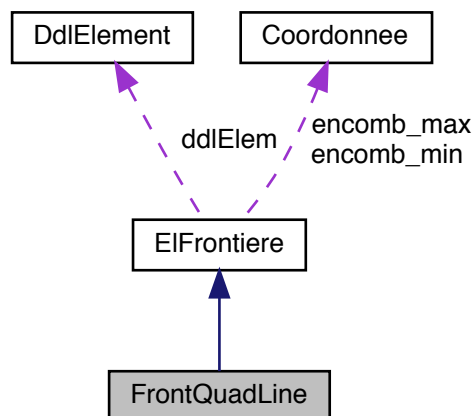
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuad↔  
CC.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuad↔  
CC.cc

## 6.320 Référence de la classe FrontQuadLine

Graphe d'héritage de FrontQuadLine:



Graphe de collaboration de FrontQuadLine:



### Fonctions membres publiques

- **FrontQuadLine** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **FrontQuadLine** (const [FrontQuadLine](#) &a)
- [EIFrontiere](#) & **operator=** (const [EIFrontiere](#) &a)
- string [TypeFrontiere](#) () const
- [ElemGeomC0](#) const & [ElementGeometrique](#) () const
- [EIFrontiere](#) \* [NevezElemFront](#) () const
- [EIFrontiere](#) \* [NevezElemFront](#) (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- [Coordonnee](#) [Ref](#) ()
- void [TangentRef](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- void [Tangent](#) (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)
- void [AutreTangent](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- bool [InSurf](#) (const double &eps) const
- [Tableau](#)< [Coordonnee](#) > \* [DernierTangent](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)
- bool [BonCote\\_t](#) (const [Coordonnee](#) &a, double &r) const
- bool [BonCote\\_tdt](#) (const [Coordonnee](#) &a, double &r) const

- const [Vecteur](#) & [Phi](#) ()
- void [Affiche](#) ([Enum\\_dure](#) temp=TEMPS\_tdt) const
- [Tableau](#)< [ElFrontiere](#) \* > & [Frontiere](#) ()
- [Met\\_abstraite](#) \* [Metrique](#) ()
- void [Lecture\\_base\\_info\\_ElFrontiere\\_pour\\_projection](#) (ifstream &ent)
- void [Ecriture\\_base\\_info\\_ElFrontiere\\_pour\\_projection](#) (ofstream &sort)

## Membres hérités additionnels

### 6.320.1 Documentation des fonctions membres

#### 6.320.1.1 Affiche()

```
void FrontQuadLine::Affiche (
    Enum\_dure temp = TEMPS_tdt ) const [virtual]
```

Implémente [ElFrontiere](#).

#### 6.320.1.2 AutreTangent()

```
void FrontQuadLine::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

#### 6.320.1.3 BonCote\_t()

```
bool FrontQuadLine::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

#### 6.320.1.4 BonCote\_tdt()

```
bool FrontQuadLine::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

#### 6.320.1.5 DernierTangent()

```
Tableau< Coordonnee > * FrontQuadLine::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [ElFrontiere](#).

#### 6.320.1.6 Ecriture\_base\_info\_ElFrontiere\_pour\_projection()

```
void FrontQuadLine::Ecriture_base_info_ElFrontiere_pour_projection (
    ofstream & sort ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.320.1.7 ElementGeometrique()

`ElemGeomC0` const & FrontQuadLine::ElementGeometrique ( ) const [inline], [virtual]  
Implémente [ElFrontiere](#).

### 6.320.1.8 Frontiere()

`Tableau< ElFrontiere * >` & FrontQuadLine::Frontiere ( ) [virtual]  
Implémente [ElFrontiere](#).

### 6.320.1.9 InSurf()

`bool` FrontQuadLine::InSurf (   
                  const double & *eps* ) const [virtual]  
Implémente [ElFrontiere](#).

### 6.320.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()

`void` FrontQuadLine::Lecture\_base\_info\_ElFrontiere\_pour\_projection (   
                  ifstream & *ent* ) [virtual]  
Implémente [ElFrontiere](#).

### 6.320.1.11 Metrique()

`Met_abstraite *` FrontQuadLine::Metrrique ( ) [inline], [virtual]  
Implémente [ElFrontiere](#).

### 6.320.1.12 NevezElemFront() [1/2]

`ElFrontiere *` FrontQuadLine::NevezElemFront ( ) const [virtual]  
Implémente [ElFrontiere](#).

### 6.320.1.13 NevezElemFront() [2/2]

`ElFrontiere *` FrontQuadLine::NevezElemFront (   
          const `Tableau< Noeud * >` & *tab*,   
          const `DdlElement` & *ddlElem* ) const [virtual]  
Implémente [ElFrontiere](#).

### 6.320.1.14 operator=()

`ElFrontiere` & FrontQuadLine::operator= (   
          const `ElFrontiere` & *a* ) [virtual]  
Réimplémentée à partir de [ElFrontiere](#).

### 6.320.1.15 Phi()

`const Vecteur` & FrontQuadLine::Phi ( ) [virtual]  
Implémente [ElFrontiere](#).

### 6.320.1.16 Ref()

```
Coordonnee FrontQuadLine::Ref ( ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.320.1.17 Tangent()

```
void FrontQuadLine::Tangent (
    const Coordonnee & M,
    Coordonnee & M1,
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.320.1.18 TangentRef()

```
void FrontQuadLine::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.320.1.19 TypeFrontiere()

```
string FrontQuadLine::TypeFrontiere ( ) const [virtual]
```

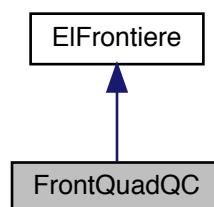
Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuad↔  
Line.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuad↔  
Line.cc

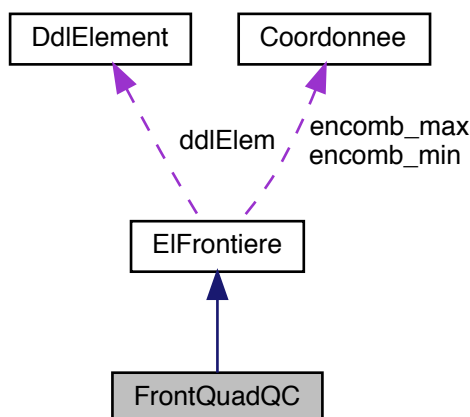
## 6.321 Référence de la classe FrontQuadQC

Graphe d'héritage de FrontQuadQC:





Graphe de collaboration de FrontQuadQC:



## Fonctions membres publiques

- **FrontQuadQC** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **FrontQuadQC** (const [FrontQuadQC](#) &a)
- **EIFrontiere** & **operator=** (const [EIFrontiere](#) &a)
- string **TypeFrontiere** () const
- **ElemGeomC0** const & **ElementGeometrique** () const
- **EIFrontiere** \* **NevezElemFront** () const
- **EIFrontiere** \* **NevezElemFront** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- **Coordonnee** Ref ()
- void **TangentRef** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **Tangent** (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **AutreTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- bool **InSurf** (const double &eps) const
- [Tableau](#)< [Coordonnee](#) > \* **DernierTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)
- bool **BonCote\_t** (const [Coordonnee](#) &a, double &r) const
- bool **BonCote\_tdt** (const [Coordonnee](#) &a, double &r) const
- const [Vecteur](#) & **Phi** ()
- void **Affiche** ([Enum\\_dure](#) temp=TEMPS\_tdt) const
- [Tableau](#)< [EIFrontiere](#) \* > & **Frontiere** ()
- **Met\_abstraite** \* **Metrique** ()
- void **Lecture\_base\_info{EIFrontiere\_pour\_projection}** (ifstream &ent)
- void **Ecriture\_base\_info{EIFrontiere\_pour\_projection}** (ofstream &sort)

## Membres hérités additionnels

### 6.321.1 Documentation des fonctions membres

#### 6.321.1.1 Affiche()

```
void FrontQuadQC::Affiche (
    Enum\_dure temp = TEMPS_tdt ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.2 AutreTangent()

```
void FrontQuadQC::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.3 BonCote\_t()

```
bool FrontQuadQC::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.4 BonCote\_tdt()

```
bool FrontQuadQC::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.5 DernierTangent()

```
Tableau< Coordonnee > * FrontQuadQC::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.6 Ecriture\_base\_info{EIFrontiere\_pour\_projection}()

```
void FrontQuadQC::Ecriture_base_info{EIFrontiere_pour_projection} (
    ofstream & sort ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.7 ElementGeometrique()

```
ElemGeomC0 const & FrontQuadQC::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.8 Frontiere()

```
Tableau<{EIFrontiere} * > & FrontQuadQC::Frontiere ( ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.9 InSurf()

```
bool FrontQuadQC::InSurf (
    const double & eps ) const [virtual]
```

Implémente [EIFrontiere](#).

**6.321.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontQuadQC::Lecture_base_info_ElFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [ElFrontiere](#).

**6.321.1.11 Metrique()**

```
Met_abstraite * FrontQuadQC::Metrrique ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.321.1.12 NevezElemFront() [1/2]**

```
ElFrontiere * FrontQuadQC::NevezElemFront ( ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.321.1.13 NevezElemFront() [2/2]**

```
ElFrontiere * FrontQuadQC::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.321.1.14 operator=()**

```
ElFrontiere & FrontQuadQC::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.321.1.15 Phi()**

```
const Vecteur & FrontQuadQC::Phi ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.321.1.16 Ref()**

```
Coordonnee FrontQuadQC::Ref ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.321.1.17 Tangent()**

```
void FrontQuadQC::Tangent (
    const Coordonnee & M,
    Coordonnee & M1,
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.321.1.18 TangentRef()

```
void FrontQuadQC::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.321.1.19 TypeFrontiere()

```
string FrontQuadQC::TypeFrontiere ( ) const [virtual]
```

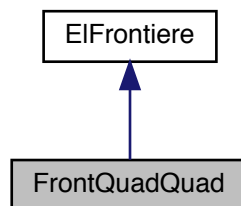
Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

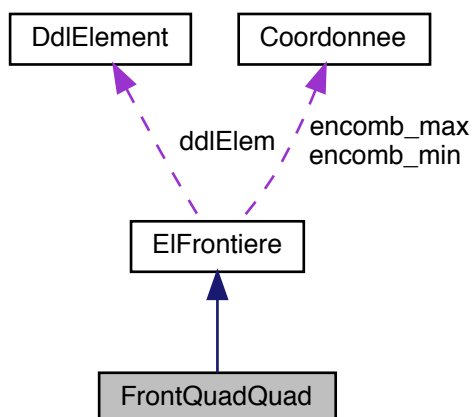
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuadQC.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuadQC.cc

## 6.322 Référence de la classe FrontQuadQuad

Grphe d'héritage de FrontQuadQuad:



Graphes de collaboration de FrontQuadQuad:



## Fonctions membres publiques

- **FrontQuadQuad** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **FrontQuadQuad** (const [FrontQuadQuad](#) &a)  
*de copie*
- [EIFrontiere](#) & **operator=** (const [EIFrontiere](#) &a)
- string [TypeFrontiere](#) () const
- [ElemGeomC0](#) const & [ElementGeometrique](#) () const
- [EIFrontiere](#) \* [NevezElemFront](#) () const
- [EIFrontiere](#) \* [NevezElemFront](#) (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- [Coordonnee](#) [Ref](#) ()
- void [TangentRef](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- void [Tangent](#) (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)
- void [AutreTangent](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- bool [InSurf](#) (const double &eps) const
- [Tableau](#)< [Coordonnee](#) > \* [DernierTangent](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)
- bool [BonCote\\_t](#) (const [Coordonnee](#) &a, double &r) const
- bool [BonCote\\_tdt](#) (const [Coordonnee](#) &a, double &r) const
- const [Vecteur](#) & [Phi](#) ()
- void [Affiche](#) ([Enum\\_dure](#) temp=TEMPS\_tdt) const
- [Tableau](#)< [EIFrontiere](#) \* > & [Frontiere](#) ()
- [Met\\_abstraite](#) \* [Metrique](#) ()
- void [Lecture\\_base\\_info EIFrontiere\\_pour\\_projection](#) (ifstream &ent)
- void [Ecriture\\_base\\_info EIFrontiere\\_pour\\_projection](#) (ofstream &sort)

## Membres hérités additionnels

### 6.322.1 Documentation des fonctions membres

#### 6.322.1.1 Affiche()

```
void FrontQuadQuad::Affiche (
    Enum\_dure temp = TEMPS_tdt ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.322.1.2 AutreTangent()

```
void FrontQuadQuad::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.322.1.3 BonCote\_t()

```
bool FrontQuadQuad::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.322.1.4 BonCote\_tdt()

```
bool FrontQuadQuad::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.322.1.5 DernierTangent()

```
Tableau< Coordonnee > * FrontQuadQuad::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.322.1.6 Ecriture\_base\_info\_ElFrontiere\_pour\_projection()

```
void FrontQuadQuad::Ecriture_base_info_ElFrontiere_pour_projection (
    ofstream & sort ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.322.1.7 ElementGeometrique()

```
ElemGeomC0 const & FrontQuadQuad::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [ElFrontiere](#).

### 6.322.1.8 Frontiere()

```
Tableau< ElFrontiere * > & FrontQuadQuad::Frontiere ( ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.322.1.9 InSurf()

```
bool FrontQuadQuad::InSurf (
    const double & eps ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.322.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontQuadQuad::Lecture_base_info_ElFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [ElFrontiere](#).

**6.322.1.11 Metrique()**

```
Met_abstraite * FrontQuadQuad::Metrrique ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.322.1.12 NevezElemFront() [1/2]**

```
ElFrontiere * FrontQuadQuad::NevezElemFront ( ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.322.1.13 NevezElemFront() [2/2]**

```
ElFrontiere * FrontQuadQuad::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.322.1.14 operator=()**

```
ElFrontiere & FrontQuadQuad::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.322.1.15 Phi()**

```
const Vecteur & FrontQuadQuad::Phi ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.322.1.16 Ref()**

```
Coordonnee FrontQuadQuad::Ref ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.322.1.17 Tangent()**

```
void FrontQuadQuad::Tangent (
    const Coordonnee & M,
    Coordonnee & M1,
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.322.1.18 TangentRef()

```
void FrontQuadQuad::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.322.1.19 TypeFrontiere()

```
string FrontQuadQuad::TypeFrontiere ( ) const [virtual]
```

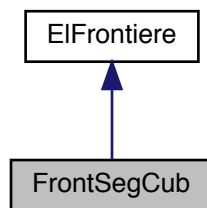
Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuadQuad.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontQuadQuad.cc

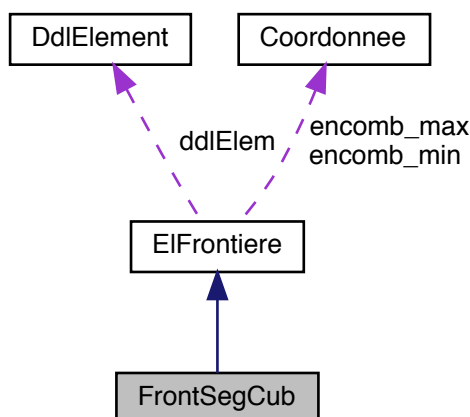
## 6.323 Référence de la classe FrontSegCub

Graphe d'héritage de FrontSegCub:





Graphe de collaboration de FrontSegCub:



## Fonctions membres publiques

- **FrontSegCub** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **FrontSegCub** (const [FrontSegCub](#) &a)
- **EIFFrontiere** & **operator=** (const [EIFFrontiere](#) &a)
- string **TypeFrontiere** () const
- **ElemGeomC0** const & **ElementGeometrique** () const
- **EIFFrontiere** \* **NevezElemFront** () const
- **EIFFrontiere** \* **NevezElemFront** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- **Coordonnee** Ref ()
- void **TangentRef** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **Tangent** (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **AutreTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- bool **InSurf** (const double &eps) const
- [Tableau](#)< [Coordonnee](#) > \* **DernierTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)
- bool **BonCote\_t** (const [Coordonnee](#) &a, double &r) const
- bool **BonCote\_tdt** (const [Coordonnee](#) &a, double &r) const
- const [Vecteur](#) & **Phi** ()
- void **Affiche** ([Enum\\_dure](#) temp=TEMPS\_tdt) const
- [Tableau](#)< [EIFFrontiere](#) \* > & **Frontiere** ()
- **Met\_abstraite** \* **Metrique** ()
- double **LongueurApprox** ()
- void **Lecture\_base\_info{EIFFrontiere\_pour\_projection}** (ifstream &ent)
- void **Ecriture\_base\_info{EIFFrontiere\_pour\_projection}** (ofstream &sort)

## Membres hérités additionnels

### 6.323.1 Documentation des fonctions membres

#### 6.323.1.1 Affiche()

```
void FrontSegCub::Affiche (
    Enum\_dure temp = TEMPS_tdt ) const [virtual]
```

Implémente [EIFFrontiere](#).

### 6.323.1.2 AutreTangent()

```
void FrontSegCub::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.323.1.3 BonCote\_t()

```
bool FrontSegCub::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.323.1.4 BonCote\_tdt()

```
bool FrontSegCub::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.323.1.5 DernierTangent()

```
Tableau< Coordonnee > * FrontSegCub::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.323.1.6 Ecriture\_base\_info{EIFrontiere\_pour\_projection}()

```
void FrontSegCub::Ecriture_base_info{EIFrontiere_pour_projection} (
    ofstream & sort ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.323.1.7 ElementGeometrique()

```
ElemGeomC0 const & FrontSegCub::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [EIFrontiere](#).

### 6.323.1.8 Frontiere()

```
Tableau<{EIFrontiere} * > & FrontSegCub::Frontiere ( ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.323.1.9 InSurf()

```
bool FrontSegCub::InSurf (
    const double & eps ) const [virtual]
```

Implémente [EIFrontiere](#).

**6.323.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontSegCub::Lecture_base_info_ElFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [ElFrontiere](#).

**6.323.1.11 LongueurApprox()**

```
double FrontSegCub::LongueurApprox ( ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.323.1.12 Metrique()**

```
Met_abstraite * FrontSegCub::Metrrique ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.323.1.13 NevezElemFront() [1/2]**

```
ElFrontiere * FrontSegCub::NevezElemFront ( ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.323.1.14 NevezElemFront() [2/2]**

```
ElFrontiere * FrontSegCub::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.323.1.15 operator=()**

```
ElFrontiere & FrontSegCub::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.323.1.16 Phi()**

```
const Vecteur & FrontSegCub::Phi ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.323.1.17 Ref()**

```
Coordonnee FrontSegCub::Ref ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.323.1.18 Tangent()**

```
void FrontSegCub::Tangent (
    const Coordonnee & M,
    Coordonnee & M1,
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

#### 6.323.1.19 TangentRef()

```
void FrontSegCub::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

#### 6.323.1.20 TypeFrontiere()

```
string FrontSegCub::TypeFrontiere ( ) const [virtual]
```

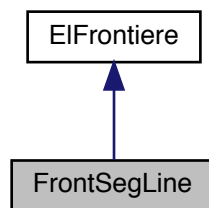
Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

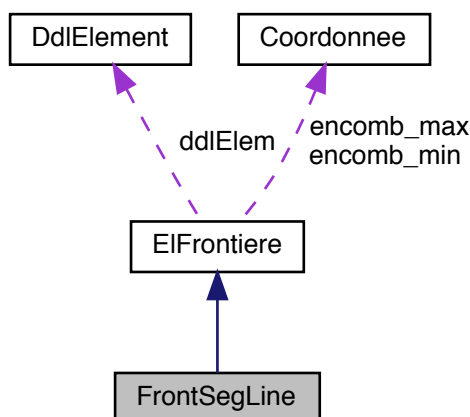
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSeg↵  
Cub.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSeg↵  
Cub.cc

## 6.324 Référence de la classe FrontSegLine

Graphe d'héritage de FrontSegLine:



Graphe de collaboration de FrontSegLine:



## Fonctions membres publiques

- **FrontSegLine** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **FrontSegLine** (const [FrontSegLine](#) &a)
- **EIFrontiere** & **operator=** (const [EIFrontiere](#) &a)
- string **TypeFrontiere** () const
- **ElemGeomC0** const & **ElementGeometrique** () const
- **EIFrontiere** \* **NevezElemFront** () const
- **EIFrontiere** \* **NevezElemFront** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- **Coordonnee** Ref ()
- void **TangentRef** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **Tangent** (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **AutreTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- bool **InSurf** (const double &eps) const
- [Tableau](#)< [Coordonnee](#) > \* **DernierTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)
- bool **BonCote\_t** (const [Coordonnee](#) &a, double &r) const
- bool **BonCote\_tdt** (const [Coordonnee](#) &a, double &r) const
- const [Vecteur](#) & **Phi** ()
- void **Affiche** ([Enum\\_dure](#) temp=TEMPS\_tdt) const
- [Tableau](#)< [EIFrontiere](#) \* > & **Frontiere** ()
- [Met\\_abstraite](#) \* **Metrique** ()
- double **LongueurApprox** ()
- void **Lecture\_base\_info EIFrontiere\_pour\_projection** (ifstream &ent)
- void **Ecriture\_base\_info EIFrontiere\_pour\_projection** (ofstream &sort)

## Membres hérités additionnels

### 6.324.1 Documentation des fonctions membres

#### 6.324.1.1 Affiche()

```
void FrontSegLine::Affiche (
    Enum\_dure temp = TEMPS_tdt ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.324.1.2 AutreTangent()

```
void FrontSegLine::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.324.1.3 BonCote\_t()

```
bool FrontSegLine::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.324.1.4 BonCote\_tdt()

```
bool FrontSegLine::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.324.1.5 DernierTangent()

```
Tableau< Coordonnee > * FrontSegLine::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.324.1.6 Ecriture\_base\_info\_ElFrontiere\_pour\_projection()

```
void FrontSegLine::Ecriture_base_info_ElFrontiere_pour_projection (
    ofstream & sort ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.324.1.7 ElementGeometrique()

```
ElemGeomC0 const & FrontSegLine::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [ElFrontiere](#).

### 6.324.1.8 Frontiere()

```
Tableau< ElFrontiere * > & FrontSegLine::Frontiere ( ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.324.1.9 InSurf()

```
bool FrontSegLine::InSurf (
    const double & eps ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.324.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontSegLine::Lecture_base_info_ElFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [ElFrontiere](#).

**6.324.1.11 LongueurApprox()**

```
double FrontSegLine::LongueurApprox ( ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.324.1.12 Metrique()**

```
Met_abstraite * FrontSegLine::Metrrique ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.324.1.13 NevezElemFront() [1/2]**

```
ElFrontiere * FrontSegLine::NevezElemFront ( ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.324.1.14 NevezElemFront() [2/2]**

```
ElFrontiere * FrontSegLine::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.324.1.15 operator=()**

```
ElFrontiere & FrontSegLine::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.324.1.16 Phi()**

```
const Vecteur & FrontSegLine::Phi ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.324.1.17 Ref()**

```
Coordonnee FrontSegLine::Ref ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.324.1.18 Tangent()**

```
void FrontSegLine::Tangent (
    const Coordonnee & M,
    Coordonnee & M1,
    Droite & dr,
    Plan & p1,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

#### 6.324.1.19 TangentRef()

```
void FrontSegLine::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

#### 6.324.1.20 TypeFrontiere()

```
string FrontSegLine::TypeFrontiere ( ) const [virtual]
```

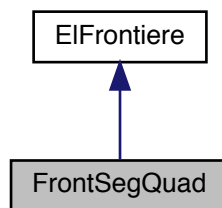
Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSegLine.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSegLine.cc

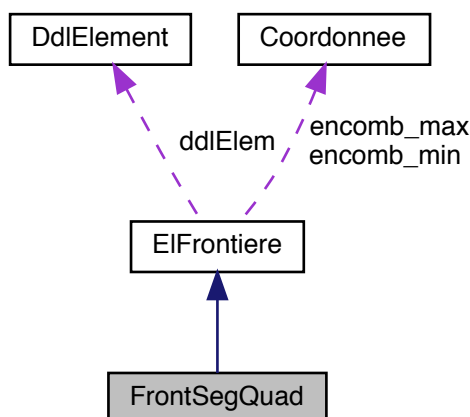
## 6.325 Référence de la classe FrontSegQuad

Graphe d'héritage de FrontSegQuad:





Graphique de collaboration de FrontSegQuad:



## Fonctions membres publiques

- **FrontSegQuad** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **FrontSegQuad** (const [FrontSegQuad](#) &a)
- **EIFrontiere** & **operator=** (const [EIFrontiere](#) &a)
- string [TypeFrontiere](#) () const
- [ElemGeomC0](#) const & [ElementGeometrique](#) () const
- [EIFrontiere](#) \* [NevezElemFront](#) () const
- [EIFrontiere](#) \* [NevezElemFront](#) (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- [Coordonnee](#) Ref ()
- void [TangentRef](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- void [Tangent](#) (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)
- void [AutreTangent](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- bool [InSurf](#) (const double &eps) const
- [Tableau](#)< [Coordonnee](#) > \* [DernierTangent](#) ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)
- bool [BonCote\\_t](#) (const [Coordonnee](#) &a, double &r) const
- bool [BonCote\\_tdt](#) (const [Coordonnee](#) &a, double &r) const
- const [Vecteur](#) & [Phi](#) ()
- void [Affiche](#) ([Enum\\_dure](#) temp=TEMPS\_tdt) const
- [Tableau](#)< [EIFrontiere](#) \* > & [Frontiere](#) ()
- [Met\\_abstraite](#) \* [Metrique](#) ()
- double [LongueurApprox](#) ()
- void [Lecture\\_base\\_info EIFrontiere\\_pour\\_projection](#) (ifstream &ent)
- void [Ecriture\\_base\\_info EIFrontiere\\_pour\\_projection](#) (ofstream &sort)

## Membres hérités additionnels

### 6.325.1 Documentation des fonctions membres

#### 6.325.1.1 Affiche()

```
void FrontSegQuad::Affiche (
    Enum\_dure temp = TEMPS_tdt ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.325.1.2 AutreTangent()

```
void FrontSegQuad::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.325.1.3 BonCote\_t()

```
bool FrontSegQuad::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.325.1.4 BonCote\_tdt()

```
bool FrontSegQuad::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.325.1.5 DernierTangent()

```
Tableau< Coordonnee > * FrontSegQuad::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.325.1.6 Ecriture\_base\_info\_ElFrontiere\_pour\_projection()

```
void FrontSegQuad::Ecriture_base_info_ElFrontiere_pour_projection (
    ofstream & sort ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.325.1.7 ElementGeometrique()

```
ElemGeomC0 const & FrontSegQuad::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [ElFrontiere](#).

### 6.325.1.8 Frontiere()

```
Tableau< ElFrontiere * > & FrontSegQuad::Frontiere ( ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.325.1.9 InSurf()

```
bool FrontSegQuad::InSurf (
    const double & eps ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.325.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontSegQuad::Lecture_base_info_ElFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [ElFrontiere](#).

**6.325.1.11 LongueurApprox()**

```
double FrontSegQuad::LongueurApprox ( ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.325.1.12 Metrique()**

```
Met_abstraite * FrontSegQuad::Metrrique ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.325.1.13 NevezElemFront() [1/2]**

```
ElFrontiere * FrontSegQuad::NevezElemFront ( ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.325.1.14 NevezElemFront() [2/2]**

```
ElFrontiere * FrontSegQuad::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.325.1.15 operator=()**

```
ElFrontiere & FrontSegQuad::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.325.1.16 Phi()**

```
const Vecteur & FrontSegQuad::Phi ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.325.1.17 Ref()**

```
Coordonnee FrontSegQuad::Ref ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.325.1.18 Tangent()**

```
void FrontSegQuad::Tangent (
    const Coordonnee & M,
    Coordonnee & M1,
    Droite & dr,
    Plan & p1,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

#### 6.325.1.19 TangentRef()

```
void FrontSegQuad::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

#### 6.325.1.20 TypeFrontiere()

```
string FrontSegQuad::TypeFrontiere ( ) const [virtual]
```

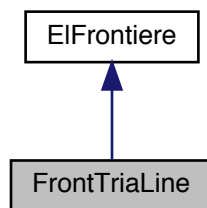
Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

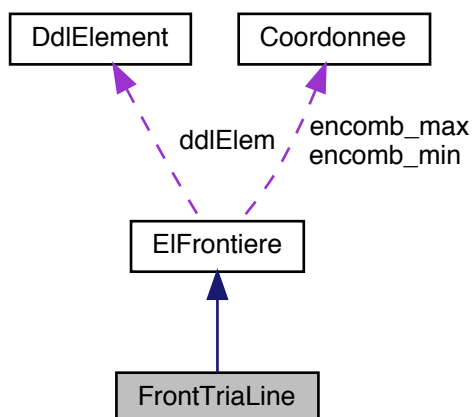
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSegQuad.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Ligne/FrontSegQuad.cc

## 6.326 Référence de la classe FrontTriaLine

Graphe d'héritage de FrontTriaLine:



Graphe de collaboration de FrontTriaLine:



## Fonctions membres publiques

- **FrontTriaLine** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **FrontTriaLine** (const [FrontTriaLine](#) &a)
- **EIFrontiere** & **operator=** (const [EIFrontiere](#) &a)
- string **TypeFrontiere** () const
- **ElemGeomC0** const & **ElementGeometrique** () const
- **EIFrontiere** \* **NevezElemFront** () const
- **EIFrontiere** \* **NevezElemFront** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- **Coordonnee** Ref ()
- void **TangentRef** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **Tangent** (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **AutreTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- bool **InSurf** (const double &eps) const
- [Tableau](#)< [Coordonnee](#) > \* **DernierTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)
- bool **BonCote\_t** (const [Coordonnee](#) &a, double &r) const
- bool **BonCote\_tdt** (const [Coordonnee](#) &a, double &r) const
- const [Vecteur](#) & **Phi** ()
- void **Affiche** ([Enum\\_dure](#) temp=TEMPS\_tdt) const
- [Tableau](#)< [EIFrontiere](#) \* > & **Frontiere** ()
- **Met\_abstraite** \* **Metrique** ()
- void **Lecture\_base\_info{EIFrontiere\_pour\_projection}** (ifstream &ent)
- void **Ecriture\_base\_info{EIFrontiere\_pour\_projection}** (ofstream &sort)

## Membres hérités additionnels

### 6.326.1 Documentation des fonctions membres

#### 6.326.1.1 Affiche()

```
void FrontTriaLine::Affiche (
    Enum\_dure temp = TEMPS_tdt ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.326.1.2 AutreTangent()

```
void FrontTriaLine::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.326.1.3 BonCote\_t()

```
bool FrontTriaLine::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.326.1.4 BonCote\_tdt()

```
bool FrontTriaLine::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.326.1.5 DernierTangent()

```
Tableau< Coordonnee > * FrontTriaLine::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.326.1.6 Ecriture\_base\_info\_ElFrontiere\_pour\_projection()

```
void FrontTriaLine::Ecriture_base_info_ElFrontiere_pour_projection (
    ofstream & sort ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.326.1.7 ElementGeometrique()

```
ElemGeomC0 const & FrontTriaLine::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [ElFrontiere](#).

### 6.326.1.8 Frontiere()

```
Tableau< ElFrontiere * > & FrontTriaLine::Frontiere ( ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.326.1.9 InSurf()

```
bool FrontTriaLine::InSurf (
    const double & eps ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.326.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontTriaLine::Lecture_base_info_ElFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [ElFrontiere](#).

**6.326.1.11 Metrique()**

```
Met_abstraite * FrontTriaLine::Metrrique ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.326.1.12 NevezElemFront() [1/2]**

```
ElFrontiere * FrontTriaLine::NevezElemFront ( ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.326.1.13 NevezElemFront() [2/2]**

```
ElFrontiere * FrontTriaLine::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.326.1.14 operator=()**

```
ElFrontiere & FrontTriaLine::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.326.1.15 Phi()**

```
const Vecteur & FrontTriaLine::Phi ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.326.1.16 Ref()**

```
Coordonnee FrontTriaLine::Ref ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.326.1.17 Tangent()**

```
void FrontTriaLine::Tangent (
    const Coordonnee & M,
    Coordonnee & M1,
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.326.1.18 TangentRef()

```
void FrontTriaLine::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.326.1.19 TypeFrontiere()

```
string FrontTriaLine::TypeFrontiere ( ) const [virtual]
```

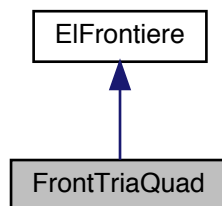
Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontTriaLine.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontTriaLine.cc

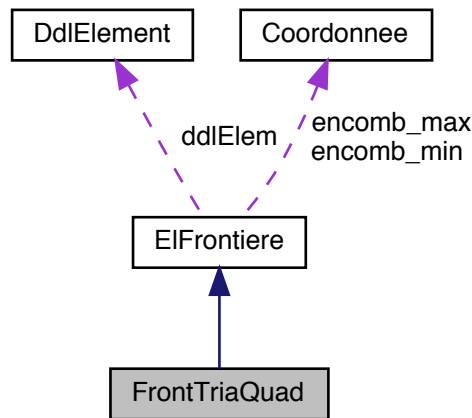
## 6.327 Référence de la classe FrontTriaQuad

Graphe d'héritage de FrontTriaQuad:





Graphe de collaboration de FrontTriaQuad:



## Fonctions membres publiques

- **FrontTriaQuad** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem)
- **FrontTriaQuad** (const [FrontTriaQuad](#) &a)
- **EIFrontiere** & **operator=** (const [EIFrontiere](#) &a)
- string **TypeFrontiere** () const
- **ElemGeomC0** const & **ElementGeometrique** () const
- **EIFrontiere** \* **NevezElemFront** () const
- **EIFrontiere** \* **NevezElemFront** (const [Tableau](#)< [Noeud](#) \* > &tab, const [DdlElement](#) &ddlElem) const
- **Coordonnee** Ref ()
- void **TangentRef** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **Tangent** (const [Coordonnee](#) &M, [Coordonnee](#) &M1, [Droite](#) &dr, [Plan](#) &pl, int &indic)
- void **AutreTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic)
- bool **InSurf** (const double &eps) const
- [Tableau](#)< [Coordonnee](#) > \* **DernierTangent** ([Droite](#) &dr, [Plan](#) &pl, int &indic, bool avec\_var=false)
- bool **BonCote\_t** (const [Coordonnee](#) &a, double &r) const
- bool **BonCote\_tdt** (const [Coordonnee](#) &a, double &r) const
- const [Vecteur](#) & **Phi** ()
- void **Affiche** ([Enum\\_dure](#) temp=TEMPS\_tdt) const
- [Tableau](#)< [EIFrontiere](#) \* > & **Frontiere** ()
- **Met\_abstraite** \* **Metrique** ()
- void **Lecture\_base\_info\_EIFrontiere\_pour\_projection** (ifstream &ent)
- void **Ecriture\_base\_info\_EIFrontiere\_pour\_projection** (ofstream &sort)

## Membres hérités additionnels

### 6.327.1 Documentation des fonctions membres

#### 6.327.1.1 Affiche()

```
void FrontTriaQuad::Affiche (
    Enum\_dure temp = TEMPS_tdt ) const [virtual]
```

Implémente [EIFrontiere](#).

### 6.327.1.2 AutreTangent()

```
void FrontTriaQuad::AutreTangent (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.327.1.3 BonCote\_t()

```
bool FrontTriaQuad::BonCote_t (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.327.1.4 BonCote\_tdt()

```
bool FrontTriaQuad::BonCote_tdt (
    const Coordonnee & a,
    double & r ) const [virtual]
```

Implémente [ElFrontiere](#).

### 6.327.1.5 DernierTangent()

```
Tableau< Coordonnee > * FrontTriaQuad::DernierTangent (
    Droite & dr,
    Plan & pl,
    int & indic,
    bool avec_var = false ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.327.1.6 Ecriture\_base\_info\_ElFrontiere\_pour\_projection()

```
void FrontTriaQuad::Ecriture_base_info_ElFrontiere_pour_projection (
    ofstream & sort ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.327.1.7 ElementGeometrique()

```
ElemGeomC0 const & FrontTriaQuad::ElementGeometrique ( ) const [inline], [virtual]
```

Implémente [ElFrontiere](#).

### 6.327.1.8 Frontiere()

```
Tableau< ElFrontiere * > & FrontTriaQuad::Frontiere ( ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.327.1.9 InSurf()

```
bool FrontTriaQuad::InSurf (
    const double & eps ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.327.1.10 Lecture\_base\_info\_ElFrontiere\_pour\_projection()**

```
void FrontTriaQuad::Lecture_base_info_ElFrontiere_pour_projection (
    ifstream & ent ) [virtual]
```

Implémente [ElFrontiere](#).

**6.327.1.11 Metrique()**

```
Met_abstraite * FrontTriaQuad::Metrrique ( ) [inline], [virtual]
```

Implémente [ElFrontiere](#).

**6.327.1.12 NevezElemFront() [1/2]**

```
ElFrontiere * FrontTriaQuad::NevezElemFront ( ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.327.1.13 NevezElemFront() [2/2]**

```
ElFrontiere * FrontTriaQuad::NevezElemFront (
    const Tableau< Noeud * > & tab,
    const DdlElement & ddlElem ) const [virtual]
```

Implémente [ElFrontiere](#).

**6.327.1.14 operator=()**

```
ElFrontiere & FrontTriaQuad::operator= (
    const ElFrontiere & a ) [virtual]
```

Réimplémentée à partir de [ElFrontiere](#).

**6.327.1.15 Phi()**

```
const Vecteur & FrontTriaQuad::Phi ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.327.1.16 Ref()**

```
Coordonnee FrontTriaQuad::Ref ( ) [virtual]
```

Implémente [ElFrontiere](#).

**6.327.1.17 Tangent()**

```
void FrontTriaQuad::Tangent (
    const Coordonnee & M,
    Coordonnee & M1,
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [ElFrontiere](#).

### 6.327.1.18 TangentRef()

```
void FrontTriaQuad::TangentRef (
    Droite & dr,
    Plan & pl,
    int & indic ) [virtual]
```

Implémente [EIFrontiere](#).

### 6.327.1.19 TypeFrontiere()

```
string FrontTriaQuad::TypeFrontiere ( ) const [virtual]
```

Implémente [EIFrontiere](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontTriaQuad.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Surface/FrontTriaQuad.cc

## 6.328 Référence de la classe LesCondLim::Gene\_asso

### Fonctions membres publiques

- **Gene\_asso** (Enum\_ddl ty\_prin, const [Tableau](#)< int > &pointes)
- **Gene\_asso** (const [Gene\\_asso](#) &a)
- **Gene\_asso & operator=** (const [Gene\\_asso](#) &a)
- **bool operator==** (const [Gene\\_asso](#) &a) const
- **bool operator!=** (const [Gene\\_asso](#) &a) const

### Attributs publics

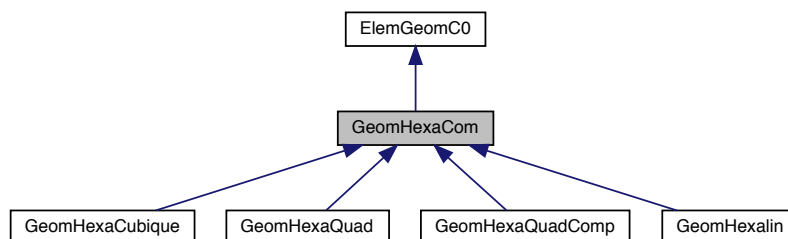
- Enum\_ddl **ty\_prin**
- [Tableau](#)< int > **pointe**

La documentation de cette classe a été générée à partir du fichier suivant :

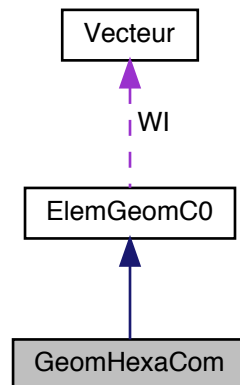
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesCondLim.h

## 6.329 Référence de la classe GeomHexaCom

Graphe d'héritage de GeomHexaCom:



Graphe de collaboration de `GeomHexaCom`:



### Fonctions membres publiques

- `GeomHexaCom` (int nbi, int nbe, `Enum_interpol` interpol)
- `GeomHexaCom` (const `GeomHexaCom` &a)
- bool `Interieur` (const `Coordonnee` &M)
- `Coordonnee Maxi_Coor_dans_directionGM` (const `Coordonnee` &M)

### Fonctions membres protégées

- void `Calcul_extrapol` (int nbi)

### Membres hérités additionnels

#### 6.329.1 Documentation des fonctions membres

##### 6.329.1.1 `Interieur()`

```
bool GeomHexaCom::Interieur (
    const Coordonnee & M ) [virtual]
```

Implémente `ElemGeomC0`.

##### 6.329.1.2 `Maxi_Coor_dans_directionGM()`

```
Coordonnee GeomHexaCom::Maxi_Coor_dans_directionGM (
    const Coordonnee & M ) [virtual]
```

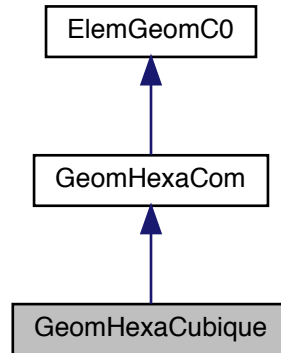
Réimplémentée à partir de `ElemGeomC0`.

La documentation de cette classe a été générée à partir du fichier suivant :

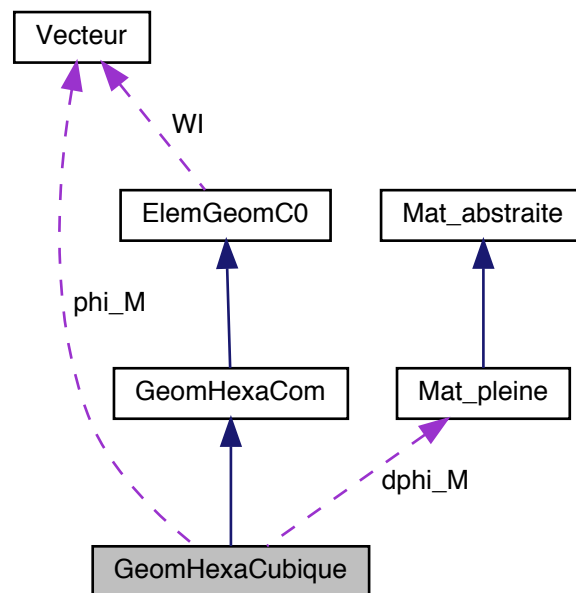
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexaCom.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexaCom.cc`

### 6.330 Référence de la classe `GeomHexaCubique`

Graphe d'héritage de `GeomHexaCubique`:



Graphe de collaboration de `GeomHexaCubique`:



#### Fonctions membres publiques

- `GeomHexaCubique` (int nbi=8)
- `GeomHexaCubique` (const [GeomHexaCubique](#) &a)
- `ElemGeomC0 * newElemGeomC0` ([ElemGeomC0](#) \*pt)
- const `Vecteur & Phi` (const [Coordonnee](#) &M)
- const `Mat_pleine & Dphi` (const [Coordonnee](#) &M)

## Fonctions membres protégées

- `double & DPHI` (int i, int j, int k)
- `double & PHI` (int i, int j)
- `void Phiphi` ()
- `void DphiDphi` ()
- `void Calcul_extrapol` (int nbi)

## Attributs protégés

- `Vecteur phi_M`
- `Mat_pleine dphi_M`

## Membres hérités additionnels

### 6.330.1 Documentation des fonctions membres

#### 6.330.1.1 `Dphi()`

```
const Mat_pleine & GeomHexaCubique::Dphi (  
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.330.1.2 `newElemGeomC0()`

```
ElemGeomC0 * GeomHexaCubique::newElemGeomC0 (  
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.330.1.3 `Phi()`

```
const Vecteur & GeomHexaCubique::Phi (  
    const Coordonnee & M ) [virtual]
```

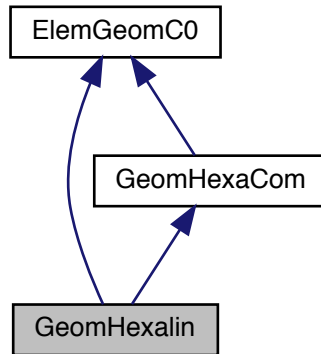
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

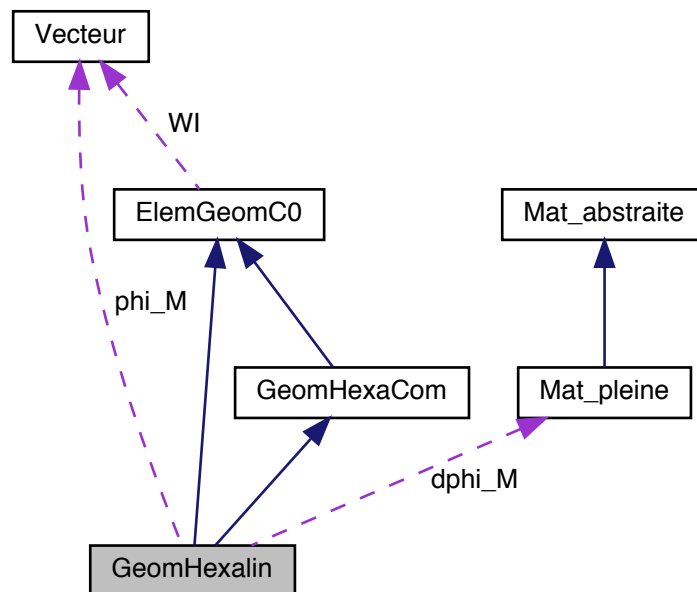
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔HexaCubique.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔HexaCubique.cc`

### 6.331 Référence de la classe GeomHexalin

Graphe d'héritage de GeomHexalin:



Graphe de collaboration de GeomHexalin:



#### Fonctions membres publiques

- **GeomHexalin** (int nbi=8)
- **GeomHexalin** (const [GeomHexalin](#) &a)
- [ElemGeomC0](#) \* **newElemGeomC0** ([ElemGeomC0](#) \*pt)
- const [Vecteur](#) & **Phi** (const [Coordonnee](#) &M)



- const `Mat_pleine` & `Dphi` (const `Coordonnee` &M)
- `GeomHexalin` (const `GeomHexalin` &a)
- const `Vecteur` & `Phi` (const `Coordonnee` &M)
- const `Mat_pleine` & `Dphi` (const `Coordonnee` &M)
- bool `Interieur` (const `Coordonnee` &M)

### Fonctions membres protégées

- double & `DPHI` (int i, int j, int k)
- double & `PHI` (int i, int j)
- void `Phiphi` ()
- void `DphiDphi` ()
- void `Calcul_extrapol` (int nbi)
- double & `DPHI` (int i, int j, int k)
- double & `PHI` (int i, int j)
- void `Phiphi` ()
- void `DphiDphi` ()

### Attributs protégés

- `Vecteur phi_M`
- `Mat_pleine dphi_M`

### Membres hérités additionnels

#### 6.331.1 Documentation des fonctions membres

##### 6.331.1.1 `Dphi()` [1/2]

```
const Mat_pleine & GeomHexalin::Dphi (
    const Coordonnee & M ) [virtual]
Implémente ElemGeomC0.
```

##### 6.331.1.2 `Dphi()` [2/2]

```
const Mat_pleine & GeomHexalin::Dphi (
    const Coordonnee & M ) [virtual]
Implémente ElemGeomC0.
```

##### 6.331.1.3 `Interieur()`

```
bool GeomHexalin::Interieur (
    const Coordonnee & M ) [virtual]
Implémente ElemGeomC0.
```

##### 6.331.1.4 `newElemGeomC0()`

```
ElemGeomC0 * GeomHexalin::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
Implémente ElemGeomC0.
```

### 6.331.1.5 Phi() [1/2]

```
const Vecteur & GeomHexalin::Phi (  
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.331.1.6 Phi() [2/2]

```
const Vecteur & GeomHexalin::Phi (  
    const Coordonnee & M ) [virtual]
```

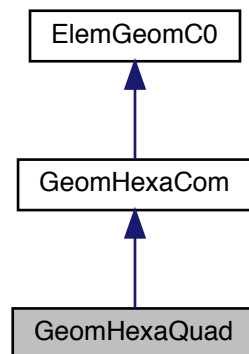
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

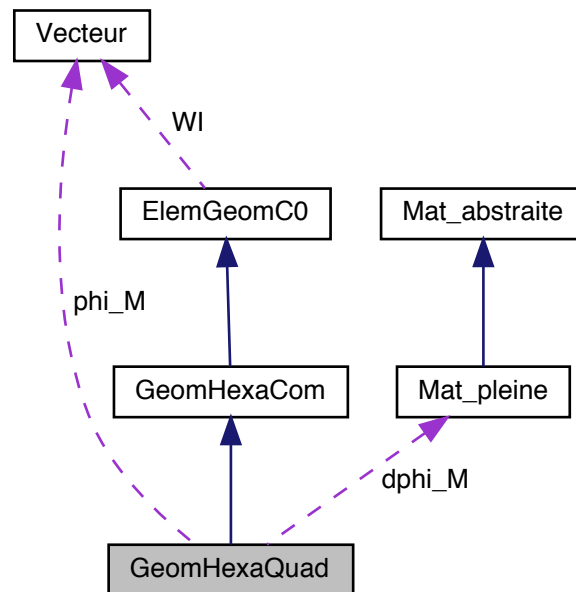
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔Hexalin.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔Tetra.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔Hexalin.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔Hexalin1.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔Hexalin2.cc

## 6.332 Référence de la classe GeomHexaQuad

Graphe d'héritage de GeomHexaQuad:



Graphe de collaboration de `GeomHexaQuad`:



### Fonctions membres publiques

- `GeomHexaQuad` (int nbi=8)
- `GeomHexaQuad` (const `GeomHexaQuad` &a)
- `ElemGeomC0 * newElemGeomC0` (`ElemGeomC0 *pt`)
- const `Vecteur` & `Phi` (const `Coordonnee` &M)
- const `Mat_pleine` & `Dphi` (const `Coordonnee` &M)

### Fonctions membres protégées

- double & `DPHI` (int i, int j, int k)
- double & `PHI` (int i, int j)
- void `Phiphi` ()
- void `DphiDphi` ()
- void `Calcul_extrapol` (int nbi)

### Attributs protégés

- `Vecteur` `phi_M`
- `Mat_pleine` `dphi_M`

### Membres hérités additionnels

#### 6.332.1 Documentation des fonctions membres

##### 6.332.1.1 `Dphi()`

```
const Mat_pleine & GeomHexaQuad::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.332.1.2 newElemGeomC0()

```
ElemGeomC0 * GeomHexaQuad::newElemGeomC0 (  
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.332.1.3 Phi()

```
const Vecteur & GeomHexaQuad::Phi (  
    const Coordonnee & M ) [virtual]
```

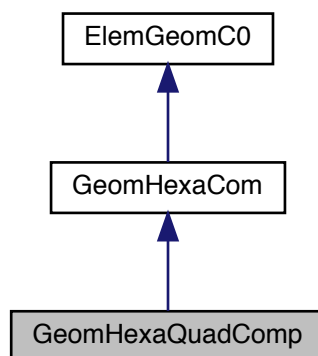
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

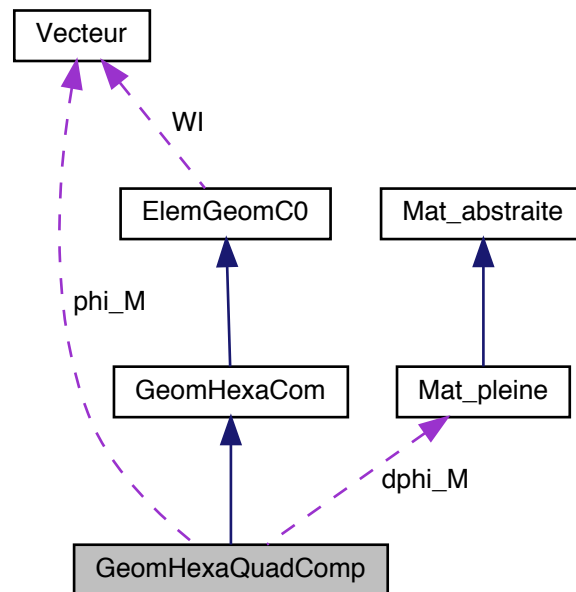
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔HexaQuad.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔HexaQuad.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔HexaQuad2.cc

## 6.333 Référence de la classe GeomHexaQuadComp

Grappe d'héritage de GeomHexaQuadComp:



Graphe de collaboration de `GeomHexaQuadComp`:



### Fonctions membres publiques

- `GeomHexaQuadComp` (int nbi=8)
- `GeomHexaQuadComp` (const `GeomHexaQuadComp` &a)
- `ElemGeomC0 * newElemGeomC0` (`ElemGeomC0 *pt`)
- const `Vecteur & Phi` (const `Coordonnee` &M)
- const `Mat_pleine & Dphi` (const `Coordonnee` &M)

### Fonctions membres protégées

- double & `DPHI` (int i, int j, int k)
- double & `PHI` (int i, int j)
- void `Phiphi` ()
- void `DphiDphi` ()
- void `Calcul_extrapol` (int nbi)

### Attributs protégés

- `Vecteur phi_M`
- `Mat_pleine dphi_M`

### Membres hérités additionnels

#### 6.333.1 Documentation des fonctions membres

##### 6.333.1.1 `Dphi()`

```
const Mat_pleine & GeomHexaQuadComp::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.333.1.2 newElemGeomC0()

```
ElemGeomC0 * GeomHexaQuadComp::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.333.1.3 Phi()

```
const Vecteur & GeomHexaQuadComp::Phi (
    const Coordonnee & M ) [virtual]
```

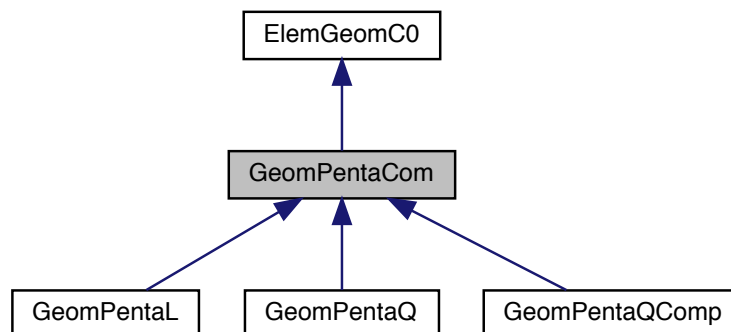
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

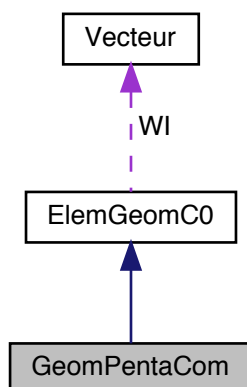
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexaQuadComp.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomHexaQuadComp.cc

## 6.334 Référence de la classe GeomPentaCom

Graphe d'héritage de GeomPentaCom:



Grappe de collaboration de `GeomPentaCom`:



### Fonctions membres publiques

- `GeomPentaCom` (int nbi, int nbn, `Enum_interpol` interpol)
- `GeomPentaCom` (const `GeomPentaCom` &a)
- bool `Interieur` (const `Coordonnee` &M)
- `Coordonnee Maxi_Coor_dans_directionGM` (const `Coordonnee` &M)

### Fonctions membres protégées

- void `Calcul_extrapol` (int nbi)

### Membres hérités additionnels

#### 6.334.1 Documentation des fonctions membres

##### 6.334.1.1 `Interieur()`

```
bool GeomPentaCom::Interieur (
    const Coordonnee & M ) [virtual]
```

Implémente `ElemGeomC0`.

##### 6.334.1.2 `Maxi_Coor_dans_directionGM()`

```
Coordonnee GeomPentaCom::Maxi_Coor_dans_directionGM (
    const Coordonnee & M ) [virtual]
```

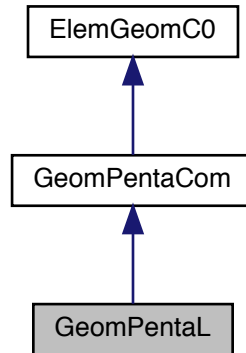
Réimplémentée à partir de `ElemGeomC0`.

La documentation de cette classe a été générée à partir du fichier suivant :

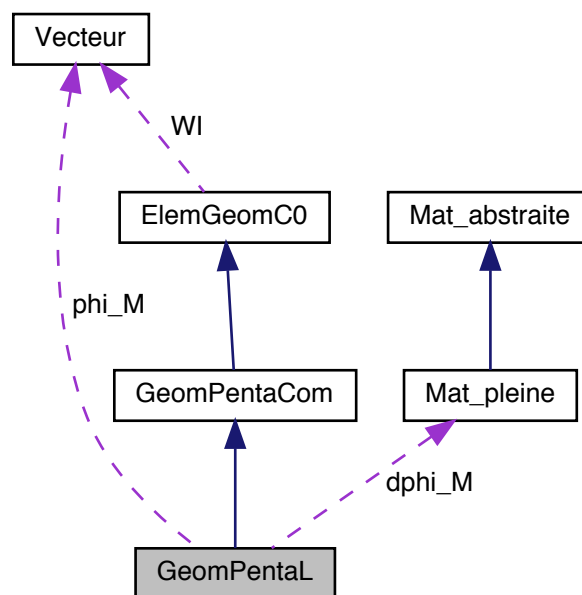
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomPentaCom.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomPentaCom.cc`

## 6.335 Référence de la classe `GeomPentaL`

Graphe d'héritage de `GeomPentaL`:



Graphe de collaboration de `GeomPentaL`:



### Fonctions membres publiques

- `GeomPentaL` (int nbi=2)
- `GeomPentaL` (const `GeomPentaL` &a)
- `ElemGeomC0 * newElemGeomC0` (`ElemGeomC0` \*pt)
- const `Vecteur` & `Phi` (const `Coordonnee` &M)
- const `Mat_pleine` & `Dphi` (const `Coordonnee` &M)



## Fonctions membres protégées

- void `Calcul_extrapol` (int nbi)

## Attributs protégés

- Vecteur `phi_M`
- Mat\_pleine `dphi_M`

## Membres hérités additionnels

### 6.335.1 Documentation des fonctions membres

#### 6.335.1.1 `Dphi()`

```
const Mat_pleine & GeomPentaL::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.335.1.2 `newElemGeomC0()`

```
ElemGeomC0 * GeomPentaL::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.335.1.3 `Phi()`

```
const Vecteur & GeomPentaL::Phi (
    const Coordonnee & M ) [virtual]
```

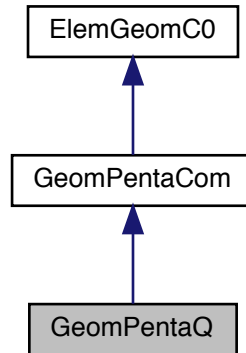
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

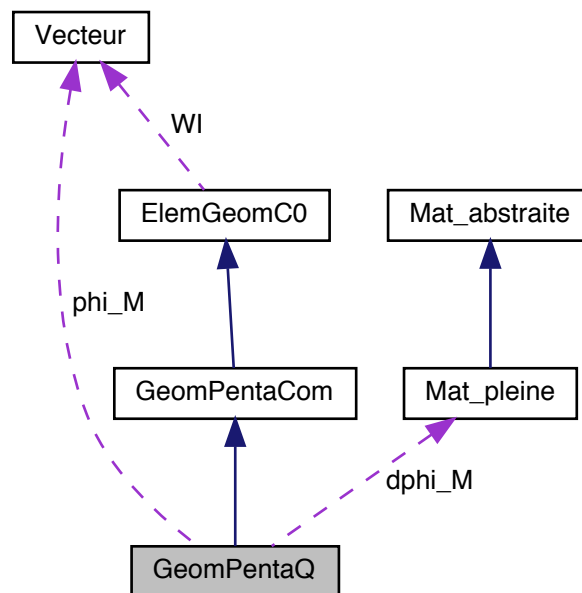
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔PentaL.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔PentaL.cc`

## 6.336 Référence de la classe GeomPentaQ

Graphe d'héritage de GeomPentaQ:



Graphe de collaboration de GeomPentaQ:



### Fonctions membres publiques

- **GeomPentaQ** (int nbi=6)
- **GeomPentaQ** (const [GeomPentaQ](#) &a)
- [ElemGeomC0](#) \* **newElemGeomC0** ([ElemGeomC0](#) \*pt)
- const [Vecteur](#) & **Phi** (const [Coordonnee](#) &M)
- const [Mat\\_pleine](#) & **Dphi** (const [Coordonnee](#) &M)

## Fonctions membres protégées

- double & **DPHI** (int i, int j, int k)
- void **Calcul\_extrapol** (int nbi)

## Attributs protégés

- Vecteur **phi\_M**
- Mat\_pleine **dphi\_M**

## Membres hérités additionnels

### 6.336.1 Documentation des fonctions membres

#### 6.336.1.1 Dphi()

```
const Mat_pleine & GeomPentaQ::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.336.1.2 newElemGeomC0()

```
ElemGeomC0 * GeomPentaQ::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.336.1.3 Phi()

```
const Vecteur & GeomPentaQ::Phi (
    const Coordonnee & M ) [virtual]
```

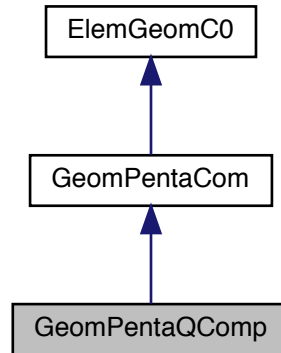
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

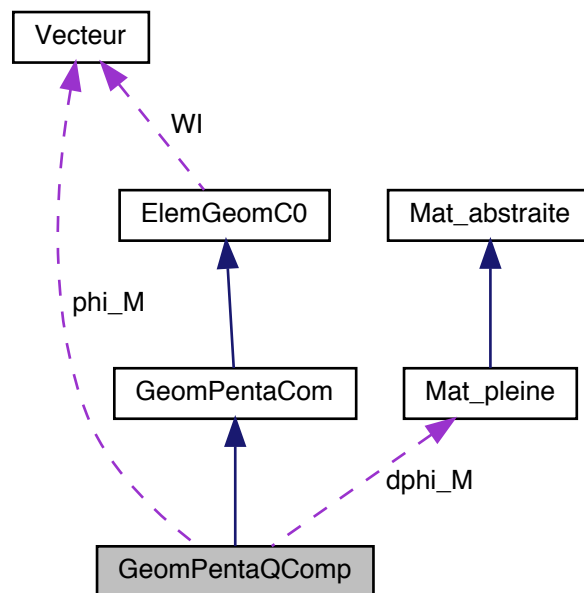
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔PentaQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔PentaQ.cc

### 6.337 Référence de la classe `GeomPentaQComp`

Graphe d'héritage de `GeomPentaQComp`:



Graphe de collaboration de `GeomPentaQComp`:



#### Fonctions membres publiques

- `GeomPentaQComp` (int nbi=6)
- `GeomPentaQComp` (const [GeomPentaQComp](#) &a)
- `ElemGeomC0 * newElemGeomC0` ([ElemGeomC0](#) \*pt)
- const `Vecteur` & `Phi` (const [Coordonnee](#) &M)
- const `Mat_pleine` & `Dphi` (const [Coordonnee](#) &M)

## Fonctions membres protégées

- `double & DPHI` (int i, int j, int k)
- `void Calcul_extrapol` (int nbi)

## Attributs protégés

- `Vecteur phi_M`
- `Mat_pleine dphi_M`

## Membres hérités additionnels

### 6.337.1 Documentation des fonctions membres

#### 6.337.1.1 `Dphi()`

```
const Mat_pleine & GeomPentaQComp::Dphi (  
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.337.1.2 `newElemGeomC0()`

```
ElemGeomC0 * GeomPentaQComp::newElemGeomC0 (  
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.337.1.3 `Phi()`

```
const Vecteur & GeomPentaQComp::Phi (  
    const Coordonnee & M ) [virtual]
```

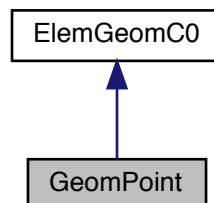
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

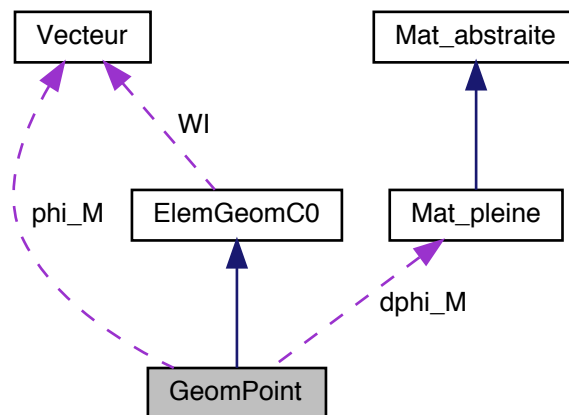
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔PentaQComp.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔PentaQComp.cc`

## 6.338 Référence de la classe `GeomPoint`

Graphe d'héritage de `GeomPoint`:



Graphe de collaboration de `GeomPoint`:



## Fonctions membres publiques

- `GeomPoint` (const `GeomPoint` &a)
- `Tableau< Mat_pleine > & taD2phi` ()
- `ElemGeomC0 * newElemGeomC0` (`ElemGeomC0` \*pt)
- const `Vecteur` & `Phi` (const `Coordonnee` &M)
- const `Mat_pleine` & `Dphi` (const `Coordonnee` &M)
- bool `Interieur` (const `Coordonnee` &M)
- `Coordonnee Maxi_Coor_dans_directionGM` (const `Coordonnee` &M)
- double & `KSI` (int i)

## Attributs protégés

- `Vecteur` `phi_M`
- `Mat_pleine` `dphi_M`
- `Tableau< Mat_pleine >` `tabD2Phi`

## Membres hérités additionnels

### 6.338.1 Documentation des fonctions membres

#### 6.338.1.1 `Dphi()`

```
const Mat_pleine & GeomPoint::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente `ElemGeomC0`.

#### 6.338.1.2 `Interieur()`

```
bool GeomPoint::Interieur (
    const Coordonnee & M ) [virtual]
```

Implémente `ElemGeomC0`.

### 6.338.1.3 `Maxi_Coor_dans_directionGM()`

```
Coordonnee GeomPoint::Maxi_Coor_dans_directionGM (
    const Coordonnee & M ) [virtual]
```

Réimplémentée à partir de [ElemGeomC0](#).

### 6.338.1.4 `newElemGeomC0()`

```
ElemGeomC0 * GeomPoint::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.338.1.5 `Phi()`

```
const Vecteur & GeomPoint::Phi (
    const Coordonnee & M ) [virtual]
```

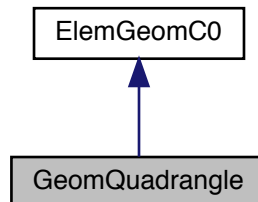
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

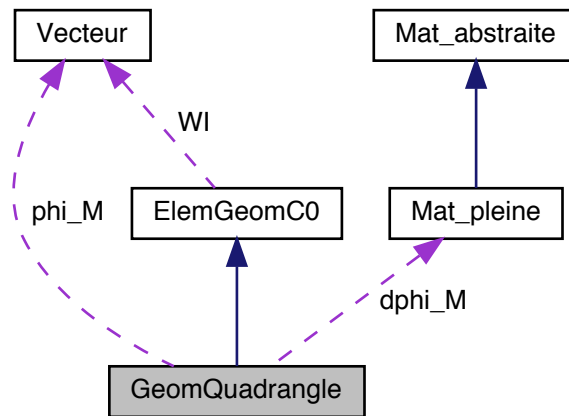
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/Point/GeomPoint.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/Point/GeomPoint.cc`

## 6.339 Référence de la classe `GeomQuadrangle`

Grappe d'héritage de `GeomQuadrangle`:



Graphe de collaboration de GeomQuadrangle:



## Fonctions membres publiques

- **GeomQuadrangle** (int nbi=4, int nbne=4, int sans\_extrapole=0)
- **GeomQuadrangle** (const [GeomQuadrangle](#) &a)
- [ElemGeomC0](#) \* **newElemGeomC0** ([ElemGeomC0](#) \*pt)
- const [Vecteur](#) & **Phi** (const [Coordonnee](#) &M)
- const [Mat\\_pleine](#) & **Dphi** (const [Coordonnee](#) &M)
- bool **Interieur** (const [Coordonnee](#) &M)
- [Coordonnee](#) **Maxi\_Coor\_dans\_directionGM** (const [Coordonnee](#) &M)

## Fonctions membres protégées

- void **Calcul\_extrapol** (int nbi)

## Attributs protégés

- [Vecteur](#) **phi\_M**
- [Mat\\_pleine](#) **dphi\_M**

## Membres hérités additionnels

### 6.339.1 Documentation des fonctions membres

#### 6.339.1.1 Dphi()

```
const Mat\_pleine & GeomQuadrangle::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.339.1.2 Interieur()

```
bool GeomQuadrangle::Interieur (
    const Coordonnee & M ) [virtual]
```



Implémente [ElemGeomC0](#).

#### 6.339.1.3 `Maxi_Coor_dans_directionGM()`

```
Coordonnee GeomQuadrangle::Maxi_Coor_dans_directionGM (
    const Coordonnee & M ) [virtual]
```

Réimplémentée à partir de [ElemGeomC0](#).

#### 6.339.1.4 `newElemGeomC0()`

```
ElemGeomC0 * GeomQuadrangle::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.339.1.5 `Phi()`

```
const Vecteur & GeomQuadrangle::Phi (
    const Coordonnee & M ) [virtual]
```

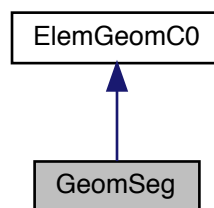
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

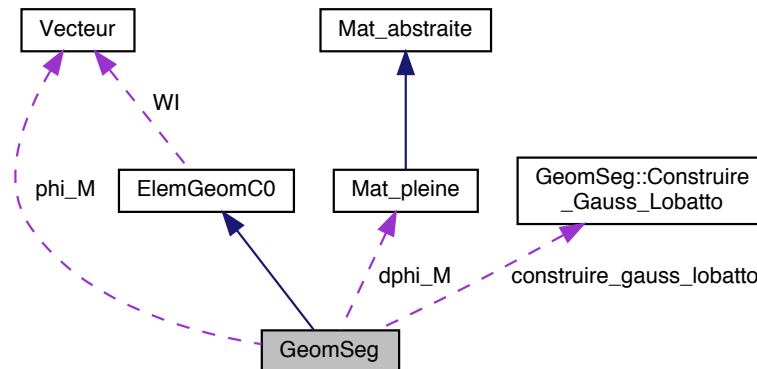
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/surface/Geom↔Quadrangle.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/surface/Geom↔Quadrangle.cc`

## 6.340 Référence de la classe `GeomSeg`

Grappe d'héritage de `GeomSeg`:



Graphe de collaboration de GeomSeg:



## Classes

- class [Construire\\_Gauss\\_Lobatto](#)

## Fonctions membres publiques

- **GeomSeg** (int code\_nbi=1, int nbe=2, [Enum\\_type\\_pt\\_integ](#) type\_pti=PTI\_GAUSS)
- **GeomSeg** (const [GeomSeg](#) &a)
- [Tableau](#)< [Mat\\_pleine](#) > & **taD2phi** ()
- [ElemGeomC0](#) \* **newElemGeomC0** ([ElemGeomC0](#) \*pt)
- const [Vecteur](#) & **Phi** (const [Coordonnee](#) &M)
- const [Mat\\_pleine](#) & **Dphi** (const [Coordonnee](#) &M)
- bool **Interieur** (const [Coordonnee](#) &M)
- [Coordonnee](#) **Maxi\_Coor\_dans\_directionGM** (const [Coordonnee](#) &M)
- double & **KSI** (int i)

## Fonctions membres protégées

- void **Calcul\_extrapol** (int nbi)

## Attributs protégés

- [Tableau](#)< [Mat\\_pleine](#) > **tabD2Phi**
- [Vecteur](#) **phi\_M**
- [Mat\\_pleine](#) **dphi\_M**

## Attributs protégés statiques

- static [Tableau](#)< [Tableau](#)< double > > **ksi\_gl**
- static [Tableau](#)< [Tableau](#)< double > > **wi\_gl**
- static [Construire\\_Gauss\\_Lobatto](#) **construire\_gauss\_lobatto**

## Membres hérités additionnels

### 6.340.1 Documentation des fonctions membres

### 6.340.1.1 Dphi()

```
const Mat_pleine & GeomSeg::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.340.1.2 Interieur()

```
bool GeomSeg::Interieur (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.340.1.3 Maxi\_Coor\_dans\_directionGM()

```
Coordonnee GeomSeg::Maxi_Coor_dans_directionGM (
    const Coordonnee & M ) [virtual]
```

Réimplémentée à partir de [ElemGeomC0](#).

### 6.340.1.4 newElemGeomC0()

```
ElemGeomC0 * GeomSeg::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.340.1.5 Phi()

```
const Vecteur & GeomSeg::Phi (
    const Coordonnee & M ) [virtual]
```

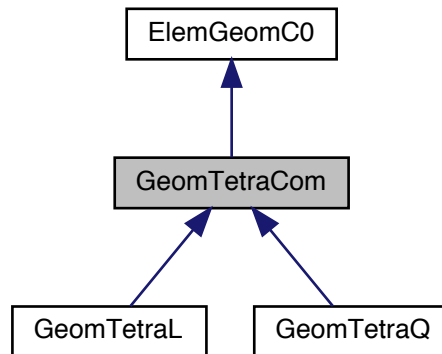
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

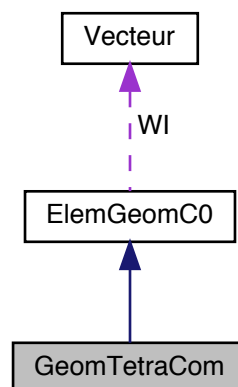
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/Ligne/GeomSeg.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/Ligne/GeomSeg.cc

## 6.341 Référence de la classe `GeomTetraCom`

Graphe d'héritage de `GeomTetraCom`:



Graphe de collaboration de `GeomTetraCom`:



### Fonctions membres publiques

- `GeomTetraCom` (int nbi=1, int nbe=4, `Enum_interpol` interpol=RIEN\_INTERPOL)
- `GeomTetraCom` (const `GeomTetraCom` &a)
- bool `Interieur` (const `Coordonnee` &M)
- `Coordonnee Maxi_Coor_dans_directionGM` (const `Coordonnee` &M)

### Fonctions membres protégées

- void `Calcul_extrapol` (int nbi)

## Membres hérités additionnels

### 6.341.1 Documentation des fonctions membres

#### 6.341.1.1 `Interieur()`

```
bool GeomTetraCom::Interieur (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

#### 6.341.1.2 `Maxi_Coor_dans_directionGM()`

```
Coordonnee GeomTetraCom::Maxi_Coor_dans_directionGM (
    const Coordonnee & M ) [virtual]
```

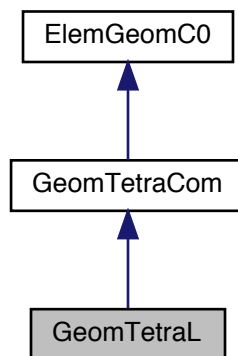
Réimplémentée à partir de [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

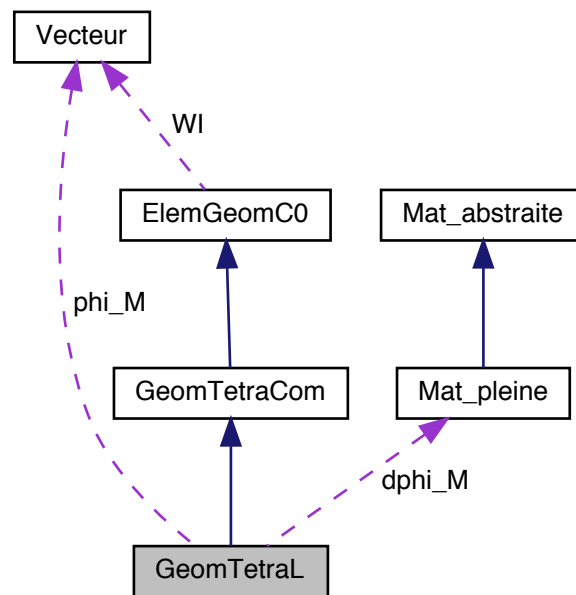
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔TetraCom.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/Geom↔TetraCom.cc`

## 6.342 Référence de la classe `GeomTetraL`

Grappe d'héritage de `GeomTetraL`:



Graphe de collaboration de GeomTetraL:



### Fonctions membres publiques

- `GeomTetraL` (int nbi=1)
- `GeomTetraL` (const `GeomTetraL` &a)
- `ElemGeomC0 * newElemGeomC0` (`ElemGeomC0 *pt`)
- const `Vecteur & Phi` (const `Coordonnee &M`)
- const `Mat_pleine & Dphi` (const `Coordonnee &M`)
- bool `Interieur` (const `Coordonnee &M`)

### Fonctions membres protégées

- void `Calcul_extrapol` (int nbi)

### Attributs protégés

- `Vecteur phi_M`
- `Mat_pleine dphi_M`

### Membres hérités additionnels

#### 6.342.1 Documentation des fonctions membres

##### 6.342.1.1 Dphi()

```
const Mat_pleine & GeomTetraL::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente `ElemGeomC0`.

### 6.342.1.2 `Interieur()`

```
bool GeomTetraL::Interieur (
    const Coordonnee & M ) [virtual]
```

Réimplémentée à partir de [GeomTetraCom](#).

### 6.342.1.3 `newElemGeomC0()`

```
ElemGeomC0 * GeomTetraL::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.342.1.4 `Phi()`

```
const Vecteur & GeomTetraL::Phi (
    const Coordonnee & M ) [virtual]
```

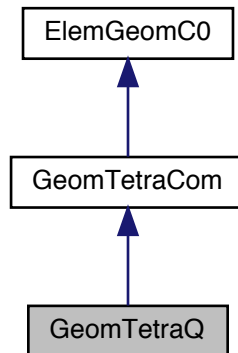
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

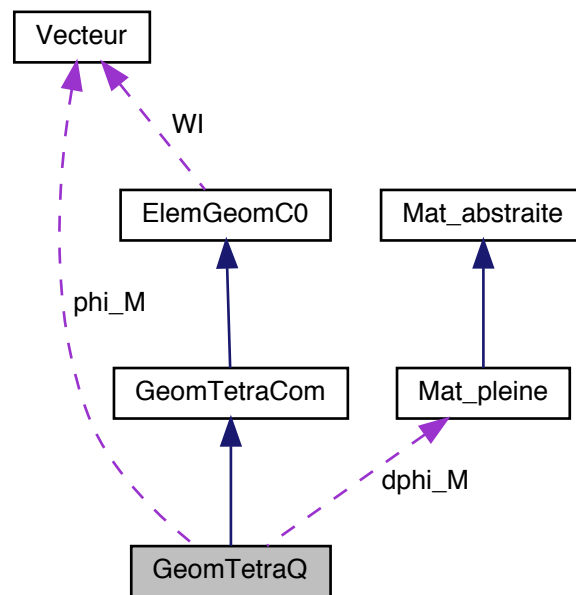
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomTetraL.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomTetraL.cc](#)

## 6.343 Référence de la classe `GeomTetraQ`

Graphe d'héritage de `GeomTetraQ`:



Grphe de collaboration de GeomTetraQ:



### Fonctions membres publiques

- **GeomTetraQ** (int nbi=4)
- **GeomTetraQ** (const [GeomTetraQ](#) &a)
- [ElemGeomC0](#) \* **newElemGeomC0** ([ElemGeomC0](#) \*pt)
- const [Vecteur](#) & **Phi** (const [Coordonnee](#) &M)
- const [Mat\\_pleine](#) & **Dphi** (const [Coordonnee](#) &M)
- bool **Interieur** (const [Coordonnee](#) &M)

### Fonctions membres protégées

- double & **DPHI** (int i, int j, int k)
- double & **PHI** (int i, int j)
- void **Phiphi** ()
- void **DphiDphi** ()
- void **Calcul\_extrapol** (int nbi)

### Attributs protégés

- [Vecteur](#) **phi\_M**
- [Mat\\_pleine](#) **dphi\_M**

### Membres hérités additionnels

#### 6.343.1 Documentation des fonctions membres



### 6.343.1.1 `Dphi()`

```
const Mat_pleine & GeomTetraQ::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.343.1.2 `Interieur()`

```
bool GeomTetraQ::Interieur (
    const Coordonnee & M ) [virtual]
```

Réimplémentée à partir de [GeomTetraCom](#).

### 6.343.1.3 `newElemGeomC0()`

```
ElemGeomC0 * GeomTetraQ::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

### 6.343.1.4 `Phi()`

```
const Vecteur & GeomTetraQ::Phi (
    const Coordonnee & M ) [virtual]
```

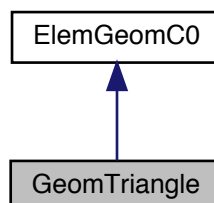
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

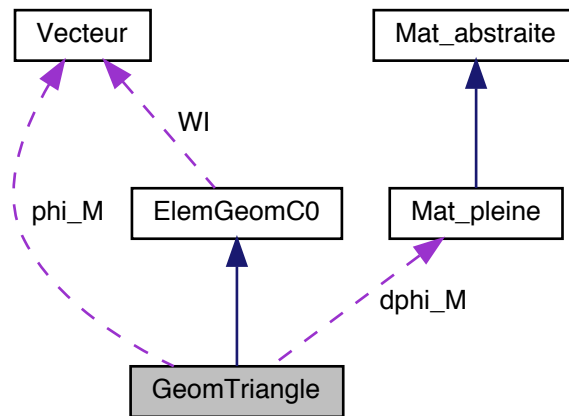
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomTetraQ.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/volume/GeomTetraQ.cc`

## 6.344 Référence de la classe `GeomTriangle`

Graphe d'héritage de `GeomTriangle`:



Graphe de collaboration de GeomTriangle:



## Fonctions membres publiques

- **GeomTriangle** (int nbi=1, int nbne=3)
- **GeomTriangle** (const [GeomTriangle](#) &a)
- [ElemGeomC0](#) \* **newElemGeomC0** ([ElemGeomC0](#) \*pt)
- const [Vecteur](#) & **Phi** (const [Coordonnee](#) &M)
- const [Mat\\_pleine](#) & **Dphi** (const [Coordonnee](#) &M)
- bool **Interieur** (const [Coordonnee](#) &M)
- [Coordonnee](#) **Maxi\_Coor\_dans\_directionGM** (const [Coordonnee](#) &M)

## Fonctions membres protégées

- double & **KSI** (int i)
- double & **ETA** (int i)
- void **Calcul\_extrapol** (int nbi)

## Attributs protégés

- [Vecteur](#) **phi\_M**
- [Mat\\_pleine](#) **dphi\_M**

## Membres hérités additionnels

### 6.344.1 Documentation des fonctions membres

#### 6.344.1.1 Dphi()

```
const Mat\_pleine & GeomTriangle::Dphi (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

**6.344.1.2 Interieur()**

```
bool GeomTriangle::Interieur (
    const Coordonnee & M ) [virtual]
```

Implémente [ElemGeomC0](#).

**6.344.1.3 Maxi\_Coor\_dans\_directionGM()**

```
Coordonnee GeomTriangle::Maxi_Coor_dans_directionGM (
    const Coordonnee & M ) [virtual]
```

Réimplémentée à partir de [ElemGeomC0](#).

**6.344.1.4 newElemGeomC0()**

```
ElemGeomC0 * GeomTriangle::newElemGeomC0 (
    ElemGeomC0 * pt ) [virtual]
```

Implémente [ElemGeomC0](#).

**6.344.1.5 Phi()**

```
const Vecteur & GeomTriangle::Phi (
    const Coordonnee & M ) [virtual]
```

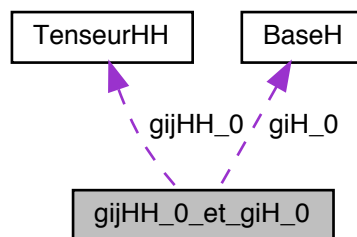
Implémente [ElemGeomC0](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/surface/Geom↔Triangle.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/ElemGeom/surface/Geom↔Triangle.cc

**6.345 Référence de la classe gijHH\_0\_et\_giH\_0**

Graphes de collaboration de gijHH\_0\_et\_giH\_0:

**Fonctions membres publiques**

- `gijHH_0_et_giH_0` (`BaseH *ggiH_0`, `TenseurHH *ggijHH_0`)
- `gijHH_0_et_giH_0` (`const gijHH_0_et_giH_0 &ex`)
- `gijHH_0_et_giH_0 &operator=` (`const gijHH_0_et_giH_0 &ex`)
- `void Mise_a_jour_grandeur` (`BaseH *ggiH_0`, `TenseurHH *ggijHH_0`)

### Attributs publics

- [TenseurHH](#) \* `gijHH_0`
- [BaseH](#) \* `giH_0`

La documentation de cette classe a été générée à partir du fichier suivant :

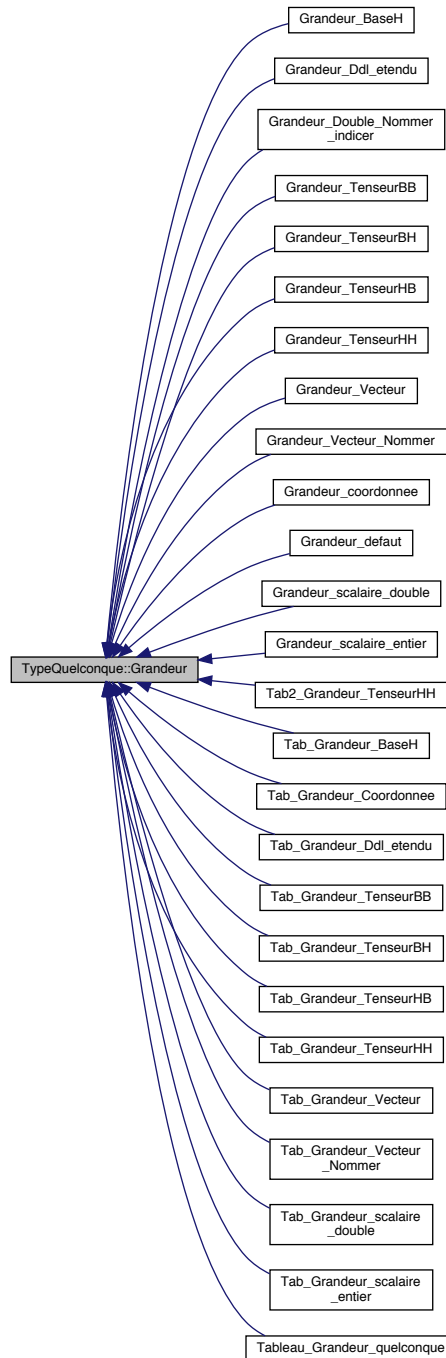
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Met_↔  
abstraite_struc_donnees.h`

## 6.346 Référence de la classe `TypeQuelconque::Grandeur`

définition de la classe virtuelle qui serait déclinée en une grandeur particulière il s'agit ici de la classe conteneur

```
#include <TypeQuelconque.h>
```

Graphe d'héritage de TypeQuelconque::Grandeur:



## Fonctions membres publiques

- virtual `Grandeur * New_idem_grandeur ()` const =0
- virtual `TypeQuelconque::Grandeur & operator= (const TypeQuelconque::Grandeur &a)=0`
- virtual void `Affectation_numerique (const TypeQuelconque::Grandeur &a)=0`
- virtual void `operator+= (const Grandeur &c)=0`
- virtual void `operator-= (const Grandeur &c)=0`
- virtual void `operator*= (double val)=0`
- virtual void `operator/= (double val)=0`

- virtual const string \* **Nom\_ref** () const
- virtual double **GrandeurNumOrdre** (int num) const =0
- virtual int **NbMaxiNumeroOrdre** () const =0
- virtual void **Grandeur\_brut** (ostream &sort, int nbcarr) const =0
- virtual [EnumTypeGrandeur](#) **Type\_grandeurAssocie** () const =0
- virtual [EnumType2Niveau](#) **Type\_structure\_grandeurAssocie** () const =0
- virtual [EnuTypeQuelParticulier](#) **Type\_enumGrandeurParticuliere** () const =0
- virtual void **Change\_repere** (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)=0
- virtual istream & **Lecture\_grandeur** (istream &ent)=0
- virtual ostream & **Ecriture\_grandeur** (ostream &sort) const =0
- virtual void **InitParDefaut** ()=0

## Amis

- istream & **operator**>> (istream &ent, [Grandeur](#) &a)
- ostream & **operator**<< (ostream &sort, const [Grandeur](#) &a)

### 6.346.1 Description détaillée

définition de la classe virtuelle qui serait déclinée en une grandeur particulière il s'agit ici de la classe conteneur  
La documentation de cette classe a été générée à partir du fichier suivant :

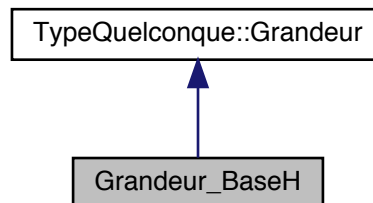
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

### 6.347 Référence de la classe [Grandeur\\_BaseH](#)

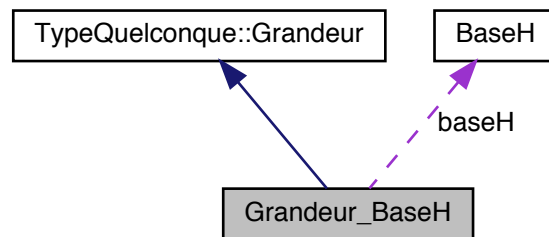
grandeur [BaseH](#): `TypeQuelconque::Grandeur::Grandeur_BaseH`

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de [Grandeur\\_BaseH](#):



Graphe de collaboration de `Grandeur_BaseH`:



## Fonctions membres publiques

- `Grandeur_BaseH` (int dim, int nb\_vec)
- `Grandeur_BaseH` (const `BaseH` &basH)
- `Grandeur_BaseH` (istream &ent)
- `Grandeur_BaseH` (const `Grandeur_BaseH` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur` & `operator=` (const `Grandeur` &a)
- `Grandeur` & `operator=` (const `Grandeur_BaseH` &a)
- `Grandeur` & `operator=` (const `BaseH` &a)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator+=` (const `Grandeur_BaseH` &aa)
- void `operator-=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur_BaseH` &aa)
- void `operator*= (double val)`
- void `operator/=` (double val)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnumTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()
- `BaseH * ConteneurBaseH` ()
- const `BaseH` & `RefConteneurBaseH` () const

## Attributs protégés

- `BaseH` baseH

## Amis

- class `Tab_Grandeur_BaseH`
- istream & `operator>>` (istream &ent, `Grandeur_BaseH` &a)
- ostream & `operator<<` (ostream &sort, const `Grandeur_BaseH` &a)

### 6.347.1 Description détaillée

grandeur `BaseH`: `TypeQuelconque::Grandeur::Grandeur_BaseH`

## 6.347.2 Documentation des fonctions membres

### 6.347.2.1 Affectation\_numerique()

```
void Grandeur_BaseH::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.347.2.2 Change\_repere()

```
void Grandeur_BaseH::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.347.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_BaseH::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.347.2.4 Grandeur\_brut()

```
void Grandeur_BaseH::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.347.2.5 GrandeurNumOrdre()

```
double Grandeur_BaseH::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.347.2.6 InitParDefaut()

```
void Grandeur_BaseH::InitParDefaut ( ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.347.2.7 Lecture\_grandeur()

```
istream & Grandeur_BaseH::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.347.2.8 NbMaxiNumeroOrdre()

```
int Grandeur_BaseH::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).



### 6.347.2.9 `New_idem_grandeur()`

`TypeQuelconque::Grandeur * Grandeur_BaseH::New_idem_grandeur ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.347.2.10 `operator*=( )`

`void Grandeur_BaseH::operator*= ( double val ) [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.347.2.11 `operator/=( )`

`void Grandeur_BaseH::operator/= ( double val ) [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.347.2.12 `Type_enumGrandeurParticuliere()`

`EnumTypeQuelParticulier Grandeur_BaseH::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.347.2.13 `Type_grandeurAssocie()`

`EnumTypeGrandeur Grandeur_BaseH::Type_grandeurAssocie ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.347.2.14 `Type_structure_grandeurAssocie()`

`EnumType2Niveau Grandeur_BaseH::Type_structure_grandeurAssocie ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

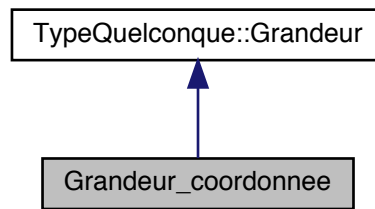
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier_2.cc`

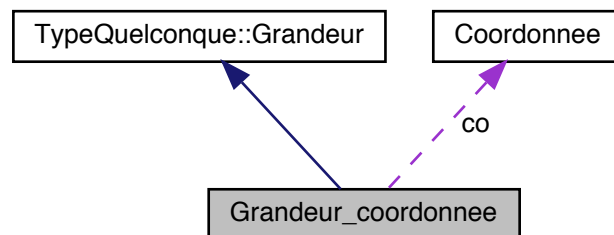
## 6.348 Référence de la classe `Grandeur_cooronnee`

grandeur coordonnée: `TypeQuelconque::Grandeur::Grandeur_cooronnee`  
`#include <TypeQuelconqueParticulier.h>`

Graphe d'héritage de Grandeur\_cooronnee:



Graphe de collaboration de Grandeur\_cooronnee:



## Fonctions membres publiques

- **Grandeur\_cooronnee** (const [Coordonnee](#) &v)
- **Grandeur\_cooronnee** (istream &ent)
- **Grandeur\_cooronnee** (const [Grandeur\\_cooronnee](#) &a)
- [TypeQuelconque::Grandeur](#) \* **New\_idem\_grandeur** () const
- void **EcriturePourLectureAvecCreation** (ostream &sort)
- [Grandeur](#) & **operator=** (const [Grandeur](#) &a)
- [Grandeur](#) & **operator=** (const [Coordonnee](#) &b)
- void **Affectation\_numerique** (const [TypeQuelconque::Grandeur](#) &a)
- void **operator+=** (const [Grandeur](#) &c)
- void **operator-=** (const [Grandeur](#) &c)
- void **operator\*=** (double val)
- void **operator/=** (double val)
- double **GrandeurNumOrdre** (int num) const
- int **NbMaxiNumeroOrdre** () const
- void **Grandeur\_brut** (ostream &sort, int nbcarr) const
- [EnumTypeGrandeur](#) **Type\_grandeurAssocie** () const
- [EnumType2Niveau](#) **Type\_structure\_grandeurAssocie** () const
- [EnuTypeQuelParticulier](#) **Type\_enumGrandeurParticuliere** () const
- void **Change\_repere** (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & **Lecture\_grandeur** (istream &ent)
- virtual ostream & **Ecriture\_grandeur** (ostream &sort) const
- void **InitParDefaut** ()
- [Coordonnee](#) \* **ConteneurCoordonnee** ()
- const [Coordonnee](#) & **ConteneurCoordonnee\_const** () const

## Attributs protégés

- `Coordonnee` `co`

## Amis

- class `Tab_Grandeur_Coorдонnee`
- `istream & operator>>` (`istream &ent`, `Grandeur_coorдонnee &a`)
- `ostream & operator<<` (`ostream &sort`, `const Grandeur_coorдонnee &a`)

### 6.348.1 Description détaillée

grandeur coordonnée: `TypeQuelconque::Grandeur::Grandeur_coorдонnee`

### 6.348.2 Documentation des fonctions membres

#### 6.348.2.1 Affectation\_numerique()

```
void Grandeur_coorдонnee::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.348.2.2 Change\_repere()

```
void Grandeur_coorдонnee::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.348.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_coorдонnee::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.348.2.4 Grandeur\_brut()

```
void Grandeur_coorдонnee::Grandeur_brut (
    ostream & sort,
    int nbcар ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.348.2.5 GrandeurNumOrdre()

```
double Grandeur_coorдонnee::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.348.2.6 InitParDefaut()

```
void Grandeur_coorдонnee::InitParDefaut ( ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

**6.348.2.7 Lecture\_grandeur()**

```
istream & Grandeur_cooronnee::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.348.2.8 NbMaxiNumeroOrdre()**

```
int Grandeur_cooronnee::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.348.2.9 New\_idem\_grandeur()**

```
TypeQuelconque::Grandeur * Grandeur_cooronnee::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.348.2.10 operator\*=( )**

```
void Grandeur_cooronnee::operator*= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.348.2.11 operator/=( )**

```
void Grandeur_cooronnee::operator/= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.348.2.12 Type\_enumGrandeurParticuliere()**

```
EnumTypeQuelParticulier Grandeur_cooronnee::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.348.2.13 Type\_grandeurAssocie()**

```
EnumTypeGrandeur Grandeur_cooronnee::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.348.2.14 Type\_structure\_grandeurAssocie()**

```
EnumType2Niveau Grandeur_cooronnee::Type_structure_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

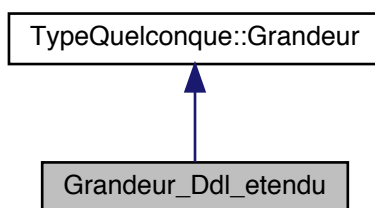
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

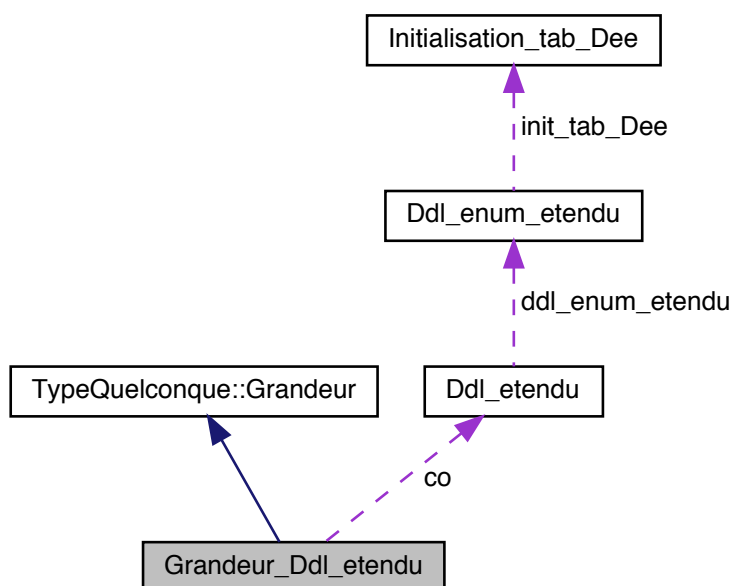
**6.349 Référence de la classe Grandeur\_Ddl\_etendu**

```
grandeur Ddl_etendu: TypeQuelconque::Grandeur::Grandeur_Ddl_etendu
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Grandeur\_Ddl\_etendu:



Graphe de collaboration de Grandeur\_Ddl\_etendu:



### Fonctions membres publiques

- **Grandeur\_Ddl\_etendu** (const [Ddl\\_etendu](#) &v, const string &nom)
- **Grandeur\_Ddl\_etendu** (istream &ent)
- **Grandeur\_Ddl\_etendu** (const [Grandeur\\_Ddl\\_etendu](#) &a)
- [TypeQuelconque::Grandeur](#) \* **New\_idem\_grandeur** () const
- void **EcriturePourLectureAvecCreation** (ostream &sort)
- [Grandeur](#) & **operator=** (const [Grandeur](#) &a)
- [Grandeur](#) & **operator=** (const [Grandeur\\_Ddl\\_etendu](#) &b)
- void **Affectation\_numerique** (const [TypeQuelconque::Grandeur](#) &a)
- void **operator+=** (const [Grandeur](#) &c)
- void **operator-=** (const [Grandeur](#) &c)
- void **operator\*=** (double val)
- void **operator/=** (double val)

- virtual const string \* [Nom\\_ref](#) () const
- double [GrandeurNumOrdre](#) (int num) const
- int [NbMaxiNumeroOrdre](#) () const
- void [Grandeur\\_brut](#) (ostream &sort, int nbcар) const
- [EnumTypeGrandeur](#) [Type\\_grandeurAssocie](#) () const
- [EnumType2Niveau](#) [Type\\_structure\\_grandeurAssocie](#) () const
- [EnuTypeQuelParticulier](#) [Type\\_enumGrandeurParticuliere](#) () const
- void [Change\\_repere](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & [Lecture\\_grandeur](#) (istream &ent)
- virtual ostream & [Ecriture\\_grandeur](#) (ostream &sort) const
- void [InitParDefaut](#) ()
- [Ddl\\_etendu](#) \* [ConteneurDdl\\_etendu](#) ()
- void [Change\\_nom\\_ref](#) (const string &nom)

### Attributs protégés

- [Ddl\\_etendu](#) **co**
- string **nom\_ref**

### Amis

- class [Tab\\_Grandeur\\_Ddl\\_etendu](#)
- istream & **operator**>> (istream &ent, [Grandeur\\_Ddl\\_etendu](#) &a)
- ostream & **operator**<< (ostream &sort, const [Grandeur\\_Ddl\\_etendu](#) &a)

## 6.349.1 Description détaillée

grandeur [Ddl\\_etendu](#): `TypeQuelconque::Grandeur::Grandeur_Ddl_etendu`

## 6.349.2 Documentation des fonctions membres

### 6.349.2.1 Affectation\_numerique()

```
void Grandeur_Ddl_etendu::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.2 Change\_repere()

```
void Grandeur_Ddl_etendu::Change_repere (
    const Mat\_pleine & beta,
    const Mat\_pleine & gamma ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_Ddl_etendu::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.4 Grandeur\_brut()

```
void Grandeur_Ddl_etendu::Grandeur_brut (
    ostream & sort,
    int nbcар ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.5 `GrandeurNumOrdre()`

```
double Grandeur_Ddl_etendu::GrandeurNumOrdre (
    int num ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.6 `InitParDefaut()`

```
void Grandeur_Ddl_etendu::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.7 `Lecture_grandeur()`

```
istream & Grandeur_Ddl_etendu::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.8 `NbMaxiNumeroOrdre()`

```
int Grandeur_Ddl_etendu::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.9 `New_idem_grandeur()`

```
TypeQuelconque::Grandeur * Grandeur_Ddl_etendu::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.10 `Nom_ref()`

```
virtual const string * Grandeur_Ddl_etendu::Nom_ref ( ) const [inline], [virtual]
```

Réimplémentée à partir de [TypeQuelconque::Grandeur](#).

### 6.349.2.11 `operator*=( )`

```
void Grandeur_Ddl_etendu::operator*=(
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.12 `operator/=( )`

```
void Grandeur_Ddl_etendu::operator/=(
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.13 `Type_enumGrandeurParticuliere()`

```
EnuTypeQuelParticulier Grandeur_Ddl_etendu::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.349.2.14 Type\_grandeurAssocie()

`EnumTypeGrandeur` `Grandeur_Ddl_etendu::Type_grandeurAssocie ( ) const [inline], [virtual]`

Implémente `TypeQuelconque::Grandeur`.

### 6.349.2.15 Type\_structure\_grandeurAssocie()

`EnumType2Niveau` `Grandeur_Ddl_etendu::Type_structure_grandeurAssocie ( ) const [inline], [virtual]`

Implémente `TypeQuelconque::Grandeur`.

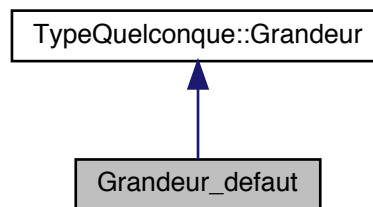
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

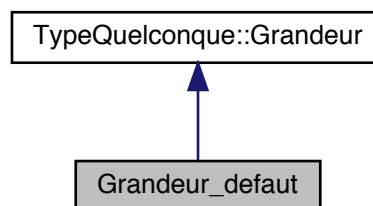
## 6.350 Référence de la classe Grandeur\_defaut

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de `Grandeur_defaut`:



Graphe de collaboration de `Grandeur_defaut`:



### Fonctions membres publiques

- `TypeQuelconque::Grandeur * New_idem_grandeur ( ) const`
- `Grandeur & operator= (const Grandeur &a)`
- `void Affectation_numerique (const TypeQuelconque::Grandeur &a)`
- `void operator+= (const Grandeur &c)`
- `void operator-= (const Grandeur &c)`



- void `operator*=(double val)`
- void `operator/=(double val)`
- double `GrandeurNumOrdre(int num) const`
- int `NbMaxiNumeroOrdre() const`
- void `Grandeur_brut(ostream &sort, int nbcар) const`
- `EnumTypeGrandeur Type_grandeurAssocie() const`
- `EnumType2Niveau Type_structure_grandeurAssocie() const`
- `EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const`
- void `Change_repere(const Mat_pleine &beta, const Mat_pleine &gamma)`
- `istream & Lecture_grandeur(istream &ent)`
- virtual `ostream & Ecriture_grandeur(ostream &sort) const`
- void `InitParDefaut()`

## Amis

- `istream & operator>>(istream &ent, Grandeur_defaut &a)`
- `ostream & operator<<(ostream &sort, const Grandeur_defaut &a)`

## 6.350.1 Description détaillée

### 6.350.1.1 grandeur par défaut: `TypeQuelconque::Grandeur::Grandeur_defaut`

## 6.350.2 Documentation des fonctions membres

### 6.350.2.1 Affectation\_numerique()

```
void Grandeur_defaut::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.2 Change\_repere()

```
void Grandeur_defaut::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_defaut::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.4 Grandeur\_brut()

```
void Grandeur_defaut::Grandeur_brut (
    ostream & sort,
    int nbcар ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.5 GrandeurNumOrdre()

```
double Grandeur_defaut::GrandeurNumOrdre (
    int num ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.6 InitParDefault()

```
void Grandeur_defaut::InitParDefault ( ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.7 Lecture\_grandeur()

```
istream & Grandeur_defaut::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.8 NbMaxiNumeroOrdre()

```
int Grandeur_defaut::NbMaxiNumeroOrdre ( ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Grandeur_defaut::New_idem_grandeur ( ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.10 operator\*=( )

```
void Grandeur_defaut::operator*= (
    double val ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.11 operator/=( )

```
void Grandeur_defaut::operator/= (
    double val ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Grandeur_defaut::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.13 Type\_grandeurAssocie()

```
EnumTypeGrandeur Grandeur_defaut::Type_grandeurAssocie ( ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.350.2.14 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Grandeur_defaut::Type_structure_grandeurAssocie ( ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

La documentation de cette classe a été générée à partir du fichier suivant :

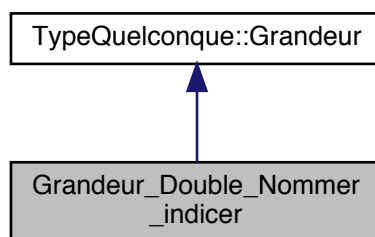
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

## 6.351 Référence de la classe Grandeur\_Double\_Nommer\_indicer

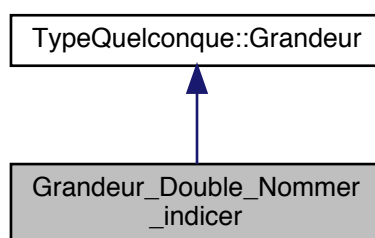
grandeur scalaire double nommé + indice: TypeQuelconque::Grandeur::Grandeur\_Double\_Nommer\_indicer contient un nom d'identificateur et un indice qui est sensé resté fixe pendant les opérations la seule grandeur qui varie c'est le double: c'est donc équivalent à un scalaire double l'indice et le nom\_ref, servent pour la gestion

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Grandeur\_Double\_Nommer\_indicer:



Graphe de collaboration de Grandeur\_Double\_Nommer\_indicer:



### Fonctions membres publiques

- **Grandeur\_Double\_Nommer\_indicer** (const string &nom\_ref\_, const double &va, int indice\_=1)
- **Grandeur\_Double\_Nommer\_indicer** (istream &ent)
- **Grandeur\_Double\_Nommer\_indicer** (const [Grandeur\\_Double\\_Nommer\\_indicer](#) &a)
- [TypeQuelconque::Grandeur](#) \* **New\_idem\_grandeur** () const
- void **EcriturePourLectureAvecCreation** (ostream &sort)
- [Grandeur](#) & **operator=** (const [Grandeur](#) &a)
- [Grandeur](#) & **operator=** (const [Grandeur\\_Double\\_Nommer\\_indicer](#) &b)
- void **Affectation\_numerique** (const [TypeQuelconque::Grandeur](#) &a)
- void **operator+=** (const [Grandeur](#) &c)
- void **operator-=** (const [Grandeur](#) &c)
- void **operator\*=** (double x)
- void **operator/=** (double x)
- double **GrandeurNumOrdre** (int num) const
- virtual const string \* **Nom\_ref** () const
- int **NbMaxiNumeroOrdre** () const
- void **Grandeur\_brut** (ostream &sort, int nbcar) const

- [EnumTypeGrandeur Type\\_grandeurAssocie](#) () const
- [EnumType2Niveau Type\\_structure\\_grandeurAssocie](#) () const
- [EnumTypeQuelParticulier Type\\_enumGrandeurParticuliere](#) () const
- void [Change\\_repere](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & [Lecture\\_grandeur](#) (istream &ent)
- virtual ostream & [Ecriture\\_grandeur](#) (ostream &sort) const
- void [InitParDefaut](#) ()
- double \* [ConteneurDouble](#) ()
- double [ContDouble](#) () const
- int [Indice\\_const](#) () const
- int & [Indice](#) ()

### Attributs protégés

- string [nom\\_ref](#)
- double [val](#)
- int [indice](#)

### Amis

- istream & [operator>>](#) (istream &ent, [Grandeur\\_Double\\_Nommer\\_indicer](#) &a)
- ostream & [operator<<](#) (ostream &sort, const [Grandeur\\_Double\\_Nommer\\_indicer](#) &a)

#### 6.351.1 Description détaillée

grandeur scalaire double nommé + indice: `TypeQuelconque::Grandeur::Grandeur_Double_Nommer_indicer` contient un nom d'identificateur et un indice qui est sensé resté fixe pendant les opérations la seule grandeur qui varie c'est le double: c'est donc équivalent à un scalaire double l'indice et le `nom_ref`, servent pour la gestion

#### 6.351.2 Documentation des fonctions membres

##### 6.351.2.1 Affectation\_numerique()

```
void Grandeur_Double_Nommer_indicer::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

##### 6.351.2.2 Change\_repere()

```
void Grandeur_Double_Nommer_indicer::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

##### 6.351.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_Double_Nommer_indicer::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

##### 6.351.2.4 Grandeur\_brut()

```
void Grandeur_Double_Nommer_indicer::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.5 `GrandeurNumOrdre()`

```
double Grandeur_Double_Nommer_indicer::GrandeurNumOrdre (
    int num ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.6 `InitParDefaut()`

```
void Grandeur_Double_Nommer_indicer::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.7 `Lecture_grandeur()`

```
istream & Grandeur_Double_Nommer_indicer::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.8 `NbMaxiNumeroOrdre()`

```
int Grandeur_Double_Nommer_indicer::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.9 `New_idem_grandeur()`

```
TypeQuelconque::Grandeur * Grandeur_Double_Nommer_indicer::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.10 `Nom_ref()`

```
virtual const string * Grandeur_Double_Nommer_indicer::Nom_ref ( ) const [inline], [virtual]
```

Réimplémentée à partir de [TypeQuelconque::Grandeur](#).

### 6.351.2.11 `operator*=( )`

```
void Grandeur_Double_Nommer_indicer::operator*= (
    double x ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.12 `operator/=( )`

```
void Grandeur_Double_Nommer_indicer::operator/= (
    double x ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.13 `Type_enumGrandeurParticuliere()`

```
EnuTypeQuelParticulier Grandeur_Double_Nommer_indicer::Type_enumGrandeurParticuliere ( ) const
[inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.14 Type\_grandeurAssocie()

```
EnumTypeGrandeur Grandeur_Double_Nommer_indicer::Type_grandeurAssocie ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.351.2.15 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Grandeur_Double_Nommer_indicer::Type_structure_grandeurAssocie ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

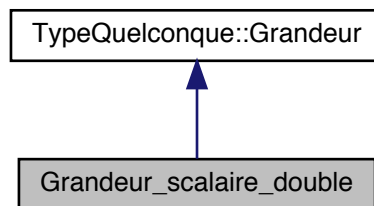
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_3.cc

## 6.352 Référence de la classe Grandeur\_scalaire\_double

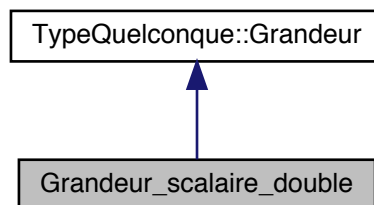
grandeur scalaire réel: [TypeQuelconque::Grandeur::Grandeur\\_scalaire\\_double](#)

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de [Grandeur\\_scalaire\\_double](#):



Graphe de collaboration de [Grandeur\\_scalaire\\_double](#):



### Fonctions membres publiques

- [Grandeur\\_scalaire\\_double](#) (const double &va)
- [Grandeur\\_scalaire\\_double](#) (istream &ent)

- `Grandeur_scalaire_double` (const `Grandeur_scalaire_double` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Grandeur_scalaire_double` &a)
- `Grandeur & operator=` (const double &b)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur` &c)
- void `operator*=` (double x)
- void `operator/=` (double x)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnuTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()
- double \* `ConteneurDouble` ()
- double `ContDouble` () const

### Attributs protégés

- double `val`

### Amis

- class `Tab_Grandeur_scalaire_double`
- istream & `operator>>` (istream &ent, `Grandeur_scalaire_double` &a)
- ostream & `operator<<` (ostream &sort, const `Grandeur_scalaire_double` &a)

## 6.352.1 Description détaillée

grandeur scalaire réel: `TypeQuelconque::Grandeur::Grandeur_scalaire_double`

## 6.352.2 Documentation des fonctions membres

### 6.352.2.1 `Affectation_numerique()`

```
void Grandeur_scalaire_double::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.352.2.2 `Change_repere()`

```
void Grandeur_scalaire_double::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.352.2.3 `Ecriture_grandeur()`

```
virtual ostream & Grandeur_scalaire_double::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.352.2.4 Grandeur\_brut()

```
void Grandeur_scalaire_double::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.352.2.5 GrandeurNumOrdre()

```
double Grandeur_scalaire_double::GrandeurNumOrdre (
    int num ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.352.2.6 InitParDefaut()

```
void Grandeur_scalaire_double::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.352.2.7 Lecture\_grandeur()

```
istream & Grandeur_scalaire_double::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.352.2.8 NbMaxiNumeroOrdre()

```
int Grandeur_scalaire_double::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.352.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Grandeur_scalaire_double::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.352.2.10 operator\*=( )

```
void Grandeur_scalaire_double::operator*= (
    double x ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.352.2.11 operator/=( )

```
void Grandeur_scalaire_double::operator/= (
    double x ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.352.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Grandeur_scalaire_double::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).



**6.352.2.13 Type\_grandeurAssocie()**

`EnumTypeGrandeur` `Grandeur_scalaire_double::Type_grandeurAssocie ( ) const [inline], [virtual]`  
 Implémente `TypeQuelconque::Grandeur`.

**6.352.2.14 Type\_structure\_grandeurAssocie()**

`EnumType2Niveau` `Grandeur_scalaire_double::Type_structure_grandeurAssocie ( ) const [inline], [virtual]`

Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

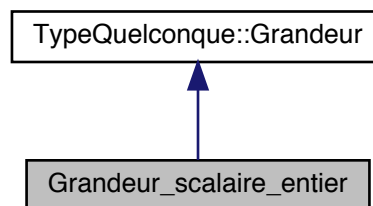
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

**6.353 Référence de la classe Grandeur\_scalaire\_entier**

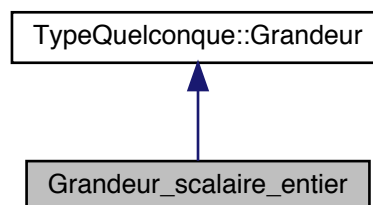
grandeur scalaire réel: `TypeQuelconque::Grandeur::Grandeur_scalaire_double`

`#include <TypeQuelconqueParticulier.h>`

Graphe d'héritage de `Grandeur_scalaire_entier`:



Graphe de collaboration de `Grandeur_scalaire_entier`:

**Fonctions membres publiques**

- `Grandeur_scalaire_entier` (const double &va)

- `Grandeur_scalaire_entier` (istream &ent)
- `Grandeur_scalaire_entier` (const `Grandeur_scalaire_entier` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Grandeur_scalaire_entier` &a)
- `Grandeur & operator=` (const double &b)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur` &c)
- void `operator*=` (double x)
- void `operator/=` (double x)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcar) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnumTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- void `Change_repere` (const `Mat_pleine` &, const `Mat_pleine` &)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()
- int \* `ConteneurEntier` ()

### Attributs protégés

- int `val`

### Amis

- class `Tab_Grandeur_scalaire_entier`
- istream & `operator>>` (istream &ent, `Grandeur_scalaire_entier` &a)
- ostream & `operator<<` (ostream &sort, const `Grandeur_scalaire_entier` &a)

## 6.353.1 Description détaillée

grandeur scalaire réel: `TypeQuelconque::Grandeur::Grandeur_scalaire_double`

## 6.353.2 Documentation des fonctions membres

### 6.353.2.1 Affectation\_numerique()

```
void Grandeur_scalaire_entier::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.353.2.2 Change\_repere()

```
void Grandeur_scalaire_entier::Change_repere (
    const Mat_pleine & ,
    const Mat_pleine & ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.353.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_scalaire_entier::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.353.2.4 `Grandeur_brut()`

```
void Grandeur_scalaire_entier::Grandeur_brut (
    ostream & sort,
    int nbcars ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.353.2.5 `GrandeurNumOrdre()`

```
double Grandeur_scalaire_entier::GrandeurNumOrdre (
    int num ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.353.2.6 `InitParDefaut()`

```
void Grandeur_scalaire_entier::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.353.2.7 `Lecture_grandeur()`

```
istream & Grandeur_scalaire_entier::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.353.2.8 `NbMaxiNumeroOrdre()`

```
int Grandeur_scalaire_entier::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.353.2.9 `New_idem_grandeur()`

```
TypeQuelconque::Grandeur * Grandeur_scalaire_entier::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.353.2.10 `operator*=( )`

```
void Grandeur_scalaire_entier::operator*=(
    double x ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.353.2.11 `operator/=( )`

```
void Grandeur_scalaire_entier::operator/=(
    double x ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.353.2.12 `Type_enumGrandeurParticuliere()`

```
EnumTypeQuelParticulier Grandeur_scalaire_entier::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.353.2.13 Type\_grandeurAssocie()

`EnumTypeGrandeur` `Grandeur_scalaire_entier::Type_grandeurAssocie ( ) const [inline], [virtual]`  
 Implémente `TypeQuelconque::Grandeur`.

### 6.353.2.14 Type\_structure\_grandeurAssocie()

`EnumType2Niveau` `Grandeur_scalaire_entier::Type_structure_grandeurAssocie ( ) const [inline], [virtual]`

Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

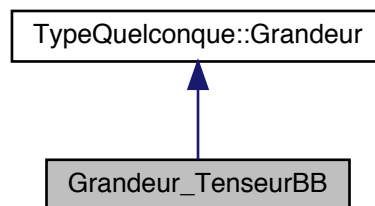
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

## 6.354 Référence de la classe Grandeur\_TenseurBB

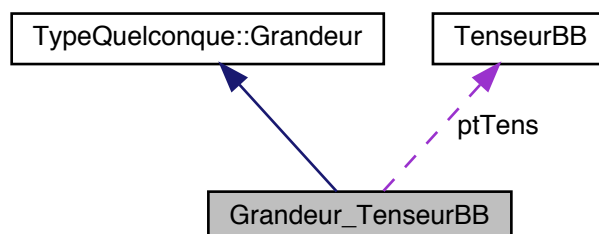
grandeur `TenseurBB`: `TypeQuelconque::Grandeur::Grandeur_TenseurBB|`

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de `Grandeur_TenseurBB`:



Graphe de collaboration de `Grandeur_TenseurBB`:



## Fonctions membres publiques

- `Grandeur_TenseurBB` (const `TenseurBB` &tens)
- `Grandeur_TenseurBB` (istream &ent)
- `Grandeur_TenseurBB` (const `Grandeur_TenseurBB` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Grandeur_TenseurBB` &a)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator+=` (const `Grandeur_TenseurBB` &aa)
- void `operator-=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur_TenseurBB` &aa)
- void `operator*=` (double val)
- void `operator/=` (double val)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnumTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()
- `TenseurBB * ConteneurTenseur` () const
- `TenseurBB & RefConteneurTenseur` () const

## Attributs protégés

- `TenseurBB * ptTens`

## Amis

- class `Tab_Grandeur_TenseurBB`
- istream & `operator>>` (istream &ent, `Grandeur_TenseurBB` &a)
- ostream & `operator<<` (ostream &sort, const `Grandeur_TenseurBB` &a)

### 6.354.1 Description détaillée

grandeur `TenseurBB`: `TypeQuelconque::Grandeur::Grandeur_TenseurBB`

### 6.354.2 Documentation des fonctions membres

#### 6.354.2.1 `Affectation_numerique()`

```
void Grandeur_TenseurBB::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.354.2.2 `Change_repere()`

```
void Grandeur_TenseurBB::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.354.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_TenseurBB::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.354.2.4 Grandeur\_brut()

```
void Grandeur_TenseurBB::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.354.2.5 GrandeurNumOrdre()

```
double Grandeur_TenseurBB::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.354.2.6 InitParDefaut()

```
void Grandeur_TenseurBB::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.354.2.7 Lecture\_grandeur()

```
istream & Grandeur_TenseurBB::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.354.2.8 NbMaxiNumeroOrdre()

```
int Grandeur_TenseurBB::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.354.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Grandeur_TenseurBB::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.354.2.10 operator\*=( )

```
void Grandeur_TenseurBB::operator*= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.354.2.11 operator/=( )

```
void Grandeur_TenseurBB::operator/= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.354.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Grandeur_TenseurBB::Type_enumGrandeurParticuliere ( ) const [inline],  
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.354.2.13 Type\_grandeurAssocie()

```
EnumTypeGrandeur Grandeur_TenseurBB::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.354.2.14 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Grandeur_TenseurBB::Type_structure_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

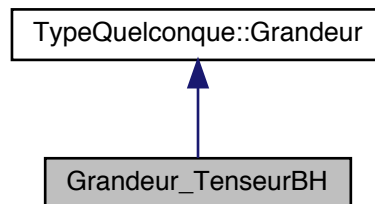
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

## 6.355 Référence de la classe Grandeur\_TenseurBH

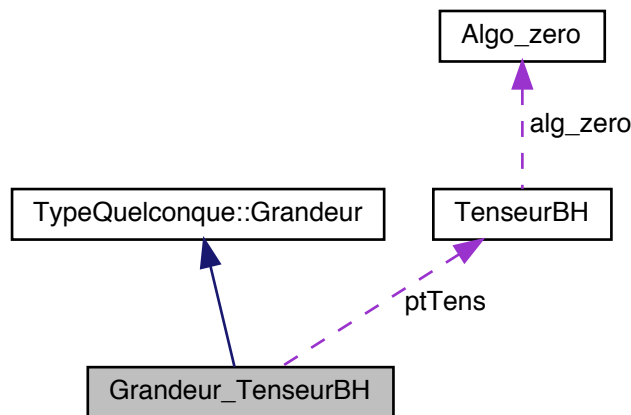
```
grandeur TenseurBH: TypeQuelconque::Grandeur::Grandeur\_TenseurBH|
```

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de [Grandeur\\_TenseurBH](#):



Graphe de collaboration de Grandeur\_TenseurBH:



## Fonctions membres publiques

- **Grandeur\_TenseurBH** (const [TenseurBH](#) &tens)
- **Grandeur\_TenseurBH** (istream &ent)
- **Grandeur\_TenseurBH** (const [Grandeur\\_TenseurBH](#) &a)
- [TypeQuelconque::Grandeur](#) \* **New\_idem\_grandeur** () const
- void **EcriturePourLectureAvecCreation** (ostream &sort)
- [Grandeur](#) & **operator=** (const [Grandeur](#) &a)
- [Grandeur](#) & **operator=** (const [Grandeur\\_TenseurBH](#) &a)
- void **Affectation\_numerique** (const [TypeQuelconque::Grandeur](#) &a)
- void **operator+=** (const [Grandeur](#) &c)
- void **operator+=** (const [Grandeur\\_TenseurBH](#) &aa)
- void **operator-=** (const [Grandeur](#) &c)
- void **operator-=** (const [Grandeur\\_TenseurBH](#) &aa)
- void **operator\*=** (double val)
- void **operator/=** (double val)
- double **GrandeurNumOrdre** (int num) const
- int **NbMaxiNumeroOrdre** () const
- void **Grandeur\_brut** (ostream &sort, int nbcarr) const
- [EnumTypeGrandeur](#) **Type\_grandeurAssocie** () const
- [EnumType2Niveau](#) **Type\_structure\_grandeurAssocie** () const
- [EnumTypeQuelParticulier](#) **Type\_enumGrandeurParticuliere** () const
- void **Change\_repere** (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & **Lecture\_grandeur** (istream &ent)
- virtual ostream & **Ecriture\_grandeur** (ostream &sort) const
- void **InitParDefaut** ()
- [TenseurBH](#) \* **ConteneurTenseur** () const
- [TenseurBH](#) & **RefConteneurTenseur** () const

## Attributs protégés

- [TenseurBH](#) \* **ptTens**

## Amis

- class **Tab\_Grandeur\_TenseurBH**
- istream & **operator>>** (istream &ent, [Grandeur\\_TenseurBH](#) &a)
- ostream & **operator<<** (ostream &sort, const [Grandeur\\_TenseurBH](#) &a)



### 6.355.1 Description détaillée

grandeur `TenseurBH`: `TypeQuelconque::Grandeur::Grandeur_TenseurBH`|

### 6.355.2 Documentation des fonctions membres

#### 6.355.2.1 Affectation\_numerique()

```
void Grandeur_TenseurBH::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.355.2.2 Change\_repere()

```
void Grandeur_TenseurBH::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.355.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_TenseurBH::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.355.2.4 Grandeur\_brut()

```
void Grandeur_TenseurBH::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.355.2.5 GrandeurNumOrdre()

```
double Grandeur_TenseurBH::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.355.2.6 InitParDefaut()

```
void Grandeur_TenseurBH::InitParDefaut ( ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.355.2.7 Lecture\_grandeur()

```
istream & Grandeur_TenseurBH::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

**6.355.2.8 NbMaxiNumeroOrdre()**

```
int Grandeur_TenseurBH::NbMaxiNumeroOrdre ( ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.355.2.9 New\_idem\_grandeur()**

```
TypeQuelconque::Grandeur * Grandeur_TenseurBH::New_idem_grandeur ( ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.355.2.10 operator\*=( )**

```
void Grandeur_TenseurBH::operator*= (
    double val ) [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.355.2.11 operator/=( )**

```
void Grandeur_TenseurBH::operator/= (
    double val ) [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.355.2.12 Type\_enumGrandeurParticuliere()**

```
EnumTypeQuelParticulier Grandeur_TenseurBH::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.355.2.13 Type\_grandeurAssocie()**

```
EnumTypeGrandeur Grandeur_TenseurBH::Type_grandeurAssocie ( ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.355.2.14 Type\_structure\_grandeurAssocie()**

```
EnumType2Niveau Grandeur_TenseurBH::Type_structure_grandeurAssocie ( ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

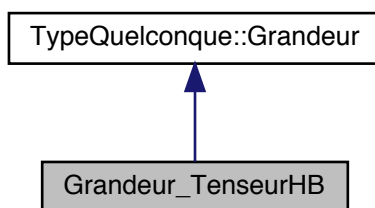
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

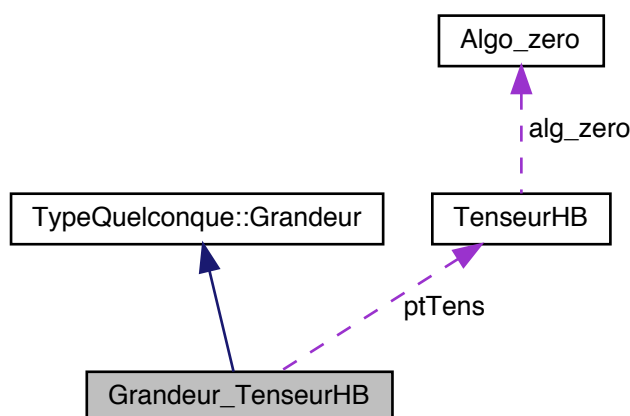
**6.356 Référence de la classe Grandeur\_TenseurHB**

```
grandeur TenseurHB: TypeQuelconque::Grandeur::Grandeur\_TenseurHB|
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Grandeur\_TenseurHB:



Graphe de collaboration de Grandeur\_TenseurHB:



## Fonctions membres publiques

- `Grandeur_TenseurHB` (const `TenseurHB` &tens)
- `Grandeur_TenseurHB` (istream &ent)
- `Grandeur_TenseurHB` (const `Grandeur_TenseurHB` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Grandeur_TenseurHB` &a)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator+=` (const `Grandeur_TenseurHB` &aa)
- void `operator-=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur_TenseurHB` &aa)
- void `operator*=` (double val)
- void `operator/=` (double val)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const

- [EnumType2Niveau](#) [Type\\_structure\\_grandeurAssocie](#) () const
- [EnumTypeQuelParticulier](#) [Type\\_enumGrandeurParticuliere](#) () const
- void [Change\\_repere](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & [Lecture\\_grandeur](#) (istream &ent)
- virtual ostream & [Ecriture\\_grandeur](#) (ostream &sort) const
- void [InitParDefaut](#) ()
- [TenseurHB](#) \* [ConteneurTenseur](#) () const
- [TenseurHB](#) & [RefConteneurTenseur](#) () const

### Attributs protégés

- [TenseurHB](#) \* [ptTens](#)

### Amis

- class [Tab\\_Grandeur\\_TenseurHB](#)
- istream & [operator](#)>>> (istream &ent, [Grandeur\\_TenseurHB](#) &a)
- ostream & [operator](#)<<< (ostream &sort, const [Grandeur\\_TenseurHB](#) &a)

## 6.356.1 Description détaillée

grandeur [TenseurHB](#): [TypeQuelconque::Grandeur::Grandeur\\_TenseurHB](#)|

## 6.356.2 Documentation des fonctions membres

### 6.356.2.1 Affectation\_numerique()

```
void Grandeur_TenseurHB::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.2 Change\_repere()

```
void Grandeur_TenseurHB::Change_repere (
    const Mat\_pleine & beta,
    const Mat\_pleine & gamma ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_TenseurHB::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.4 Grandeur\_brut()

```
void Grandeur_TenseurHB::Grandeur_brut (
    ostream & sort,
    int nbcars ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.5 `GrandeurNumOrdre()`

```
double Grandeur_TenseurHB::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.6 `InitParDefaut()`

```
void Grandeur_TenseurHB::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.7 `Lecture_grandeur()`

```
istream & Grandeur_TenseurHB::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.8 `NbMaxiNumeroOrdre()`

```
int Grandeur_TenseurHB::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.9 `New_idem_grandeur()`

```
TypeQuelconque::Grandeur * Grandeur_TenseurHB::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.10 `operator*=( )`

```
void Grandeur_TenseurHB::operator*= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.11 `operator/=( )`

```
void Grandeur_TenseurHB::operator/= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.12 `Type_enumGrandeurParticuliere()`

```
EnumTypeQuelParticulier Grandeur_TenseurHB::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.13 `Type_grandeurAssocie()`

```
EnumTypeGrandeur Grandeur_TenseurHB::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.356.2.14 Type\_structure\_grandeurAssocie()

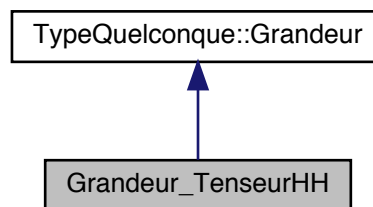
`EnumType2Niveau` `Grandeur_TenseurHB::Type_structure_grandeurAssocie ( ) const [inline], [virtual]`  
 Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

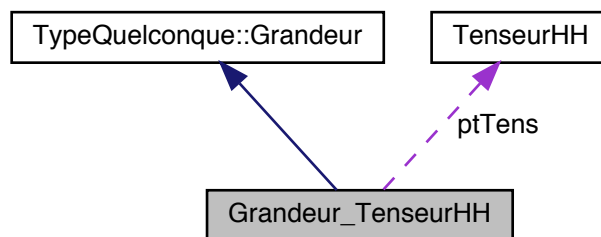
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

## 6.357 Référence de la classe Grandeur\_TenseurHH

grandeur `TenseurHH`: `TypeQuelconque::Grandeur::Grandeur_TenseurHH|`  
`#include <TypeQuelconqueParticulier.h>`  
 Graphe d'héritage de `Grandeur_TenseurHH`:



Graphe de collaboration de `Grandeur_TenseurHH`:



### Fonctions membres publiques

- `Grandeur_TenseurHH` (const `TenseurHH` &tens)
- `Grandeur_TenseurHH` (istream &ent)
- `Grandeur_TenseurHH` (const `Grandeur_TenseurHH` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` ( ) const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Grandeur_TenseurHH` &a)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator+=` (const `Grandeur_TenseurHH` &aa)

- void `operator-=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur_TenseurHH` &aa)
- void `operator*=` (double val)
- void `operator/=` (double val)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnuTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()
- `TenseurHH * ConteneurTenseur` () const
- `TenseurHH & RefConteneurTenseur` () const

### Attributs protégés

- `TenseurHH * ptTens`

### Amis

- class `Tab_Grandeur_TenseurHH`
- class `Tab2_Grandeur_TenseurHH`
- istream & `operator>>` (istream &ent, `Grandeur_TenseurHH` &a)
- ostream & `operator<<` (ostream &sort, const `Grandeur_TenseurHH` &a)

## 6.357.1 Description détaillée

grandeur `TenseurHH`: `TypeQuelconque::Grandeur::Grandeur_TenseurHH`

## 6.357.2 Documentation des fonctions membres

### 6.357.2.1 `Affectation_numerique()`

```
void Grandeur_TenseurHH::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.357.2.2 `Change_repere()`

```
void Grandeur_TenseurHH::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.357.2.3 `Ecriture_grandeur()`

```
virtual ostream & Grandeur_TenseurHH::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.357.2.4 Grandeur\_brut()

```
void Grandeur_TenseurHH::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.357.2.5 GrandeurNumOrdre()

```
double Grandeur_TenseurHH::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.357.2.6 InitParDefaut()

```
void Grandeur_TenseurHH::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.357.2.7 Lecture\_grandeur()

```
istream & Grandeur_TenseurHH::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.357.2.8 NbMaxiNumeroOrdre()

```
int Grandeur_TenseurHH::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.357.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Grandeur_TenseurHH::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.357.2.10 operator\*=( )

```
void Grandeur_TenseurHH::operator*=(
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.357.2.11 operator/=( )

```
void Grandeur_TenseurHH::operator/=(
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.357.2.12 Type\_enumGrandeurParticuliere()

```
EnuTypeQuelParticulier Grandeur_TenseurHH::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).



### 6.357.2.13 Type\_grandeurAssocie()

`EnumTypeGrandeur` `Grandeur_TenseurHH::Type_grandeurAssocie ( ) const [inline], [virtual]`  
 Implémente `TypeQuelconque::Grandeur`.

### 6.357.2.14 Type\_structure\_grandeurAssocie()

`EnumType2Niveau` `Grandeur_TenseurHH::Type_structure_grandeurAssocie ( ) const [inline], [virtual]`  
 Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

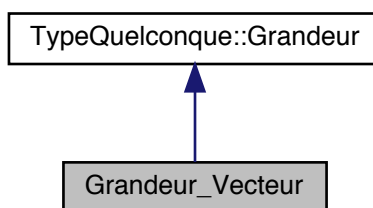
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

## 6.358 Référence de la classe Grandeur\_Vecteur

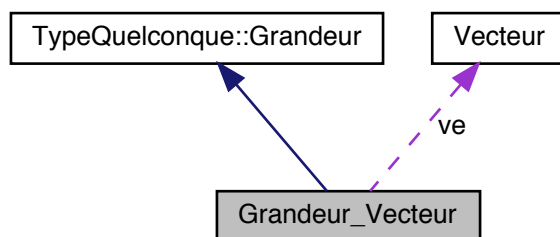
`grandeur vecteur: TypeQuelconque::Grandeur::Grandeur_vecteur`

`#include <TypeQuelconqueParticulier.h>`

Graphe d'héritage de `Grandeur_Vecteur`:



Graphe de collaboration de `Grandeur_Vecteur`:



### Fonctions membres publiques

- `Grandeur_Vecteur` (const `Vecteur` &v)
- `Grandeur_Vecteur` (int n)
- `Grandeur_Vecteur` (istream &ent)

- **Grandeur\_Vecteur** (const [Grandeur\\_Vecteur](#) &a)
- [TypeQuelconque::Grandeur](#) \* [New\\_idem\\_grandeur](#) () const
- void **EcriturePourLectureAvecCreation** (ostream &sort)
- [Grandeur](#) & **operator=** (const [Grandeur](#) &a)
- [Grandeur](#) & **operator=** (const [Vecteur](#) &b)
- void [Affectation\\_numerique](#) (const [TypeQuelconque::Grandeur](#) &a)
- void **operator+=** (const [Grandeur](#) &c)
- void **operator-=** (const [Grandeur](#) &c)
- void **operator\*=** (double val)
- void **operator/=** (double val)
- double [GrandeurNumOrdre](#) (int num) const
- int [NbMaxiNumeroOrdre](#) () const
- void [Grandeur\\_brut](#) (ostream &sort, int nbcarr) const
- [EnumTypeGrandeur](#) [Type\\_grandeurAssocie](#) () const
- [EnumType2Niveau](#) [Type\\_structure\\_grandeurAssocie](#) () const
- [EnuTypeQuelParticulier](#) [Type\\_enumGrandeurParticuliere](#) () const
- void [Change\\_repere](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & [Lecture\\_grandeur](#) (istream &ent)
- virtual ostream & [Ecriture\\_grandeur](#) (ostream &sort) const
- void [InitParDefaut](#) ()
- [Vecteur](#) \* **ConteneurVecteur** ()
- const [Vecteur](#) & **ConteneurVecteur\_const** () const

## Attributs protégés

- [Vecteur](#) **ve**

## Amis

- class [Tab\\_Grandeur\\_Vecteur](#)
- istream & **operator>>** (istream &ent, [Grandeur\\_Vecteur](#) &a)
- ostream & **operator<<** (ostream &sort, const [Grandeur\\_Vecteur](#) &a)

### 6.358.1 Description détaillée

grandeur vecteur: [TypeQuelconque::Grandeur::Grandeur\\_vecteur](#)

### 6.358.2 Documentation des fonctions membres

#### 6.358.2.1 Affectation\_numerique()

```
void Grandeur_Vecteur::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.2 Change\_repere()

```
void Grandeur_Vecteur::Change_repere (
    const Mat\_pleine & beta,
    const Mat\_pleine & gamma ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_Vecteur::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.4 `Grandeur_brut()`

```
void Grandeur_Vecteur::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.5 `GrandeurNumOrdre()`

```
double Grandeur_Vecteur::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.6 `InitParDefaut()`

```
void Grandeur_Vecteur::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.7 `Lecture_grandeur()`

```
istream & Grandeur_Vecteur::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.8 `NbMaxiNumeroOrdre()`

```
int Grandeur_Vecteur::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.9 `New_idem_grandeur()`

```
TypeQuelconque::Grandeur * Grandeur_Vecteur::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.10 `operator*=( )`

```
void Grandeur_Vecteur::operator*= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.11 `operator/=( )`

```
void Grandeur_Vecteur::operator/= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.358.2.12 `Type_enumGrandeurParticuliere()`

```
EnuTypeQuelParticulier Grandeur_Vecteur::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.358.2.13 Type\_grandeurAssocie()

`EnumTypeGrandeur` `Grandeur_Vecteur::Type_grandeurAssocie ( ) const [inline], [virtual]`  
 Implémente `TypeQuelconque::Grandeur`.

### 6.358.2.14 Type\_structure\_grandeurAssocie()

`EnumType2Niveau` `Grandeur_Vecteur::Type_structure_grandeurAssocie ( ) const [inline], [virtual]`  
 Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

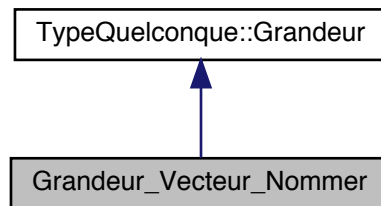
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_3.cc

## 6.359 Référence de la classe Grandeur\_Vecteur\_Nommer

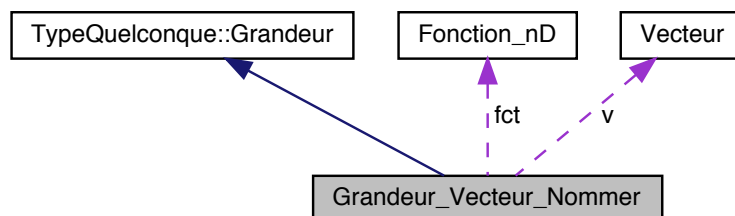
grandeur vecteur nommé : `TypeQuelconque::Grandeur::Grandeur_Vecteur_Nommer` contient un nom d'identificateur (utilisé par exemple pour un nom de fonc

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de `Grandeur_Vecteur_Nommer`:



Graphe de collaboration de `Grandeur_Vecteur_Nommer`:



### Fonctions membres publiques

- `Grandeur_Vecteur_Nommer` (`const string &nom_ref_`, `const int taille`, `Fonction_nD *fct_=NULL`)
- `Grandeur_Vecteur_Nommer` (`istream &ent`)
- `Grandeur_Vecteur_Nommer` (`const Grandeur_Vecteur_Nommer &a`)

- `TypeQuelconque::Grandeur * New_idem_grandeur ()` const
- `void EcriturePourLectureAvecCreation (ostream &sort)`
- `Grandeur & operator= (const Grandeur &a)`
- `Grandeur & operator= (const Grandeur_Vecteur_Nommer &b)`
- `void Affectation_numerique (const TypeQuelconque::Grandeur &a)`
- `void operator+= (const Grandeur &c)`
- `void operator-= (const Grandeur &c)`
- `void operator*= (double val)`
- `void operator/= (double val)`
- `double GrandeurNumOrdre (int num)` const
- `virtual const string * Nom_ref ()` const
- `int NbMaxiNumeroOrdre ()` const
- `void Grandeur_brut (ostream &sort, int nbcar)` const
- `EnumTypeGrandeur Type_grandeurAssocie ()` const
- `EnumType2Niveau Type_structure_grandeurAssocie ()` const
- `EnumTypeQuelParticulier Type_enumGrandeurParticuliere ()` const
- `void Change_repere (const Mat_pleine &beta, const Mat_pleine &gamma)`
- `istream & Lecture_grandeur (istream &ent)`
- `virtual ostream & Ecriture_grandeur (ostream &sort)` const
- `void InitParDefaut ()`
- `Vecteur & ConteneurVecteur ()`
- `Fonction_nD * Fct ()` const
- `void Change_fonction_nD (Fonction_nD *fct_)`

### Attributs protégés

- `Fonction_nD * fct`
- `string nom_ref`
- `Vecteur v`

### Amis

- `istream & operator>> (istream &ent, Grandeur_Vecteur_Nommer &a)`
- `ostream & operator<< (ostream &sort, const Grandeur_Vecteur_Nommer &a)`

## 6.359.1 Description détaillée

grandeur vecteur nommé : `TypeQuelconque::Grandeur::Grandeur_Vecteur_Nommer` contient un nom d'identificateur (utilisé par exemple pour un nom de fonc

## 6.359.2 Documentation des fonctions membres

### 6.359.2.1 Affectation\_numerique()

```
void Grandeur_Vecteur_Nommer::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.359.2.2 Change\_repere()

```
void Grandeur_Vecteur_Nommer::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.359.2.3 Ecriture\_grandeur()

```
virtual ostream & Grandeur_Vecteur_Nommer::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.359.2.4 Grandeur\_brut()

```
void Grandeur_Vecteur_Nommer::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.359.2.5 GrandeurNumOrdre()

```
double Grandeur_Vecteur_Nommer::GrandeurNumOrdre (
    int num ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.359.2.6 InitParDefaut()

```
void Grandeur_Vecteur_Nommer::InitParDefaut ( ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.359.2.7 Lecture\_grandeur()

```
istream & Grandeur_Vecteur_Nommer::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.359.2.8 NbMaxiNumeroOrdre()

```
int Grandeur_Vecteur_Nommer::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.359.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Grandeur_Vecteur_Nommer::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.359.2.10 Nom\_ref()

```
virtual const string * Grandeur_Vecteur_Nommer::Nom_ref ( ) const [inline], [virtual]
```

Réimplémentée à partir de [TypeQuelconque::Grandeur](#).

### 6.359.2.11 operator\*=( )

```
void Grandeur_Vecteur_Nommer::operator*= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.359.2.12 operator/=( )**

```
void Grandeur_Vecteur_Nommer::operator/= (
    double val ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.359.2.13 Type\_enumGrandeurParticuliere()**

```
EnumTypeQuelParticulier Grandeur_Vecteur_Nommer::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.359.2.14 Type\_grandeurAssocie()**

```
EnumTypeGrandeur Grandeur_Vecteur_Nommer::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.359.2.15 Type\_structure\_grandeurAssocie()**

```
EnumType2Niveau Grandeur_Vecteur_Nommer::Type_structure_grandeurAssocie ( ) const [inline],
[virtual]
```

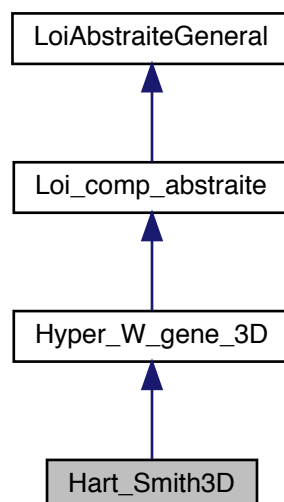
Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

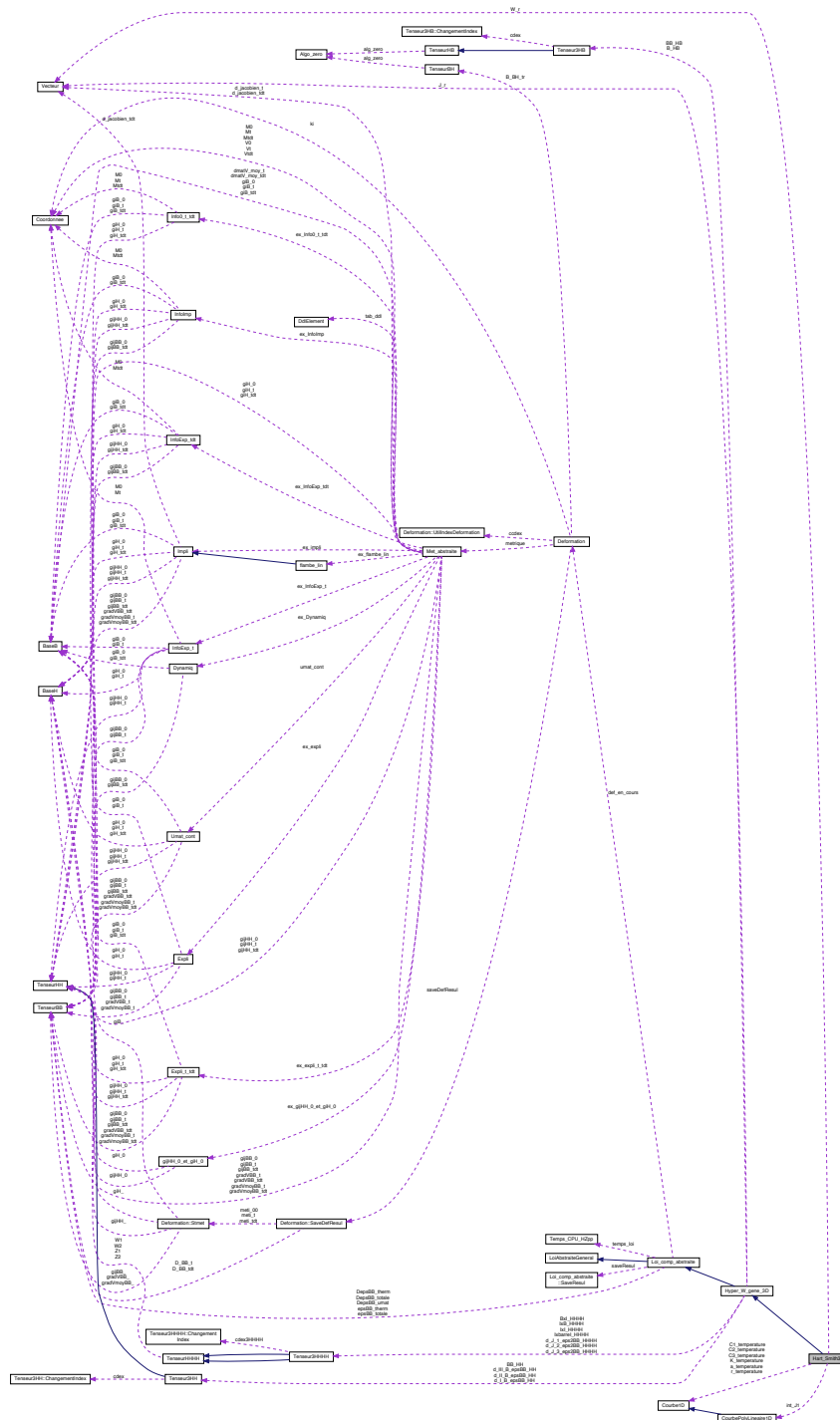
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

**6.360 Référence de la classe Hart\_Smith3D**

Grappe d'héritage de Hart\_Smith3D:



Graphe de collaboration de Hart\_Smith3D:



## Fonctions membres publiques

- **Hart\_Smith3D** (const [Hart\\_Smith3D](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)



- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- double `Module_young_equivalent` (Enum\_dure temps, const Deformation &, SaveResul \*)
- virtual double `HsurH0` (SaveResul \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- void `Info_commande_LoisDeComp` (UtilLecture &lec)

### Fonctions membres protégées

- void `Calcul_SigmaHH` (TenseurHH &sigHH\_t, TenseurBB &DepsBB, DdlElement &tab\_ddl, TenseurBB &gij←BB\_t, TenseurHH &gijHH\_t, BaseB &giB, BaseH &gi\_H, TenseurBB &epsBB\_, TenseurBB &delta\_epsBB, TenseurBB &gijBB\_, TenseurHH &gijHH\_, Tableau< TenseurBB \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, TenseurHH &sigHH, EnergieMeca &energ, const EnergieMeca &energ\_t, double &module\_←compressibilite, double &module\_cisaillement, const Met\_abstraite::Expli\_t\_tdt &ex)
- void `Calcul_DsigmaHH_tdt` (TenseurHH &sigHH\_t, TenseurBB &DepsBB, DdlElement &tab\_ddl, BaseB &giB\_t, TenseurBB &gijBB\_t, TenseurHH &gijHH\_t, BaseB &giB\_tdt, Tableau< BaseB > &d\_giB\_tdt, BaseH &giH\_tdt, Tableau< BaseH > &d\_giH\_tdt, TenseurBB &epsBB\_tdt, Tableau< TenseurBB \* > &d\_eps←BB, TenseurBB &delta\_epsBB, TenseurBB &gijBB\_tdt, TenseurHH &gijHH\_tdt, Tableau< TenseurBB \* > &d\_gijBB\_tdt, Tableau< TenseurHH \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, Vecteur &d\_jacobien\_tdt, TenseurHH &sigHH, Tableau< TenseurHH \* > &d\_sigHH, EnergieMeca &energ, const EnergieMeca &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const Met\_←abstraite::Impli &ex)
- virtual void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, TenseurHH &sigHH\_t, TenseurBB &DepsBB, TenseurBB &epsBB\_tdt, TenseurBB &delta\_epsBB, double &jacobien\_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d\_sigma\_deps, EnergieMeca &energ, const EnergieMeca &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const Met\_abstraite::Umat\_cont &ex)
- virtual void `CalculGrandeurTravail` (const PtIntegMecalInterne &, const Deformation &, Enum\_dure, const ThermoDonnee &, const Met\_abstraite::Impli \*ex\_impli, const Met\_abstraite::Expli\_t\_tdt \*ex\_expli\_tdt, const Met\_abstraite::Umat\_cont \*ex\_umat, const List\_io< Ddl\_etendu > \*exclure\_dd\_etend, const List\_io< const TypeQuelconque \* > \*exclure\_Q)

### Attributs protégés

- double `C1`
- double `C2`
- double `C3`
- double `K`
- `Courbe1D` \* `C1_temperature`
- `Courbe1D` \* `C2_temperature`
- `Courbe1D` \* `C3_temperature`
- `Courbe1D` \* `K_temperature`
- int `type_pot_vol`
- bool `avec_courbure`
- double `a_courbure`
- double `r_courbure`
- `Courbe1D` \* `a_temperature`
- `Courbe1D` \* `r_temperature`
- `CourbePolyLineaire1D` int\_J1
- double `W_d`
- double `W_v`
- Vecteur `W_r`
- double `W_d_J1`
- double `W_d_J2`
- double `W_d_J1_2`
- double `W_d_J1_J2`
- double `W_d_J2_2`
- double `W_v_J3`
- double `W_v_J3J3`
- Tableau2< double > `W_rs`
- double `W_c`
- double `W_c_J1`
- double `W_c_J3`
- double `W_c_J1_2`
- double `W_c_J3_2`
- double `W_c_J1_J3`

## Membres hérités additionnels

### 6.360.1 Documentation des fonctions membres

#### 6.360.1.1 Affiche()

void Hart\_Smith3D::Affiche ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.360.1.2 Calcul\_dsigma\_deps()

```
void Hart_Smith3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
    Tenseur3HHHH d_sigma_depsHHHH; d_sigma_depsHHHH.TransfertDunTenseurGeneral(dSigdepsHHHH.↔
    Symetrise1et2_3et4());
    Réimplémentée à partir de Loi\_comp\_abstraite.
```

#### 6.360.1.3 Calcul\_DsigmaHH\_tdt()

```
void Hart_Smith3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
```

```

    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.360.1.4 Calcul\_SigmaHH()

```

void Hart_Smith3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.360.1.5 CalculGrandeurTravail()

```

virtual void Hart_Smith3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.360.1.6 Ecriture\_base\_info\_loi()

```

void Hart_Smith3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

### 6.360.1.7 HsurH0()

```
virtual double Hart_Smith3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.360.1.8 Info\_commande\_LoisDeComp()

```
void Hart_Smith3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.360.1.9 Lecture\_base\_info\_loi()

```
void Hart_Smith3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.360.1.10 LectureDonneesParticulieres()

```
void Hart_Smith3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.360.1.11 Module\_young\_equivalent()

```
double Hart_Smith3D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.360.1.12 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Hart_Smith3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.360.1.13 TestComplet()

```
int Hart_Smith3D::TestComplet ( ) [virtual]
```

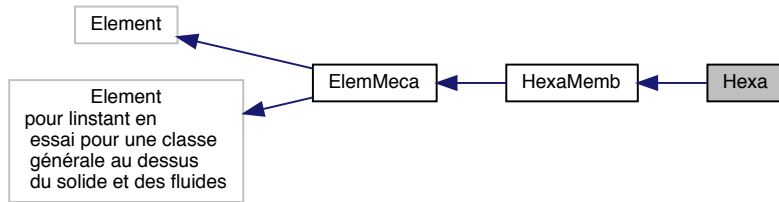
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

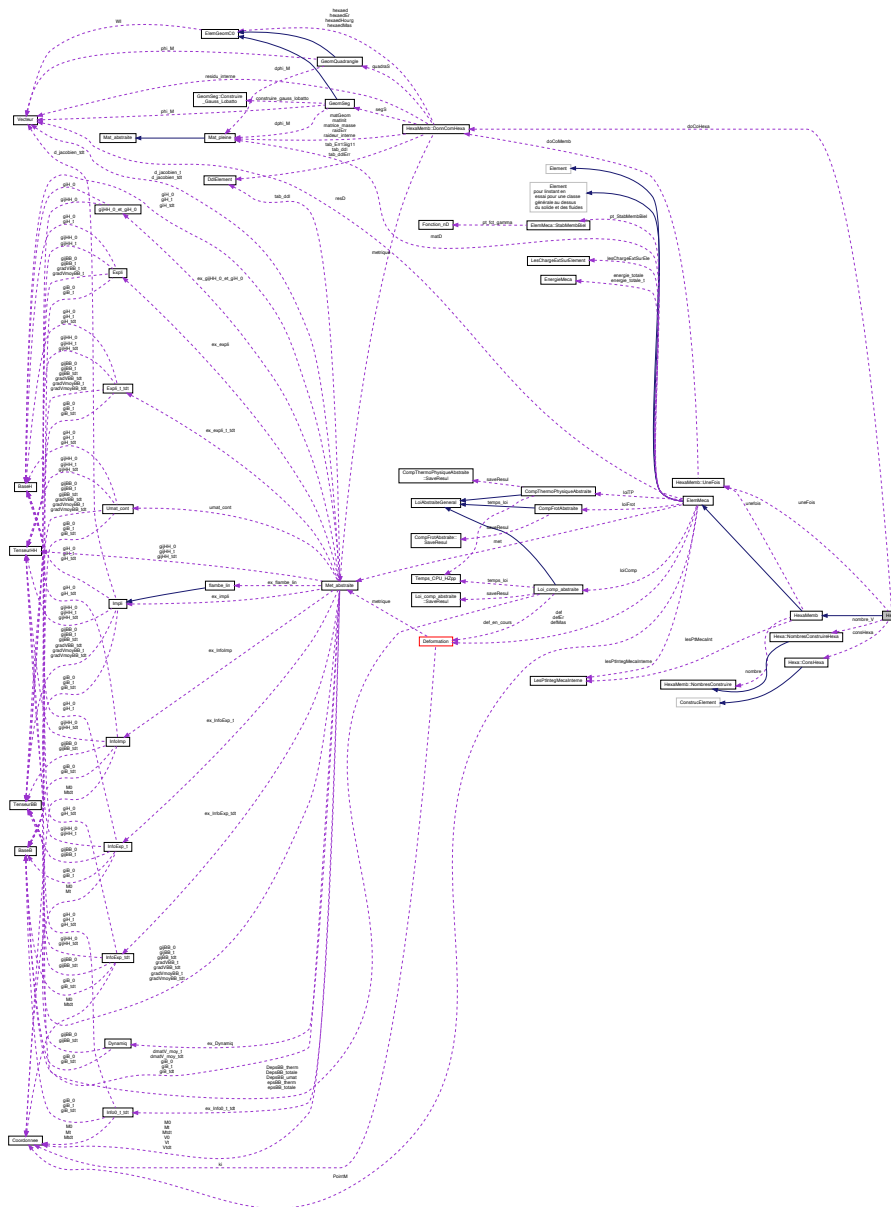
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hart\_Smith3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hart\_Smith3D.cc

## 6.361 Référence de la classe Hexa

Graphe d'héritage de Hexa:



Graphe de collaboration de Hexa:



## Classes

- class [ConsHexa](#)
- class [NombresConstruireHexa](#)

## Fonctions membres publiques

- [Hexa](#) (int num\_mail, int num\_id)
- [Hexa](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [Hexa](#) (const [Hexa](#) &hexa)
- [Element](#) \* [Nevez\\_copie](#) () const
- [Hexa](#) & [operator=](#) ([Hexa](#) &hexa)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* [doCoHexa](#) = NULL
- static [HexaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireHexa](#) [nombre\\_V](#)
- static [ConsHexa](#) [consHexa](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.361.1 Documentation des fonctions membres

#### 6.361.1.1 [new\\_frontiere\\_lin\(\)](#)

```
ElFrontiere * Hexa::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.361.1.2 [new\\_frontiere\\_surf\(\)](#)

```
ElFrontiere * Hexa::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaLMemb.cc



## Classes

- class [ConsHexa\\_cm1pti](#)
- class [NombresConstruireHexa\\_cm1pti](#)

## Fonctions membres publiques

- [Hexa\\_cm1pti](#) (int num\_mail, int num\_id)
- [Hexa\\_cm1pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [Hexa\\_cm1pti](#) (const [Hexa\\_cm1pti](#) &hexa\_cm1pti)
- [Element](#) \* [Nevez\\_copie](#) () const
- [Hexa\\_cm1pti](#) & [operator=](#) ([Hexa\\_cm1pti](#) &hexa\_cm1pti)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* [doCoHexa\\_cm1pti](#) = NULL
- static [HexaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireHexa\\_cm1pti](#) [nombre\\_V](#)
- static [ConsHexa\\_cm1pti](#) [consHexa\\_cm1pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.362.1 Documentation des fonctions membres

#### 6.362.1.1 [new\\_frontiere\\_lin\(\)](#)

```
ElFrontiere * Hexa_cm1pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.362.1.2 [new\\_frontiere\\_surf\(\)](#)

```
ElFrontiere * Hexa_cm1pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

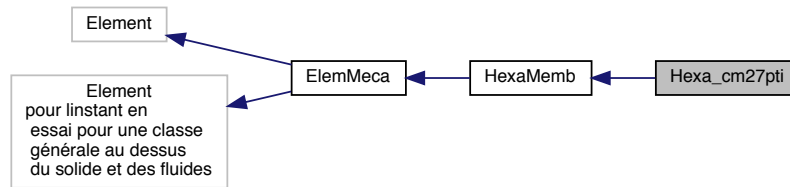
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm1pti.cc

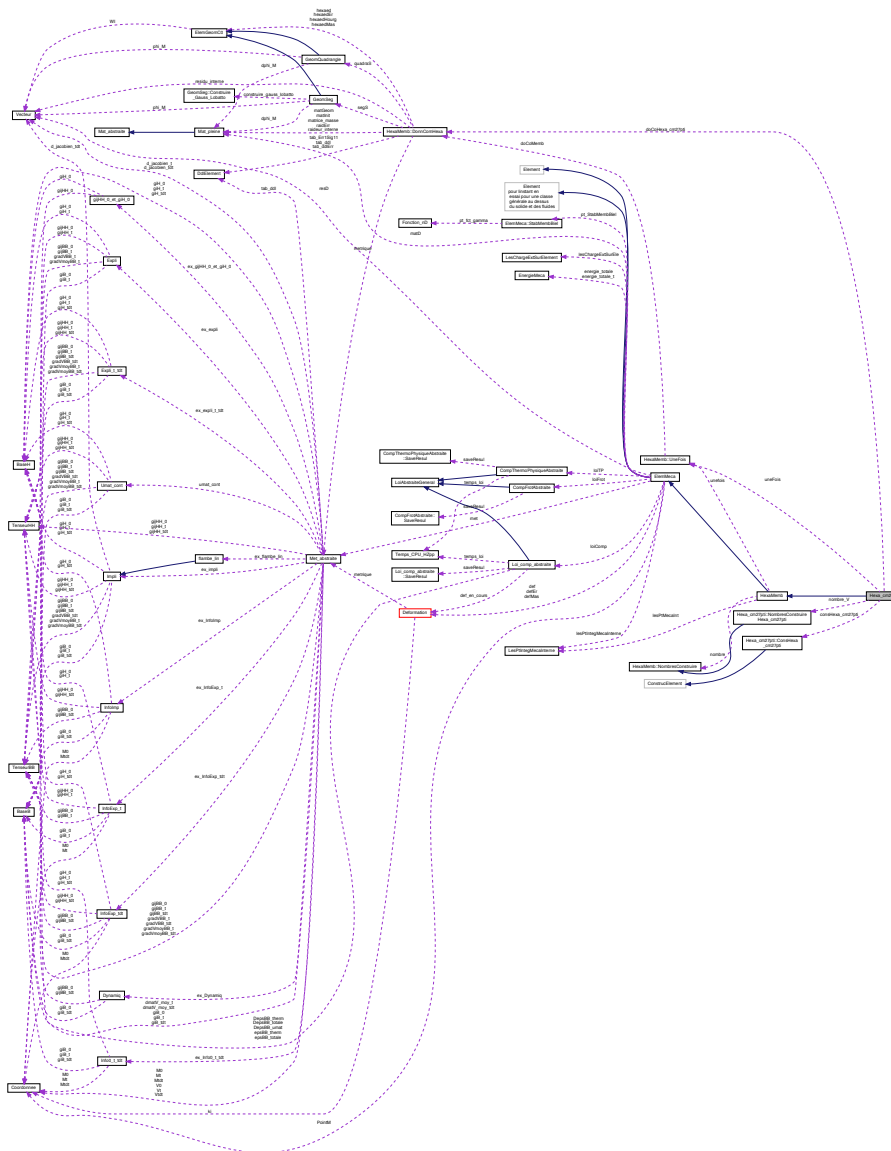


## 6.363 Référence de la classe Hexa\_cm27pti

Grappe d'héritage de Hexa\_cm27pti:



Grappe de collaboration de Hexa\_cm27pti:



## Classes

- class [ConsHexa\\_cm27pti](#)
- class [NombresConstruireHexa\\_cm27pti](#)

## Fonctions membres publiques

- [Hexa\\_cm27pti](#) (int num\_mail, int num\_id)
- [Hexa\\_cm27pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [Hexa\\_cm27pti](#) (const [Hexa\\_cm27pti](#) &[Hexa\\_cm27pti](#))
- [Element](#) \* [Nevez\\_copie](#) () const
- [Hexa\\_cm27pti](#) & [operator=](#) ([Hexa\\_cm27pti](#) &[Hexa\\_cm27pti](#))
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* [doCoHexa\\_cm27pti](#) = NULL
- static [HexaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireHexa\\_cm27pti](#) [nombre\\_V](#)
- static [ConsHexa\\_cm27pti](#) [consHexa\\_cm27pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.363.1 Documentation des fonctions membres

#### 6.363.1.1 [new\\_frontiere\\_lin\(\)](#)

```
ElFrontiere * Hexa\_cm27pti::new\_frontiere\_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.363.1.2 [new\\_frontiere\\_surf\(\)](#)

```
ElFrontiere * Hexa\_cm27pti::new\_frontiere\_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

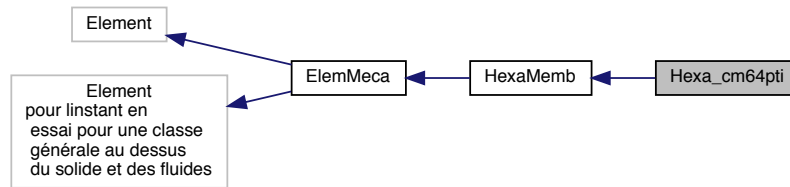
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

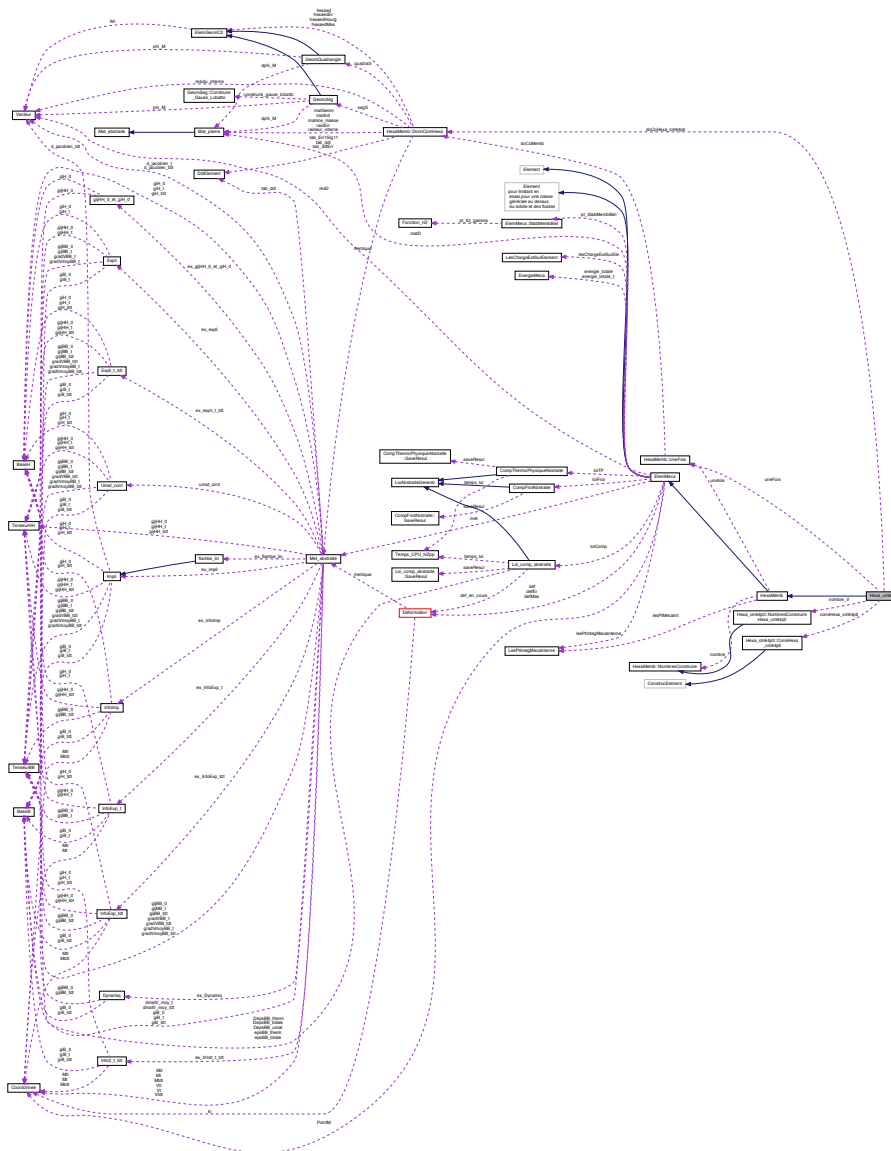
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm27pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm27pti.cc

## 6.364 Référence de la classe Hexa\_cm64pti

Grphe d'héritage de Hexa\_cm64pti:



Grphe de collaboration de Hexa\_cm64pti:



## Classes

- class [ConsHexa\\_cm64pti](#)
- class [NombresConstruireHexa\\_cm64pti](#)

## Fonctions membres publiques

- [Hexa\\_cm64pti](#) (int num\_mail, int num\_id)
- [Hexa\\_cm64pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [Hexa\\_cm64pti](#) (const [Hexa\\_cm64pti](#) &[Hexa\\_cm64pti](#))
- [Element](#) \* [Nevez\\_copie](#) () const
- [Hexa\\_cm64pti](#) & [operator=](#) ([Hexa\\_cm64pti](#) &[Hexa\\_cm64pti](#))
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* [doCoHexa\\_cm64pti](#) = NULL
- static [HexaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireHexa\\_cm64pti](#) [nombre\\_V](#)
- static [ConsHexa\\_cm64pti](#) [consHexa\\_cm64pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.364.1 Documentation des fonctions membres

#### 6.364.1.1 [new\\_frontiere\\_lin\(\)](#)

```
ElFrontiere * Hexa\_cm64pti::new\_frontiere\_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.364.1.2 [new\\_frontiere\\_surf\(\)](#)

```
ElFrontiere * Hexa\_cm64pti::new\_frontiere\_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

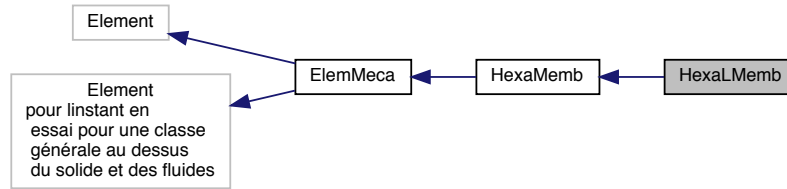
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

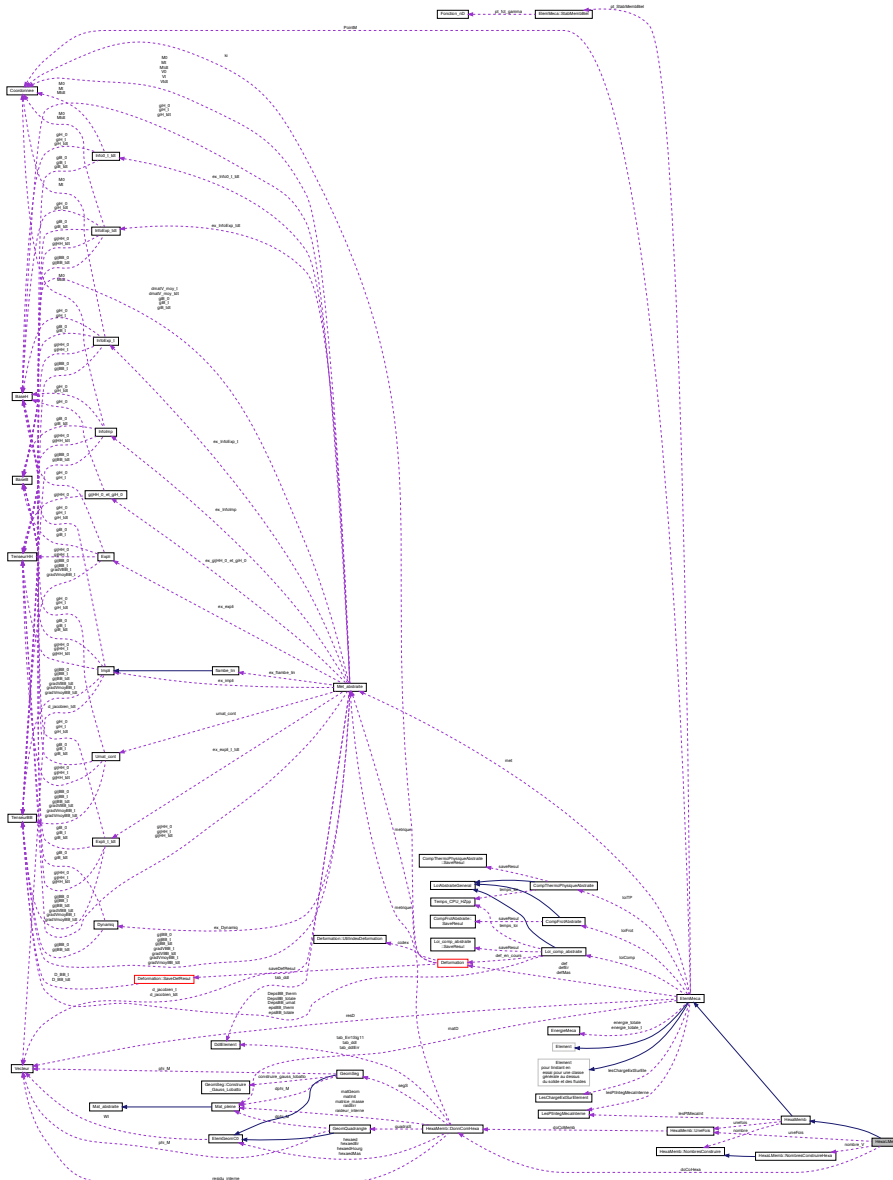
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm64pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm64pti.cc

## 6.365 Référence de la classe HexaLMemb

Grphe d'héritage de HexaLMemb:



Grphe de collaboration de HexaLMemb:



## Classes

- class [NombresConstruireHexa](#)

## Fonctions membres publiques

- [HexaLMemb](#) (int num\_id)
- [HexaLMemb](#) (int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [HexaLMemb](#) (const [HexaLMemb](#) &hexa)
- [Hexa](#) & **operator=** ([Hexa](#) &hexa)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [NombresConstruire](#) **ConstruireLesNombres** ()

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* **doCoHexa**
- static [HexaMemb::UneFois](#) **uneFois**
- static [NombresConstruireHexa](#) **nombre\_V**

## Membres hérités additionnels

### 6.365.1 Documentation des fonctions membres

#### 6.365.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaLMemb::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.365.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaLMemb::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

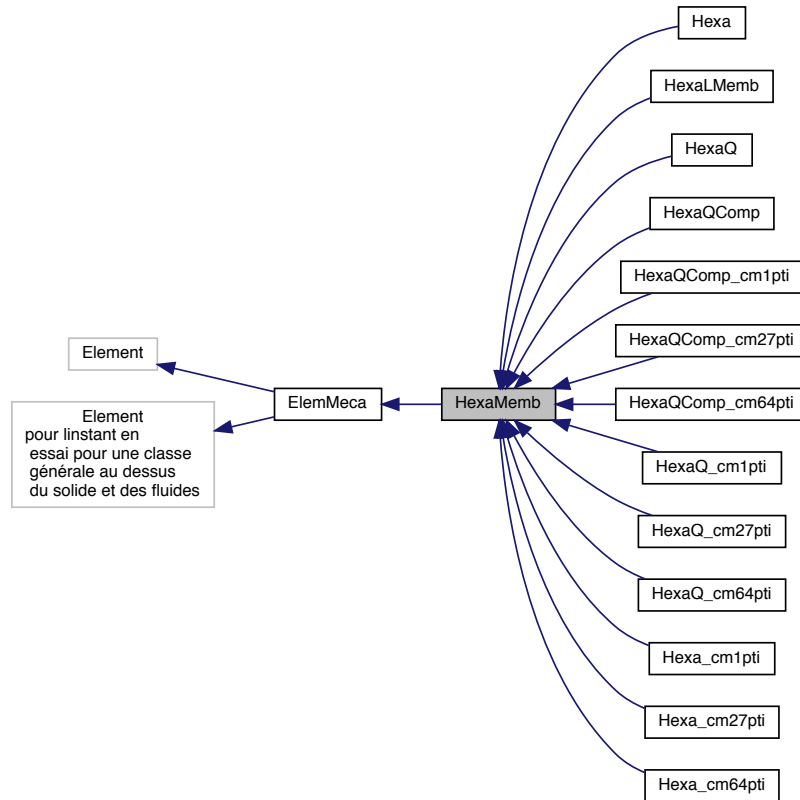
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

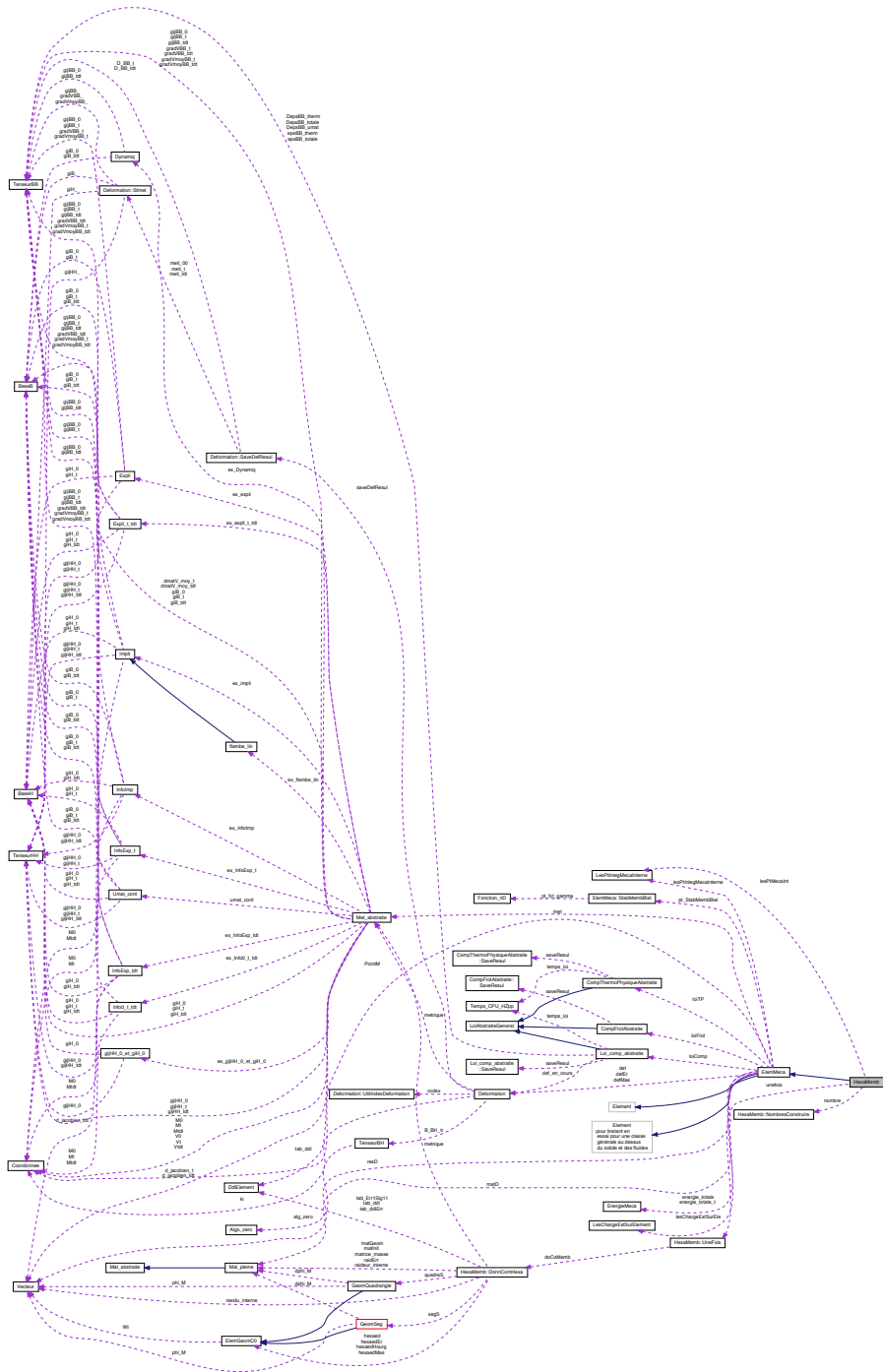
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaLMemb.h

## 6.366 Référence de la classe HexaMemb

Grphe d'héritage de HexaMemb:



Graphe de collaboration de HexaMemb:



### Classes

- class [DonnComHexa](#)
- class [NombresConstruire](#)
- class [UneFois](#)

### Fonctions membres publiques

- **HexaMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")



- **HexaMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, const [Tableau](#)< [Noeud](#) \* > &tab, string info="")
- **HexaMemb** (const [HexaMemb](#) &hexaMem)
- **HexaMemb** & **operator=** ([HexaMemb](#) &hexaMem)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- **ElemGeomC0** & **ElementGeometrique** () const
- const **ElemGeomC0** & **ElementGeometrique\_const** () const
- **Coordonnee** & **Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComplet** ()
- [Element](#) \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- [Element](#) \* **Complet\_Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- bool **SurfExiste** (int ns) const
- bool **AreteExiste** (int na) const
- const **DdlElement** & **TableauDdl** () const
- [Element::ResRaid](#) **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- [Element::Er\\_ResRaid](#) **ContrainteAuNoeud\_ResRaid** ()
- [Element::Er\\_ResRaid](#) **ErreurAuNoeud\_ResRaid** ()
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- **DdlElement** & **Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [ResRaid](#) **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [ResRaid](#) **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_presUniDir\_E\_t** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_presUniDir\_E\_tdt** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)

- ResRaid **SMR\_charge\_presUniDir\_I** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_lineique\\_E\\_t](#) (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_lineique\\_E\\_tdt](#) (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_pression\\_E\\_t](#) (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_pression\\_E\\_tdt](#) (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_hydrostatique\\_E\\_t](#) (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- [Vecteur SM\\_charge\\_hydrostatique\\_E\\_tdt](#) (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- [Vecteur SM\\_charge\\_hydrodynamique\\_E\\_t](#) ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_hydrodynamique\\_E\\_tdt](#) ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Tableau](#)< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- void **ConstTabDdl** ()

## Fonctions membres protégées

- [HexaMemb::DonnComHexa](#) \* **Init** ([ElemGeomC0](#) \*hexa, [ElemGeomC0](#) \*hexaEr, [ElemGeomC0](#) \*hexaMas, [ElemGeomC0](#) \*hexaeHourg, bool sans\_init\_noeud=false)
- void **Destruction** ()
- virtual [EIFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)=0
- virtual [EIFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)=0
- int **Dim\_sig\_eps** () const

## Attributs protégés

- [UneFois](#) \* **unefois**
- [LesPtIntegMecalInterne](#) **lesPtMecalnt**
- [NombresConstruire](#) \* **nombre**

## Membres hérités additionnels

### 6.366.1 Documentation des fonctions membres

#### 6.366.1.1 Active\_ddl\_Sigma()

void [HexaMemb::Active\\_ddl\\_Sigma](#) ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

#### 6.366.1.2 Active\_premier\_ddl\_Sigma()

void [HexaMemb::Active\\_premier\\_ddl\\_Sigma](#) ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

### 6.366.1.3 ContraintesAbsolues()

```
bool HexaMemb::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.366.1.4 Dim\_sig\_eps()

```
int HexaMemb::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.366.1.5 ErreurElement()

```
void HexaMemb::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

### 6.366.1.6 Inactive\_ddl\_Sigma()

```
void HexaMemb::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.366.1.7 LectureContraintes()

```
void HexaMemb::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.366.1.8 Long\_arrete\_mini\_sur\_c()

```
double HexaMemb::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.366.1.9 new\_frontiere\_lin()

```
virtual ElFrontiere * HexaMemb::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [protected], [pure virtual]
```

Implémente [ElemMeca](#).

### 6.366.1.10 new\_frontiere\_surf()

```
virtual ElFrontiere * HexaMemb::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [protected], [pure virtual]
```

Implémente [ElemMeca](#).

### 6.366.1.11 Plus\_ddl\_Sigma()

```
void HexaMemb::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.366.1.12 Tableau\_de\_Sig1()

```
DdlElement & HexaMemb::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

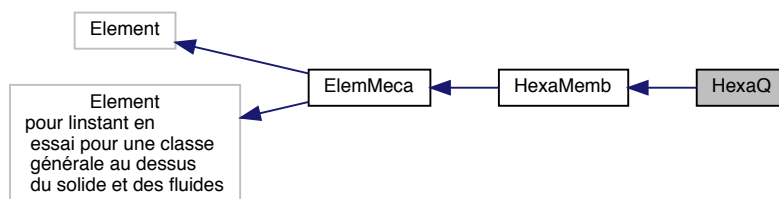
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

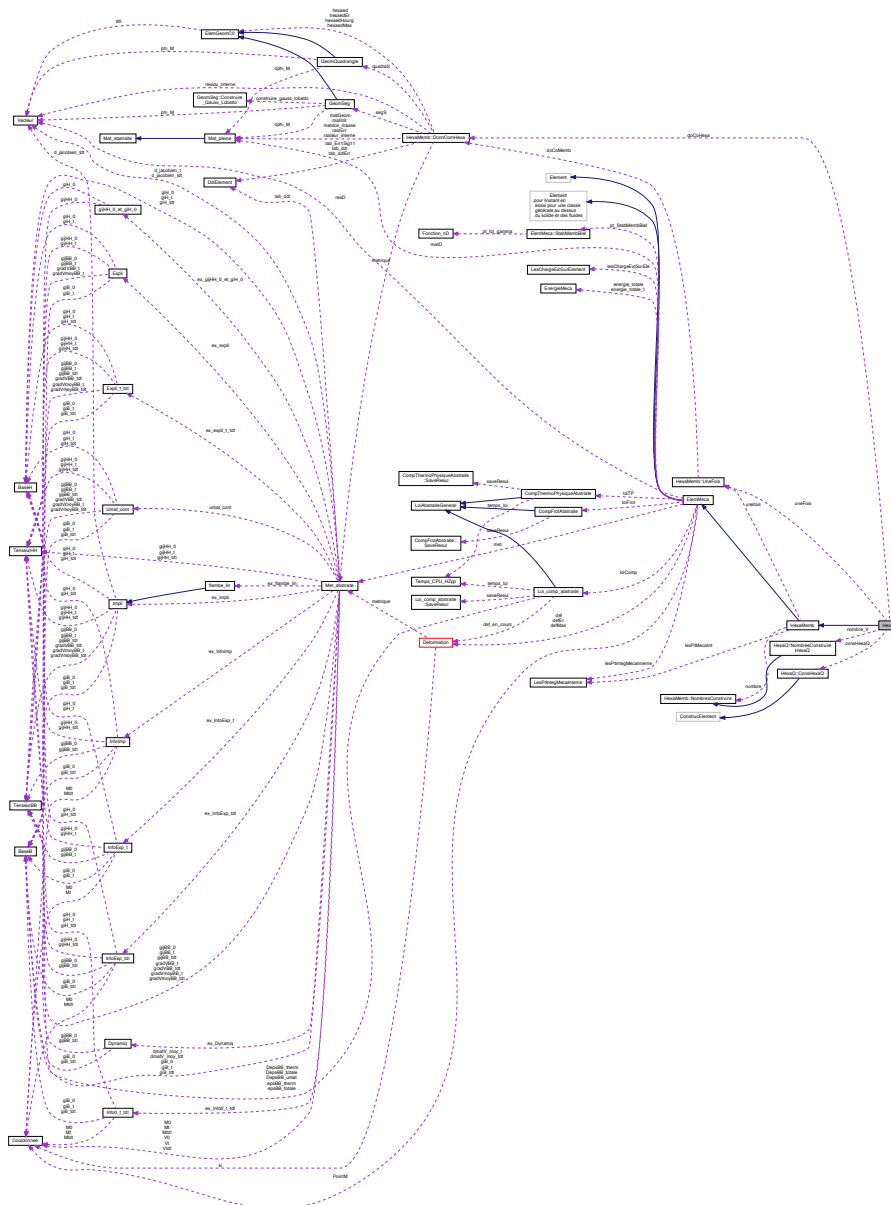
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaMemb.cc

## 6.367 Référence de la classe HexaQ

Grphe d'héritage de HexaQ:



Graphe de collaboration de HexaQ:



## Classes

- class [ConsHexaQ](#)
- class [NombresConstruireHexaQ](#)

## Fonctions membres publiques

- [HexaQ](#) (int num\_mail, int num\_id)
- [HexaQ](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [HexaQ](#) (const [HexaQ](#) &hexa)
- [Element](#) \* [Nevez\\_copie](#) () const
- [HexaQ](#) & [operator=](#) ([HexaQ](#) &hexa)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)
- list< [Noeud](#) \* > [Construct\\_from\\_lineaire](#) (const [Element](#) &elem, list< [DeuxEntiers](#) > &li\_bornes, int nbnt)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `HexaMemb::DonnComHexa` \* doCoHexa = NULL
- static `HexaMemb::UneFois` uneFois
- static `NombresConstruireHexaQ` nombre\_V
- static `ConsHexaQ` consHexaQ
- static int bidon

## Membres hérités additionnels

### 6.367.1 Documentation des fonctions membres

#### 6.367.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaQ::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.367.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaQ::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

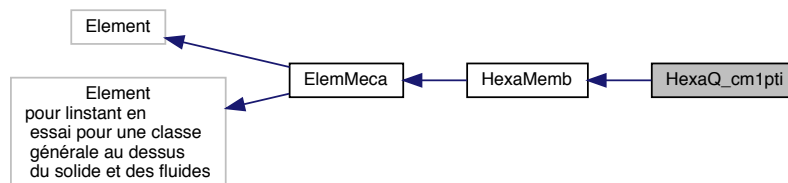
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

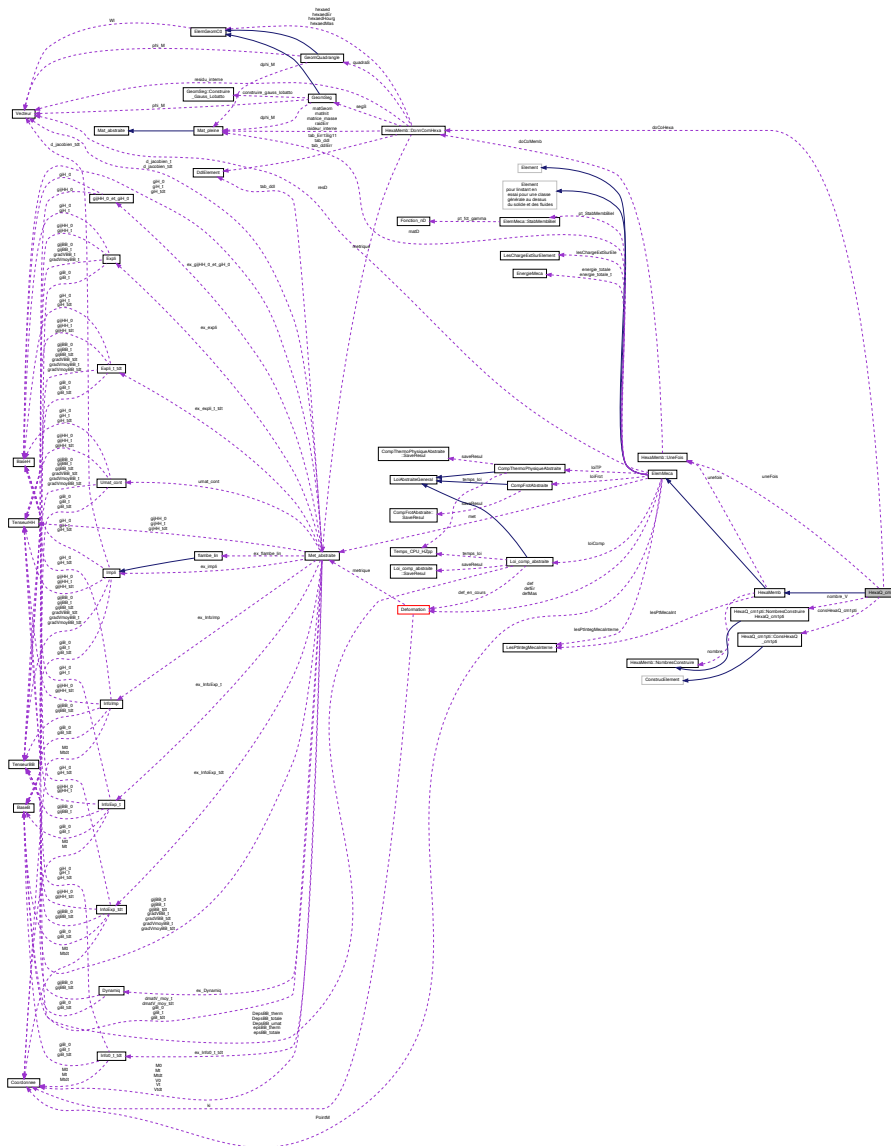
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ.cc`

## 6.368 Référence de la classe HexaQ\_cm1pti

Graphe d'héritage de HexaQ\_cm1pti:



Graphe de collaboration de HexaQ\_cm1pti:



## Classes

- class [ConsHexaQ\\_cm1pti](#)
- class [NombresConstruireHexaQ\\_cm1pti](#)

## Fonctions membres publiques

- [HexaQ\\_cm1pti](#) (int num\_mail, int num\_id)
- [HexaQ\\_cm1pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [HexaQ\\_cm1pti](#) (const [HexaQ\\_cm1pti](#) &hexa)
- [Element](#) \* [Nevez\\_copie](#) () const
- [HexaQ\\_cm1pti](#) & [operator=](#) ([HexaQ\\_cm1pti](#) &hexa)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* **doCoHexa** = NULL
- static [HexaMemb::UneFois](#) **uneFois**
- static [NombresConstruireHexaQ\\_cm1pti](#) **nombre\_V**
- static [ConsHexaQ\\_cm1pti](#) **consHexaQ\_cm1pti**
- static int **bidon**

## Membres hérités additionnels

### 6.368.1 Documentation des fonctions membres

#### 6.368.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaQ_cm1pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.368.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaQ_cm1pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

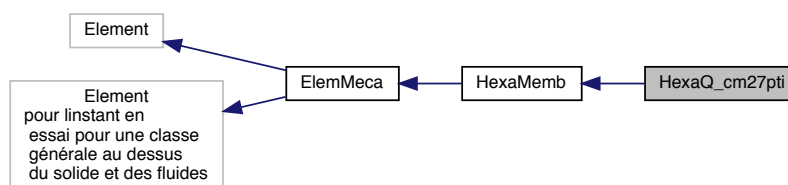
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm1pti.cc

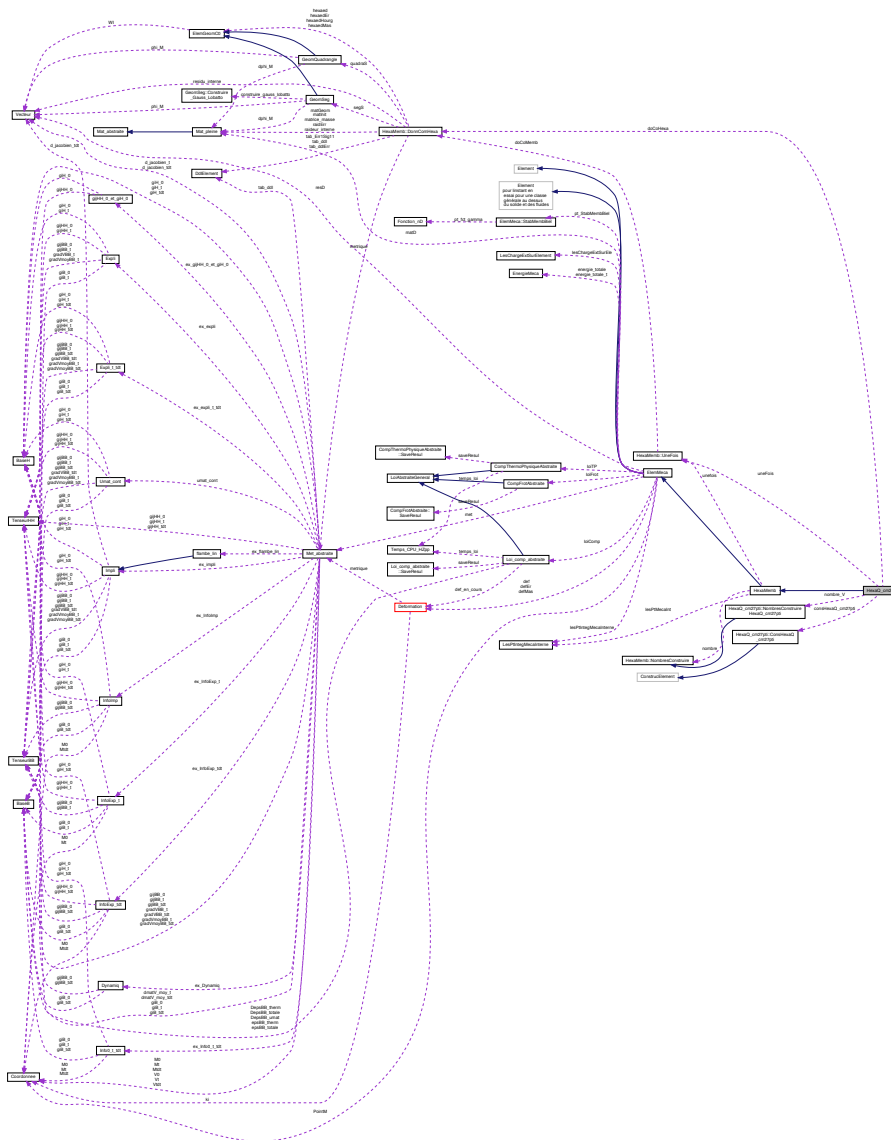
## 6.369 Référence de la classe HexaQ\_cm27pti

Graphe d'héritage de HexaQ\_cm27pti:





Graphe de collaboration de HexaQ\_cm27pti:



## Classes

- class [ConsHexaQ\\_cm27pti](#)
- class [NombresConstruireHexaQ\\_cm27pti](#)

## Fonctions membres publiques

- [HexaQ\\_cm27pti](#) (int num\_mail, int num\_id)
- [HexaQ\\_cm27pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [HexaQ\\_cm27pti](#) (const [HexaQ\\_cm27pti](#) &hexa)
- [Element](#) \* [Nevez\\_copie](#) () const
- [HexaQ\\_cm27pti](#) & [operator=](#) ([HexaQ\\_cm27pti](#) &hexa)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* [doCoHexa](#) = NULL
- static [HexaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireHexaQ\\_cm27pti](#) [nombre\\_V](#)
- static [ConsHexaQ\\_cm27pti](#) [consHexaQ\\_cm27pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.369.1 Documentation des fonctions membres

#### 6.369.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaQ_cm27pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.369.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaQ_cm27pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

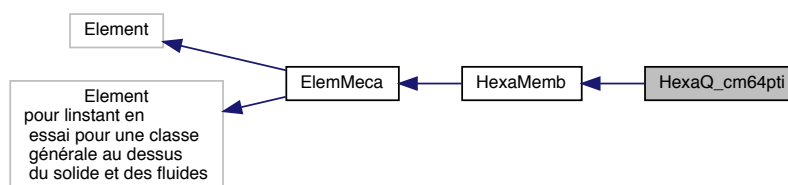
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

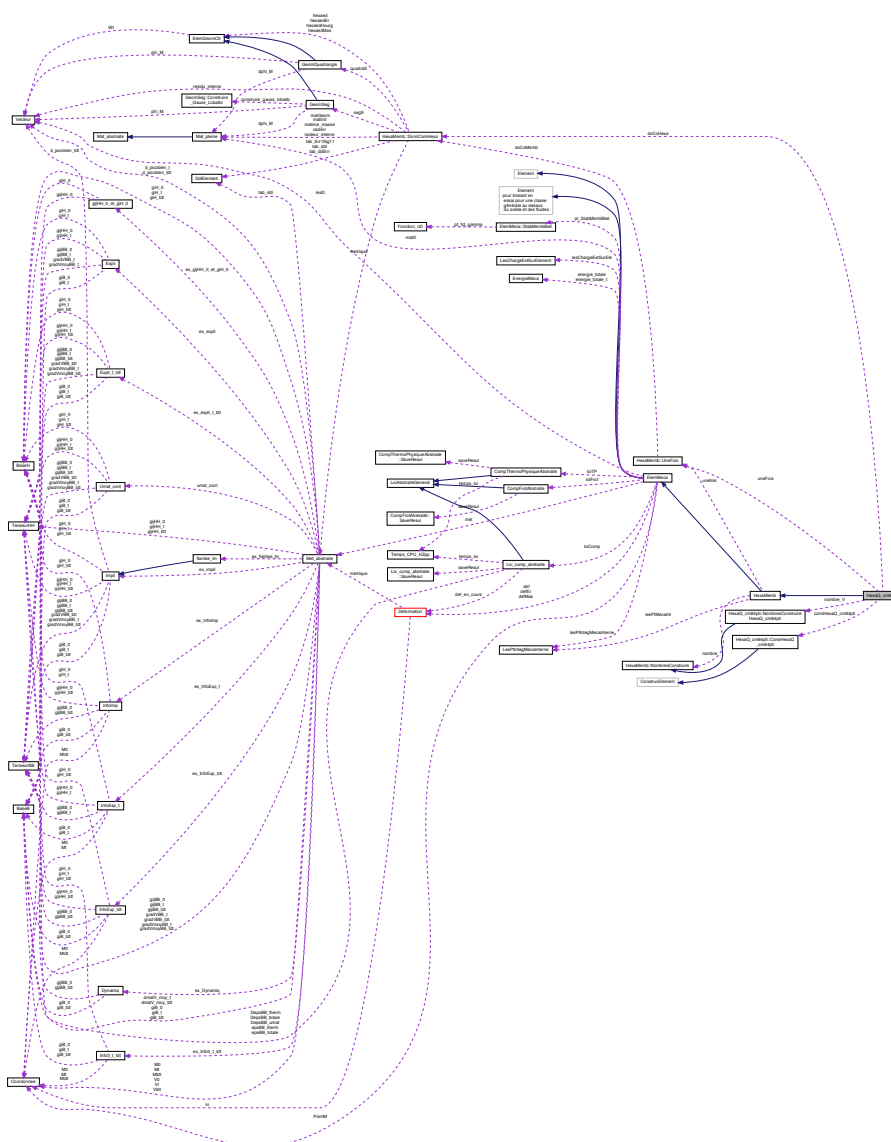
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm27pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm27pti.cc

## 6.370 Référence de la classe HexaQ\_cm64pti

Graphe d'héritage de HexaQ\_cm64pti:



Graphe de collaboration de HexaQ\_cm64pti:



## Classes

- class [ConsHexaQ\\_cm64pti](#)
- class [NombresConstruireHexaQ\\_cm64pti](#)

## Fonctions membres publiques

- [HexaQ\\_cm64pti](#) (int num\_mail, int num\_id)
- [HexaQ\\_cm64pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [HexaQ\\_cm64pti](#) (const [HexaQ\\_cm64pti](#) &hexa)
- [Element](#) \* [Nevez\\_copie](#) () const
- [HexaQ\\_cm64pti](#) & [operator=](#) ([HexaQ\\_cm64pti](#) &hexa)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* **doCoHexa** = NULL
- static [HexaMemb::UneFois](#) **uneFois**
- static [NombresConstruireHexaQ\\_cm64pti](#) **nombre\_V**
- static [ConsHexaQ\\_cm64pti](#) **consHexaQ\_cm64pti**
- static int **bidon**

## Membres hérités additionnels

### 6.370.1 Documentation des fonctions membres

#### 6.370.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaQ_cm64pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.370.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaQ_cm64pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

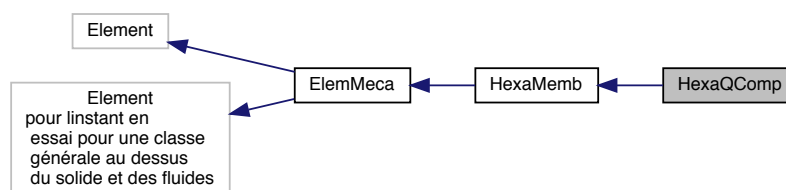
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

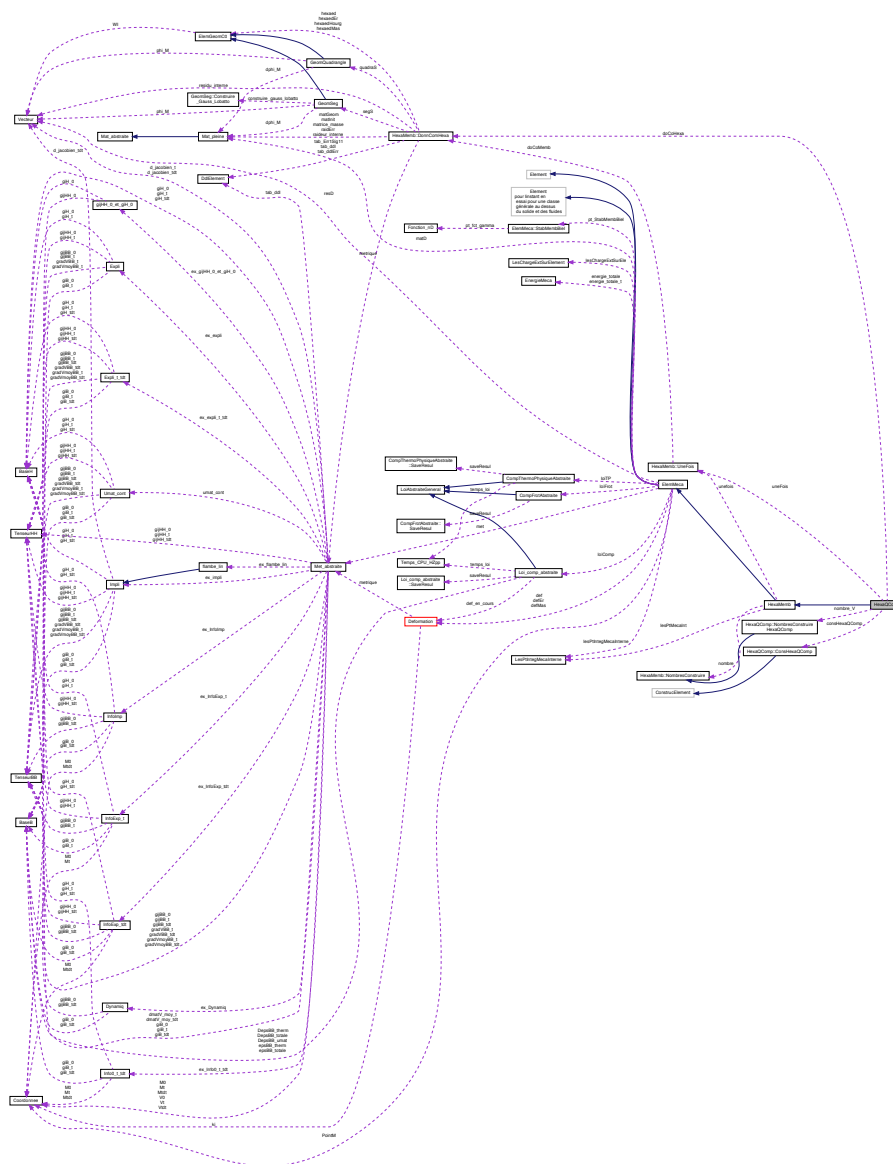
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm64pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm64pti.cc

## 6.371 Référence de la classe HexaQComp

Graphe d'héritage de HexaQComp:



Graphe de collaboration de HexaQComp:



## Classes

- class [ConsHexaQComp](#)
- class [NombresConstruireHexaQComp](#)

## Fonctions membres publiques

- **HexaQComp** (int num\_mail, int num\_id)
- **HexaQComp** (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **HexaQComp** (const [HexaQComp](#) &hexa)
- [Element](#) \* **Nevez\_copie** () const
- [HexaQComp](#) & **operator=** ([HexaQComp](#) &hexa)
- list< [Noeud](#) \* > **Construct\_from\_imcomplet** (const [Element](#) &elem, list< [DeuxEntiers](#) > &li\_bornes, int nbnt)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `HexaMemb::DonnComHexa` \* `doCoHexa` = NULL
- static `HexaMemb::UneFois` `uneFois`
- static `NombresConstruireHexaQComp` `nombre_V`
- static `ConsHexaQComp` `consHexaQComp`
- static int `bidon`

## Membres hérités additionnels

### 6.371.1 Documentation des fonctions membres

#### 6.371.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaQComp::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.371.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaQComp::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

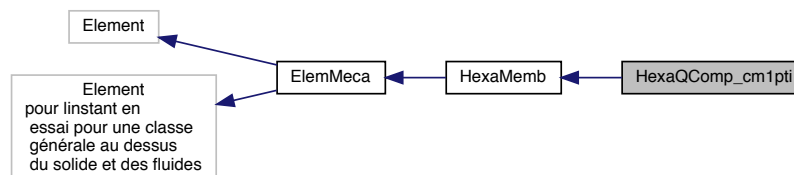
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

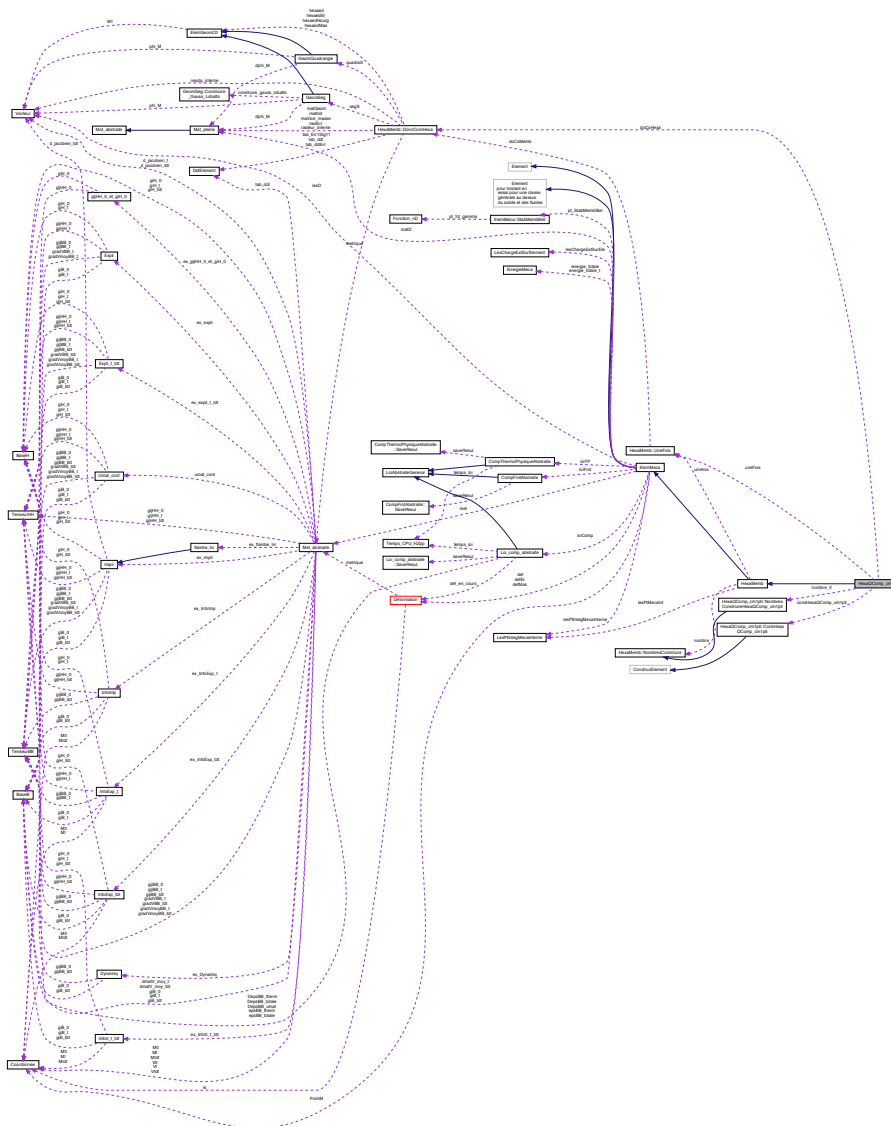
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp.cc`

## 6.372 Référence de la classe HexaQComp\_cm1pti

Grappe d'héritage de `HexaQComp_cm1pti`:



Graphe de collaboration de HexaQComp\_cm1pti:



## Classes

- class [ConsHexaQComp\\_cm1pti](#)
- class [NombresConstruireHexaQComp\\_cm1pti](#)

## Fonctions membres publiques

- [HexaQComp\\_cm1pti](#) (int num\_mail, int num\_id)
- [HexaQComp\\_cm1pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [HexaQComp\\_cm1pti](#) (const [HexaQComp\\_cm1pti](#) &hexa)
- [Element](#) \* [Nevez\\_copie](#) () const
- [HexaQComp\\_cm1pti](#) & [operator=](#) ([HexaQComp\\_cm1pti](#) &hexa)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* **doCoHexa** = NULL
- static [HexaMemb::UneFois](#) **uneFois**
- static [NombresConstruireHexaQComp\\_cm1pti](#) **nombre\_V**
- static [ConsHexaQComp\\_cm1pti](#) **consHexaQComp\_cm1pti**
- static int **bidon**

## Membres hérités additionnels

### 6.372.1 Documentation des fonctions membres

#### 6.372.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaQComp_cm1pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.372.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaQComp_cm1pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

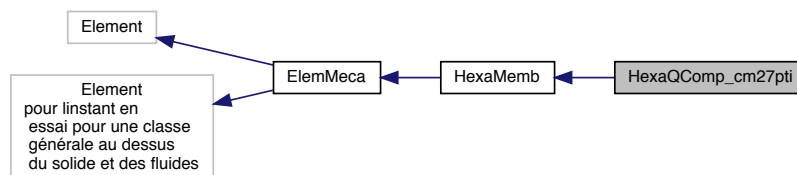
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm1pti.cc

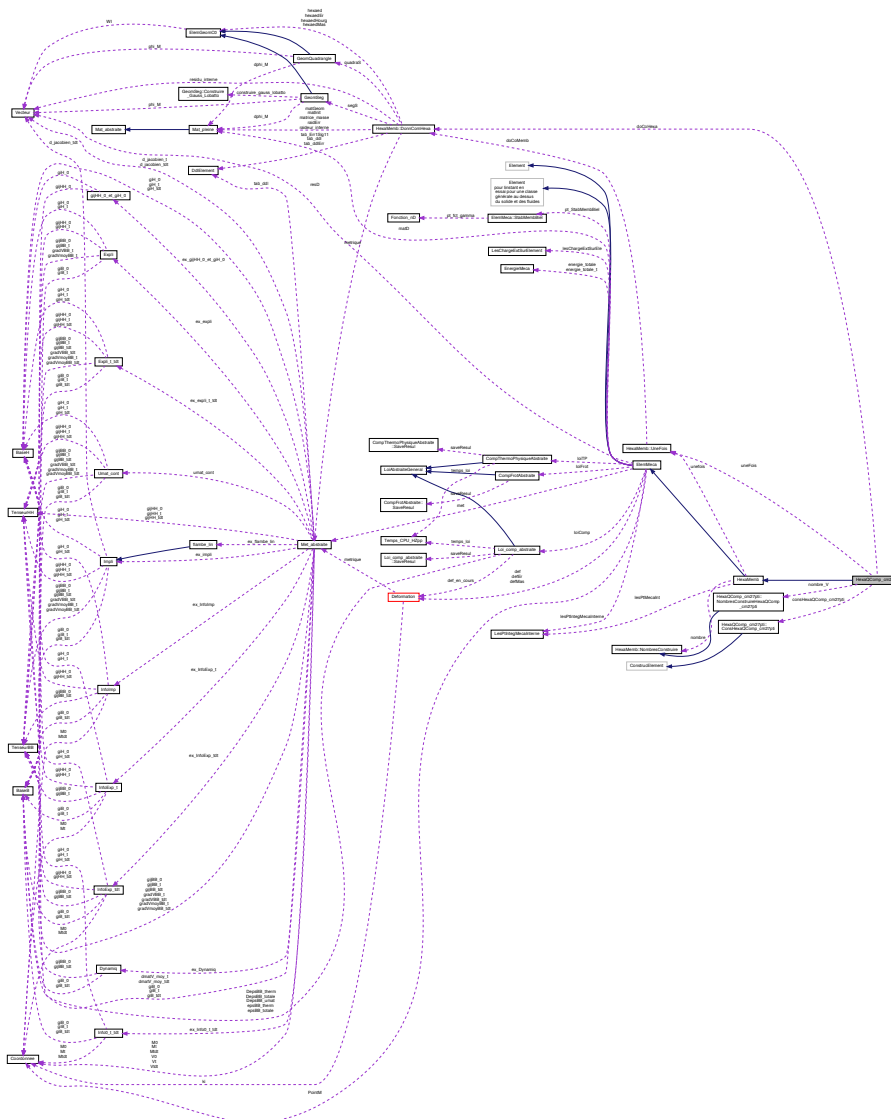
## 6.373 Référence de la classe HexaQComp\_cm27pti

Graphe d'héritage de HexaQComp\_cm27pti:





Graphe de collaboration de HexaQComp\_cm27pti:



## Classes

- class [ConsHexaQComp\\_cm27pti](#)
- class [NombresConstruireHexaQComp\\_cm27pti](#)

## Fonctions membres publiques

- [HexaQComp\\_cm27pti](#) (int num\_mail, int num\_id)
- [HexaQComp\\_cm27pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [HexaQComp\\_cm27pti](#) (const [HexaQComp\\_cm27pti](#) &hexa)
- [Element](#) \* [Nevez\\_copie](#) () const
- [HexaQComp\\_cm27pti](#) & [operator=](#) ([HexaQComp\\_cm27pti](#) &hexa)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* **doCoHexa** = NULL
- static [HexaMemb::UneFois](#) **uneFois**
- static [NombresConstruireHexaQComp\\_cm27pti](#) **nombre\_V**
- static [ConsHexaQComp\\_cm27pti](#) **consHexaQComp\_cm27pti**
- static int **bidon**

## Membres hérités additionnels

### 6.373.1 Documentation des fonctions membres

#### 6.373.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaQComp_cm27pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.373.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaQComp_cm27pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

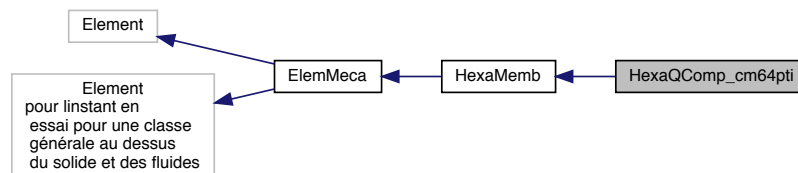
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

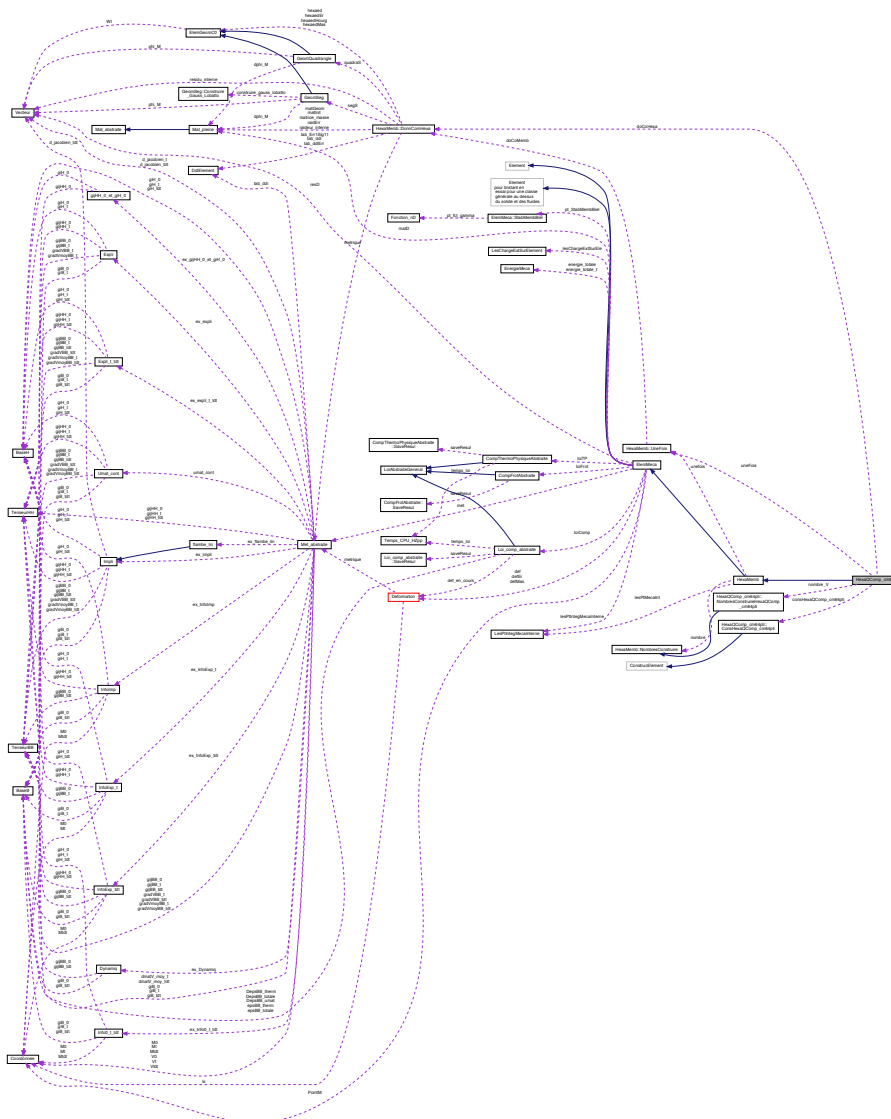
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_↵  
cm27pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_↵  
27pti.cc

## 6.374 Référence de la classe HexaQComp\_cm64pti

Graphe d'héritage de HexaQComp\_cm64pti:



Graphe de collaboration de HexaQComp\_cm64pti:



## Classes

- class [ConsHexaQComp\\_cm64pti](#)
- class [NombresConstruireHexaQComp\\_cm64pti](#)

## Fonctions membres publiques

- [HexaQComp\\_cm64pti](#) (int num\_mail, int num\_id)
- [HexaQComp\\_cm64pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [HexaQComp\\_cm64pti](#) (const [HexaQComp\\_cm64pti](#) &hexa)
- Element \* [Nevez\\_copie](#) () const
- [HexaQComp\\_cm64pti](#) & [operator=](#) ([HexaQComp\\_cm64pti](#) &hexa)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [HexaMemb::DonnComHexa](#) \* [doCoHexa](#) = NULL
- static [HexaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireHexaQComp\\_cm64pti](#) [nombre\\_V](#)
- static [ConsHexaQComp\\_cm64pti](#) [consHexaQComp\\_cm64pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.374.1 Documentation des fonctions membres

#### 6.374.1.1 new\_frontiere\_lin()

```
ElFrontiere * HexaQComp_cm64pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [HexaMemb](#).

#### 6.374.1.2 new\_frontiere\_surf()

```
ElFrontiere * HexaQComp_cm64pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

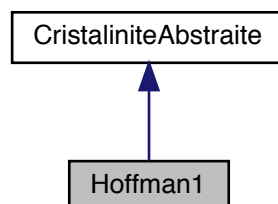
Implémente [HexaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

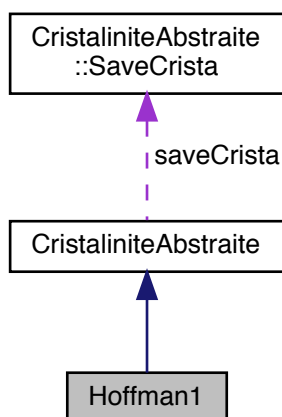
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_↔  
cm64pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_↔  
64pti.cc

## 6.375 Référence de la classe Hoffman1

Graphe d'héritage de Hoffman1:



Graphe de collaboration de Hoffman1:



## Classes

— class [SaveCrista\\_Hoffman1](#)

## Fonctions membres publiques

— **Hoffman1** (const [Hoffman1](#) &co)  
 — [SaveCrista](#) \* [New\\_et\\_Initialise](#) ()  
 — void [LectureDonneesLoiCrista](#) ([UtilLecture](#) \*entreePrinc, [LesCourbes1D](#) &, [LesFonctions\\_nD](#) &les←  
 FonctionsnD)  
 — void [Affiche](#) () const  
 — void [Info\\_commande\\_LoisCrista](#) ([UtilLecture](#) &entreePrinc)  
 — double [fct\\_KT](#) (const double &P, const double &T) const  
 — double [Cristalinite](#) (const double &P\_t, const double &T\_t, [CristaliniteAbstraite::SaveCrista](#) \*saveTP, const  
 double &P, const double &T, [Enum\\_dure](#) temps)  
 — double [Cristalinite](#) ([CristaliniteAbstraite::SaveCrista](#) \*saveTP, const double &P, const double &T)  
 — void [Lecture\\_don\\_base\\_info](#) (ifstream &, const int cas, [LesCourbes1D](#) &, [LesFonctions\\_nD](#) &)  
 — void [Ecriture\\_don\\_base\\_info](#) (ofstream &sort, const int cas) const

## Membres hérités additionnels

### 6.375.1 Documentation des fonctions membres

#### 6.375.1.1 Affiche()

void Hoffman1::Affiche ( ) const [virtual]  
 Implémente [CristaliniteAbstraite](#).

#### 6.375.1.2 Cristalinite() [1/2]

```
double Hoffman1::Cristalinite (
    const double & P_t,
    const double & T_t,
```

```
    CristaliniteAbstraite::SaveCrista * saveTP,  
    const double & P,  
    const double & T,  
    Enum_dure temps ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.375.1.3 Cristalinite() [2/2]

```
double Hoffman1::Cristalinite (   
    CristaliniteAbstraite::SaveCrista * saveTP,  
    const double & P,  
    const double & T ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.375.1.4 Ecriture\_don\_base\_info()

```
void Hoffman1::Ecriture_don_base_info (   
    ofstream & sort,  
    const int cas ) const [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.375.1.5 fct\_KT()

```
double Hoffman1::fct_KT (   
    const double & P,  
    const double & T ) const [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.375.1.6 Info\_commande\_LoisCrista()

```
void Hoffman1::Info_commande_LoisCrista (   
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.375.1.7 Lecture\_don\_base\_info()

```
void Hoffman1::Lecture_don_base_info (   
    ifstream & ent,  
    const int cas,  
    LesCourbes1D & ,  
    LesFonctions_nD & ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.375.1.8 LectureDonneesLoiCrista()

```
void Hoffman1::LectureDonneesLoiCrista (   
    UtilLecture * entreePrinc,  
    LesCourbes1D & ,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

### 6.375.1.9 New\_et\_Initialise()

```
SaveCrista * Hoffman1::New_et_Initialise ( ) [inline], [virtual]
```

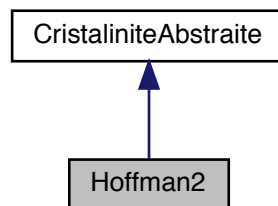
Réimplémentée à partir de [CristaliniteAbstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

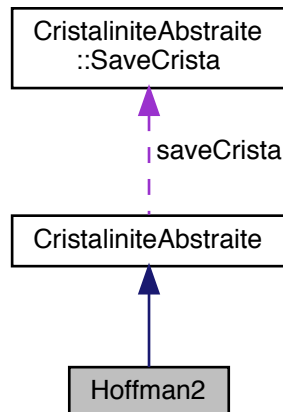
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Hoffman1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Hoffman1.cc

## 6.376 Référence de la classe Hoffman2

Graphe d'héritage de Hoffman2:



Graphe de collaboration de Hoffman2:



### Classes

- class [SaveCrista\\_Hoffman2](#)

### Fonctions membres publiques

- `Hoffman2` (const [Hoffman2](#) &co)

- `SaveCrista * New_et_Initialise ()`
- `void LectureDonneesLoiCrista (UtilLecture *entreePrinc, LesCourbes1D &, LesFonctions_nD &)`
- `void Affiche () const`
- `void Info_commande_LoisCrista (UtilLecture &entreePrinc)`
- `double fct_KT (const double &P, const double &T) const`
- `double Cristalinite (const double &P_t, const double &T_t, CristaliniteAbstraite::SaveCrista *saveTP, const double &P, const double &T, Enum_dure temps)`
- `double Cristalinite (CristaliniteAbstraite::SaveCrista *saveTP, const double &P, const double &T)`
- `void Lecture_don_base_info (ifstream &, const int cas, LesCourbes1D &, LesFonctions_nD &)`
- `void Ecriture_don_base_info (ofstream &sort, const int cas) const`

## Membres hérités additionnels

### 6.376.1 Documentation des fonctions membres

#### 6.376.1.1 Affiche()

```
void Hoffman2::Affiche ( ) const [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.376.1.2 Cristalinite() [1/2]

```
double Hoffman2::Cristalinite (
    const double & P_t,
    const double & T_t,
    CristaliniteAbstraite::SaveCrista * saveTP,
    const double & P,
    const double & T,
    Enum_dure temps ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.376.1.3 Cristalinite() [2/2]

```
double Hoffman2::Cristalinite (
    CristaliniteAbstraite::SaveCrista * saveTP,
    const double & P,
    const double & T ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.376.1.4 Ecriture\_don\_base\_info()

```
void Hoffman2::Ecriture_don_base_info (
    ofstream & sort,
    const int cas ) const [virtual]
```

Implémente [CristaliniteAbstraite](#).

#### 6.376.1.5 fct\_KT()

```
double Hoffman2::fct_KT (
    const double & P,
    const double & T ) const [virtual]
```

Implémente [CristaliniteAbstraite](#).



**6.376.1.6 Info\_commande\_LoisCrista()**

```
void Hoffman2::Info_commande_LoisCrista (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

**6.376.1.7 Lecture\_don\_base\_info()**

```
void Hoffman2::Lecture_don_base_info (
    ifstream & ent,
    const int cas,
    LesCourbes1D & ,
    LesFonctions_nD & ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

**6.376.1.8 LectureDonneesLoiCrista()**

```
void Hoffman2::LectureDonneesLoiCrista (
    UtilLecture * entreePrinc,
    LesCourbes1D & ,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [CristaliniteAbstraite](#).

**6.376.1.9 New\_et\_Initialise()**

```
SaveCrista * Hoffman2::New_et_Initialise ( ) [inline], [virtual]
```

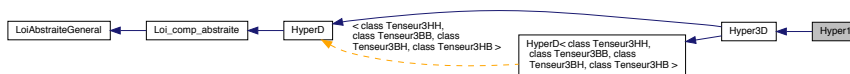
Réimplémentée à partir de [CristaliniteAbstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

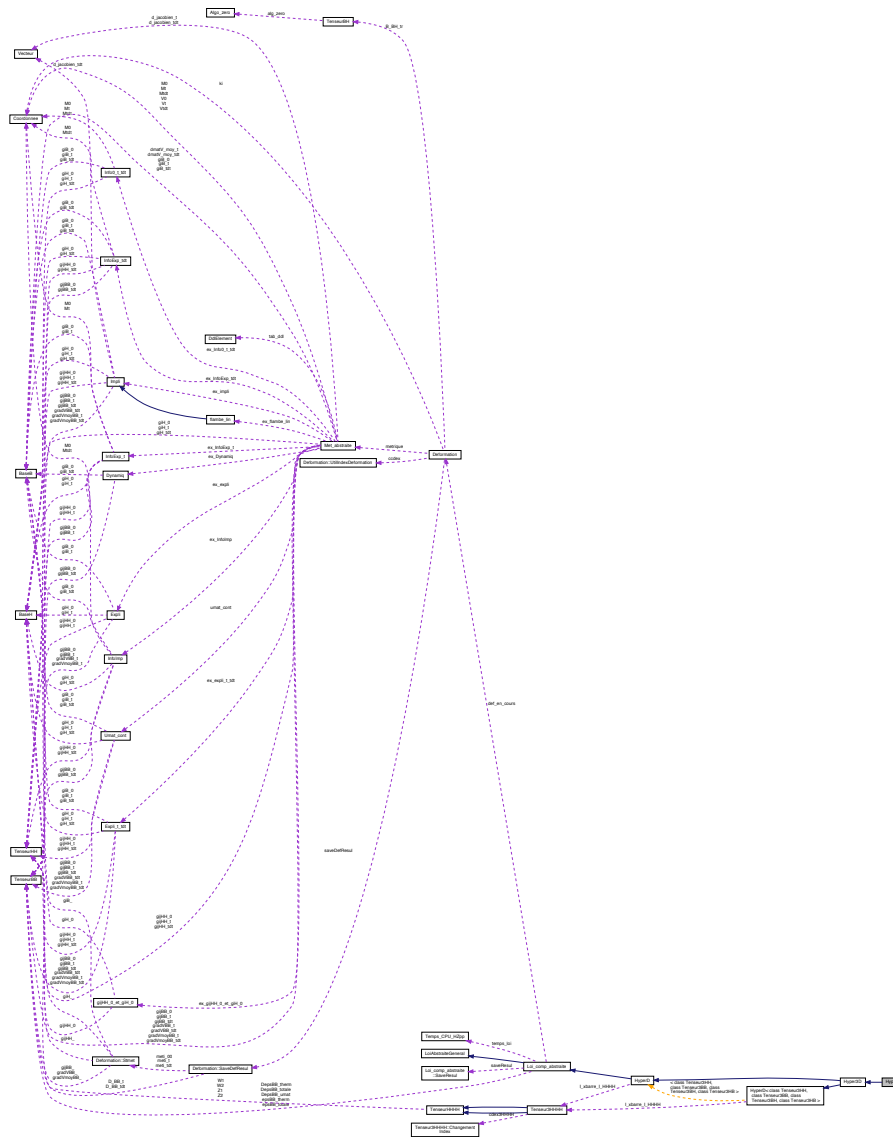
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Hoffman2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Hoffman2.cc

**6.377 Référence de la classe Hyper1**

Grappe d'héritage de Hyper1:



Graphe de collaboration de Hyper1:



## Fonctions membres publiques

- **Hyper1** (const [Hyper1](#) &loi)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*)
- void **Affiche** ()
- int **TestComple** ()
- void **Alpha** (double &E, double &EV, double &EQeps, double &leps, double &V, double &Qeps, double &alpha\_0, double &alpha\_1, double &alpha\_2)
- void **Alpha\_var** (double &E, [Tableau](#)< double > &dE, double &EV, [Tableau](#)< double > &dEV, double &EQeps, [Tableau](#)< double > &dEQeps, double &EVV, [Tableau](#)< double > &dEVV, double &EQQ, [Tableau](#)< double > &dEQQ, double &EVQ, [Tableau](#)< double > &dEVQ, double &leps, [Tableau](#)< double > &d←leps, double &V, [Tableau](#)< double > &dV, double &Qeps, [Tableau](#)< double > &dQeps, double &alpha←\_0, [Tableau](#)< double > &dalpha\_0, double &alpha\_1, [Tableau](#)< double > &dalpha\_1, double &alpha\_2, [Tableau](#)< double > &dalpha\_2)
- void **Potentiel** (double &leps, double &V, double &Qeps, double &E, double &EV, double &EQeps)
- void **PotentielPhase** (double &leps, double &V, double &Qeps, double &cos3phi, double &sin3phi, double &E, double &EV, double &EQeps, double &EPhi)
- void **Potentiel\_et\_var** (double &leps, [Tableau](#)< double > &dleps, double &V, [Tableau](#)< double > &dV, double &Qeps, [Tableau](#)< double > &dQeps, double &E, [Tableau](#)< double > &dE, double &EV, [Tableau](#)<

double > &dEV, double &EQeps, [Tableau](#)< double > &dEQeps, double &EVV, [Tableau](#)< double > &dEVV, double &EQQ, [Tableau](#)< double > &dEQQ, double &EVQ, [Tableau](#)< double > &dEVQ)

- void **PotentielPhase\_et\_var** (double &leps, [Tableau](#)< double > &dleps, double &V, [Tableau](#)< double > &dV, double &Qeps, [Tableau](#)< double > &dQeps, double &cos3phi, [Tableau](#)< double > &dcos3phi, double &sin3phi, [Tableau](#)< double > &dsin3phi, double &E, [Tableau](#)< double > &dE, double &EV, [Tableau](#)< double > &dEV, double &EQeps, [Tableau](#)< double > &dEQeps, double &EPhi, [Tableau](#)< double > &dEPhi, double &EVV, [Tableau](#)< double > &dEVV, double &EQQ, [Tableau](#)< double > &dEQQ, double &EVQ, [Tableau](#)< double > &dEVQ, double &Ephphi, [Tableau](#)< double > &dEphphi, double &EQphi, [Tableau](#)< double > &dEQphi, double &EVphi, [Tableau](#)< double > &dEVphi)

## Attributs publics

- double **K**
- double **Qor**
- double **mur**
- double **mu\_inf**

## Membres hérités additionnels

### 6.377.1 Documentation des fonctions membres

#### 6.377.1.1 TestComplet()

```
int Hyper1::TestComplet ( ) [virtual]
```

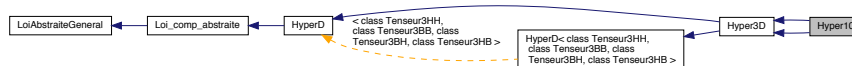
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

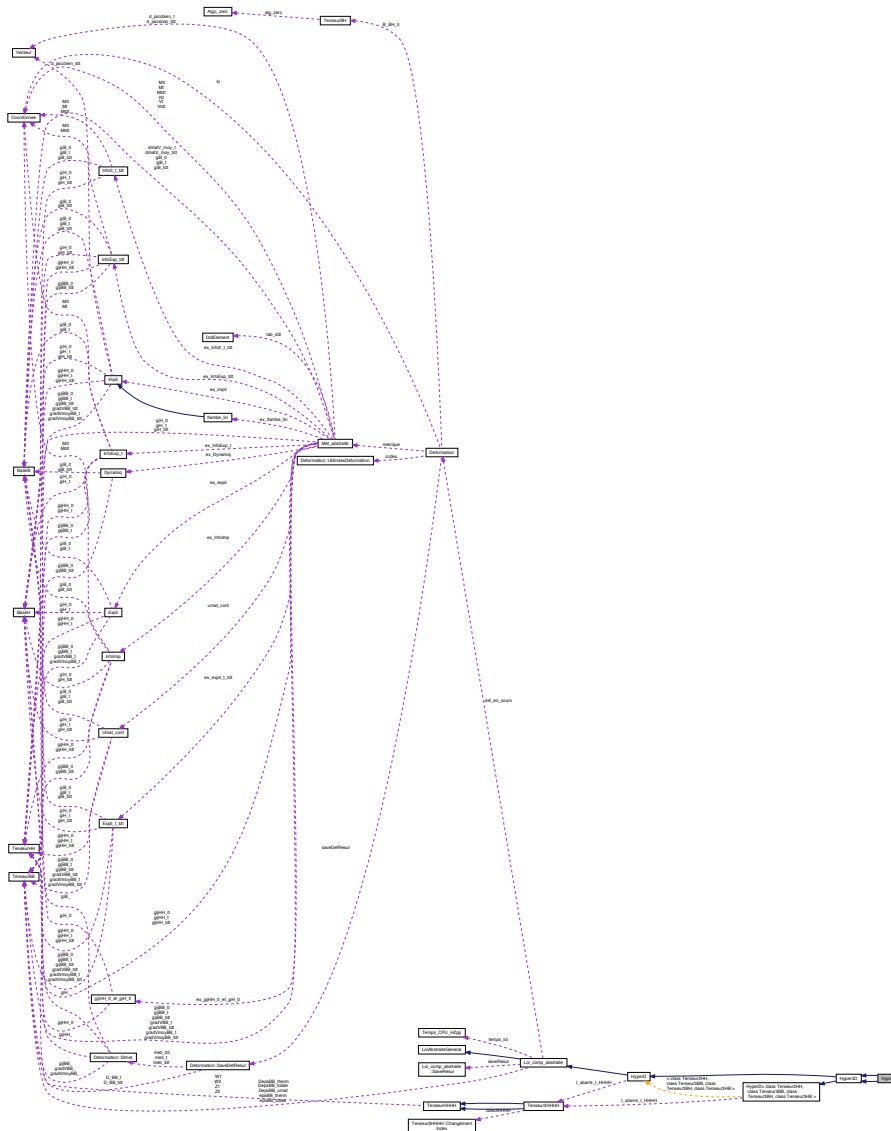
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper1.cc

## 6.378 Référence de la classe Hyper10

Graphe d'héritage de Hyper10:



Graphe de collaboration de Hyper10:



## Fonctions membres publiques

- **Hyper10** (const [Hyper10](#) &loi)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*)
- void **Affiche** ()
- int **TestComple** ()
- void **Potentiel** (double &leps, double &V, double &Qeps, double &E, double &EV, double &EQeps)
- void **PotentielPhase** (double &leps, double &V, double &Qeps, double &cos3phi, double &sin3phi, double &E, double &EV, double &EQeps, double &EPhi)
- void **Potentiel\_et\_var** (double &leps, [Tableau](#)< double > &dleps, double &V, [Tableau](#)< double > &dV, double &Qeps, [Tableau](#)< double > &dQeps, double &E, [Tableau](#)< double > &dE, double &EV, [Tableau](#)< double > &dEV, double &EQeps, [Tableau](#)< double > &dEQeps, double &EVV, [Tableau](#)< double > &dEVV, double &EQQ, [Tableau](#)< double > &dEQQ, double &EVQ, [Tableau](#)< double > &dEVQ)
- void **PotentielPhase\_et\_var** (double &leps, [Tableau](#)< double > &dleps, double &V, [Tableau](#)< double > &dV, double &Qeps, [Tableau](#)< double > &dQeps, double &cos3phi, [Tableau](#)< double > &dcos3phi, double &sin3phi, [Tableau](#)< double > &dsin3phi, double &E, [Tableau](#)< double > &dE, double &EV, [Tableau](#)< double > &dEV, double &EQeps, [Tableau](#)< double > &dEQeps, double &EPhi, [Tableau](#)< double > &dEPhi, double &EVV, [Tableau](#)< double > &dEVV, double &EQQ, [Tableau](#)< double > &dEQQ, double &EVQ, [Tableau](#)<

- double > &dEVQ, double &Ephiphi, [Tableau](#)< double > &dEphiphi, double &EQphi, [Tableau](#)< double > &dEQphi, double &EVphi, [Tableau](#)< double > &dEVphi)
- **Hyper10** (const [Hyper10](#) &loi)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*)
- void **Affiche** ()
- int **TestComple** ()
- void **Potentiel** (double &leps, double &V, double &Qeps, double &E, double &EV, double &EQeps)
- void **PotentielPhase** (double &leps, double &V, double &Qeps, double &cos3phi, double &sin3phi, double &E, double &EV, double &EQeps, double &EPhi)
- void **Potentiel\_et\_var** (double &leps, [Tableau](#)< double > &dleps, double &V, [Tableau](#)< double > &dV, double &Qeps, [Tableau](#)< double > &dQeps, double &E, [Tableau](#)< double > &dE, double &EV, [Tableau](#)< double > &dEV, double &EQeps, [Tableau](#)< double > &dEQeps, double &EVV, [Tableau](#)< double > &dEVV, double &EQQ, [Tableau](#)< double > &dEQQ, double &EVQ, [Tableau](#)< double > &dEVQ)
- void **PotentielPhase\_et\_var** (double &leps, [Tableau](#)< double > &dleps, double &V, [Tableau](#)< double > &dV, double &Qeps, [Tableau](#)< double > &dQeps, double &cos3phi, [Tableau](#)< double > &dcos3phi, double &sin3phi, [Tableau](#)< double > &dsin3phi, double &E, [Tableau](#)< double > &dE, double &EV, [Tableau](#)< double > &dEV, double &EQeps, [Tableau](#)< double > &dEQeps, double &EPhi, [Tableau](#)< double > &dEPhi, double &EVV, [Tableau](#)< double > &dEVV, double &EQQ, [Tableau](#)< double > &dEQQ, double &EVQ, [Tableau](#)< double > &dEVQ, double &Ephiphi, [Tableau](#)< double > &dEphiphi, double &EQphi, [Tableau](#)< double > &dEQphi, double &EVphi, [Tableau](#)< double > &dEVphi)

### Attributs publics

- double **K**
- double **Qor**
- double **mur**
- double **mu\_inf**

### Membres hérités additionnels

#### 6.378.1 Documentation des fonctions membres

##### 6.378.1.1 TestComple() [1/2]

`int Hyper10::TestComple ( ) [virtual]`  
 Réimplémentée à partir de [LoiAbstraiteGeneral](#).

##### 6.378.1.2 TestComple() [2/2]

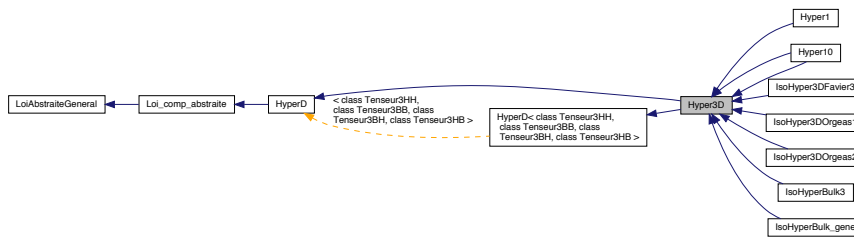
`int Hyper10::TestComple ( ) [virtual]`  
 Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

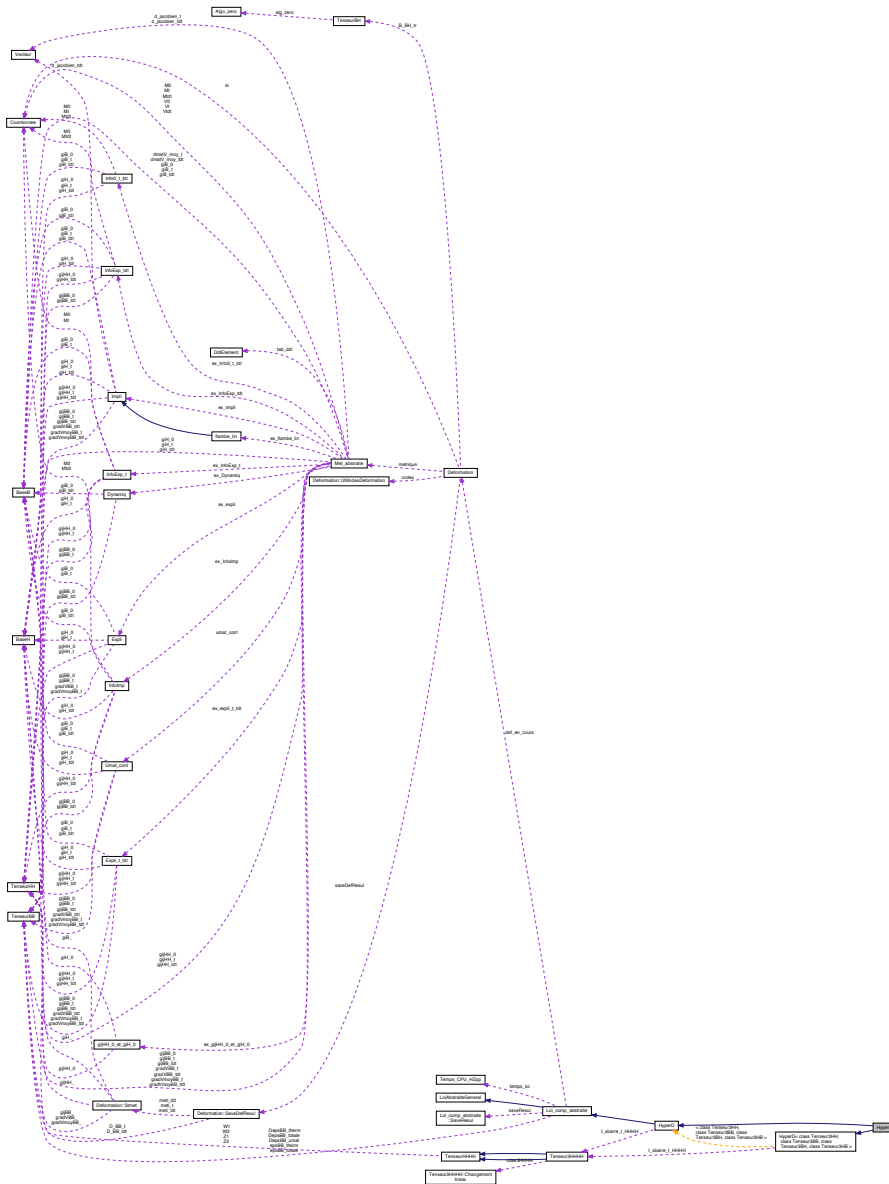
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper10.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_w1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper10.cc

### 6.379 Référence de la classe Hyper3D

Graphe d'héritage de Hyper3D:



Graphe de collaboration de Hyper3D:



## Classes

- class [Invariant0QepsCosphi](#)
- class [Invariant2Qeps](#)
- class [Invariant2QepsCosphi](#)
- class [InvariantQeps](#)
- class [InvariantQepsCosphi](#)
- class [PoGrenoble\\_V](#)
- class [PoGrenoble\\_VV](#)
- class [PoGrenobleAvecPhaseAvecVar](#)
- class [PoGrenobleAvecPhaseSansVar](#)
- class [PoGrenobleSansPhaseAvecVar](#)
- class [PoGrenobleSansPhaseSansVar](#)

## Fonctions membres publiques

- [Hyper3D](#) ([Enum\\_comp](#) id\_compor, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, bool avec\_ph)
- [Hyper3D](#) (char \*nom, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, bool avec\_ph)
- [Hyper3D](#) (const [Hyper3D](#) &loi)
- [TenseurBH \\* Invariants](#) (const [TenseurBB](#) &epsBB\_t, const [TenseurBB](#) &gijBB\_t, const [TenseurHH](#) &gijHH\_t, const double &jacobien\_0, const double &jacobien\_t, [Invariant](#) &invariant, [TenseurBH](#) &epsBH)
- [TenseurBH \\* Invariants\\_et\\_var](#) (const [TenseurBB](#) &epsBB\_tdt, const [TenseurBB](#) &gijBB\_tdt, const [Tableau](#)<[TenseurBB](#) \* > &d\_gijBB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const [Tableau](#)<[TenseurHH](#) \* > &d\_gijHH\_tdt, const double &jacobien\_0, const double &jacobien\_tdt, const [Vecteur](#) &d\_jacobien\_tdt, [InvariantVarDdl](#) &invariantVarDdl, [TenseurBH](#) &epsBH\_tdt, [Tableau](#)<[TenseurBH](#) \* > &depsBH\_tdt)
- [TenseurBH \\* Invariants\\_et\\_varEps](#) (const [TenseurBB](#) &epsBB\_tdt, const [TenseurBB](#) &gijBB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien\_tdt, [InvariantVarEps](#) &invariantVarEps, [TenseurBH](#) &epsBH\_tdt)
- [PotenSansPhaseSansVar Potentiel](#) (const [Invariant](#) &invariant, const double &jacobien0)
- [PotenAvecPhaseSansVar PotentielPhase](#) (const [Invariant](#) &invariant, const double &jacobien0)
- [PotenSansPhaseAvecVar Potentiel\\_et\\_var](#) (const [Invariant](#) &invariant, const double &jacobien0)
- [PotenAvecPhaseAvecVar PotentielPhase\\_et\\_var](#) (const [Invariant](#) &invariant, const double &jacobien0)
- [PotenSansPhaseAvecVar Potentiel\\_et\\_varEps](#) (const [Invariant](#) &invariant, const double &jacobien0)
- [Hyper3D](#) ([Enum\\_comp](#) id\_compor, bool avec\_ph)
- [Hyper3D](#) (char \*nom, bool avec\_ph)
- [Hyper3D](#) (const [Hyper3D](#) &loi)
- [Tenseur3BH \\* Invariants](#) ([TenseurBB](#) &epsBB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, double &jacobien\_0, double &jacobien\_t, double &leps, double &V, double &Qeps, [Tenseur3BH](#) &epsBH)
- [Tenseur3BH \\* InvariantsPhase](#) ([TenseurBB](#) &epsBB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, double &jacobien\_0, double &jacobien\_t, double &leps, double &V, double &Qeps, double &cos3phi, double &sin3phi, [Tenseur3BH](#) &epsBH)
- [Tenseur3BH \\* Invariants\\_et\\_var](#) ([TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &gijBB\_tdt, [Tableau](#)<[TenseurBB](#) \* > &d\_gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)<[TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien\_tdt, [Vecteur](#) &d\_jacobien\_tdt, double &leps, [Tableau](#)<double > &dleps, double &V, [Tableau](#)<double > &dV, double &Qeps, [Tableau](#)<double > &dQeps, [Tenseur3BH](#) &epsBH, [Tableau](#)<[Tenseur3BH](#) > &depsBH)
- [Tenseur3BH \\* InvariantsPhase\\_et\\_var](#) ([TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &gijBB\_tdt, [Tableau](#)<[TenseurBB](#) \* > &d\_gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)<[TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien\_tdt, [Vecteur](#) &d\_jacobien\_tdt, double &leps, [Tableau](#)<double > &dleps, double &V, [Tableau](#)<double > &dV, double &Qeps, [Tableau](#)<double > &dQeps, double &cos3phi, [Tableau](#)<double > &dcos3phi, double &sin3phi, [Tableau](#)<double > &dsin3phi, [Tenseur3BH](#) &epsBH, [Tableau](#)<[Tenseur3BH](#) > &depsBH)

## Fonctions membres protégées

- double [Invariant0Qeps](#) (const [Invariant](#) &invariant)
- [Invariant0QepsCosphi Invariant0Specif](#) (const [Invariant](#) &invariant)
- [Invariant0QepsCosphi InvariantDe\\_V\\_bllb\\_leps](#) (const [Invariant](#) &invariant)
- [InvariantQeps InvariantSpecifSansPhase](#) (const [Invariant](#) &invariant)
- [InvariantQepsCosphi InvariantSpecif](#) (const [Invariant](#) &invariant)
- [Invariant2Qeps Invariant2SpecifSansPhase](#) (const [Invariant](#) &invariantVarDdl)
- [Invariant2QepsCosphi Invariant2Specif](#) (const [Invariant](#) &invariantVarDdl)

- virtual double `PoGrenoble` (const double &Qeps, const `Invariant` &invariant)=0  
*les fonctions potentielles sont défini dans les classes dérivées*
- virtual double `PoGrenoble` (const `Invariant0QepsCosphi` &inv, const `Invariant` &invariant)=0
- virtual `PoGrenoble_V PoGrenoble_et_V` (const double &Qeps, const `Invariant` &invariant)=0
- virtual `PoGrenoble_V PoGrenoble_et_V` (const `Invariant0QepsCosphi` &inv, const `Invariant` &invariant)=0
- virtual `PoGrenoble_VV PoGrenoble_et_VV` (const double &Qeps, const `Invariant` &invariant)=0
- virtual `PoGrenoble_VV PoGrenoble_et_VV` (const `Invariant0QepsCosphi` &inv, const `Invariant` &invariant)=0
- virtual `PoGrenobleSansPhaseSansVar PoGrenoble` (const `InvariantQeps` &inv, const `Invariant` &invariant)=0
- virtual `PoGrenobleAvecPhaseSansVar PoGrenoblePhase` (const `InvariantQepsCosphi` &inv, const `Invariant` &invariant)=0
- virtual `PoGrenobleSansPhaseAvecVar PoGrenoble_et_var` (const `Invariant2Qeps` &inv, const `Invariant` &invariantVarDdl)=0
- virtual `PoGrenobleAvecPhaseAvecVar PoGrenoblePhase_et_var` (const `Invariant2QepsCosphi` &inv, const `Invariant` &invariantVarDdl)=0
- void `Verif_Potentiel_et_var` (const `InvariantVarDdl` &inv, const double &jacobien0, const `PotenSansPhaseAvecVar` &potret)
- ===== fonctions pour la vérification et la mise au point =====
- void `Verif_Potentiel_et_var` (const `InvariantVarDdl` &inv, const double &jacobien0, const `PotenAvecPhaseAvecVar` &potret)

### Attributs protégés statiques

- static int `indic_Verif_Potentiel_et_var` = 0
- static double `limite_inf_Qeps` = sqrt(2)\*6.e-5
- static double `limite_inf_bllb` = 36.e-10

### Membres hérités additionnels

#### 6.379.1 Documentation des fonctions membres

##### 6.379.1.1 `InvariantDe_V_bllb_leps()`

```
Hyper3D::Invariant0QepsCosphi Hyper3D::InvariantDe_V_bIIb_Ieps (
    const Invariant & invariant ) [protected]
```

\*\*\*\*\* en fait ça merdouille, le bllb calculé donne n'importe quoi quand c'est petit donc a éviter!!! a défaut d'autre chose pour l'instant on utilise `Invariant0Specif`

##### 6.379.1.2 `Invariants()`

```
TenseurBH * Hyper3D::Invariants (
    const TenseurBB & epsBB_t,
    const TenseurBB & gijBB_t,
    const TenseurHH & gijHH_t,
    const double & jacobien_0,
    const double & jacobien_t,
    Invariant & invariant,
    TenseurBH & epsBH ) [virtual]
```

Implémente `HyperD`.

##### 6.379.1.3 `Invariants_et_var()`

```
TenseurBH * Hyper3D::Invariants_et_var (
    const TenseurBB & epsBB_tdt,
    const TenseurBB & gijBB_tdt,
    const Tableau< TenseurBB * > & d_gijBB_tdt,
    const TenseurHH & gijHH_tdt,
    const Tableau< TenseurHH * > & d_gijHH_tdt,
```



```

    const double & jacobien_0,
    const double & jacobien_tdt,
    const Vecteur & d_jacobien_tdt,
    InvariantVarDdl & invariantVarDdl,
    TenseurBH & epsBH_tdt,
    Tableau< TenseurBH * > & depsBH_tdt ) [virtual]

```

pour info la bonne expression de blllb est donnée par tutu!!

Implémente [HyperD](#).

#### 6.379.1.4 Invariants\_et\_varEps()

```

TenseurBH * Hyper3D::Invariants_et_varEps (
    const TenseurBB & epsBB_tdt,
    const TenseurBB & gijBB_tdt,
    const TenseurHH & gijHH_tdt,
    const double & jacobien_0,
    const double & jacobien_tdt,
    InvariantVarEps & invariantVarEps,
    TenseurBH & epsBH_tdt ) [virtual]

```

Implémente [HyperD](#).

#### 6.379.1.5 PoGrenoble()

```

virtual double Hyper3D::PoGrenoble (
    const double & Qeps,
    const Invariant & invariant ) [protected], [pure virtual]

```

les fonctions potentielles sont défini dans les classes dérivées

Implémenté dans [IsoHyper3DFavier3](#), [IsoHyper3DOrgeas1](#), [IsoHyper3DOrgeas2](#), [IsoHyperBulk3](#), et [IsoHyperBulk\\_gene](#).

#### 6.379.1.6 Potentiel()

```

Hyper3D::PotenSansPhaseSansVar Hyper3D::Potentiel (
    const Invariant & invariant,
    const double & jacobien0 ) [virtual]

```

Implémente [HyperD](#).

#### 6.379.1.7 Potentiel\_et\_var()

```

Hyper3D::PotenSansPhaseAvecVar Hyper3D::Potentiel_et_var (
    const Invariant & invariant,
    const double & jacobien0 ) [virtual]

```

Implémente [HyperD](#).

#### 6.379.1.8 PotentielPhase()

```

Hyper3D::PotenAvecPhaseSansVar Hyper3D::PotentielPhase (
    const Invariant & invariant,
    const double & jacobien0 ) [virtual]

```

Implémente [HyperD](#).

#### 6.379.1.9 PotentielPhase\_et\_var()

```

Hyper3D::PotenAvecPhaseAvecVar Hyper3D::PotentielPhase_et_var (

```

```

const Invariant & invariant,
const double & jacobien0 ) [virtual]

```

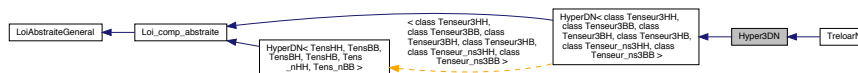
Implémente [HyperD](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D\_save.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D\_save.cc

## 6.380 Référence de la classe Hyper3DN

Graphe d'héritage de Hyper3DN:



Graphe de collaboration de Hyper3DN:



## Fonctions membres publiques

- **Hyper3DN** ([Enum\\_comp](#) id\_comp, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp)
- **Hyper3DN** (char \*nom, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp)
- **Hyper3DN** (const [Hyper3DN](#) &loi)
- **Tenseur3BH** \* [Invariants](#) ([TenseurBB](#) &epsBB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, double &jacobien\_0, double &jacobien\_t, double &leps, double &V, double &bllb, [Tenseur3BH](#) &epsBH)
- **Tenseur3BH** \* [Invariants\\_et\\_var](#) ([TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &gijBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien\_tdt, [Vecteur](#) &d\_jacobien\_tdt, double &leps, [Tableau](#)< double > &dleps, double &V, [Tableau](#)< double > &dV, double &bllb, [Tableau](#)< double > &dbllb, [Tenseur3BH](#) &epsBH, [Tableau](#)< [Tenseur3BH](#) > &depsBH)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const

## Membres hérités additionnels

### 6.380.1 Documentation des fonctions membres

#### 6.380.1.1 HsurH0()

```
virtual double Hyper3DN::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.380.1.2 Invariants()

```
Tenseur3BH * Hyper3DN::Invariants (
    TenseurBB & epsBB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    double & jacobien_0,
    double & jacobien_t,
    double & Ieps,
    double & V,
    double & bIIb,
    Tenseur3BH & epsBH ) [virtual]
```

Implémente [HyperDN](#) < class [Tenseur3HH](#), class [Tenseur3BB](#), class [Tenseur3BH](#), class [Tenseur3HB](#), class [Tenseur\\_ns3HH](#), class [T](#)

#### 6.380.1.3 Invariants\_et\_var()

```
Tenseur3BH * Hyper3DN::Invariants_et_var (
    TenseurBB & epsBB_tdt,
    TenseurBB & gijBB_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien_tdt,
    Vecteur & d_jacobien_tdt,
    double & Ieps,
    Tableau< double > & dIeps,
    double & V,
    Tableau< double > & dV,
    double & bIIb,
    Tableau< double > & dbIIb,
    Tenseur3BH & epsBH,
    Tableau< Tenseur3BH > & depsBH ) [virtual]
```

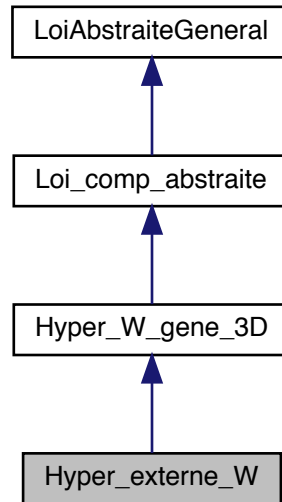
Implémente [HyperDN](#) < class [Tenseur3HH](#), class [Tenseur3BB](#), class [Tenseur3BH](#), class [Tenseur3HB](#), class [Tenseur\\_ns3HH](#), class [T](#)

La documentation de cette classe a été générée à partir du fichier suivant :

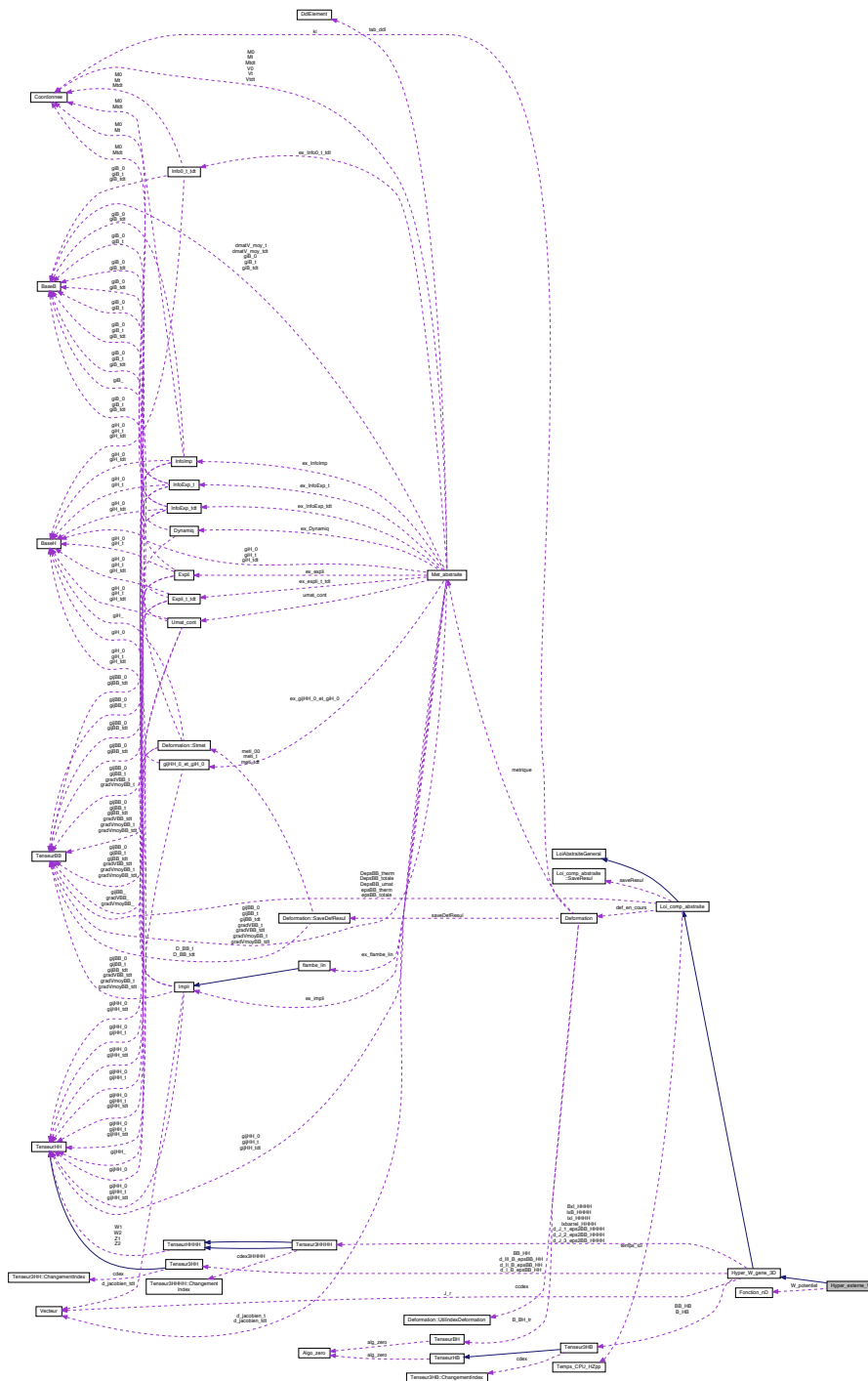
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3DN.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3DN.cc

## 6.381 Référence de la classe Hyper\_externe\_W

Grphe d'héritage de Hyper\_externe\_W:



Graphe de collaboration de Hyper\_externe\_W:



### Classes

- class [SaveResulHyper\\_externe\\_W](#)

### Fonctions membres publiques

- **Hyper\_externe\_W** (const [Hyper\\_externe\\_W](#) &loi)
- [SaveResul](#) \* **New\_et\_Initialise** ()
- void **AfficheDataSpecif** (ofstream &, [SaveResul](#) \*a) const

- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D &lesCourbes1D`, `LesFonctions_nD &lesFonctionsnD`)
- void `Affiche` () const
- int `TestComple` ()
- void `Lecture_base_info_loi` (`ifstream &ent`, const int cas, `LesReferences &lesRef`, `LesCourbes1D &lesCourbes1D`, `LesFonctions_nD &lesFonctionsnD`)
- void `Ecriture_base_info_loi` (`ofstream &sort`, const int cas)
- void `Grandeur_particuliere` (bool absolue, `List_io< TypeQuelconque > &`, `Loi_comp_abstraite::SaveResul *`, `list< int > &decal`) const
- void `ListeGrandeurs_particulieres` (bool absolue, `List_io< TypeQuelconque > &`) const
- double `Module_young_equivalent` (`Enum_dure temps`, const `Deformation &`, `SaveResul *saveResul`)
- double `Module_compressibilite_equivalent` (`Enum_dure temps`, const `Deformation &def`, `SaveResul *saveResul`)
- virtual double `HsurH0` (`SaveResul *saveResul`) const
- `Loi_comp_abstraite * Nouvelle_loi_identique` () const
- void `Info_commande LoisDeComp` (`UtilLecture &lec`)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne &`, const `Deformation &`, `Enum_dure`, const `ThermoDonnee &`, const `Met_abstraite::Impli *ex_impli`, const `Met_abstraite::Expli_t_tdt *ex_expli_tdt`, const `Met_abstraite::Umat_cont *ex_umat`, const `List_io< Ddl_etendu > *exclure_dd_etend`, const `List_io< const TypeQuelconque * > *exclure_Q`)

### Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH &sigHH_t`, `TenseurBB &DepsBB`, `DdlElement &tab_ddl`, `TenseurBB &gijBB_t`, `TenseurHH &gijHH_t`, `BaseB &giB`, `BaseH &gi_H`, `TenseurBB &epsBB_`, `TenseurBB &delta_epsBB_`, `TenseurBB &gijBB_`, `TenseurHH &gijHH_`, `Tableau< TenseurBB * > &d_gijBB_`, double `&jacobien_0`, double `&jacobien`, `TenseurHH &sigHH`, `EnergieMeca &energ`, const `EnergieMeca &energ_t`, double `&module_compressibilite`, double `&module_cisaillement`, const `Met_abstraite::Expli_t_tdt &ex`)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH &sigHH_t`, `TenseurBB &DepsBB`, `DdlElement &tab_ddl`, `BaseB &giB_t`, `TenseurBB &gijBB_t`, `TenseurHH &gijHH_t`, `BaseB &giB_tdt`, `Tableau< BaseB > &d_giB_tdt`, `BaseH &giH_tdt`, `Tableau< BaseH > &d_giH_tdt`, `TenseurBB &epsBB_tdt`, `Tableau< TenseurBB * > &d_epsBB`, `TenseurBB &delta_epsBB`, `TenseurBB &gijBB_tdt`, `TenseurHH &gijHH_tdt`, `Tableau< TenseurBB * > &d_gijBB_tdt`, `Tableau< TenseurHH * > &d_gijHH_tdt`, double `&jacobien_0`, double `&jacobien`, `Vecteur &d_jacobien_tdt`, `TenseurHH &sigHH`, `Tableau< TenseurHH * > &d_sigHH`, `EnergieMeca &energ`, const `EnergieMeca &energ_t`, double `&module_compressibilite`, double `&module_cisaillement`, const `Met_abstraite::Impli &ex`)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH &sigHH_t`, `TenseurBB &DepsBB`, `TenseurBB &epsBB_tdt`, `TenseurBB &delta_epsBB`, double `&jacobien_0`, double `&jacobien`, `TenseurHH &sigHH`, `TenseurHHHH &d_sigma_deps`, `EnergieMeca &energ`, const `EnergieMeca &energ_t`, double `&module_compressibilite`, double `&module_cisaillement`, const `Met_abstraite::Umat_cont &ex`)

### Attributs protégés

- `Fonction_nD * W_potentiel`
- double `W_d`
- double `W_v`
- double `W_d_J1`
- double `W_d_J2`
- double `W_d_J1_2`
- double `W_d_J1_J2`
- double `W_d_J2_2`
- double `W_v_J3`
- double `W_v_J3J3`
- `List_io< TypeQuelconque > invar`
- `List_io< Ddl_enum_etendu > exclure_ddenum`

### Membres hérités additionnels

#### 6.381.1 Documentation des fonctions membres

**6.381.1.1 Affiche()**

void Hyper\_externe\_W::Affiche ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

**6.381.1.2 AfficheDataSpecif()**

void Hyper\_externe\_W::AfficheDataSpecif (   
     ofstream & ,   
     SaveResul \* a ) const [inline], [virtual]  
 Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.381.1.3 Calcul\_dsigma\_deps()**

void Hyper\_externe\_W::Calcul\_dsigma\_deps (   
     bool en\_base\_orthonormee,   
     TenseurHH & sigHH\_t,   
     TenseurBB & DepsBB,   
     TenseurBB & epsBB\_tdt,   
     TenseurBB & delta\_epsBB,   
     double & jacobien\_0,   
     double & jacobien,   
     TenseurHH & sigHH,   
     TenseurHHHH & d\_sigma\_deps,   
     EnergieMeca & energ,   
     const EnergieMeca & energ\_t,   
     double & module\_compressibilite,   
     double & module\_cisaillement,   
     const Met\_abstraite::Umat\_cont & ex ) [protected], [virtual]  
 Tenseur3HHHH d\_sigma\_depsHHHH; d\_sigma\_depsHHHH.TransfertDunTenseurGeneral(dSigdepsHHHH.↔  
 Symetrise1et2\_3et4());  
 Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.381.1.4 Calcul\_DsigmaHH\_tdt()**

void Hyper\_externe\_W::Calcul\_DsigmaHH\_tdt (   
     TenseurHH & sigHH\_t,   
     TenseurBB & DepsBB,   
     DdlElement & tab\_ddl,   
     BaseB & giB\_t,   
     TenseurBB & gijBB\_t,   
     TenseurHH & gijHH\_t,   
     BaseB & giB\_tdt,   
     Tableau< BaseB > & d\_giB\_tdt,   
     BaseH & giH\_tdt,   
     Tableau< BaseH > & d\_giH\_tdt,   
     TenseurBB & epsBB\_tdt,   
     Tableau< TenseurBB \* > & d\_epsBB,   
     TenseurBB & delta\_epsBB,   
     TenseurBB & gijBB\_tdt,   
     TenseurHH & gijHH\_tdt,   
     Tableau< TenseurBB \* > & d\_gijBB\_tdt,   
     Tableau< TenseurHH \* > & d\_gijHH\_tdt,   
     double & jacobien\_0,   
     double & jacobien,   
     Vecteur & d\_jacobien\_tdt,



```

TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.381.1.5 Calcul\_SigmaHH()

```

void Hyper_externe_W::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.381.1.6 CalculGrandeurTravail()

```

virtual void Hyper_externe_W::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.381.1.7 Ecriture\_base\_info\_loi()

```

void Hyper_externe_W::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

**6.381.1.8 Grandeur\_particuliere()**

```
void Hyper_externe_W::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Hyper\\_W\\_gene\\_3D](#).

**6.381.1.9 HsurH0()**

```
virtual double Hyper_externe_W::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.381.1.10 Info\_commande\_LoisDeComp()**

```
void Hyper_externe_W::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.381.1.11 Lecture\_base\_info\_loi()**

```
void Hyper_externe_W::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.381.1.12 LectureDonneesParticulieres()**

```
void Hyper_externe_W::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.381.1.13 ListeGrandeurs\_particulieres()**

```
void Hyper_externe_W::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Hyper\\_W\\_gene\\_3D](#).

**6.381.1.14 Module\_compressibilite\_equivalent()**

```
double Hyper_externe_W::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.381.1.15 Module\_young\_equivalent()**

```
double Hyper_externe_W::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.381.1.16 New\_et\_Initialise()**

```
SaveResul * Hyper_externe_W::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.381.1.17 Nouvelle\_loi\_identique()**

```
Loi_comp_abstraite * Hyper_externe_W::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.381.1.18 TestComplet()**

```
int Hyper_externe_W::TestComplet ( ) [virtual]
```

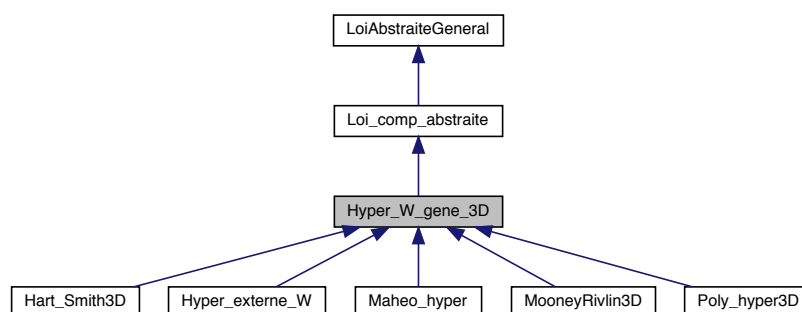
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

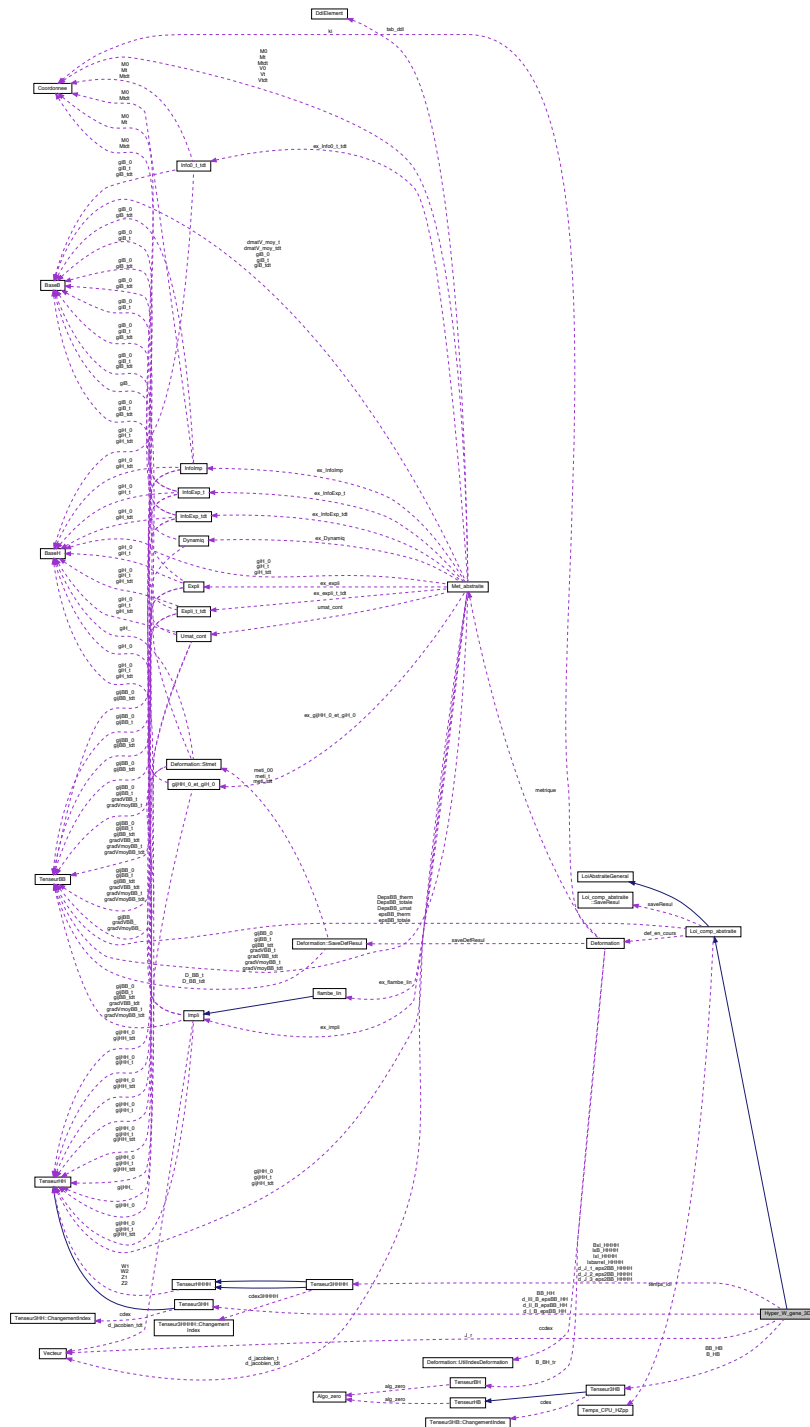
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_externe\_↔  
W.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_externe\_↔  
W.cc

**6.382 Référence de la classe Hyper\_W\_gene\_3D**

Graphe d'héritage de Hyper\_W\_gene\_3D:



Graphe de collaboration de Hyper\_W\_gene\_3D:



### Classes

- class [Invariantpost3D](#)
- class [SaveResulHyper\\_W\\_gene\\_3D](#)

### Fonctions membres publiques

- [Hyper\\_W\\_gene\\_3D](#) ([Enum\\_comp](#) id\_comp, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension, bool vit\_def=false)

- **Hyper\_W\_gene\_3D** (const [Hyper\\_W\\_gene\\_3D](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()
- virtual void [Grandeur\\_particuliere](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &litQ, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, list< int > &decal) const
- virtual void [ListeGrandeurs\\_particulieres](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &litQ) const
- void [Invariants\\_et\\_var1](#) (const [TenseurBB](#) &gijBB\_0, const [TenseurHH](#) &gijHH\_0, const [TenseurBB](#) &gij↔  
BB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien)
- void [Invariants\\_et\\_var2](#) (const [TenseurBB](#) &gijBB\_0, const [TenseurHH](#) &gijHH\_0, const [TenseurBB](#) &gij↔  
BB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien)
- const double & [I\\_B](#) () const
- const double & [I\\_BB](#) () const
- const double & [II\\_B](#) () const
- const [Vecteur](#) & [JJ\\_r](#) () const
- const [Tableau](#)< [Tenseur3HH](#) > & [D\\_J\\_r\\_epsBB\\_HH](#) () const
- const [Tenseur3HHHH](#) & [D\\_J\\_1\\_eps2BB\\_HHHH](#) () const
- const [Tenseur3HHHH](#) & [D\\_J\\_2\\_eps2BB\\_HHHH](#) () const
- const [Tenseur3HHHH](#) & [D\\_J\\_3\\_eps2BB\\_HHHH](#) () const

### Fonctions membres protégées

- void [Info\\_commande\\_LoisDeComp\\_hyper3D](#) ([UtilLecture](#) &lec)
- void [InvariantsEtVar1](#) (const [TenseurBB](#) &gijBB\_0, const [TenseurHH](#) &gijHH\_0, const [TenseurBB](#) &gijBB↔  
\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien)
- void [InvariantsEtVar2](#) (const [TenseurBB](#) &gijBB\_0, const [TenseurHH](#) &gijHH\_0, const [TenseurBB](#) &gijBB↔  
\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien)
- void [Calcul\\_derivee\\_numerique](#) (const [TenseurBB](#) &gijBB\_0, const [TenseurHH](#) &gijHH\_0, const [TenseurBB](#)  
&gijBB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien)
- void [Calcul\\_derivee\\_numerique2](#) (const [TenseurBB](#) &gijBB\_0, const [TenseurHH](#) &gijHH\_0, const  
[TenseurBB](#) &gijBB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien)
- void [Invariants\\_et\\_var1\\_deb](#) (const [TenseurBB](#) &gijBB\_0, const [TenseurHH](#) &gijHH\_0, const [TenseurBB](#)  
&gijBB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien)
- void [Invariants\\_et\\_var2\\_deb](#) (const [TenseurBB](#) &gijBB\_0, const [TenseurHH](#) &gijHH\_0, const [TenseurBB](#)  
&gijBB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien)

### Attributs protégés

- int [sortie\\_post](#)
- double [I\\_B](#)
- double [I\\_BB](#)
- double [II\\_B](#)
- double [III\\_B](#)
- [Vecteur](#) [J\\_r](#)
- [Tenseur3HH](#) [d\\_I\\_B\\_epsBB\\_HH](#)
- [Tenseur3HH](#) [d\\_II\\_B\\_epsBB\\_HH](#)
- [Tenseur3HH](#) [d\\_III\\_B\\_epsBB\\_HH](#)
- [Tableau](#)< [Tenseur3HH](#) > [d\\_J\\_r\\_epsBB\\_HH](#)
- [Tenseur3HHHH](#) [d\\_J\\_1\\_eps2BB\\_HHHH](#)
- [Tenseur3HHHH](#) [d\\_J\\_2\\_eps2BB\\_HHHH](#)
- [Tenseur3HHHH](#) [d\\_J\\_3\\_eps2BB\\_HHHH](#)
- double [V](#)
- [Tenseur3HH](#) [BB\\_HH](#)
- [Tenseur3HB](#) [B\\_HB](#)
- [Tenseur3HB](#) [BB\\_HB](#)
- [Tenseur3HHHH](#) [IxI\\_HHHH](#)
- [Tenseur3HHHH](#) [Ixbarrel\\_HHHH](#)
- [Tenseur3HHHH](#) [IxB\\_HHHH](#)
- [Tenseur3HHHH](#) [BxI\\_HHHH](#)
- double [J3\\_puiss\\_untiers](#)
- double [unSurJ3\\_puissuntiers](#)
- double [unSurJ3\\_puissuntiers2](#)
- double [unSurJ3\\_puissuntiers4](#)

## Membres hérités additionnels

### 6.382.1 Documentation des fonctions membres

#### 6.382.1.1 Calcul\_derivee\_numerique()

```
void Hyper_W_gene_3D::Calcul_derivee_numerique (
    const TenseurBB & gijBB_0,
    const TenseurHH & gijHH_0,
    const TenseurBB & gijBB_tdt,
    const TenseurHH & gijHH_tdt,
    const double & jacobien_0,
    const double & jacobien ) [protected]
```

10.;

#### 6.382.1.2 Grandeur\_particuliere()

```
void Hyper_W_gene_3D::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.382.1.3 ListeGrandeurs\_particulieres()

```
void Hyper_W_gene_3D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.382.1.4 New\_et\_Initialise()

```
SaveResul * Hyper_W_gene_3D::New_et_Initialise ( ) [inline], [virtual]
```

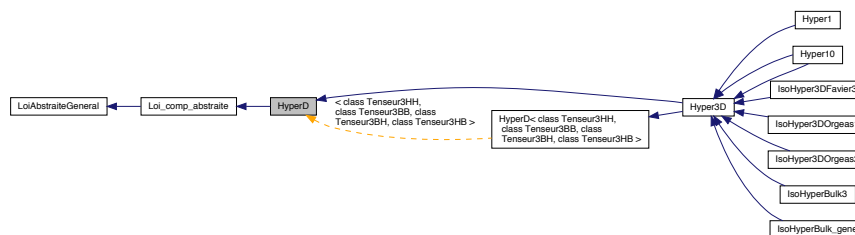
Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

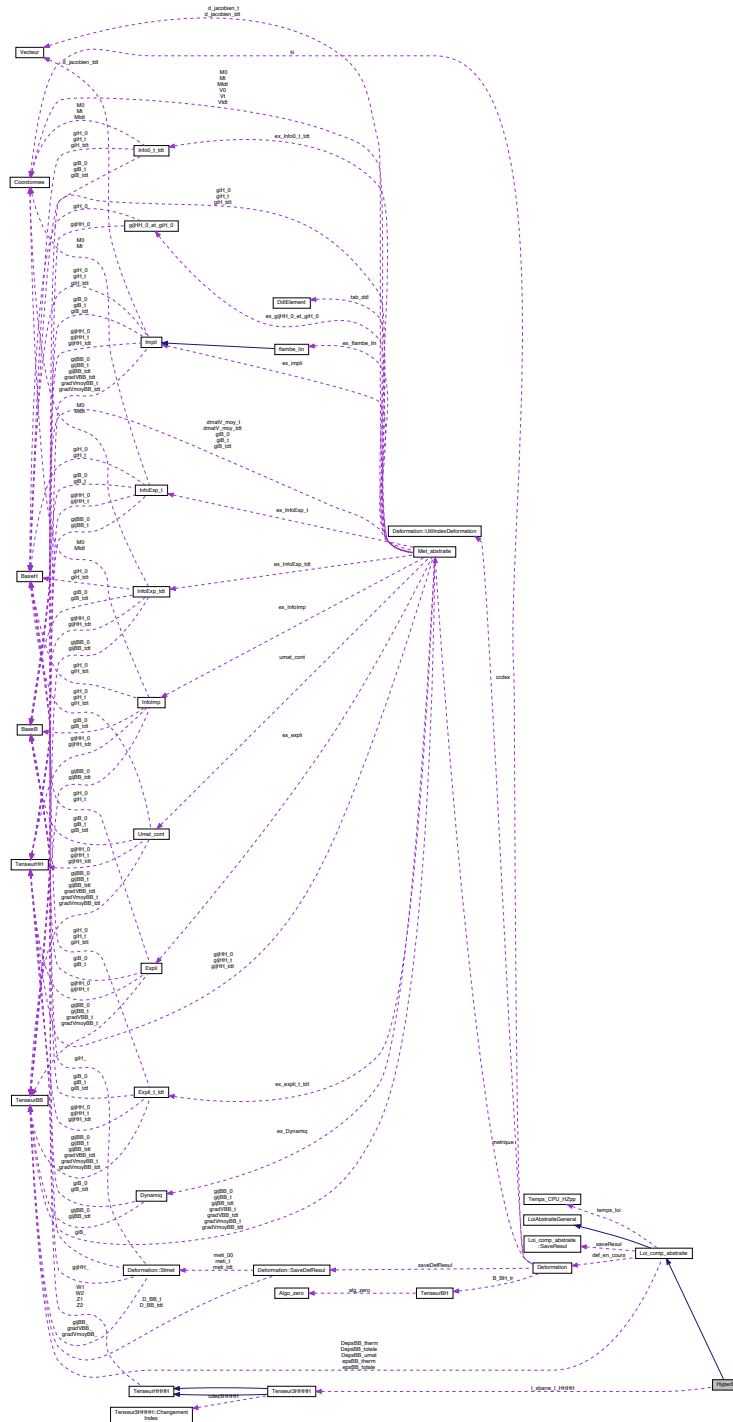
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_W\_gene\_3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_W\_gene\_3D.cc

## 6.383 Référence de la classe HyperD

Graphe d'héritage de HyperD:



Graphe de collaboration de HyperD:



### Classes

- class [A\\_i](#)
- class [A\\_iAvecVarDdl](#)
- class [A\\_iAvecVarEps](#)
- class [Invariant](#)
- class [Invariantpost3D](#)
- class [InvariantVarDdl](#)
- class [InvariantVarEps](#)

- class [PotenAvecPhaseAvecVar](#)
- class [PotenAvecPhaseSansVar](#)
- class [PotenSansPhaseAvecVar](#)
- class [PotenSansPhaseSansVar](#)
- class [SaveResulHyperD](#)

## Fonctions membres publiques

- **HyperD** ([Enum\\_comp](#) id\_compor, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension, bool avec\_ph)
- **HyperD** (char \*nom, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension, bool avec\_ph)
- **HyperD** (const [HyperD](#) &a)
- **SaveResul** \* [New\\_et\\_Initialise](#) ()
- virtual void [Grandeur\\_particuliere](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, [Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, list< int > &decal) const
- virtual void [ListeGrandeurs\\_particulieres](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ) const
- void [AfficheDataSpecif](#) (ofstream &, [SaveResul](#) \*) const
- virtual void [Activation\\_stockage\\_grandeurs\\_quelconques](#) (list< [EnumTypeQuelconque](#) > &listEnuQuelc)
- virtual void [Insertion\\_conteneur\\_dans\\_save\\_result](#) ([SaveResul](#) \*saveResul)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gij←  
BB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_,  
[TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double  
&jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module←  
compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#)  
&giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#)  
&giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_eps←  
BB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* >  
&d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#)  
&d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const  
[EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met](#)←  
abstraite::Impli &ex)
- void [Calcul\\_dsigma\\_deps](#) (bool en\_base\_orthonormee, [TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB,  
[TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, double &jacobien\_0, double &jacobien, [TenseurHH](#)  
&sigHH, [TenseurHHHH](#) &d\_sigma\_deps, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double  
&module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Umat\\_cont](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const  
[ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const  
[Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const  
[TypeQuelconque](#) \* > \*exclure\_Q)
- virtual [TenseurBH](#) \* **Invariants** (const [TenseurBB](#) &epsBB\_t, const [TenseurBB](#) &gijBB\_t, const [TenseurHH](#)  
&gijHH\_t, const double &jacobien\_0, const double &jacobien\_t, [Invariant](#) &invariant, [TenseurBH](#) &epsBH)=0
- virtual [TenseurBH](#) \* **Invariants\_et\_var** (const [TenseurBB](#) &epsBB\_tdt, const [TenseurBB](#) &gijBB\_tdt, const  
[Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_tdt, const [TenseurHH](#) &gijHH\_tdt, const [Tableau](#)< [TenseurHH](#) \*  
> &d\_gijHH\_tdt, const double &jacobien\_0, const double &jacobien\_tdt, const [Vecteur](#) &d\_jacobien\_tdt,  
[InvariantVarDdl](#) &invariantVarDdl, [TenseurBH](#) &epsBH\_tdt, [Tableau](#)< [TenseurBH](#) \* > &depsBH\_tdt)=0
- virtual [TenseurBH](#) \* **Invariants\_et\_varEps** (const [TenseurBB](#) &epsBB\_tdt, const [TenseurBB](#) &gijBB←  
tdt, const [TenseurHH](#) &gijHH\_tdt, const double &jacobien\_0, const double &jacobien\_tdt, [InvariantVarEps](#)  
&invariantVarEps, [TenseurBH](#) &epsBH\_tdt)=0
- virtual [PotenSansPhaseSansVar](#) **Potentiel** (const [Invariant](#) &invariant, const double &jacobien0)=0
- virtual [PotenAvecPhaseSansVar](#) **PotentielPhase** (const [Invariant](#) &invariant, const double &jacobien0)=0
- virtual [PotenSansPhaseAvecVar](#) **Potentiel\_et\_var** (const [Invariant](#) &invariant, const double &jacobien0)=0
- virtual [PotenAvecPhaseAvecVar](#) **PotentielPhase\_et\_var** (const [Invariant](#) &invariant, const double &jaco-  
bien0)=0
- void [Info\\_commande\\_LoisDeComp\\_hyper3D](#) ([UtilLecture](#) &lec)
- void **AAA\_i** (const [PotenSansPhaseSansVar](#) &potenSansPhaseSansVar, const [Invariant](#) &invariant, const  
double &jaco, [A\\_i](#) &a\_i)
- void **AAA\_iPhase** (const [PotenAvecPhaseSansVar](#) &potenAvecPhaseSansVar, const [Invariant](#) &invariant,  
const double &jaco, [A\\_i](#) &a\_i)



- void **AAA\_i\_var** (const [PotenSansPhaseAvecVar](#) &potenSansPhaseAvecVar, const double &jaco0, const [InvariantVarDdl](#) &invariantVarDdl, const double &jaco, const [Vecteur](#) &d\_jacobien\_tdt, [A\\_iAvecVarDdl](#) &a\_iAvecVarDdl)
- void **AAA\_iPhase\_var** (const [PotenAvecPhaseAvecVar](#) &potenAvecPhaseAvecVar, const double &jaco0, const [InvariantVarDdl](#) &invariantVarDdl, const double &jaco, const [Vecteur](#) &d\_jacobien\_tdt, [A\\_iAvecVarDdl](#) &a\_iAvecVarDdl)
- void **AAA\_i\_varEps** (const [PotenSansPhaseAvecVar](#) &potenSansPhaseAvecVar, const double &jaco0, const [InvariantVarEps](#) &invariantVarEps, const double &jaco, [A\\_iAvecVarEps](#) &a\_iAvecVarEps)
- void **AAA\_iPhase\_varEps** (const [PotenAvecPhaseAvecVar](#) &potenAvecPhaseAvecVar, const double &jaco0, const [InvariantVarEps](#) &invariantVarEps, const double &jaco, [A\\_iAvecVarEps](#) &a\_iAvecVarEps)

### Attributs protégés

- bool **avec\_phase**
- bool **avec\_regularisation**
- double **fact\_regularisation**
- int **sortie\_post**
- [Tenseur3HHHH](#) **I\_xbarre\_I\_HHHH**
- const [Met\\_abstraite::Impli](#) \* **ex\_impli\_hyper**
- const [Met\\_abstraite::Expli\\_t\\_tdt](#) \* **ex\_expli\_tdt\_hyper**
- const [Met\\_abstraite::Umat\\_cont](#) \* **ex\_umat\_hyper**

### Membres hérités additionnels

#### 6.383.1 Documentation des fonctions membres

##### 6.383.1.1 Activation\_stockage\_grandeurs\_quelconques()

```
void HyperD::Activation_stockage_grandeurs_quelconques (
    list< EnumTypeQuelconque > & listEnuQuelc ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

##### 6.383.1.2 AfficheDataSpecif()

```
void HyperD::AfficheDataSpecif (
    ofstream & ,
    SaveResul * ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

##### 6.383.1.3 Calcul\_dsigma\_deps()

```
void HyperD::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met\_abstraite::Umat\_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.383.1.4 Calcul\_DsigmaHH\_tdt()

```
void HyperD::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.383.1.5 Calcul\_SigmaHH()

```
void HyperD::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.383.1.6 CalculGrandeurTravail()

```
virtual void HyperD::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.383.1.7 Grandeur\_particuliere()

```
void HyperD::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.383.1.8 Insertion\_conteneur\_dans\_save\_result()

```
void HyperD::Insertion_conteneur_dans_save_result (
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.383.1.9 Invariants\_et\_var()

```
virtual TenseurBH * HyperD::Invariants_et_var (
    const TenseurBB & epsBB_tdt,
    const TenseurBB & gijBB_tdt,
    const Tableau< TenseurBB * > & d_gijBB_tdt,
    const TenseurHH & gijHH_tdt,
    const Tableau< TenseurHH * > & d_gijHH_tdt,
    const double & jacobien_0,
    const double & jacobien_tdt,
    const Vecteur & d_jacobien_tdt,
    InvariantVarDdl & invariantVarDdl,
    TenseurBH & epsBH_tdt,
    Tableau< TenseurBH * > & depsBH_tdt ) [protected], [pure virtual]
```

Implémenté dans [Hyper3D](#).

#### 6.383.1.10 ListeGrandeurs\_particulieres()

```
void HyperD::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.383.1.11 New\_et\_Initialise()

`HyperD::SaveResul * HyperD::New_et_Initialise ( ) [virtual]`

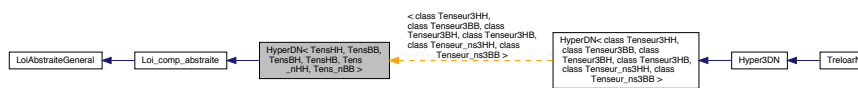
Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

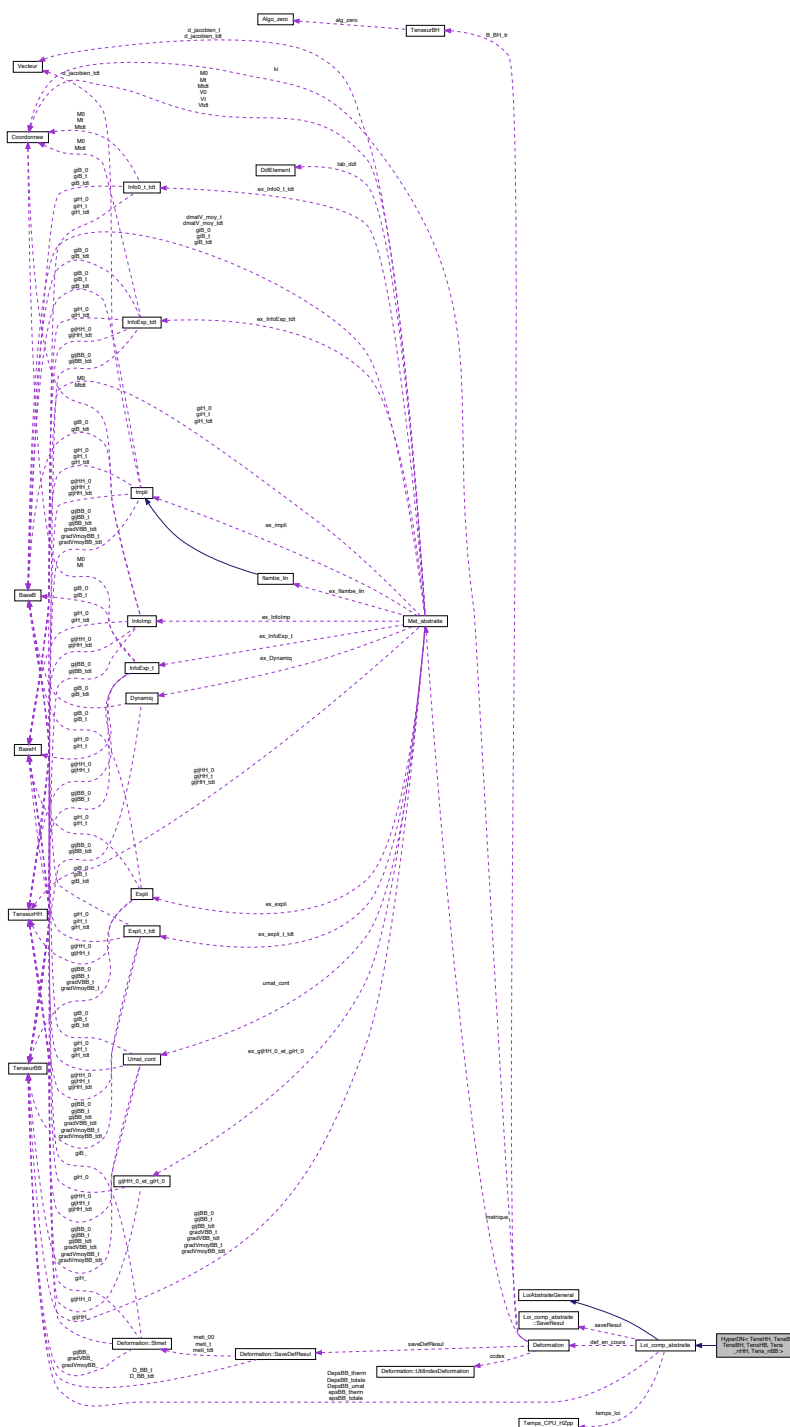
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.cc

## 6.384 Référence du modèle de la classe HyperDN< TensHH, TensBB, TensBH, TensHB, Tens\_nHH, Tens\_nBB >

Graphe d'héritage de HyperDN< TensHH, TensBB, TensBH, TensHB, Tens\_nHH, Tens\_nBB > :



Graphe de collaboration de HyperDN< TensHH, TensBB, TensBH, TensHB, Tens\_nHH, Tens\_nBB > :



## Classes

- class [SaveResulHyperDN](#)

## Fonctions membres publiques

- [HyperDN](#) (Enum\_comp id\_compor, Enum\_categorie\_loi\_comp categorie\_comp, int dimension)
- [HyperDN](#) (char \*nom, Enum\_categorie\_loi\_comp categorie\_comp, int dimension)
- [HyperDN](#) (const [HyperDN](#) &a)

- `SaveResul * New_et_Initialise ()`
- `void AfficheDataSpecif (ofstream &, SaveResul *) const`

## Fonctions membres protégées

- `void Calcul_SigmaHH (TenseurHH &sigHH_t, TenseurBB &DepsBB, DdlElement &tab_ddl, TenseurBB &gij← BB_t, TenseurHH &gijHH_t, BaseB &giB, BaseH &gi_H, TenseurBB &epsBB_, TenseurBB &delta_epsBB_, TenseurBB &gijBB_, TenseurHH &gijHH_, Tableau< TenseurBB * > &d_gijBB_, double &jacobien_0, double &jacobien, TenseurHH &sigHH, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_← compressibilite, double &module_cisaillement, const Met_abstraite::Expli_t_tdt &ex)`
- `void Calcul_DsigmaHH_tdt (TenseurHH &sigHH_t, TenseurBB &DepsBB, DdlElement &tab_ddl, BaseB &giB_t, TenseurBB &gijBB_t, TenseurHH &gijHH_t, BaseB &giB_tdt, Tableau< BaseB > &d_giB_tdt, BaseH &giH_tdt, Tableau< BaseH > &d_giH_tdt, TenseurBB &epsBB_tdt, Tableau< TenseurBB * > &d_eps← BB, TenseurBB &delta_epsBB, TenseurBB &gijBB_tdt, TenseurHH &gijHH_tdt, Tableau< TenseurBB * > &d_gijBB_tdt, Tableau< TenseurHH * > &d_gijHH_tdt, double &jacobien_0, double &jacobien, Vecteur &d_jacobien_tdt, TenseurHH &sigHH, Tableau< TenseurHH * > &d_sigHH, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_← abstraite::Impli &ex)`
- `virtual void CalculGrandeurTravail (const PtIntegMecalInterne &, const Deformation &, Enum_dure, const ThermoDonnee &, const Met_abstraite::Impli *ex_impli, const Met_abstraite::Expli_t_tdt *ex_expli_tdt, const Met_abstraite::Umat_cont *ex_umat, const List_io< Ddl_etendu > *exclure_dd_etend, const List_io< const TypeQuelconque * > *exclure_Q)`
- `virtual TensBH * Invariants (TenseurBB &epsBB_t, TenseurBB &gijBB_t, TenseurHH &gijHH_t, double &jacobien_0, double &jacobien_t, double &leps, double &V, double &bllb, TensBH &epsBH)=0`
- `virtual TensBH * Invariants_et_var (TenseurBB &epsBB_tdt, TenseurBB &gijBB_tdt, Tableau< TenseurBB * > &d_gijBB_tdt, TenseurHH &gijHH_tdt, Tableau< TenseurHH * > &d_gijHH_tdt, double &jacobien_← 0, double &jacobien_tdt, Vecteur &d_jacobien_tdt, double &leps, Tableau< double > &dleps, double &V, Tableau< double > &dV, double &bllb, Tableau< double > &dbllb, TensBH &epsBH, Tableau< TensBH > &depsBH)=0`
- `virtual void Potentiel (double &jacobien_0, double &leps, double &V, double &bllb, double &E, double &EV, double &Ebllb, double &Eleps)=0`
- `virtual void Potentiel_et_var (double &jacobien_0, double &leps, double &V, double &bllb, double &E, double &EV, double &Ebllb, double &Eleps, double &EVV, double &Ebllb2, double &Eleps2, double &EVbllb, double &EVleps, double &Ebllbleps)=0`
- `void Alpha (double &E, double &EV, double &Ebllb, double &Eleps, double &jacobien_0, double &leps, double &V, double &bllb, double &alpha_0, double &alpha_1, double &alpha_2)`
- `void Alpha_var (double &E, double &EV, double &Ebllb, double &Eleps, double &EVV, double &Ebllb2, double &Eleps2, double &EVbllb, double &EVleps, double &Ebllbleps, double &jacobien_0, double &leps, Tableau< double > &dleps, double &V, Tableau< double > &dV, double &bllb, Tableau< double > &dbllb, double &alpha_0, Tableau< double > &dalpha_0, double &alpha_1, Tableau< double > &dalpha_1, double &alpha_2, Tableau< double > &dalpha_2)`

## Membres hérités additionnels

### 6.384.1 Documentation des fonctions membres

#### 6.384.1.1 AfficheDataSpecif()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class
Tens_nBB >
void HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::AfficheDataSpecif (
    ofstream & ,
    SaveResul * ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.384.1.2 Calcul\_DsigmaHH\_tdt()**

```

template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class
Tens_nBB >
void HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

**6.384.1.3 Calcul\_SigmaHH()**

```

template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class
Tens_nBB >
void HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,

```

```
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.384.1.4 CalculGrandeurTravail()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class
Tens_nBB >
virtual void HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::CalculGrandeur←
Travail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
```

```
[virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.384.1.5 New\_et\_Initialise()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class
Tens_nBB >
SaveResul * HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::New_et_Initialise (
) [inline], [virtual]
```

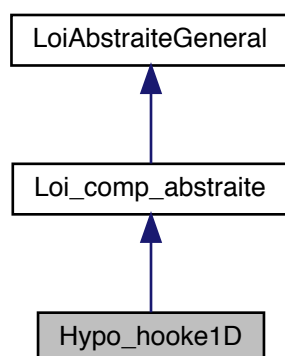
Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperDN.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperDN.cc

## 6.385 Référence de la classe Hypo\_hooke1D

Graphe d'héritage de Hypo\_hooke1D:







- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComplet` ()
- void `Lecture_base_info_loi` (`ifstream &ent`, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (`ofstream &sort`, const int cas)
- void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &, `Loi_comp_abstraite::SaveResul *`, `list`< int > &decal) const
- void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &) const
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &, `SaveResul *`saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul *`saveResul)
- double `Eps33BH` (`SaveResul *`saveDon) const
- double `Eps22BH` (`SaveResul *`saveDon) const
- double `HsurH0` (`SaveResul *`saveResul) const
- double `BsurB0` (`SaveResul *`saveResul) const
- `Loi_comp_abstraite *` `Nouvelle_loi_identique` () const
- void `Info_commande LoisDeComp` (`UtilLecture` &lec)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecaInterne` &ptintmeca, const `Deformation` &def, `Enum_dure` temps, const `ThermoDonnee` &dTP, const `Met_abstraite::Impli *`ex\_impli, const `Met_abstraite::Expli_t_tdt *`ex\_expli\_tdt, const `Met_abstraite::Umat_cont *`ex\_umat, const `List_io`< `Ddl_etendu` > `*exclure_dd_etend`, const `List_io`< const `TypeQuelconque *` > `*exclure_Q`)

## Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB *` > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB *` > &d\_epsBB\_tdt, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB *` > &d\_gijBB\_tdt, `Tableau`< `TenseurHH *` > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH *` > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)

## Attributs protégés

- double `f`
- `Courbe1D *` `f_temperature`
- `Courbe1D *` `f_lleps`
- `Fonction_nD *` `f_nD`
- double `Kc`
- `Courbe1D *` `Kc_temperature`
- `Courbe1D *` `Kc_lleps`
- `Fonction_nD *` `Kc_nD`
- bool `compress_thermophysique`
- int `type_derive`
- int `restriction_traction_compression`
- `Tenseur3HHHH` `I_x_I_HHHH`
- `Tenseur3HHHH` `I_xbarre_I_HHHH`
- `Tenseur3HHHH` `I_x_eps_HHHH`
- `Tenseur3HHHH` `I_x_D_HHHH`
- `Tenseur3HHHH` `I_xbarre_D_HHHH`
- `Tenseur3HHHH` `d_sig_t_HHHH`
- `Tenseur3HHHH` `d_spherique_sig_t_HHHH`

## Amis

— class `SaveResulLoi_Hypo1D`

## Membres hérités additionnels

### 6.385.1 Documentation des fonctions membres

#### 6.385.1.1 Affiche()

```
void Hypo_hooke1D::Affiche ( ) const [virtual]
Implémente LoiAbstraiteGeneral.
```

#### 6.385.1.2 BsurB0()

```
double Hypo_hooke1D::BsurB0 (
    SaveResul * saveResul ) const [virtual]
Réimplémentée à partir de Loi\_comp\_abstraite.
```

#### 6.385.1.3 Calcul\_dsigma\_deps()

```
void Hypo_hooke1D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met\_abstraite::Umat\_cont & ex ) [protected], [virtual]
Réimplémentée à partir de Loi\_comp\_abstraite.
```

#### 6.385.1.4 Calcul\_DsigmaHH\_tdt()

```
void Hypo_hooke1D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
```

```

TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.385.1.5 Calcul\_SigmaHH()

```

void Hypo_hooke1D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.385.1.6 CalculGrandeurTravail()

```

virtual void Hypo_hooke1D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,
    const ThermoDonnee & dTP,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.385.1.7 Ecriture\_base\_info\_loi()

```
void Hypo_hooke1D::Ecriture_base_info_loi (
    ostream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.385.1.8 Eps22BH()

```
double Hypo_hooke1D::Eps22BH (
    SaveResul * saveDon ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.385.1.9 Eps33BH()

```
double Hypo_hooke1D::Eps33BH (
    SaveResul * saveDon ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.385.1.10 Grandeur\_particuliere()

```
void Hypo_hooke1D::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.385.1.11 HsurH0()

```
double Hypo_hooke1D::HsurH0 (
    SaveResul * saveResul ) const [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.385.1.12 Info\_commande\_LoisDeComp()

```
void Hypo_hooke1D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.385.1.13 Lecture\_base\_info\_loi()

```
void Hypo_hooke1D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.385.1.14 LectureDonneesParticulieres()**

```
void Hypo_hooke1D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.385.1.15 ListeGrandeurs\_particulieres()**

```
void Hypo_hooke1D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.385.1.16 Module\_compressibilite\_equivalent()**

```
double Hypo_hooke1D::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.385.1.17 Module\_young\_equivalent()**

```
double Hypo_hooke1D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.385.1.18 New\_et\_Initialise()**

```
Hypo_hooke1D::SaveResul * Hypo_hooke1D::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.385.1.19 Nouvelle\_loi\_identique()**

```
Loi_comp_abstraite * Hypo_hooke1D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.385.1.20 TestComplet()**

```
int Hypo_hooke1D::TestComplet ( ) [virtual]
```

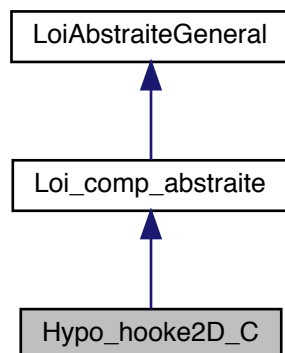
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

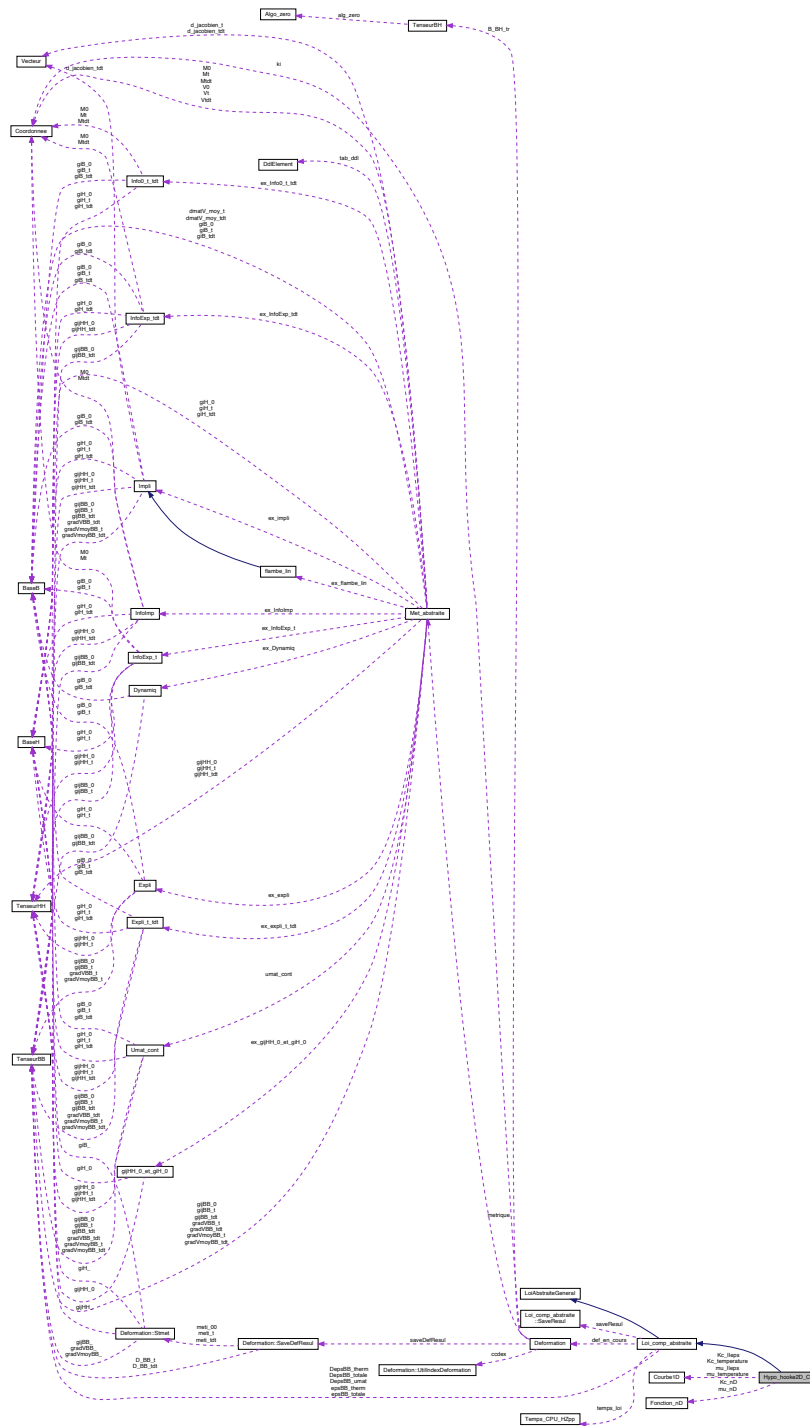
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke1D.cc

## 6.386 Référence de la classe Hypo\_hooke2D\_C

Graphe d'héritage de Hypo\_hooke2D\_C:



Graphe de collaboration de Hypo\_hooke2D\_C:



### Classes

- class [SaveResul\\_Hypo\\_hooke2D\\_C](#)

### Fonctions membres publiques

- [Hypo\\_hooke2D\\_C](#) (const [Hypo\\_hooke2D\\_C](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()



- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D &lesCourbes1D`, `LesFonctions_nD &lesFonctionsnD`)
- void `Affiche` () const
- int `TestComple` ()
- void `Lecture_base_info_loi` (`ifstream &ent`, `const int cas`, `LesReferences &lesRef`, `LesCourbes1D &lesCourbes1D`, `LesFonctions_nD &lesFonctionsnD`)
- void `Ecriture_base_info_loi` (`ofstream &sort`, `const int cas`)
- void `Grandeur_particuliere` (`bool absolue`, `List_io< TypeQuelconque > &`, `Loi_comp_abstraite::SaveResul *`, `list< int > &decal`) const
- void `ListeGrandeurs_particulieres` (`List_io< TypeQuelconque > &`) const
- double `Module_young_equivalent` (`Enum_dure temps`, `const Deformation &`, `SaveResul *saveResul`)
- double `Module_compressibilite_equivalent` (`Enum_dure temps`, `const Deformation &`, `SaveResul *saveDon`)
- virtual double `HsurH0` (`SaveResul *saveResul`) const
- `Loi_comp_abstraite * Nouvelle_loi_identique` () const
- void `Info_commande LoisDeComp` (`UtilLecture &lec`)
- virtual void `CalculGrandeurTravail` (`const PtIntegMecalInterne &ptintmeca`, `const Deformation &def`, `Enum_dure temps`, `const ThermoDonnee &dTP`, `const Met_abstraite::Impli *ex_impli`, `const Met_abstraite::Expli_t_tdt *ex_expli_tdt`, `const Met_abstraite::Umat_cont *ex_umat`, `const List_io< Ddl_etendu > *exclure_dd_etend`, `const List_io< const TypeQuelconque * > *exclure_Q`)
- virtual double `Eps33BH` (`SaveResul *saveResul`) const
- virtual bool `Contraintes_planes_de_3D` () const
- double `Deps33BH` (`TenseurBB &epsBB_`, `TenseurBB &DepsBB_`, `TenseurHH &gijHH_`)

## Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH &sigHH_t`, `TenseurBB &DepsBB`, `DdlElement &tab_ddl`, `TenseurBB &gijBB_t`, `TenseurHH &gijHH_t`, `BaseB &giB`, `BaseH &gi_H`, `TenseurBB &epsBB_`, `TenseurBB &delta_epsBB_`, `TenseurBB &gijBB_`, `TenseurHH &gijHH_`, `Tableau< TenseurBB * > &d_gijBB_`, `double &jacobien_0`, `double &jacobien`, `TenseurHH &sigHH`, `EnergieMeca &energ`, `const EnergieMeca &energ_t`, `double &module_compressibilite`, `double &module_cisaillement`, `const Met_abstraite::Expli_t_tdt &ex`)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH &sigHH_t`, `TenseurBB &DepsBB`, `DdlElement &tab_ddl`, `BaseB &giB_t`, `TenseurBB &gijBB_t`, `TenseurHH &gijHH_t`, `BaseB &giB_tdt`, `Tableau< BaseB > &d_giB_tdt`, `BaseH &giH_tdt`, `Tableau< BaseH > &d_giH_tdt`, `TenseurBB &epsBB_tdt`, `Tableau< TenseurBB * > &d_epsBB_`, `TenseurBB &delta_epsBB_`, `TenseurBB &gijBB_tdt`, `TenseurHH &gijHH_tdt`, `Tableau< TenseurBB * > &d_gijBB_tdt`, `Tableau< TenseurHH * > &d_gijHH_tdt`, `double &jacobien_0`, `double &jacobien`, `Vecteur &d_jacobien_tdt`, `TenseurHH &sigHH`, `Tableau< TenseurHH * > &d_sigHH`, `EnergieMeca &energ`, `const EnergieMeca &energ_t`, `double &module_compressibilite`, `double &module_cisaillement`, `const Met_abstraite::Impli &ex`)

## Attributs protégés

- double `mu`
- `Courbe1D * mu_temperature`
- `Courbe1D * mu_ileps`
- `Fonction_nD * mu_nD`
- double `Kc`
- `Courbe1D * Kc_temperature`
- `Courbe1D * Kc_ileps`
- `Fonction_nD * Kc_nD`
- bool `compress_thermophysique`
- int `type_derive`
- short int `cas_calcul`

## Membres hérités additionnels

### 6.386.1 Documentation des fonctions membres

**6.386.1.1 Affiche()**

```
void Hypo_hooke2D_C::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.386.1.2 Calcul\_DsigmaHH\_tdt()**

```
void Hypo_hooke2D_C::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.386.1.3 Calcul\_SigmaHH()**

```
void Hypo_hooke2D_C::Calcul_SigmaHH(
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
```

```

double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.386.1.4 CalculGrandeurTravail()

```

virtual void Hypo_hooke2D_C::CalculGrandeurTravail (
const PtIntegMecaInterne & ptintmeca,
const Deformation & def,
Enum_dure temps,
const ThermoDonnee & dTP,
const Met_abstraite::Impli * ex_impli,
const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
const Met_abstraite::Umat_cont * ex_umat,
const List_io< Ddl_etendu > * exclure_dd_etend,
const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.386.1.5 Contraintes\_planes\_de\_3D()

```

virtual bool Hypo_hooke2D_C::Contraintes_planes_de_3D ( ) const [inline], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.386.1.6 Ecriture\_base\_info\_loi()

```

void Hypo_hooke2D_C::Ecriture_base_info_loi (
ofstream & sort,
const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.386.1.7 Eps33BH()

```

double Hypo_hooke2D_C::Eps33BH (
SaveResul * saveResul ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.386.1.8 Grandeur\_particuliere()

```

void Hypo_hooke2D_C::Grandeur_particuliere (
bool absolue,
List_io< TypeQuelconque > & liTQ,
Loi_comp_abstraite::SaveResul * saveDon,
list< int > & decal ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.386.1.9 HsurH0()

```

double Hypo_hooke2D_C::HsurH0 (
SaveResul * saveResul ) const [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.386.1.10 Info\_commande\_LoisDeComp()

```
void Hypo_hooke2D_C::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.386.1.11 Lecture\_base\_info\_loi()

```
void Hypo_hooke2D_C::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.386.1.12 LectureDonneesParticulieres()

```
void Hypo_hooke2D_C::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.386.1.13 Module\_compressibilite\_equivalent()

```
double Hypo_hooke2D_C::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveDon ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.386.1.14 Module\_young\_equivalent()

```
double Hypo_hooke2D_C::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.386.1.15 New\_et\_Initialise()

```
Hypo_hooke2D_C::SaveResul * Hypo_hooke2D_C::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.386.1.16 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Hypo_hooke2D_C::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.386.1.17 TestComplet()

```
int Hypo_hooke2D_C::TestComplet ( ) [virtual]
```

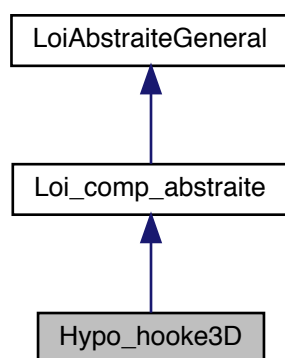
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

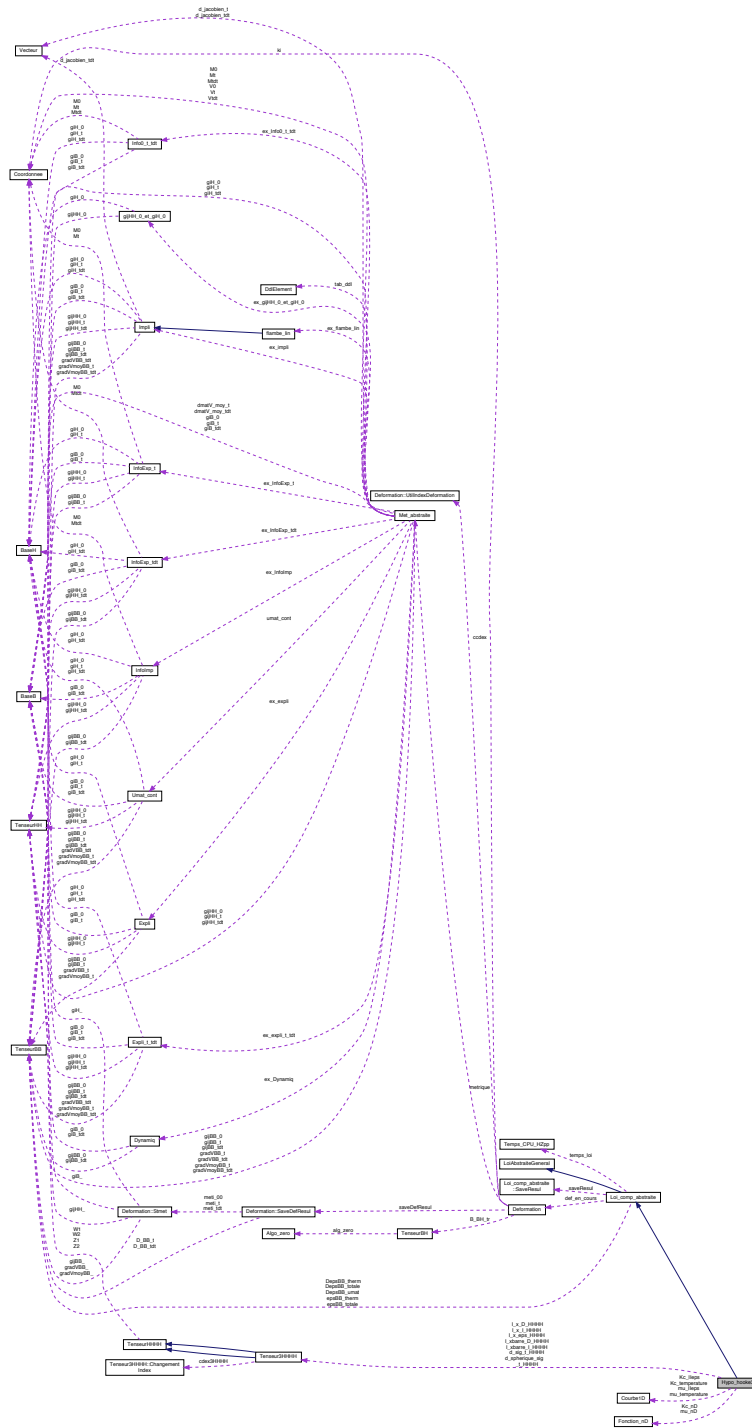
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke2D\_C.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke2D\_C.cc

## 6.387 Référence de la classe Hypo\_hooke3D

Grphe d'héritage de Hypo\_hooke3D:



Graphe de collaboration de Hypo\_hooke3D:



### Classes

- class [SaveResulLoi\\_Hypo3D](#)

### Fonctions membres publiques

- **Hypo\_hooke3D** (const [Hypo\\_hooke3D](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()

- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &, `Loi_comp_abstraite::SaveResul *`, `list`< int > &decal) const
- void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &) const
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &, `SaveResul *`saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul *`saveResul)
- virtual double `HsurH0` (`SaveResul *`saveResul) const
- `Loi_comp_abstraite *` `Nouvelle_loi_identique` () const
- void `Info_commande LoisDeComp` (`UtilLecture` &lec)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &ptintmeca, const `Deformation` &def, `Enum_dure` temps, const `ThermoDonnee` &dTP, const `Met_abstraite::Impli *`ex\_impli, const `Met_abstraite::Expli_t_tdt *`ex\_expli\_tdt, const `Met_abstraite::Umat_cont *`ex\_umat, const `List_io`< `Ddl_etendu` > `*exclure_dd_etend`, const `List_io`< const `TypeQuelconque *` > `*exclure_Q`)

### Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB *` > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB *` > &d\_epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB *` > &d\_gijBB\_tdt, `Tableau`< `TenseurHH *` > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH *` > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)

### Attributs protégés

- double `mu`
- `Courbe1D *` `mu_temperature`
- `Courbe1D *` `mu_lleps`
- `Fonction_nD *` `mu_nD`
- double `Kc`
- `Courbe1D *` `Kc_temperature`
- `Courbe1D *` `Kc_lleps`
- `Fonction_nD *` `Kc_nD`
- bool `compress_thermophysique`
- int `type_derive`
- short int `cas_calcul`
- `Tenseur3HHHH` `I_x_I_HHHH`
- `Tenseur3HHHH` `I_xbarre_I_HHHH`
- `Tenseur3HHHH` `I_x_eps_HHHH`
- `Tenseur3HHHH` `I_x_D_HHHH`
- `Tenseur3HHHH` `I_xbarre_D_HHHH`
- `Tenseur3HHHH` `d_sig_t_HHHH`
- `Tenseur3HHHH` `d_spherique_sig_t_HHHH`

## Membres hérités additionnels

### 6.387.1 Documentation des fonctions membres

#### 6.387.1.1 Affiche()

```
void Hypo_hooke3D::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.387.1.2 Calcul\_dsigma\_deps()

```
void Hypo_hooke3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.387.1.3 Calcul\_DsigmaHH\_tdt()

```
void Hypo_hooke3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
```



```

    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.387.1.4 Calcul\_SigmaHH()

```

void Hypo_hooke3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.387.1.5 CalculGrandeurTravail()

```

virtual void Hypo_hooke3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,
    const ThermoDonnee & dTP,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etendu,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.387.1.6 Ecriture\_base\_info\_loi()

```

void Hypo_hooke3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.387.1.7 Grandeur\_particuliere()

```

void Hypo_hooke3D::Grandeur_particuliere (
    bool absolue,

```

```
List_io< TypeQuelconque > & liTQ,  
Loi_comp_abstraite::SaveResul * saveDon,  
list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.387.1.8 HsurH0()

```
virtual double Hypo_hooke3D::HsurH0 (  
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.387.1.9 Info\_commande\_LoisDeComp()

```
void Hypo_hooke3D::Info_commande_LoisDeComp (  
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.387.1.10 Lecture\_base\_info\_loi()

```
void Hypo_hooke3D::Lecture_base_info_loi (  
    ifstream & ent,  
    const int cas,  
    LesReferences & lesRef,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.387.1.11 LectureDonneesParticulieres()

```
void Hypo_hooke3D::LectureDonneesParticulieres (  
    UtilLecture * entreePrinc,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.387.1.12 ListeGrandeurs\_particulieres()

```
void Hypo_hooke3D::ListeGrandeurs_particulieres (  
    bool absolue,  
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.387.1.13 Module\_compressibilite\_equivalent()

```
double Hypo_hooke3D::Module_compressibilite_equivalent (  
    Enum_dure temps,  
    const Deformation & def,  
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.387.1.14 Module\_young\_equivalent()

```
double Hypo_hooke3D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.387.1.15 New\_et\_Initialise()

```
Hypo_hooke3D::SaveResul * Hypo_hooke3D::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.387.1.16 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Hypo_hooke3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.387.1.17 TestComplet()

```
int Hypo_hooke3D::TestComplet ( ) [virtual]
```

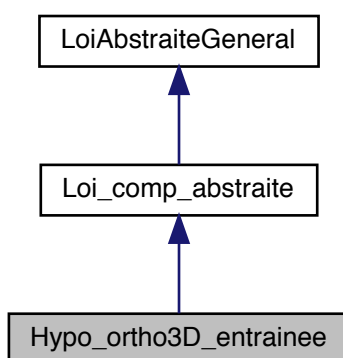
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

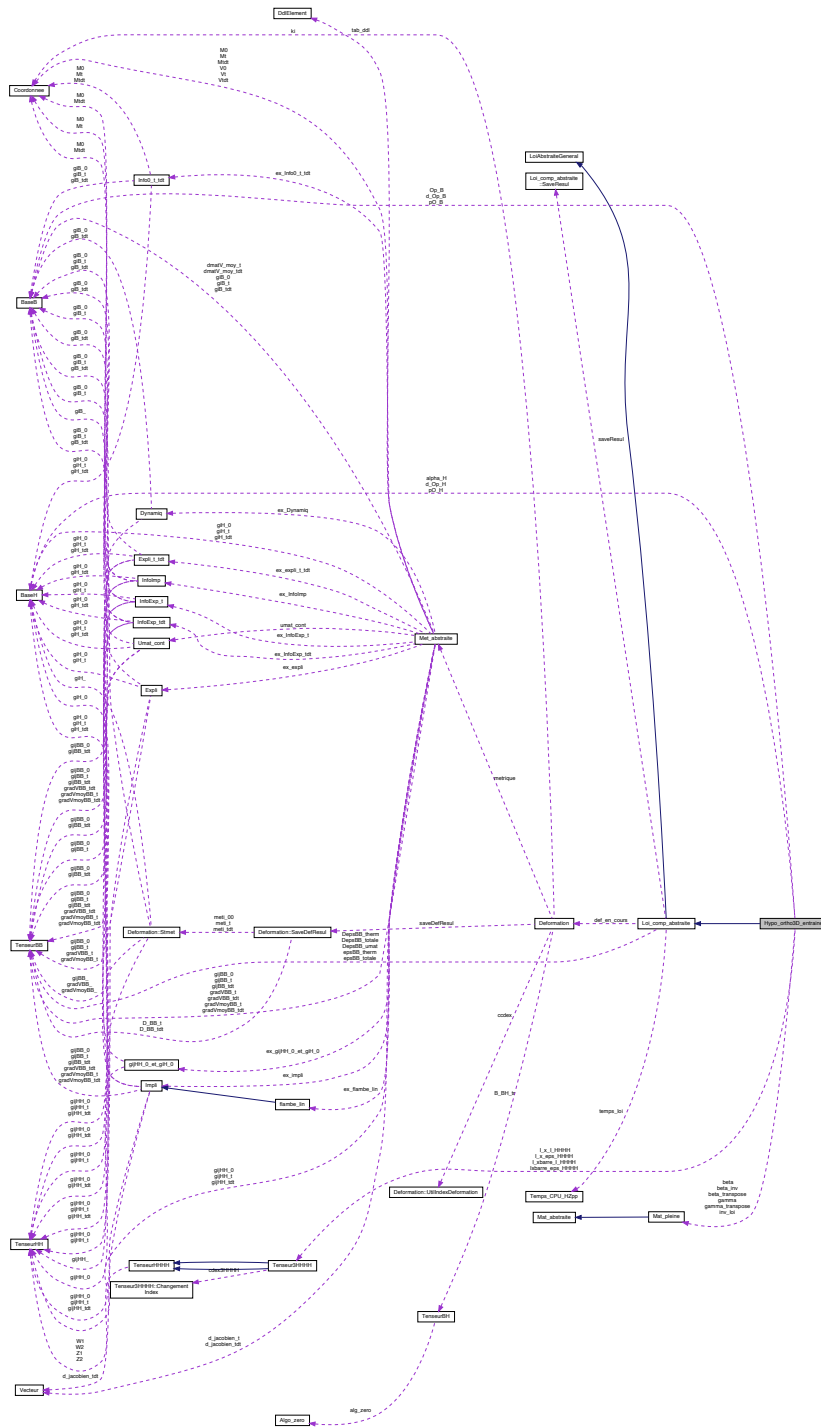
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke3D.cc

## 6.388 Référence de la classe Hypo\_ortho3D\_entrainee

Grappe d'héritage de Hypo\_ortho3D\_entrainee:



Graphe de collaboration de Hypo\_ortho3D\_entrainee:



**Classes**

- class [SaveResulHypo\\_ortho3D\\_entrainee](#)

**Fonctions membres publiques**

- **Hypo\_ortho3D\_entrainee** (const double &EE1, const double &EE2, const double &EE3, const double &nu12, const double &nu13, const double &nu23, const double &GG12, const double &GG13, const double &GG23, const string &nom\_rep)

- **Hypo\_ortho3D\_entrainee** (const [Hypo\\_ortho3D\\_entrainee](#) &loi)
- [SaveResul \\* New\\_et\\_Initialise](#) ()
- void [LectureDonneesParticulieres](#) ([UtilLecture \\*](#), [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &, [SaveResul \\*saveResul](#))
- double [Module\\_compressibilite\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul \\*saveResul](#))
- virtual double [HsurH0](#) ([SaveResul \\*saveResul](#)) const
- [Loi\\_comp\\_abstraite \\* Nouvelle\\_loi\\_identique](#) () const
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)
- virtual void [Grandeur\\_particuliere](#) (bool absolue, [List\\_io< TypeQuelconque >](#) &, [Loi\\_comp\\_abstraite::SaveResul \\*](#), list< int > &decal) const
- virtual void [ListeGrandeurs\\_particulieres](#) (bool absolue, [List\\_io< TypeQuelconque >](#) &) const
- const string & [NomRepere](#) () const
- const int & [Type\\_transport](#) () const

### Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_, [TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau< TenseurBB \\* >](#) &d\_gijBB\_, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#) &giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau< BaseB >](#) &d\_giB\_tdt, [BaseH](#) &giH\_tdt, [Tableau< BaseH >](#) &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau< TenseurBB \\* >](#) &d\_epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau< TenseurBB \\* >](#) &d\_gijBB\_tdt, [Tableau< TenseurHH \\* >](#) &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau< TenseurHH \\* >](#) &d\_sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Impli](#) &ex)
- void [Calcul\\_dsigma\\_deps](#) (bool en\_base\_orthonormee, [TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [TenseurHHHH](#) &d\_sigma\_deps, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Umat\\_cont](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite::Impli \\*ex\\_impli](#), const [Met\\_abstraite::Expli\\_t\\_tdt \\*ex\\_expli\\_tdt](#), const [Met\\_abstraite::Umat\\_cont \\*ex\\_umat](#), const [List\\_io< Ddl\\_etendu >](#) \*exclure\_dd\_etend, const [List\\_io< const TypeQuelconque \\* >](#) \*exclure\_Q)

### Attributs protégés

- double **E1**
- double **E2**
- double **E3**
- double **nu12**
- double **nu13**
- double **nu23**
- double **G12**
- double **G13**
- double **G23**
- [Tableau< Fonction\\_nD \\* >](#) **fct\_para**
- bool **null\_fct\_para**
- int **type\_derive**
- string **nom\_repere**
- int **cas\_calcul**
- double **ratio\_inf\_module\_compressibilite**

- [Mat\\_pleine](#) `inv_loi`
- `int` `type_transport`
- [BaseB](#) `Op_B`
- [BaseB](#) `d_Op_B`
- [BaseB](#) `pO_B`
- [BaseH](#) `d_Op_H`
- [BaseH](#) `pO_H`
- [BaseH](#) `alpha_H`
- [Mat\\_pleine](#) `beta`
- [Mat\\_pleine](#) `gamma`
- [Mat\\_pleine](#) `beta_transpose`
- [Mat\\_pleine](#) `gamma_transpose`
- [Mat\\_pleine](#) `beta_inv`
- `int` `sortie_post`
- [Tenseur3HHHH](#) `I_x_I_HHHH`
- [Tenseur3HHHH](#) `I_xbarre_I_HHHH`
- [Tenseur3HHHH](#) `I_x_eps_HHHH`
- [Tenseur3HHHH](#) `Ixbarre_eps_HHHH`

## Amis

- `class` [SaveResulHypo\\_ortho3D\\_entrainee](#)

## Membres hérités additionnels

### 6.388.1 Documentation des fonctions membres

#### 6.388.1.1 Affiche()

`void` [Hypo\\_ortho3D\\_entrainee::Affiche](#) ( ) `const` [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.388.1.2 Calcul\_dsigma\_deps()

```
void Hypo_ortho3D_entrainee::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.388.1.3 Calcul\_DsigmaHH\_tdt()

```
void Hypo_ortho3D_entrainee::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
```

```

BaseB & giB_t,
TenseurBB & gijBB_t,
TenseurHH & gijHH_t,
BaseB & giB_tdt,
Tableau< BaseB > & d_giB_tdt,
BaseH & giH_tdt,
Tableau< BaseH > & d_giH_tdt,
TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.388.1.4 Calcul\_SigmaHH()

```

void Hypo_ortho3D_entrainee::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.388.1.5 CalculGrandeurTravail()

```

virtual void Hypo_ortho3D_entrainee::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,

```

```

const ThermoDonnee & ,
const Met_abstraite::Impli * ex_impli,
const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
const Met_abstraite::Umat_cont * ex_umat,
const List_io< Ddl_etendu > * exclure_dd_etend,
const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],

```

[virtual]

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.388.1.6 Ecriture\_base\_info\_loi()

```

void Hypo_ortho3D_entrainee::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.388.1.7 Grandeur\_particuliere()

```

void Hypo_ortho3D_entrainee::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.388.1.8 HsurH0()

```

virtual double Hypo_ortho3D_entrainee::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.388.1.9 Info\_commande\_LoisDeComp()

```

void Hypo_ortho3D_entrainee::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.388.1.10 Lecture\_base\_info\_loi()

```

void Hypo_ortho3D_entrainee::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.388.1.11 LectureDonneesParticulieres()

```

void Hypo_ortho3D_entrainee::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```



Implémente [LoiAbstraiteGeneral](#).

#### 6.388.1.12 ListeGrandeurs\_particulieres()

```
void Hypo_ortho3D_entrainee::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.388.1.13 Module\_compressibilite\_equivalent()

```
double Hypo_ortho3D_entrainee::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.388.1.14 Module\_young\_equivalent()

```
double Hypo_ortho3D_entrainee::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.388.1.15 New\_et\_Initialise()

```
SaveResul * Hypo_ortho3D_entrainee::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.388.1.16 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Hypo_ortho3D_entrainee::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.388.1.17 TestComplet()

```
int Hypo_ortho3D_entrainee::TestComplet ( ) [virtual]
```

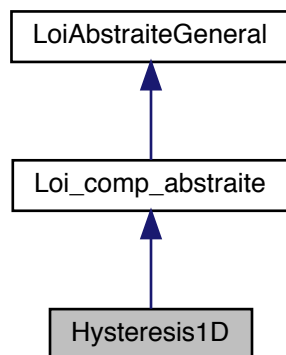
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Hypo\_ortho3D\_entrainee.↵  
h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Hypo\_ortho3D\_entrainee.↵  
cc

## 6.389 Référence de la classe Hysteresis1D

Graphe d'héritage de Hysteresis1D:





- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &, `Loi_comp_abstraite::SaveResul` \*, list< int > &decal) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &) const
- `Vecteur` & `Residu_constitutif` (const double &alpha, const `Vecteur` &x, int &test)
- `Mat_abstraite` & `Mat_tangente_constitutif` (const double &alpha, const `Vecteur` &x, `Vecteur` &resi, int &test)
- `TenseurQ1geneBHBH` & `Dsig_depsilon` (`TenseurQ1geneBHBH` &dsig\_deps)
- `Vecteur` & `Sigma_point` (const double &tau, const `Vecteur` &sigma\_tau, `Vecteur` &sig\_point, int &erreur)
- void `Verif_integrite_Sigma` (const double &tau, const `Vecteur` &sigma\_tau, int &erreur)

## Fonctions membres protégées

- bool `Coincidence` (double &unSur\_wprimeCarre, bool premiere\_charge, `SaveResulHysteresis1D` &save\_resul, double &W\_a, `List_io`< `Tenseur1BH` >::iterator &iatens, `List_io`< double >::iterator &iafct, bool &pt\_sur\_principal, `List_io`< `Tenseur1BH` >::iterator &iatens\_princ, `List_io`< double >::iterator &iafct\_princ, double &delta\_W\_a)
- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gij\_BB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &, const `Deformation` &, `Enum_dure`, const `ThermoDonnee` &, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)
- void `Init_thermo_dependance` ()
- void `CalculContrainte_tdt` (`Tableau`< double > &indicateurs\_resolution)
- void `Avancement_temporel` (const `Tenseur1BB` &delta\_epsBB, const `Tenseur1HH` &gijHH, `SaveResulHysteresis1D` &save\_resul, `Tenseur1HH` &sigHH)

## Attributs protégés

- double `xnp`
- `Courbe1D` \* `xnp_temperature`
- double `Qzero`
- `Courbe1D` \* `Qzero_temperature`
- double `xmu`
- `Courbe1D` \* `xmu_temperature`
- double `tolerance_residu`
- double `tolerance_residu_rel`
- double `maxi_delta_var_sig_sur_iter_pour_Newton`
- double `tolerance_coincidence`
- int `nb_boucle_maxi`
- int `nb_sous_increment`

- int `type_resolution_equa_constitutive`
- int `nb_maxInvCoinSurUnPas`
- int `sortie_post`
- `Tenseur1BH` `sigma_t_barre_tdt`
- `Tenseur1BH` `sigma_i_barre_BH`
- `Tenseur1BH` `sigma_barre_BH_R`
- `Tenseur1BH` `delta_sigma_barre_BH_Rat`
- `Tenseur1BH` `delta_sigma_barre_BH_Ratdt`
- `Tenseur1BH` `delta_sigma_barre_tdt_BH`
- `Tenseur1BH` `residuBH`
- `Tenseur1BH` `delta_barre_epsBH`
- `Tenseur1BH` `delta_barre_alpha_epsBH`
- double `wprime`
- `Vecteur` `residu`
- `Mat_pleine` `derResidu`
- `Algo_zero` `alg_zero`
- `Algo_edp` `alg_edp`
- int `cas_kutta`
- double `erreurAbsolue`
- double `erreurRelative`
- int `nbMaxiAppel`
- `Vecteur` `sig_point`
- `Tenseur1BH` `sigma_pointBH`
- `Tenseur1BH` `sigma_tauBH`
- `Vecteur` `sigma_tau`
- `Tenseur1BH` `delta_sigma_barre_R_a_tauBH`
- `Tenseur1BH` `betaphideltasigHB`
- `Tenseur1BH` `deuxmudeltaepsHB`
- `Tenseur1BH` `rdelta_sigma_barre_BH_Ratdt`
- `Tenseur1BH` `rdelta_sigma_barre_tdt_BH`

## Membres hérités additionnels

### 6.389.1 Documentation des fonctions membres

#### 6.389.1.1 Affiche()

`void Hysteresis1D::Affiche ( ) const [virtual]`  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.389.1.2 Calcul\_DsigmaHH\_tdt()

```
void Hysteresis1D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
```

```

Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.389.1.3 Calcul\_SigmaHH()

```

void Hysteresis1D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.389.1.4 CalculGrandeurTravail()

```

virtual void Hysteresis1D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.389.1.5 Ecriture\_base\_info\_loi()

```

void Hysteresis1D::Ecriture_base_info_loi (

```

```
    ofstream & sort,  
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.389.1.6 Grandeur\_particuliere()

```
void Hysteresis1D::Grandeur_particuliere (  
    bool absolue,  
    List_io< TypeQuelconque > & liTQ,  
    Loi_comp_abstraite::SaveResul * saveDon,  
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.389.1.7 HsurH0()

```
virtual double Hysteresis1D::HsurH0 (  
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.389.1.8 Info\_commande\_LoisDeComp()

```
void Hysteresis1D::Info_commande_LoisDeComp (  
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.389.1.9 Lecture\_base\_info\_loi()

```
void Hysteresis1D::Lecture_base_info_loi (  
    ifstream & ent,  
    const int cas,  
    LesReferences & lesRef,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.389.1.10 LectureDonneesParticulieres()

```
void Hysteresis1D::LectureDonneesParticulieres (  
    UtilLecture * entreePrinc,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.389.1.11 ListeGrandeurs\_particulieres()

```
void Hysteresis1D::ListeGrandeurs_particulieres (  
    bool absolue,  
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.389.1.12 Module\_young\_equivalent()

```
double Hysteresis1D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.389.1.13 New\_et\_Initialise()

```
SaveResul * Hysteresis1D::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.389.1.14 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Hysteresis1D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.389.1.15 TestComplet()

```
int Hysteresis1D::TestComplet ( ) [virtual]
```

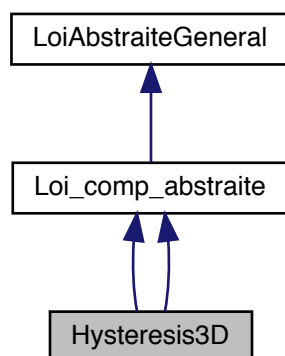
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis1D.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis1D\_2.cc

## 6.390 Référence de la classe Hysteresis3D

Graphe d'héritage de Hysteresis3D:







- [SaveResul \\* New\\_et\\_Initialise](#) ()
- void [LectureDonneesParticulieres](#) ([UtilLecture \\*](#), [LesCourbes1D](#) &lesCourbes1D)
- void [Affiche](#) () const
- int [TestComplet](#) ()
- double [Module\\_young\\_equivalent](#) ([Deformation](#) &def)
- [Loi\\_comp\\_abstraite \\* Nouvelle\\_loi\\_identique](#) () const
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &les←  
Courbes1D)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- void [Info\\_commande LoisDeComp](#) ([UtilLecture](#) &lec)
- [Vecteur & Residu\\_constitutif](#) (const double &alpha, const [Vecteur](#) &x)
- [Mat\\_pleine & Mat\\_tangente\\_constitutif](#) (const double &alpha, const [Vecteur](#) &x, [Vecteur](#) &resi)
- [TenseurQ1geneBHBH & Dsig\\_depsilon](#) ([TenseurQ1geneBHBH](#) &dsig\_deps)
- [Hysteresis3D](#) (const [Hysteresis3D](#) &loi)
- [SaveResul \\* New\\_et\\_Initialise](#) ()
- void [LectureDonneesParticulieres](#) ([UtilLecture \\*](#), [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &les←  
FonctionsnD)
- void [Affiche](#) () const
- int [TestComplet](#) ()
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul \\*saveResul](#))
- virtual double [HsurH0](#) ([SaveResul \\*saveResul](#)) const
- [Loi\\_comp\\_abstraite \\* Nouvelle\\_loi\\_identique](#) () const
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &les←  
Courbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- void [Info\\_commande LoisDeComp](#) ([UtilLecture](#) &lec)
- virtual void [Grandeur\\_particuliere](#) (bool absolue, [List\\_io< TypeQuelconque >](#) &, [Loi\\_comp\\_abstraite::SaveResul](#)  
&, [list< int >](#) &decal) const
- virtual void [ListeGrandeurs\\_particulieres](#) (bool absolue, [List\\_io< TypeQuelconque >](#) &) const
- [Vecteur & Residu\\_constitutif](#) (const double &alpha, const [Vecteur](#) &x, int &test)
- [Mat\\_abstraite & Mat\\_tangente\\_constitutif](#) (const double &alpha, const [Vecteur](#) &x, [Vecteur](#) &resi, int &test)
- [TenseurHHHH & Dsig\\_depsilon](#) (bool en\_base\_orthonormee, const [Tenseur3HH](#) &gijHH\_tdt, [TenseurHHHH](#)  
&dsig\_deps, [TenseurHH](#) &sigHH)
- void [Dsig\\_d\\_ddl](#) ([Tableau< TenseurHH \\* >](#) &d\_sigHH, [TenseurBB](#) &epsBB\_tdt, [TenseurHH](#) &gijHH\_tdt,  
[Tableau< TenseurBB \\* >](#) &d\_gijBB\_tdt, [Tableau< TenseurHH \\* >](#) &d\_gijHH\_tdt, [Tableau< TenseurBB \\* >](#)  
&d\_epsBB)
- [Vecteur & Sigma\\_point](#) (const double &tau, const [Vecteur](#) &sigma\_tau, [Vecteur](#) &sig\_point, int &erreur)
- void [Verif\\_integrite\\_Sigma](#) (const double &tau, const [Vecteur](#) &sigma\_tau, int &erreur)
- int [Type\\_de\\_transport\\_tenseur](#) () const

## Fonctions membres protégées

- bool [Coincidence](#) (double &unSur\_wBaseCarre, bool premiere\_charge, [SaveResulHysteresis3D](#) &save\_←  
resul, double &W\_a, [List\\_io< Tenseur3BH >::iterator](#) &iatens, [List\\_io< double >::iterator](#) &iafct, bool &pt\_←  
sur\_principal, [List\\_io< Tenseur3BH >::iterator](#) &iatens\_princ, [List\\_io< double >::iterator](#) &iafct\_princ, double  
&delta\_W\_a)
- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#)  
&gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_eps←  
BB\_, [TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau< TenseurBB \\* >](#) &d\_gijBB\_, double &jacobien\_0,  
double &jacobien, [TenseurHH](#) &sigHH)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#)  
&gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau< BaseB >](#) &d\_giB\_tdt, [BaseH](#) &giH\_tdt, [Tableau<](#)  
[BaseH >](#) &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau< TenseurBB \\* >](#) &d\_epsBB, [TenseurBB](#) &delta\_←  
epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau< TenseurBB \\* >](#) &d\_gijBB\_tdt, [Tableau<](#)  
[TenseurHH \\* >](#) &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [TenseurHH](#)  
&sigHH, [Tableau< TenseurHH \\* >](#) &d\_sigHH)
- void [Cal\\_wprime](#) ()
- void [Cal\\_wprime\\_et\\_der](#) ()
- bool [Coincidence](#) (bool bascule\_fct\_aide, double &unSur\_wBaseCarre, const [Tenseur3HH](#) &gijHH, const  
[Tenseur3BB](#) &gijBB, [SaveResulHysteresis3D](#) &save\_resul, double &W\_a, [List\\_io< Tenseur3BH >::iterator](#)  
&iatens, [List\\_io< double >::iterator](#) &iadef\_equi, [List\\_io< double >::iterator](#) &iafct, bool &pt\_sur\_principal,  
[List\\_io< Tenseur3BH >::iterator](#) &iaOi, bool &oi\_sur\_principal, [List\\_io< Tenseur3BH >::iterator](#) &iatens\_←  
princ, [List\\_io< double >::iterator](#) &iadef\_equi\_princ, [List\\_io< Tenseur3BH >::iterator](#) &iaOi\_princ, [List\\_io<](#)  
[double >::iterator](#) &iafct\_princ, const double &rayon\_tdt, bool force\_coincidence)

- bool **Coincidence** (bool bascule\_fct\_aide, double &unSur\_wprimeCarre, const [Tenseur3HH](#) &gijHH, const [Tenseur3BB](#) &gijBB, [SaveResulHysteresis3D](#) &save\_resul, double &W\_a, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iatens, [List\\_io](#)< double >::iterator &iadef\_equi, [List\\_io](#)< double >::iterator &iafct, bool &pt\_sur\_principal, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iaOi, bool &oi\_sur\_principal, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iatens←\_princ, [List\\_io](#)< double >::iterator &iadef\_equi\_princ, double &position\_coincidence, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iaOi\_princ, [List\\_io](#)< double >::iterator &iafct\_princ, double &delta\_W\_a, bool force\_coincidence)
- void **Calcul\_SigmaHH** ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gij←BB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_, [TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_←compressibilite, double &module\_cisaillement, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) &ex)
- void **Calcul\_DsigmaHH\_tdt** ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#) &giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#) &giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_eps←BB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met](#)←[abstraite](#)::[Impli](#) &ex)
- void **Calcul\_dsigma\_deps** (bool en\_base\_orthonormee, [TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [TenseurHHHH](#) &d\_sigma\_deps, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite](#)::[Umat\\_cont](#) &ex)
- virtual void **CalculGrandeurTravail** (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite](#)::[Impli](#) \*ex\_impli, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite](#)::[Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)
- void **Cal\_wprime** (const double &QdeltaSiRatdt, const [Tenseur3BH](#) &delta\_sig\_barre\_BH\_Ratdt, const double &Q\_OiaR, const [Tenseur3BH](#) &delta\_sig\_barre\_BH\_OiaR, const double &w\_Base, [Tenseur3BH](#) &rdelta\_sigma\_barre\_tdt\_BH, double &w\_prime, double &w\_prime\_point, bool &trajet\_neutre) const
- void **Cal\_wprime\_et\_der** (const double &QdeltaSiRatdt, const [Tenseur3BH](#) &delta\_sig\_barre\_BH\_←Ratdt, const double &Q\_OiaR, const [Tenseur3BH](#) &delta\_sig\_barre\_BH\_OiaR, const double &w\_Base, [Tenseur3BH](#) &rdelta\_sigma\_barre\_tdt\_BH, double &w\_prime, [Tenseur3BH](#) &d\_w\_prime, double &w\_←prime\_point, [Tenseur3BH](#) &d\_wprime\_point, bool &trajet\_neutre) const
- int **CentreCercle** ([Tenseur3BH](#) &sig0i\_BH, const [Tenseur3BH](#) &oc\_BH\_R, const [Tenseur3BH](#) &delta\_←sigma\_barre\_BH\_OiaR, const [Tenseur3BH](#) &delta\_sigma\_barre\_BH\_Rat)
- void **Gestion\_pointeur\_coincidence** (double &unSur\_wBaseCarre, [SaveResulHysteresis3D](#) &save\_resul, double &W\_a, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iatens, [List\\_io](#)< double >::iterator &iadef\_equi, [List\\_io](#)< double >::iterator &iafct, bool &pt\_sur\_principal, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iaOi, bool &oi\_sur←\_principal, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iatens\_princ, [List\\_io](#)< double >::iterator &iadef\_equi\_princ, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iaOi\_princ, [List\\_io](#)< double >::iterator &iafct\_princ)
- void **Gestion\_para\_Saveresult\_Modif** (const bool &pt\_sur\_principal, [SaveResulHysteresis3D](#) &save\_resul, const bool &inversion)
- bool **Autre\_coincidence\_a\_la\_tolerance\_pres** (bool bascule\_fct\_aide, const [Tenseur3HH](#) &gijHH, const [Tenseur3BB](#) &gijBB, const bool &pt\_sur\_principal, const bool &oi\_sur\_principal, [List\\_io](#)< [Tenseur3BH](#) >←::iterator &iatens, [List\\_io](#)< [Tenseur3BH](#) >::iterator &iaOi)
- void **Dsig\_d** ([MatLapack](#) &MPC)
- void **Dsig\_d\_Runge** ([MatLapack](#) &MPC)
- void **Avancement\_temporel** (const [Tenseur3BB](#) &delta\_epsBB, const [Tenseur3HH](#) &gijHH, const [Tenseur3BB](#) &gijBB, int cas, [SaveResulHysteresis3D](#) &save\_resul, [Tenseur3HH](#) &sigHH, const [EnergieMeca](#) &energ\_t, [EnergieMeca](#) &energ)
- void **Init\_thermo\_dependance** ()
- void **CalculContrainte\_tdt** (double &phi\_2\_dt, [Tableau](#)< double > &indicateurs\_resolution)
- void **UpdateEnergieHyste** ([EnergieMeca](#) &energ, const [Tenseur3BH](#) &sigma\_deb\_BH, const [Tenseur3BH](#) &sigma\_fin\_BH, const [Tenseur3BH](#) &delta\_sigma\_sur\_tau\_BH, const [Tenseur3BH](#) &delta\_sigma\_R\_tau\_←BH, const double &phi\_2\_dt, const [Tenseur3BH](#) &delta\_eps\_sur\_deltaTau\_BH, double &QdeltaSigmaRatdt, const double &wprime, const double &w\_prime\_point, const bool &trajet\_neutre, const double &def\_equi\_R, const double &def\_i\_equi, double &def\_equi)
- void **DefEqui\_et\_maxi** (double &QdeltaSigmaRatdt, const bool &trajet\_neutre, double &defEquiMaxi, const double &def\_equi\_R, const double &def\_i\_equi, double &def\_equi, const double &phi\_2\_dt, const double &wprime)

- void **ParaMateriauxAvecPhaseDeltaSig** (double &xnp, const [Tenseur3BH](#) &delta\_sigma\_barre\_BH\_OiaR, const [Tenseur3BH](#) &delta\_sig\_barre\_BH\_Ratdt, double &xmu, double &Qzero)
- void **ParaMateriauxAvecDefEqui** (double &xnp, const double &QdeltaSigmaRatdt, const double &phi\_2←\_dt, const double &wprime, const double &def\_equi\_R, const bool &trajet\_neutre, const double &def\_i\_equi, const double &defEqui\_maxi)
- void **ParaMateriauxAvecPhaseEpsilon** (const [Tenseur3HH](#) &gijHH, const [Tenseur3BB](#) &epsBB, double &xnp, double &xmu, double &Qzero)
- void **Affiche\_debug** (double &unSur\_wBaseCarre, [SaveResulHysteresis3D](#) &save\_resul, [List\\_io](#)<[Tenseur3BH](#)>::iterator &iatens, [List\\_io](#)<double>::iterator &iadef\_equi, bool &pt\_sur\_principal, [List\\_io](#)<[Tenseur3BH](#)>::iterator &iaOi, bool &oi\_sur\_principal, const [List\\_io](#)<[Tenseur3BH](#)>::iterator &iatens\_princ, const [List\\_io](#)<double>::iterator &iadef\_equi\_princ, const [List\\_io](#)<[Tenseur3BH](#)>::iterator &iaOi\_princ)
- void **Verif\_coincidence** (const [Tenseur3BB](#) &delta\_epsBB, const [Tenseur3HH](#) &gijHH, const [Tenseur3BB](#) &gijBB, int cas, [SaveResulHysteresis3D](#) &save\_resul, [Tenseur3HH](#) &sigHH, const [EnergieMeca](#) &energ\_t, [EnergieMeca](#) &energ, bool premiere\_fois)
- void **Transport\_grandeur** (const [Tenseur3BB](#) &gijBB, const [Tenseur3HH](#) &gijHH, const [Tenseur3BH](#) &aBH, [Tenseur3BH](#) &rBH)
- void **Transport\_grandeur** (const [Tenseur3BB](#) &gijBB, const [Tenseur3HH](#) &gijHH, [SaveResulHysteresis3D](#) &save\_resul)

## Attributs protégés

- double **xnp**
- int **cas\_prager**
- double **Qzero**
- double **xmu**
- double **tolerance\_residu**
- double **tolerance\_coincidence**
- int **nb\_boucle\_maxi**
- int **nb\_sous\_increment**
- [Tenseur3BH](#) **sigma\_t\_barre\_tdt**
- [Tenseur3BH](#) **sigma\_i\_barre\_BH**
- [Tenseur3BH](#) **sigma\_barre\_BH\_R**
- [Tenseur3BH](#) **delta\_sigma\_barre\_BH\_Rat**
- [Tenseur3BH](#) **delta\_sigma\_barre\_BH\_Ratdt**
- [Tenseur3BH](#) **delta\_sigma\_barre\_tdt\_BH**
- [Tenseur3BH](#) **residuBH**
- [Tenseur3BH](#) **delta\_barre\_epsBH**
- [Tenseur3BH](#) **delta\_barre\_alpha\_epsBH**
- int **wBase**
- double **wprime**
- [Tenseur3BH](#) **d\_wprime**
- [Vecteur](#) **residu**
- [Mat\\_pleine](#) **derResidu**
- [Algo\\_zero](#) **alg\_zero**
- [Tenseur3BH](#) **rdelta\_sigma\_barre\_BH\_Ratdt**
- [Tenseur3BH](#) **rdelta\_sigma\_barre\_tdt\_BH**
- double **xnp\_lue**
- [Courbe1D](#) \* **xnp\_temperature**
- [Courbe1D](#) \* **xnp\_phase**
- [Courbe1D](#) \* **xnp\_defEqui**
- double **Qzero\_lue**
- [Courbe1D](#) \* **Qzero\_temperature**
- [Courbe1D](#) \* **Qzero\_phase**
- [Courbe1D](#) \* **Qzero\_defEqui**
- double **xmu\_lue**
- [Courbe1D](#) \* **xmu\_temperature**
- [Courbe1D](#) \* **xmu\_phase**
- bool **avec\_phaseDeltaSig**
- bool **avec\_phaseEps**
- bool **avec\_defEqui**
- bool **avec\_phase\_paradefEqui**
- [Courbe1D](#) \* **coef\_paradefEqui**
- double **val\_coef\_paradefEqui**

- double `mini_QepsilonBH`
- double `tolerance_residu_rel`
- double `maxi_delta_var_sig_sur_iter_pour_Newton`
- double `tolerance_centre`
- int `nb_dichotomie`
- double `mini_rayon`
- int `type_resolution_equa_constitutive`
- int `type_calcul_comportement_tangent`
- bool `avecVarWprimeS`
- int `nb_maxInvCoinSurUnPas`
- double `min_angle_trajet_neutre`
- bool `possibilite_cosAlphaNegatif`
- double `mini_Qsig_pour_phase_sigma_Oi_tdt`
- double `force_phi_zero_si_negatif`
- double `depassement_Q0`
- int `type_de_transport_memorisation`
- double `niveau_bascule_fct_aide`
- int `sortie_post`
- double `def_equi_R`
- double `def_i_equi`
- double `defEqui_maxi`
- double `def_equi_atdt`
- `Tenseur3BH` `delta_sigma_barre_BH_Ra_i`
- `Tenseur3BH` `delta_sigma_barre_BH_OiaR`
- double `Q_OiaR`
- double `QdeltaSigmaRatdt`
- `Tenseur3BH` `oc_BH_R`
- `Vecteur` `val_initiale`
- `Vecteur` `racine`
- `MatLapack` `der_at_racine`
- bool `centre_initial`
- bool `aucun_pt_inversion`
- `Tenseur3BH` `delta_epsBH`
- double `wprime_point`
- bool `trajet_neutre`
- `Tenseur3BH` `d_wprime_point`
- `MatLapack` `derResidu`
- `MatLapack` `derResidu_adeConstant`
- `Algo_edp` `alg_edp`
- int `cas_kutta`
- double `erreurAbsolue`
- double `erreurRelative`
- int `nbMaxiAppel`
- `Vecteur` `sig_point`
- `Tenseur3BH` `sigma_pointBH`
- `Tenseur3BH` `sigma_tauBH`
- `Vecteur` `sigma_tau`
- `Tenseur3BH` `delta_sigma_barre_R_a_tauBH`
- `Tenseur3BH` `delta_sigma_barre_t_a_tauBH`
- `Tenseur3BH` `betaphideltasigHB`
- `Tenseur3BH` `deuxmudeltaepsHB`
- `Vecteur` `sig_initiale`
- `Vecteur` `dersig_initiale`
- `Vecteur` `sig_finale`
- `Vecteur` `dersig_finale`
- `Vecteur` `estime_erreur`
- bool `calcul_dResi_dsig`
- `MatLapack` `dRdsigMoinsun`
- `MatLapack` `dRdeps`
- `MatLapack` `MPC`
- `TenseurQ3geneHHHH` `dsig_ojef_HHHH`
- `MatLapack` `matriceH`
- `MatLapack` `matriceM`
- `MatLapack` `matHmoinsun`

## Amis

— class **SaveResulHysteresis3D**

## Membres hérités additionnels

### 6.390.1 Documentation des fonctions membres

#### 6.390.1.1 Affiche() [1/2]

void Hysteresis3D::Affiche ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.390.1.2 Affiche() [2/2]

void Hysteresis3D::Affiche ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.390.1.3 Calcul\_dsigma\_deps()

```
void Hysteresis3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.390.1.4 Calcul\_DsigmaHH\_tdt()

```
void Hysteresis3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
```

```

Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.390.1.5 Calcul\_SigmaHH()

```

void Hysteresis3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.390.1.6 CalculGrandeurTravail()

```

virtual void Hysteresis3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

**6.390.1.7 Ecriture\_base\_info\_loi() [1/2]**

```
void Hysteresis3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.390.1.8 Ecriture\_base\_info\_loi() [2/2]**

```
void Hysteresis3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.390.1.9 Grandeur\_particuliere()**

```
void Hysteresis3D::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.390.1.10 HsurH0()**

```
virtual double Hysteresis3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.390.1.11 Info\_commande\_LoisDeComp() [1/2]**

```
void Hysteresis3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.390.1.12 Info\_commande\_LoisDeComp() [2/2]**

```
void Hysteresis3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.390.1.13 Lecture\_base\_info\_loi()**

```
void Hysteresis3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).



**6.390.1.14 LectureDonneesParticulieres()**

```
void Hysteresis3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.390.1.15 ListeGrandeurs\_particulieres()**

```
void Hysteresis3D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.390.1.16 Module\_young\_equivalent()**

```
double Hysteresis3D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.390.1.17 New\_et\_Initialise() [1/2]**

```
SaveResul * Hysteresis3D::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.390.1.18 New\_et\_Initialise() [2/2]**

```
SaveResul * Hysteresis3D::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.390.1.19 Nouvelle\_loi\_identique() [1/2]**

```
Loi_comp_abstraite * Hysteresis3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.390.1.20 Nouvelle\_loi\_identique() [2/2]**

```
Loi_comp_abstraite * Hysteresis3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.390.1.21 TestComplet() [1/2]**

```
int Hysteresis3D::TestComplet ( ) [virtual]
```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

**6.390.1.22 TestComplet() [2/2]**

```
int Hysteresis3D::TestComplet ( ) [virtual]
```

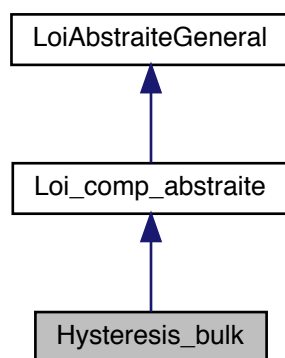
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

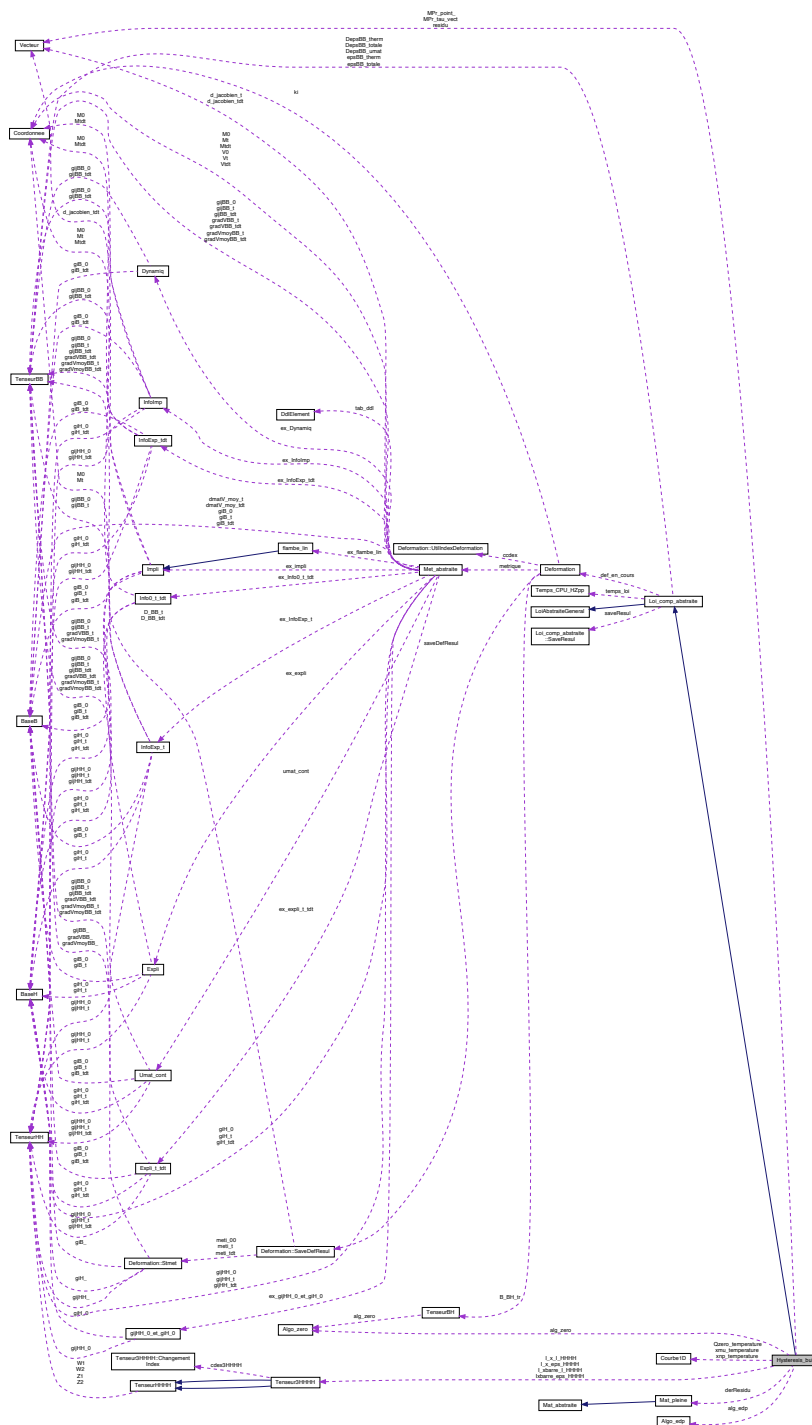
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Copie\_de\_Hysteresis3↔D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Copie\_de\_Hysteresis3↔D.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis3D.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis3D\_2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis3D\_3.cc

## 6.391 Référence de la classe Hysteresis\_bulk

Graphe d'héritage de Hysteresis\_bulk:



Graphe de collaboration de Hysteresis\_bulk:



### Classes

- class [SaveResulHysteresis\\_bulk](#)

### Fonctions membres publiques

- [Hysteresis\\_bulk](#) (const [Hysteresis\\_bulk](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()

- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &, `Loi_comp_abstraite::SaveResul` \*, list< int > &decal) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &) const
- virtual void `Activation_stockage_grandeurs_quelconques` (list< `EnumTypeQuelconque` > &listEnuQuelc)
- virtual void `Insertion_conteneur_dans_save_result` (`SaveResul` \*saveResul)
- `Vecteur` & `Residu_constitutif` (const double &alpha, const `Vecteur` &x, int &test)
- `Mat_abstraite` & `Mat_tangente_constitutif` (const double &alpha, const `Vecteur` &x, `Vecteur` &resi, int &test)
- void `Dsig_depsilon` (double &T\_d\_pres\_d\_V, bool en\_base\_orthonormee, const `Tenseur3HH` &gijHH\_tdt, `TenseurHHHH` \*dsig\_deps, const double &V\_tdt)
- `Vecteur` & `Pression_point` (const double &tau, const `Vecteur` &sigma\_tau, `Vecteur` &sig\_point, int &erreur)
- void `Verif_integrite_Pression` (const double &tau, const `Vecteur` &MPr\_tau, int &erreur)
- void `Affiche_debug` (double &unSur\_wBaseCarre, `SaveResulHysteresis_bulk` &save\_resul, `List_io`< double >::iterator &iatens, bool &pt\_sur\_principal, const `List_io`< double >::iterator &iatens\_princ)

## Fonctions membres protégées

- bool `Coincidence` (bool &aucun\_pt\_inversion, double &unSur\_wprimeCarre, bool premiere\_charge, `SaveResulHysteresis_bulk` &save\_resul, double &W\_a, `List_io`< double >::iterator &iatens, `List_io`< double >::iterator &iafct, bool &pt\_sur\_principal, `List_io`< double >::iterator &iatens\_princ, `List_io`< double >::iterator &iafct\_princ, double &delta\_W\_a, bool force\_coincidence, const double &MPr\_tdt)
- bool `Inversion_et_Coincidence` (bool &aucun\_pt\_inversion, double &unSur\_wprimeCarre, `SaveResulHysteresis_bulk` &save\_resul, double &W\_a, `List_io`< double >::iterator &iatens, const double &delta\_MPr\_tatdt, `List_io`< double >::iterator &iafct, bool &pt\_sur\_principal, `List_io`< double >::iterator &iatens\_princ, `List_io`< double >::iterator &iafct\_princ, double &delta\_W\_a, bool force\_coincidence, const double &MPr\_tdt)
- void `Gestion_pointeur_coincidence` (double &unSur\_wprimeCarre, `SaveResulHysteresis_bulk` &save\_resul, double &W\_a, bool &aucun\_pt\_inversion, `List_io`< double >::iterator &iatens, `List_io`< double >::iterator &iafct, bool &pt\_sur\_principal, `List_io`< double >::iterator &iatens\_princ, `List_io`< double >::iterator &iafct\_princ, const double &MPr\_tdt)
- void `Gestion_pointeur_Inversion_et_Coincidence` (double &unSur\_wprimeCarre, `SaveResulHysteresis_bulk` &save\_resul, double &W\_a, bool &aucun\_pt\_inversion, `List_io`< double >::iterator &iatens, `List_io`< double >::iterator &iafct, bool &pt\_sur\_principal, `List_io`< double >::iterator &iatens\_princ, `List_io`< double >::iterator &iafct\_princ, const double &MPr\_tdt)
- void `Gestion_para_Saveresult_Modif` (const bool &pt\_sur\_principal, `SaveResulHysteresis_bulk` &save\_resul, const bool &inversion)
- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB\_tdt, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH`

- &sigHH, [TenseurHHHH](#) &d\_sigma\_deps, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Umat\\_cont](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecaInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)
- void [Init\\_thermo\\_dependance](#) ()
- void [CalculPression\\_tdt](#) ([Tableau](#)< double > &indicateurs\_resolution)
- void [Avancement\\_temporel](#) (const [Tenseur3HH](#) &gijHH, const [Tenseur3BB](#) &gijBB, int cas, [SaveResulHysteresis\\_bulk](#) &save\_resul, [Tenseur3HH](#) &sigHH, const [EnergieMeca](#) &energ\_t, [EnergieMeca](#) &energ)

## Attributs protégés

- double [xnp](#)
- [Courbe1D](#) \* [xnp\\_temperature](#)
- double [Qzero](#)
- [Courbe1D](#) \* [Qzero\\_temperature](#)
- double [xmu](#)
- [Courbe1D](#) \* [xmu\\_temperature](#)
- double [tolerance\\_residu](#)
- double [tolerance\\_residu\\_rel](#)
- double [maxi\\_delta\\_var\\_sig\\_sur\\_iter\\_pour\\_Newton](#)
- double [tolerance\\_coincidence](#)
- int [nb\\_boucle\\_maxi](#)
- int [nb\\_sous\\_increment](#)
- int [type\\_resolution\\_equa\\_constitutive](#)
- int [nb\\_maxInvCoinSurUnPas](#)
- double [depassement\\_Q0](#)
- int [sortie\\_post](#)
- [Tenseur3HHHH](#) [I\\_x\\_I\\_HHHH](#)
- [Tenseur3HHHH](#) [I\\_xbarre\\_I\\_HHHH](#)
- [Tenseur3HHHH](#) [I\\_x\\_eps\\_HHHH](#)
- [Tenseur3HHHH](#) [I\\_xbarre\\_eps\\_HHHH](#)
- double [MPr\\_t\\_tdt](#)
- double [MPr\\_i](#)
- double [MPr\\_R](#)
- double [delta\\_MPr\\_Rat](#)
- double [delta\\_MPr\\_Ratdt](#)
- double [delta\\_MPr\\_tatdt](#)
- double [residuBH](#)
- double [delta\\_V](#)
- double [delta\\_alpha\\_V](#)
- double [wprime](#)
- [Vecteur](#) [residu](#)
- [Mat\\_pleine](#) [derResidu](#)
- [Algo\\_zero](#) [alg\\_zero](#)
- [Algo\\_edp](#) [alg\\_edp](#)
- int [cas\\_kutta](#)
- double [erreurAbsolue](#)
- double [erreurRelative](#)
- int [nbMaxiAppel](#)
- [Vecteur](#) [MPr\\_point](#)
- double [MPr\\_point](#)
- double [MPr\\_tau](#)
- [Vecteur](#) [MPr\\_tau\\_vect](#)
- double [delta\\_MPr\\_R\\_a\\_tau](#)
- double [rdelta\\_MPr\\_Ratdt](#)
- double [rdelta\\_MPr\\_tatdt](#)
- bool [aucun\\_pt\\_inversion](#)

## Membres hérités additionnels

### 6.391.1 Documentation des fonctions membres

**6.391.1.1 Activation\_stockage\_grandeurs\_quelconques()**

```
void Hysteresis_bulk::Activation_stockage_grandeurs_quelconques (
    list< EnumTypeQuelconque > & listEnuQuelc ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.391.1.2 Affiche()**

```
void Hysteresis_bulk::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.391.1.3 Calcul\_dsigma\_deps()**

```
void Hysteresis_bulk::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.391.1.4 Calcul\_DsigmaHH\_tdt()**

```
void Hysteresis_bulk::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
```

```

    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.391.1.5 Calcul\_SigmaHH()

```

void Hysteresis_bulk::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.391.1.6 CalculGrandeurTravail()

```

virtual void Hysteresis_bulk::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.391.1.7 Ecriture\_base\_info\_loi()

```

void Hysteresis_bulk::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

### 6.391.1.8 Grandeur\_particuliere()

```
void Hysteresis_bulk::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.391.1.9 HsurH0()

```
virtual double Hysteresis_bulk::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.391.1.10 Info\_commande\_LoisDeComp()

```
void Hysteresis_bulk::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.391.1.11 Insertion\_conteneur\_dans\_save\_result()

```
void Hysteresis_bulk::Insertion_conteneur_dans_save_result (
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.391.1.12 Lecture\_base\_info\_loi()

```
void Hysteresis_bulk::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.391.1.13 LectureDonneesParticulieres()

```
void Hysteresis_bulk::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.391.1.14 ListeGrandeurs\_particulieres()

```
void Hysteresis_bulk::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).



**6.391.1.15 Module\_young\_equivalent()**

```
double Hysteresis_bulk::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.391.1.16 New\_et\_Initialise()**

```
Hysteresis_bulk::SaveResul * Hysteresis_bulk::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.391.1.17 Nouvelle\_loi\_identique()**

```
Loi_comp_abstraite * Hysteresis_bulk::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.391.1.18 TestComplet()**

```
int Hysteresis_bulk::TestComplet ( ) [virtual]
```

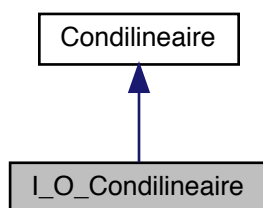
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

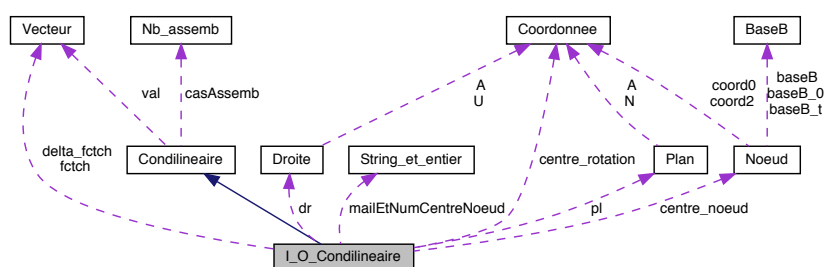
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis\_bulk.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis\_bulk.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis\_bulk\_2.cc

**6.392 Référence de la classe I\_O\_Condilinaire**

Graphe d'héritage de I\_O\_Condilinaire:



Graphe de collaboration de I\_O\_Condilinaire:



## Fonctions membres publiques

- **I\_O\_Condilinaire** (const [I\\_O\\_Condilinaire](#) &nd)
- const string & **NomRef** () const
- const string \* **NomMaillage** () const
- const [Tableau](#)< string > & **ReferenceAssociees** () const
- const [Tableau](#)< string > & **NomMaillageAssociees** () const
- bool **MemeCibleMaisDataDifferentes** ([I\\_O\\_Condilinaire](#) &d)
- void **Affiche** () const
- [I\\_O\\_Condilinaire](#) & **operator=** (const [I\\_O\\_Condilinaire](#) &d)
- bool **operator==** ([I\\_O\\_Condilinaire](#) &a) const
- bool **operator!=** ([I\\_O\\_Condilinaire](#) &a) const
- void **Lecture** ([UtilLecture](#) &entreePrinc)
- bool **ConditionRelative** () const
- bool **PlanDroite** () const
- bool **Coherence\_condi\_PlanDroite** (int indi) const
- bool **StricteEgalite** () const
- const [String\\_et\\_entier](#) \* **RefNoeudRotation** () const
- void **ChangeCentreNoeudRotation** ([Noeud](#) \*noe)
- void **ActualiseDirectionPlanDroite** ()
- [Coordonnee](#) & **ProjectionSurPlanDroite\_et\_calValCondiLin** ([Coordonnee](#) &M)
- int **ExisteFonctionChargeCoef** () const
- const string & **NomFctch** (int i) const
- void **Valeur\_fonctionChargeCoef** (const [Vecteur](#) &fctch\_, const [Vecteur](#) &delta\_fctch\_)
- [Condilinaire](#) & **ConstructionCondition** ([Tableau](#)< [Noeud](#) \* > &t\_noe, [Condilinaire](#) &condi\_actuelle)
- void **Info\_commande\_conditionLineaire** (ofstream &sort, bool plusieurs\_maillages)
- Enum\_ddl **TypeEnu** () const
- bool **Temps\_actif** (const double &temps) const
- bool **Pas\_a\_prendre\_en\_compte** (const double &temps) const
- bool **Pas\_a\_prendre\_en\_compte\_dans\_intervalle** (const double &temps) const
- void **Validation** (const double &temps)
- bool **Etat\_validation** () const
- void **Initialisation\_condil** ()

## Attributs protégés

- string \* **nom\_maillage**
- string **refe**
- [Tableau](#)< string > **refs\_associe**
- [Tableau](#)< string > **nom\_mail\_associe**
- double **t\_min**
- double **t\_max**
- double **echelle**
- Enum\_ddl **enu**
- int **nbddlffamille**
- int **condition\_relative**
- int **precedent**

- double **valeur\_precedente\_t**
- [Tableau](#)< string > **tab\_co\_charge**
- [Vecteur](#) **fctch**
- [Vecteur](#) **delta\_fctch**
- bool **def\_auto\_par\_rotation**
- int **type\_centre**
- [Coordonnee](#) **centre\_rotation**
- [String\\_et\\_entier](#) **mailEtNumCentreNoeud**
- [Noeud](#) \* **centre\_noeud**
- [Plan](#) **pl**
- [Droite](#) **dr**
- bool **stricte\_egalite**

### Amis

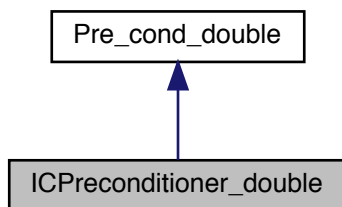
- istream & **operator**>> (istream &, [I\\_O\\_Condilinaire](#) &)
- ostream & **operator**<< (ostream &, const [I\\_O\\_Condilinaire](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

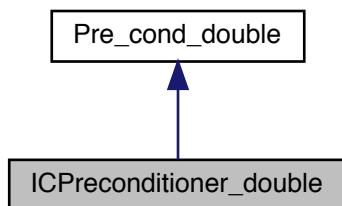
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/I\_O\_Condilinaire.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/I\_O\_Condilinaire.cc

## 6.393 Référence de la classe ICPreconditioner\_double

Grappe d'héritage de ICPreconditioner\_double:



Grappe de collaboration de ICPreconditioner\_double:



## Fonctions membres publiques

- `ICPreconditioner_double` (const `CompCol_Mat_double` &A)
- `ICPreconditioner_double` (const `CompRow_Mat_double` &A)
- `ICPreconditioner_double` (const `Mat_abstraite` &A)
- `VECTOR_double solve` (const `VECTOR_double` &x) const
- `VECTOR_double trans_solve` (const `VECTOR_double` &x) const

## Fonctions membres protégées

- void `Init_CompCol_Mat_double` (const `CompCol_Mat_double` &A)
- void `Init_CompRow_Mat_double` (const `CompRow_Mat_double` &A)
- void `Init_Mat_creuse_CompCol` (const `Mat_abstraite` &A)

## 6.393.1 Documentation des fonctions membres

### 6.393.1.1 solve()

```
VECTOR_double ICPreconditioner_double::solve (
    const VECTOR_double & x ) const [virtual]
```

Implémente [Pre\\_cond\\_double](#).

### 6.393.1.2 trans\_solve()

```
VECTOR_double ICPreconditioner_double::trans_solve (
    const VECTOR_double & x ) const [virtual]
```

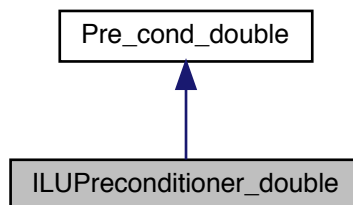
Implémente [Pre\\_cond\\_double](#).

La documentation de cette classe a été générée à partir du fichier suivant :

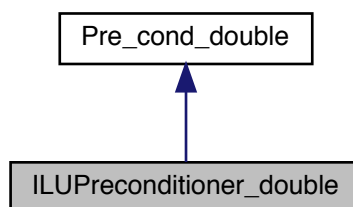
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/icpre_double_GR.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/icpre_double_GR.cc`

## 6.394 Référence de la classe ILUPreconditioner\_double

Graphe d'héritage de `ILUPreconditioner_double`:



Graphe de collaboration de ILUPreconditioner\_double:



## Fonctions membres publiques

- **ILUPreconditioner\_double** (const [Mat\\_abstraite](#) &A)
- **VECTOR\_double** [solve](#) (const **VECTOR\_double** &x) const
- **VECTOR\_double** [trans\\_solve](#) (const **VECTOR\_double** &x) const

### 6.394.1 Documentation des fonctions membres

#### 6.394.1.1 solve()

```
VECTOR_double ILUPreconditioner_double::solve (  
    const VECTOR_double & x ) const [inline], [virtual]
```

Implémente [Pre\\_cond\\_double](#).

#### 6.394.1.2 trans\_solve()

```
VECTOR_double ILUPreconditioner_double::trans_solve (  
    const VECTOR_double & x ) const [inline], [virtual]
```

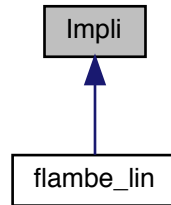
Implémente [Pre\\_cond\\_double](#).

La documentation de cette classe a été générée à partir du fichier suivant :

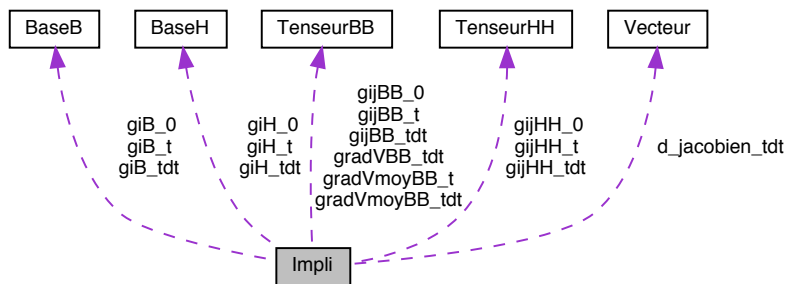
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/ilupre\_double\_GR.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/ilupre\_double\_GR.cc

## 6.395 Référence de la classe Impli

Graphe d'héritage de Impli:



Graphe de collaboration de Impli:



### Fonctions membres publiques

- **Impli** (*BaseB* \*ggiB\_0, *BaseH* \*ggiH\_0, *BaseB* \*ggiB\_t, *BaseH* \*ggiH\_t, *BaseB* \*ggiB\_tdt, *Tableau*< *BaseB* > \*d\_ggiB\_tdt, *BaseH* \*ggiH\_tdt, *Tableau*< *BaseH* > \*d\_ggiH\_tdt, *TenseurBB* \*ggijBB\_0, *TenseurHH* \*ggijHH\_0, *TenseurBB* \*ggijBB\_t, *TenseurHH* \*ggijHH\_t, *TenseurBB* \*ggijBB\_tdt, *TenseurHH* \*ggijHH\_tdt, *TenseurBB* \*ggradVmoyBB\_t, *TenseurBB* \*ggradVmoyBB\_tdt, *TenseurBB* \*ggradVBB\_tdt, *Tableau*< *TenseurBB* \* > \*gd\_gijBB\_tdt, *Tableau2*< *TenseurBB* \* > \*gd2\_gijBB\_tdt, *Tableau*< *TenseurHH* \* > \*gd\_gijHH\_tdt, double \*gjacobien, double \*gjacobien\_t, double \*gjacobien\_0, *Vecteur* \*gd\_jacobien\_tdt, *Tableau*< *TenseurBB* \* > \*gd\_gradVmoyBB\_t, *Tableau*< *TenseurBB* \* > \*gd\_gradVmoyBB\_tdt, *Tableau*< *TenseurBB* \* > \*gd\_gradVBB\_t, *Tableau*< *TenseurBB* \* > \*gd\_gradVBB\_tdt)
- **Impli** (const *Impli* &ex)
- **Impli** & **operator=** (const *Impli* &ex)
- void **Mise\_a\_jour\_grandeur** (*BaseB* \*ggiB\_0, *BaseH* \*ggiH\_0, *BaseB* \*ggiB\_t, *BaseH* \*ggiH\_t, *BaseB* \*ggiB\_tdt, *Tableau*< *BaseB* > \*d\_ggiB\_tdt, *BaseH* \*ggiH\_tdt, *Tableau*< *BaseH* > \*d\_ggiH\_tdt, *TenseurBB* \*ggijBB\_0, *TenseurHH* \*ggijHH\_0, *TenseurBB* \*ggijBB\_t, *TenseurHH* \*ggijHH\_t, *TenseurBB* \*ggijBB\_tdt, *TenseurHH* \*ggijHH\_tdt, *TenseurBB* \*ggradVmoyBB\_t, *TenseurBB* \*ggradVmoyBB\_tdt, *TenseurBB* \*ggradVBB\_tdt, *Tableau*< *TenseurBB* \* > \*gd\_gijBB\_tdt, *Tableau2*< *TenseurBB* \* > \*gd2\_gijBB\_tdt, *Tableau*< *TenseurHH* \* > \*gd\_gijHH\_tdt, double \*gjacobien, double \*gjacobien\_t, double \*gjacobien\_0, *Vecteur* \*gd\_jacobien\_tdt, *Tableau*< *TenseurBB* \* > \*gd\_gradVmoyBB\_t, *Tableau*< *TenseurBB* \* > \*gd\_gradVmoyBB\_tdt, *Tableau*< *TenseurBB* \* > \*gd\_gradVBB\_t, *Tableau*< *TenseurBB* \* > \*gd\_gradVBB\_tdt)
- void **Recup\_grandeur\_0\_t** (*BaseB* &ggiB\_0, *BaseH* &ggiH\_0, *BaseB* &ggiB\_t, *BaseH* &ggiH\_t, *TenseurBB* &ggijBB\_0, *TenseurHH* &ggijHH\_0, *TenseurBB* &ggijBB\_t, *TenseurHH* &ggijHH\_t, *TenseurBB* &, double &gjacobien\_t, double &gjacobien\_0)

- void **Passage\_de\_Ordre2\_vers\_Ordre3** (const [Impli](#) &ex, bool plusZero, int type\_recopie)
- void **Passage\_de\_Ordre3\_vers\_Ordre2** (const [Impli](#) &ex, int type\_recopie)
- void **Passage\_de\_Ordre1\_vers\_Ordre3** (const [Impli](#) &ex, bool plusZero, int type\_recopie)
- void **Passage\_de\_Ordre3\_vers\_Ordre1** (const [Impli](#) &ex, int type\_recopie)
- void **Passage\_de\_Ordre1\_vers\_Ordre2** (const [Impli](#) &ex, bool plusZero, int type\_recopie)
- void **Passage\_de\_Ordre2\_vers\_Ordre1** (const [Impli](#) &ex, int type\_recopie)
- [Umat\\_cont](#) & [Recup\\_Umat\\_cont](#) ([Umat\\_cont](#) &ex) const

### Attributs publics

- [BaseB](#) \* **giB\_0**
- [BaseH](#) \* **giH\_0**
- [BaseB](#) \* **giB\_t**
- [BaseH](#) \* **giH\_t**
- [BaseB](#) \* **giB\_tdt**
- [Tableau](#)< [BaseB](#) > \* **d\_giB\_tdt**
- [BaseH](#) \* **giH\_tdt**
- [Tableau](#)< [BaseH](#) > \* **d\_giH\_tdt**
- [TenseurBB](#) \* **gijBB\_0**
- [TenseurHH](#) \* **gijHH\_0**
- [TenseurBB](#) \* **gijBB\_t**
- [TenseurHH](#) \* **gijHH\_t**
- [TenseurBB](#) \* **gijBB\_tdt**
- [TenseurHH](#) \* **gijHH\_tdt**
- [TenseurBB](#) \* **gradVmoyBB\_t**
- [TenseurBB](#) \* **gradVmoyBB\_tdt**
- [TenseurBB](#) \* **gradVBB\_tdt**
- [Tableau](#)< [TenseurBB](#) \* > \* **d\_gijBB\_tdt**
- [Tableau2](#)< [TenseurBB](#) \* > \* **d2\_gijBB\_tdt**
- [Tableau](#)< [TenseurHH](#) \* > \* **d\_gijHH\_tdt**
- double \* **jacobien\_tdt**
- double \* **jacobien\_t**
- double \* **jacobien\_0**
- [Vecteur](#) \* **d\_jacobien\_tdt**
- [Tableau](#)< [TenseurBB](#) \* > \* **d\_gradVmoyBB\_t**
- [Tableau](#)< [TenseurBB](#) \* > \* **d\_gradVmoyBB\_tdt**
- [Tableau](#)< [TenseurBB](#) \* > \* **d\_gradVBB\_t**
- [Tableau](#)< [TenseurBB](#) \* > \* **d\_gradVBB\_tdt**

La documentation de cette classe a été générée à partir du fichier suivant :

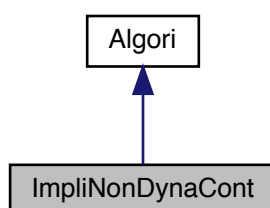
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔  
abstraite\_struc\_donnees.h

## 6.396 Référence de la classe ImpliNonDynaCont

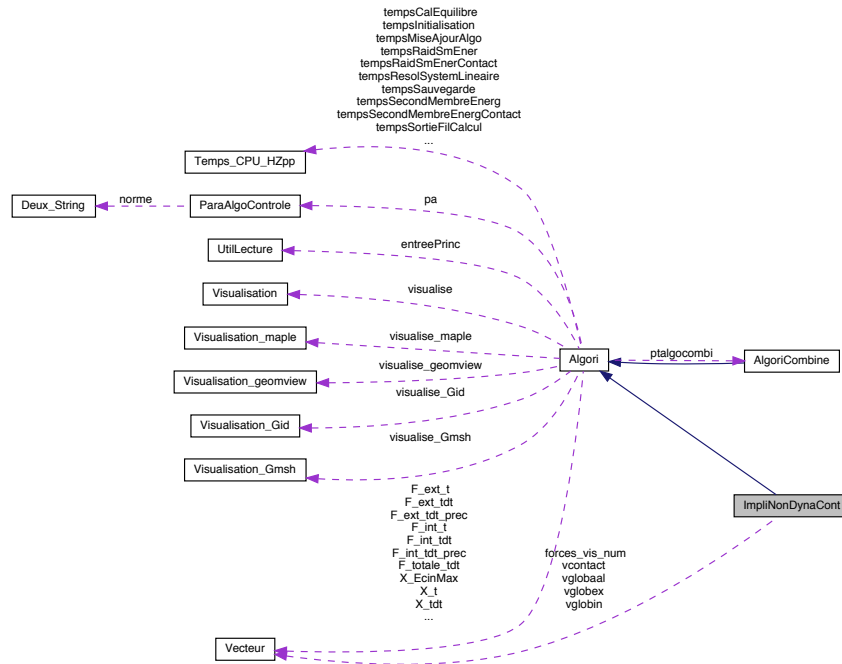
BUT: Algorithme de calcul implicite non dynamique avec contact , pour de la mecanique en coordonnees materielles entrainees.

```
#include <ImpliNonDynaCont.h>
```

Grphe d'héritage de ImpliNonDynaCont:



Graphe de collaboration de ImpliNonDynaCont:



## Fonctions membres publiques

- **ImpliNonDynaCont** (const bool avec\_typeDeCal, const list< [EnumSousTypeCalcul](#) > &soustype, const list< bool > &avec\_soustypeDeCal, [UtilLecture](#) &entreePrinc)
- **ImpliNonDynaCont** (const [ImpliNonDynaCont](#) &algo)
- **Algori \* New\_idem** (const [Algori](#) \*algo) const
- void **Execution** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*varExpor, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **InitAlgorithme** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **MiseAJourAlgo** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **CalEquilibre** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [Tableau](#)< [Fonction\\_nD](#) > \*tb\_combiner)
- void **FinCalcul** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesCourbes1D](#) \*, [LesFonctions\\_nD](#) \*, [VariablesExporter](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*)
- void **SchemaXML\_Algori** (ofstream &, const [Enum\\_IO\\_XML](#)) const

## Attributs protégés

- [Vecteur](#) **vglobin**
- [Vecteur](#) **vglobex**
- [Vecteur](#) **vcontact**
- [Vecteur](#) **vglobaal**
- [Vecteur](#) **forces\_vis\_num**



## Membres hérités additionnels

### 6.396.1 Description détaillée

BUT: Algorithme de calcul implicite non dynamique avec contact , pour de la mecanique en coordonnees materielles entrainees.

### 6.396.2 Documentation des fonctions membres

#### 6.396.2.1 CalEquilibre()

```
void ImpliNonDynaCont::CalEquilibre (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    Tableau< Fonction_nD * > * tb_combiner ) [inline], [virtual]
```

Implémente [Algori](#).

#### 6.396.2.2 Execution()

```
void ImpliNonDynaCont::Execution (
    ParaGlob * paraGlob,
    LesMaillages * lesMail,
    LesReferences * lesRef,
    LesCourbes1D * lesCourbes1D,
    LesFonctions_nD * lesFonctionsnD,
    VariablesExporter * varExpor,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * divStock,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats ) [virtual]
```

Implémente [Algori](#).

#### 6.396.2.3 FinCalcul()

```
void ImpliNonDynaCont::FinCalcul (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
```

```

    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]

```

Implémente [Algori](#).

#### 6.396.2.4 InitAlgorithme()

```

void ImpliNonDynaCont::InitAlgorithme (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]

```

Implémente [Algori](#).

#### 6.396.2.5 MiseAJourAlgo()

```

void ImpliNonDynaCont::MiseAJourAlgo (
    ParaGlob * ,
    LesMaillages * ,
    LesReferences * ,
    LesCourbes1D * ,
    LesFonctions_nD * ,
    VariablesExporter * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ) [inline], [virtual]

```

Implémente [Algori](#).

#### 6.396.2.6 New\_idem()

```

Algori * ImpliNonDynaCont::New_idem (
    const Algori * algo ) const [inline], [virtual]

```

Implémente [Algori](#).

#### 6.396.2.7 SchemaXML\_Algori()

```

void ImpliNonDynaCont::SchemaXML_Algori (
    ofstream & ,
    const Enum_IO_XML ) const [inline], [virtual]

```

Implémente [Algori](#).

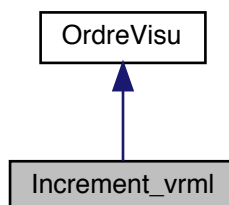
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/ImpliNon↔DynaCont.h

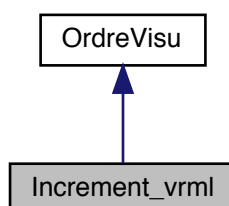
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/GalerkinContinu/AlgoStatiques/ImpliNon↔  
DynaCont.cc

## 6.397 Référence de la classe Increment\_vrml

Graphe d'héritage de Increment\_vrml:



Graphe de collaboration de Increment\_vrml:



### Fonctions membres publiques

- **Increment\_vrml** (const [Increment\\_vrml](#) &algo)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#)↔  
::EnumTypeIncre, int, bool, const map< string, const double \* , std::less< string > > &, const [List\\_io](#)<  
[TypeQuelconque](#) > &listeVecGlob)
- void **ChoixOrdre** ()
- void **Init\_list\_inc** (list< int > &li\_inc)
- void **Init\_list\_inc\_DebuEtFin** ()
- const list< int > & **List\_choisit** ()
- void **Impos\_list** (const list< int > &li\_inc)
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

### Membres hérités additionnels

#### 6.397.1 Documentation des fonctions membres

### 6.397.1.1 ChoixOrdre()

```
void Increment_vrml::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.397.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Increment_vrml::Ecriture_parametres_OrdreVisu (
```

```
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.397.1.3 ExeOrdre()

```
void Increment_vrml::ExeOrdre (
```

```
    ParaGlob * ,
```

```
    const Tableau< int > & ,
```

```
    LesMaillages * ,
```

```
    bool ,
```

```
    LesReferences * ,
```

```
    LesLoisDeComp * ,
```

```
    DiversStockage * ,
```

```
    Charge * ,
```

```
    LesCondLim * ,
```

```
    LesContacts * ,
```

```
    Resultats * ,
```

```
    UtilLecture & ,
```

```
    OrdreVisu::EnumTypeIncre ,
```

```
    int ,
```

```
    bool ,
```

```
    const map< string, const double * , std::less< string > > & ,
```

```
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.397.1.4 Lecture\_parametres\_OrdreVisu()

```
void Increment_vrml::Lecture_parametres_OrdreVisu (
```

```
    UtilLecture & entreePrinc ) [virtual]
```

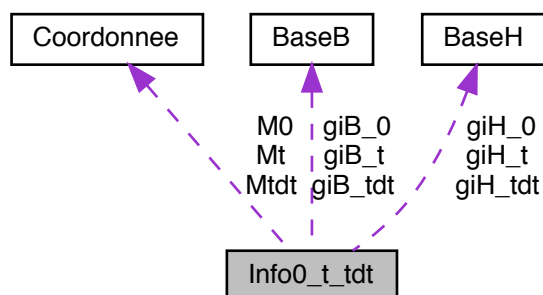
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Increment\_vrml.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Increment\_vrml.cc

## 6.398 Référence de la classe Info0\_t\_tdt

Graphe de collaboration de Info0\_t\_tdt:



### Fonctions membres publiques

- `Info0_t_tdt` (`Coordonnee *gM0`, `Coordonnee *gMt`, `Coordonnee *gMtdt`, `BaseB *ggiB_0`, `BaseB *ggiB_t`, `BaseB *ggiB_tdt`, `BaseH *ggiH_0`, `BaseH *ggiH_t`, `BaseH *ggiH_tdt`)
- `Info0_t_tdt` (`const Info0_t_tdt &ex`)
- `Info0_t_tdt &operator=` (`const Info0_t_tdt &ex`)
- `void Mise_a_jour_grandeur` (`Coordonnee *gM0`, `Coordonnee *gMt`, `Coordonnee *gMtdt`, `BaseB *ggiB_0`, `BaseB *ggiB_t`, `BaseB *ggiB_tdt`, `BaseH *ggiH_0`, `BaseH *ggiH_t`, `BaseH *ggiH_tdt`)

### Attributs publics

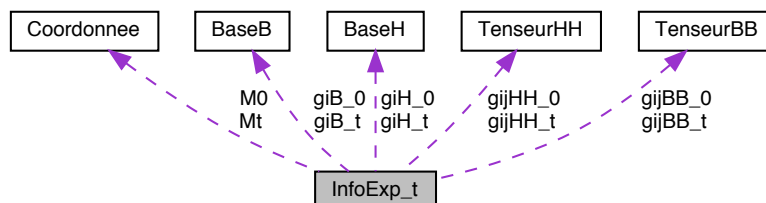
- `Coordonnee * M0`
- `Coordonnee * Mt`
- `Coordonnee * Mtdt`
- `BaseB * giB_0`
- `BaseB * giB_t`
- `BaseB * giB_tdt`
- `BaseH * giH_0`
- `BaseH * giH_t`
- `BaseH * giH_tdt`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Met_↔ abstraite_struc_donnees.h`

## 6.399 Référence de la classe InfoExp\_t

Graphe de collaboration de InfoExp\_t:



### Fonctions membres publiques

- **InfoExp\_t** ([Coordonnee](#) \*gM0, [Coordonnee](#) \*gMt, [BaseB](#) \*ggiB\_0, [BaseB](#) \*ggiB\_t, [BaseH](#) \*ggiH\_0, [BaseH](#) \*ggiH\_t, [TenseurHH](#) \*ggijHH\_t, [TenseurBB](#) \*ggijBB\_t, [TenseurBB](#) \*ggijBB\_0, [TenseurHH](#) \*ggijHH\_0)
- **InfoExp\_t** (const [InfoExp\\_t](#) &ex)
- **InfoExp\_t & operator=** (const [InfoExp\\_t](#) &ex)
- void **Mise\_a\_jour\_grandeur** ([Coordonnee](#) \*gM0, [Coordonnee](#) \*gMt, [BaseB](#) \*ggiB\_0, [BaseB](#) \*ggiB\_t, [BaseH](#) \*ggiH\_0, [BaseH](#) \*ggiH\_t, [TenseurHH](#) \*ggijHH\_t, [TenseurBB](#) \*ggijBB\_t, [TenseurBB](#) \*ggijBB\_0, [TenseurHH](#) \*ggijHH\_0)

### Attributs publics

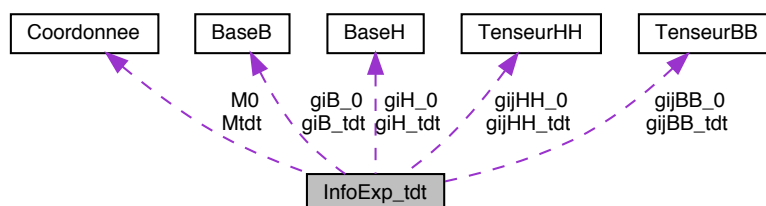
- [Coordonnee](#) \* **M0**
- [Coordonnee](#) \* **Mt**
- [BaseB](#) \* **giB\_0**
- [BaseB](#) \* **giB\_t**
- [BaseH](#) \* **giH\_0**
- [BaseH](#) \* **giH\_t**
- [TenseurHH](#) \* **gijHH\_t**
- [TenseurBB](#) \* **gijBB\_t**
- [TenseurBB](#) \* **gijBB\_0**
- [TenseurHH](#) \* **gijHH\_0**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔ abstraite\_struc\_donnees.h

## 6.400 Référence de la classe InfoExp\_tdt

Graphe de collaboration de InfoExp\_tdt:



## Fonctions membres publiques

- **InfoExp\_tdt** ([Coordonnee](#) \*gM0, [Coordonnee](#) \*gMtdt, [BaseB](#) \*ggiB\_0, [BaseB](#) \*ggiB\_tdt, [BaseH](#) \*ggiH\_0, [BaseH](#) \*ggiH\_tdt, [TenseurHH](#) \*ggijHH\_tdt, [TenseurBB](#) \*ggijBB\_tdt, [TenseurBB](#) \*ggijBB\_0, [TenseurHH](#) \*ggijHH\_0)
- **InfoExp\_tdt** (const [InfoExp\\_tdt](#) &ex)
- **InfoExp\_tdt** & **operator=** (const [InfoExp\\_tdt](#) &ex)
- void **Mise\_a\_jour\_grandeur** ([Coordonnee](#) \*gM0, [Coordonnee](#) \*gMtdt, [BaseB](#) \*ggiB\_0, [BaseB](#) \*ggiB\_tdt, [BaseH](#) \*ggiH\_0, [BaseH](#) \*ggiH\_tdt, [TenseurHH](#) \*ggijHH\_tdt, [TenseurBB](#) \*ggijBB\_tdt, [TenseurBB](#) \*ggijBB\_0, [TenseurHH](#) \*ggijHH\_0)

## Attributs publics

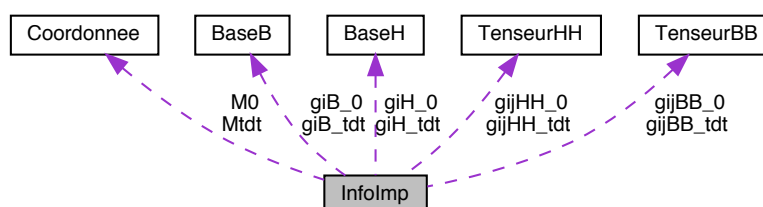
- [Coordonnee](#) \* **M0**
- [Coordonnee](#) \* **Mtdt**
- [BaseB](#) \* **giB\_0**
- [BaseB](#) \* **giB\_tdt**
- [BaseH](#) \* **giH\_0**
- [BaseH](#) \* **giH\_tdt**
- [TenseurHH](#) \* **gijHH\_tdt**
- [TenseurBB](#) \* **gijBB\_tdt**
- [TenseurBB](#) \* **gijBB\_0**
- [TenseurHH](#) \* **gijHH\_0**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔  
abstraite\_struc\_donnees.h

## 6.401 Référence de la classe Infolmp

Graphe de collaboration de Infolmp:



## Fonctions membres publiques

- **Infolmp** ([Coordonnee](#) \*gM0, [Coordonnee](#) \*gMtdt, [BaseB](#) \*ggiB\_0, [BaseB](#) \*ggiB\_tdt, [BaseH](#) \*ggiH\_0, [BaseH](#) \*ggiH\_tdt, [TenseurHH](#) \*ggijHH\_tdt, [TenseurBB](#) \*ggijBB\_tdt, [TenseurBB](#) \*ggijBB\_0, [TenseurHH](#) \*ggijHH\_0)
- **Infolmp** (const [Infolmp](#) &ex)
- **Infolmp** & **operator=** (const [Infolmp](#) &ex)
- void **Mise\_a\_jour\_grandeur** ([Coordonnee](#) \*gM0, [Coordonnee](#) \*gMtdt, [BaseB](#) \*ggiB\_0, [BaseB](#) \*ggiB\_tdt, [BaseH](#) \*ggiH\_0, [BaseH](#) \*ggiH\_tdt, [TenseurHH](#) \*ggijHH\_tdt, [TenseurBB](#) \*ggijBB\_tdt, [TenseurBB](#) \*ggijBB\_0, [TenseurHH](#) \*ggijHH\_0)

## Attributs publics

- [Coordonnee](#) \* **M0**

- [Coordonnee](#) \* [Mtdt](#)
- [BaseB](#) \* [giB\\_0](#)
- [BaseB](#) \* [giB\\_tdt](#)
- [BaseH](#) \* [giH\\_0](#)
- [BaseH](#) \* [giH\\_tdt](#)
- [TenseurHH](#) \* [gijHH\\_tdt](#)
- [TenseurBB](#) \* [gijBB\\_tdt](#)
- [TenseurBB](#) \* [gijBB\\_0](#)
- [TenseurHH](#) \* [gijHH\\_0](#)

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\\_gene/Met\\_↔ abstraite\\_struc\\_donnees.h](#)

## 6.402 Référence de la classe [Spectre::Initialisation\\_description\\_spectre](#)

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext\\_visu/spectre.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext\\_visu/spectre.cc](#)

## 6.403 Référence de la classe [Initialisation\\_tab\\_Daa](#)

[Initialisation\\_tab\\_Daa](#): définition d'une classe pour l'initialisation du tableau `tab_Daa`.

```
#include <TypeQuelconque_enum_etendu.h>
```

### 6.403.1 Description détaillée

[Initialisation\\_tab\\_Daa](#): définition d'une classe pour l'initialisation du tableau `tab_Daa`.

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque\\_enum\\_etendu.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque\\_enum\\_etendu.cc](#)

## 6.404 Référence de la classe [Initialisation\\_tab\\_Dee](#)

une classe secondaire: définition d'une classe pour l'initialisation du tableau `tab_Dee` et de la map qui relie les string et les [Ddl\\_enum\\_etendu](#)

```
#include <Ddl_enum_etendu.h>
```

### 6.404.1 Description détaillée

une classe secondaire: définition d'une classe pour l'initialisation du tableau `tab_Dee` et de la map qui relie les string et les [Ddl\\_enum\\_etendu](#)

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl\\_enum\\_etendu-copie.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl\\_enum\\_etendu.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl\\_enum\\_etendu-copie.cc](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ddl\\_enum\\_etendu.cc](#)

## 6.405 Référence de la classe [ElemPoint::inNeNpti](#)

### Attributs publics

- `int` \* [incre](#)
- `int` \* [step](#)
- `int` \* [nbe](#)
- `int` \* [nbpti](#)
- `double` \* [temps\\_tdt](#)



- double \* **delta\_t**
- string \* **nom\_loi**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.h

## 6.406 Référence de la classe DdlElement::Int\_initer

### Fonctions membres publiques

- **Int\_initer** (const [Int\\_initer](#) &a)
- [Int\\_initer](#) & **operator=** (const [Int\\_initer](#) &a)
- void **operator++** (int)
- void **operator--** (int)

### Attributs publics

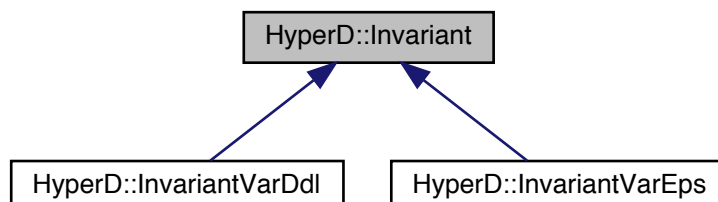
- int **vali**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/DdlElement.h

## 6.407 Référence de la classe HyperD::Invariant

Graphe d'héritage de HyperD::Invariant:



### Fonctions membres publiques

- **Invariant** (const double &l\_n, const double &V\_n, const double &bllb\_n, const double &blllb\_n)
- **Invariant** (const [Invariant](#) &a)
- [Invariant](#) & **operator=** (const [Invariant](#) &a)

### Attributs publics

- double **leps**
- double **V**
- double **bllb**
- double **blllb**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.408 Référence de la classe `Hyper3D::Invariant0QepsCosphi`

### Fonctions membres publiques

- `Invariant0QepsCosphi` (const double &Qe, const double &cos3p)

### Attributs publics

- double `Qeps`
- double `cos3phi`

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h

## 6.409 Référence de la classe `Hyper3D::Invariant2Qeps`

### Fonctions membres publiques

- `Invariant2Qeps` (const double &Q, const double &dQdbllb, const double &dQdbllb2)

### Attributs publics

- double `Qeps`
- double `dQepsdbllb`
- double `dQepsdbllb2`

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h

## 6.410 Référence de la classe `Hyper3D::Invariant2QepsCosphi`

### Attributs publics

- double `Qeps`
- double `cos3phi`
- double `dQepsdbllb`
- double `dcos3phidV`
- double `dcos3phidleps`
- double `dcos3phidbllb`
- double `dQepsdbllb2`
- double `dcos3phidV2`
- double `dcos3phidleps2`
- double `dcos3phidbllb2`
- double `dcos3phidVdbllb`
- double `dcos3phidlepsdbllb`

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h

## 6.411 Référence de la classe `Hyper_W_gene_3D::Invariantpost3D`

### Fonctions membres publiques

- `Invariantpost3D` (const double &potent)
- `Invariantpost3D` (const `Invariantpost3D` &a)
- `Invariantpost3D` & `operator=` (const `Invariantpost3D` &a)

### Attributs publics

- double `potentiel`

## Amis

- `istream & operator>>` (`istream &ent`, [Invariantpost3D &a](#))
- `ostream & operator<<` (`ostream &sort`, `const Invariantpost3D &a`)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper_W_gene_3D.h`

## 6.412 Référence de la classe HyperD::Invariantpost3D

### Fonctions membres publiques

- `Invariantpost3D` (`const double &V1`, `const double &Qe`, `const double &cos3p`, `const double &potent`)
- `Invariantpost3D` (`const Invariantpost3D &a`)
- `Invariantpost3D & operator=` (`const Invariantpost3D &a`)

### Attributs publics

- `double V`
- `double Qeps`
- `double cos3phi`
- `double potentiel`

## Amis

- `istream & operator>>` (`istream &ent`, [Invariantpost3D &a](#))
- `ostream & operator<<` (`ostream &sort`, `const Invariantpost3D &a`)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/HyperD.h`

## 6.413 Référence de la classe Hyper3D::InvariantQeps

### Attributs publics

- `double Qeps`
- `double dQepsdbllb`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper3D.h`

## 6.414 Référence de la classe Hyper3D::InvariantQepsCosphi

### Attributs publics

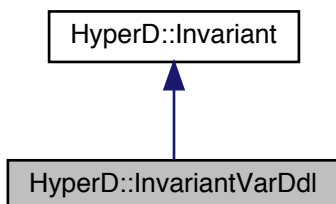
- `double Qeps`
- `double cos3phi`
- `double dQepsdbllb`
- `double dcos3phidV`
- `double dcos3phidleps`
- `double dcos3phidbllb`

La documentation de cette classe a été générée à partir du fichier suivant :

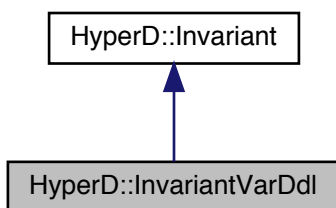
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/Hyper3D.h`

## 6.415 Référence de la classe HyperD::InvariantVarDdl

Graphe d'héritage de HyperD::InvariantVarDdl:



Graphe de collaboration de HyperD::InvariantVarDdl:



### Fonctions membres publiques

- **InvariantVarDdl** (int nddl=0)
- **InvariantVarDdl** (const [InvariantVarDdl](#) &a)
- **InvariantVarDdl** & **operator=** (const [InvariantVarDdl](#) &a)

### Attributs publics

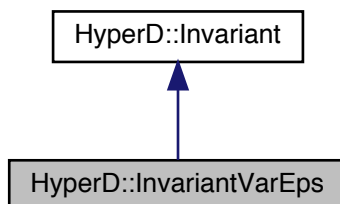
- [Tableau](#)< double > **dleps**
- [Tableau](#)< double > **dV**
- [Tableau](#)< double > **dbllb**
- [Tableau](#)< double > **dblllb**

La documentation de cette classe a été générée à partir du fichier suivant :

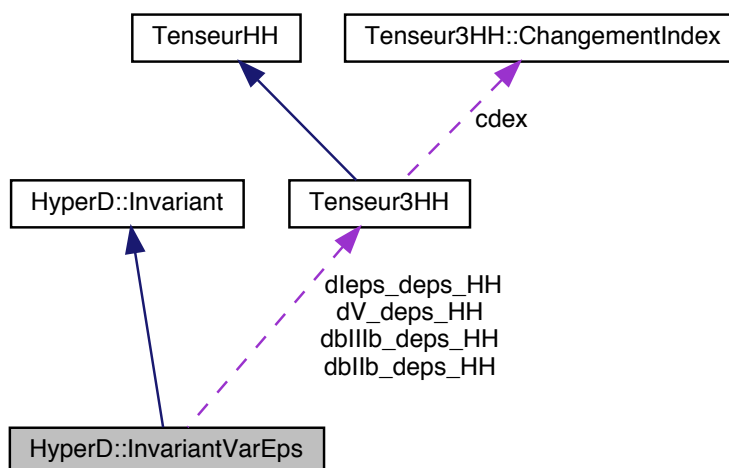
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.416 Référence de la classe HyperD::InvariantVarEps

Grphe d'héritage de HyperD::InvariantVarEps:



Grphe de collaboration de HyperD::InvariantVarEps:



### Fonctions membres publiques

- `InvariantVarEps` (const [InvariantVarEps](#) &a)
- `InvariantVarEps & operator=` (const [InvariantVarEps](#) &a)

### Attributs publics

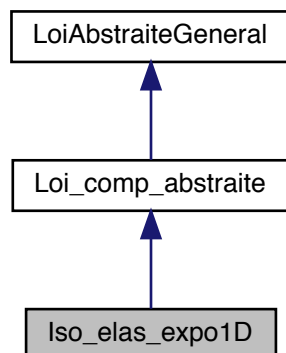
- `Tenseur3HH` `dleps_deps_HH`
- `Tenseur3HH` `dV_deps_HH`
- `Tenseur3HH` `dblllb_deps_HH`
- `Tenseur3HH` `dblllb_deps_HH`

La documentation de cette classe a été générée à partir du fichier suivant :

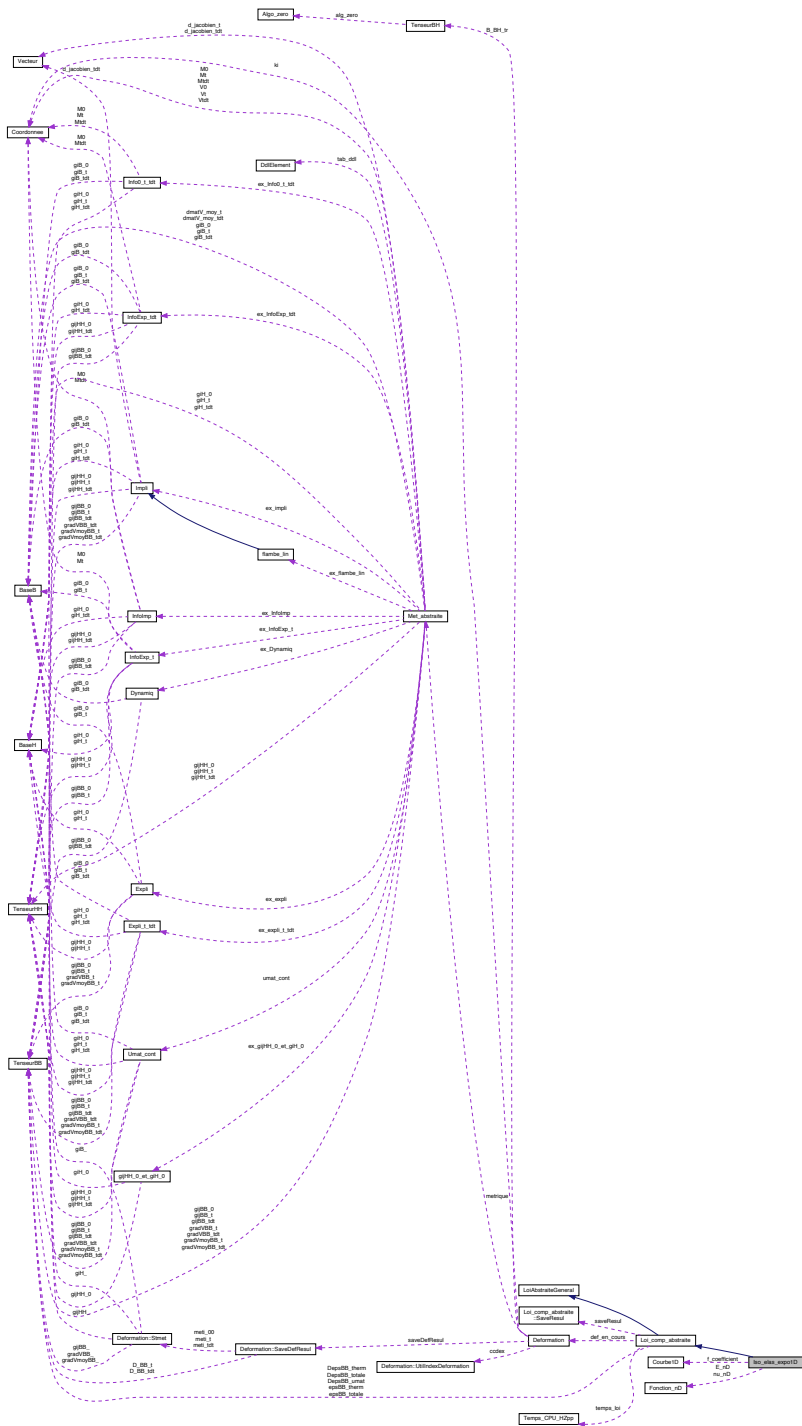
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper_elastique/HyperD.h`

## 6.417 Référence de la classe Iso\_elas\_expo1D

Graphe d'héritage de Iso\_elas\_expo1D:



Graphe de collaboration de Iso\_elas\_expo1D:



### Classes

- class [SaveResultIso\\_elas\\_expo1D](#)

### Fonctions membres publiques

- [Iso\\_elas\\_expo1D](#) (const [Iso\\_elas\\_expo1D](#) &loi)
- [SaveResul \\* New\\_et\\_Initialise](#) ()

- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComplet](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul](#) \*saveResul)
- double [Module\\_compressibilite\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul](#) \*saveResul)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)
- virtual void [Grandeur\\_particuliere](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &, [Loi\\_comp\\_abstraite::SaveResul](#) \*, [list](#)< int > &decal) const
- virtual void [ListeGrandeurs\\_particulieres](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &) const
- virtual void [Insertion\\_conteneur\\_dans\\_save\\_result](#) ([SaveResul](#) \*saveResul)
- virtual void [Activation\\_stockage\\_grandeurs\\_quelconques](#) ([list](#)< [EnumTypeQuelconque](#) > &listEnuQuelc)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_, [TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#) &giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#) &giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Impli](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- double [E](#)
- double [nu](#)
- [Courbe1D](#) \* [f\\_coefficient](#)
- [Fonction\\_nD](#) \* [E\\_nD](#)
- [Fonction\\_nD](#) \* [nu\\_nD](#)

## Amis

- class [SaveResullso\\_elas\\_expo1D](#)

## Membres hérités additionnels

### 6.417.1 Documentation des fonctions membres



**6.417.1.1 Activation\_stockage\_grandeurs\_quelconques()**

```
void Iso_elas_expo1D::Activation_stockage_grandeurs_quelconques (
    list< EnumTypeQuelconque > & listEnuQuelc ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.417.1.2 Affiche()**

```
void Iso_elas_expo1D::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.417.1.3 Calcul\_DsigmaHH\_tdt()**

```
void Iso_elas_expo1D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.417.1.4 Calcul\_SigmaHH()**

```
void Iso_elas_expo1D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
```

```

TenseurHH & gijHH_,
Tableau< TenseurBB * > & d_gijBB_,
double & jacobien_0,
double & jacobien,
TenseurHH & sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.417.1.5 CalculGrandeurTravail()

```

virtual void Iso_elas_expolD::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_imple,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.417.1.6 Ecriture\_base\_info\_loi()

```

void Iso_elas_expolD::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.417.1.7 Grandeur\_particuliere()

```

void Iso_elas_expolD::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.417.1.8 HsurH0()

```

virtual double Iso_elas_expolD::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.417.1.9 Info\_commande\_LoisDeComp()

```

void Iso_elas_expolD::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

**6.417.1.10 Insertion\_conteneur\_dans\_save\_result()**

```
void Iso_elas_expo1D::Insertion_conteneur_dans_save_result (
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.417.1.11 Lecture\_base\_info\_loi()**

```
void Iso_elas_expo1D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.417.1.12 LectureDonneesParticulieres()**

```
void Iso_elas_expo1D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.417.1.13 ListeGrandeurs\_particulieres()**

```
void Iso_elas_expo1D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.417.1.14 Module\_compressibilite\_equivalent()**

```
double Iso_elas_expo1D::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.417.1.15 Module\_young\_equivalent()**

```
double Iso_elas_expo1D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.417.1.16 New\_et\_Initialise()**

```
SaveResul * Iso_elas_expo1D::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.417.1.17 Nouvelle\_loi\_identique()

`Loi_comp_abstraite * Iso_elas_expo1D::Nouvelle_loi_identique ( ) const [inline], [virtual]`  
Implémente [Loi\\_comp\\_abstraite](#).

### 6.417.1.18 TestComplet()

`int Iso_elas_expo1D::TestComplet ( ) [virtual]`

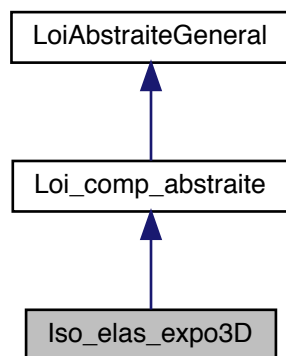
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

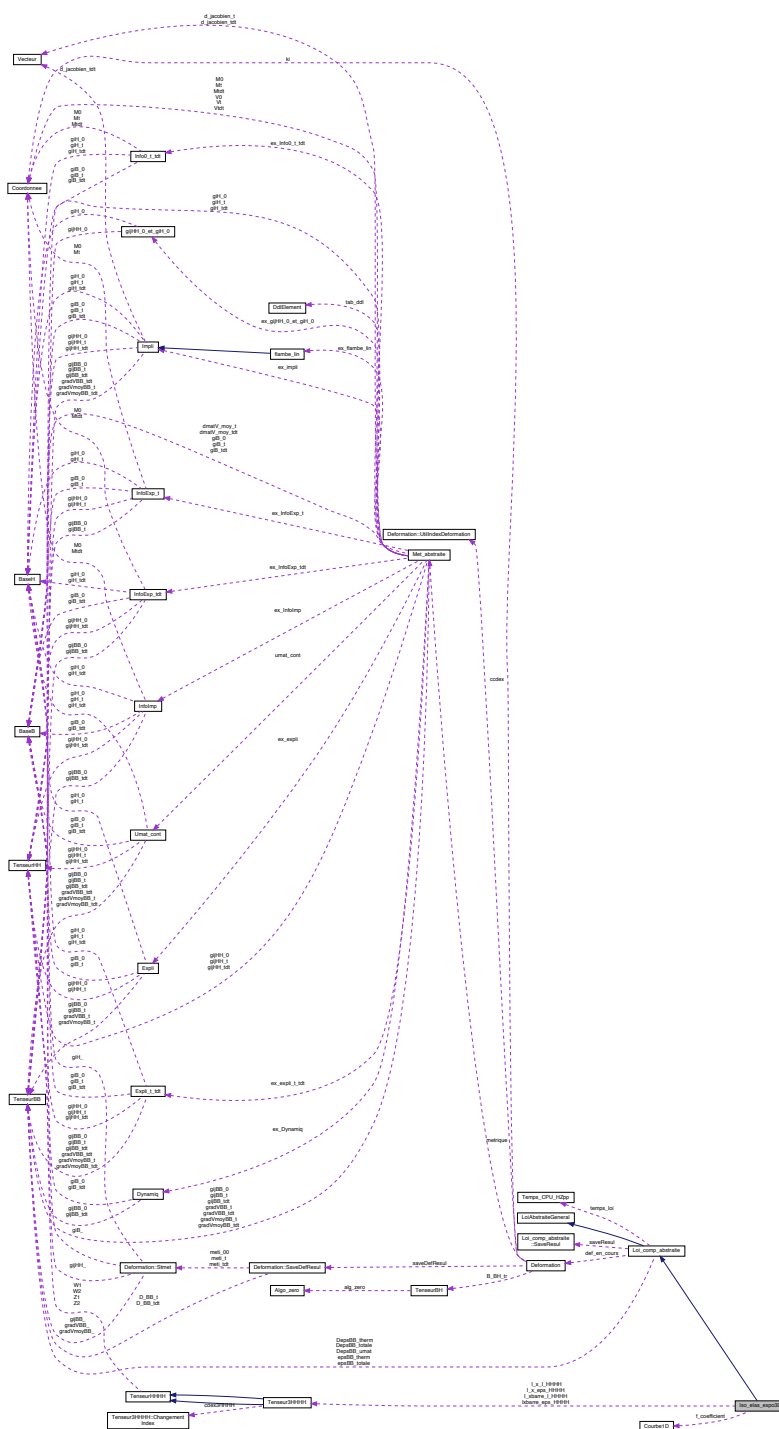
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_nonlinear/Iso\_elas\_expo1↔  
D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_nonlinear/Iso\_elas\_expo1↔  
D.cc

## 6.418 Référence de la classe Iso\_elas\_expo3D

Graphe d'héritage de Iso\_elas\_expo3D:



Graphe de collaboration de Iso\_elas\_expo3D:



## Fonctions membres publiques

- `Iso_elas_expo3D` (const `Iso_elas_expo3D` &loi)
- void `LectureDonneesParticulieres` (`UtilLecture` \*, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- void `Lecture_base_info_loi` (`ifstream` &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)

- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure temps, const [Deformation](#) &def, [SaveResul](#) \*saveResul)
- double [Module\\_compressibilite\\_equivalent](#) (Enum\_dure temps, const [Deformation](#) &def)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gij←  
BB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_,  
[TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double  
&jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_←  
compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#)  
&giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#)  
&giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_eps←  
BB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* >  
&d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#)  
&d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const  
[EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_←  
abstraite::Impli](#) &ex)
- void [Calcul\\_dsigma\\_deps](#) (bool en\_base\_orthonormee, [TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB,  
[TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, double &jacobien\_0, double &jacobien, [TenseurHH](#)  
&sigHH, [TenseurHHHH](#) &d\_sigma\_deps, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double  
&module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Umat\\_cont](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const  
[ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const  
[Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const  
[TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- double [E](#)
- double [nu](#)
- [Courbe1D](#) \* [f\\_coefficient](#)
- [Tenseur3HHHH](#) [I\\_x](#) [I\\_HHHH](#)
- [Tenseur3HHHH](#) [I\\_xbarre](#) [I\\_HHHH](#)
- [Tenseur3HHHH](#) [I\\_x\\_eps](#) [HHHH](#)
- [Tenseur3HHHH](#) [I\\_xbarre\\_eps](#) [HHHH](#)

## Membres hérités additionnels

### 6.418.1 Documentation des fonctions membres

#### 6.418.1.1 Affiche()

void [Iso\\_elas\\_expo3D::Affiche](#) ( ) const [virtual]  
Implémente [LoiAbstraiteGeneral](#).

#### 6.418.1.2 Calcul\_dsigma\_deps()

```
void Iso\_elas\_expo3D::Calcul\_dsigma\_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
```

```

TenseurBB & delta_epsBB,
double & jacobien_0,
double & jacobien,
TenseurHH & sigHH,
TenseurHHHH & d_sigma_deps,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Umat_cont & ex ) [protected], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.418.1.3 Calcul\_DsigmaHH\_tdt()

```

void Iso_elas_expo3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.418.1.4 Calcul\_SigmaHH()

```

void Iso_elas_expo3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,

```

```

TenseurHH & gijHH_,
Tableau< TenseurBB * > & d_gijBB_,
double & jacobien_0,
double & jacobien,
TenseurHH & sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.418.1.5 CalculGrandeurTravail()

```

virtual void Iso_elas_expo3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_imple,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.418.1.6 Ecriture\_base\_info\_loi()

```

void Iso_elas_expo3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.418.1.7 HsurH0()

```

virtual double Iso_elas_expo3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.418.1.8 Info\_commande\_LoisDeComp()

```

void Iso_elas_expo3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.418.1.9 Lecture\_base\_info\_loi()

```

void Iso_elas_expo3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```



Implémente [LoiAbstraiteGeneral](#).

#### 6.418.1.10 LectureDonneesParticulieres()

```
void Iso_elas_expo3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.418.1.11 Module\_young\_equivalent()

```
double Iso_elas_expo3D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.418.1.12 Nouvelle\_loi\_identique()

```
Loi\_comp\_abstraite * Iso_elas_expo3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.418.1.13 TestComplet()

```
int Iso_elas_expo3D::TestComplet ( ) [virtual]
```

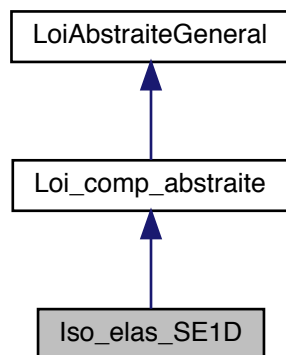
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

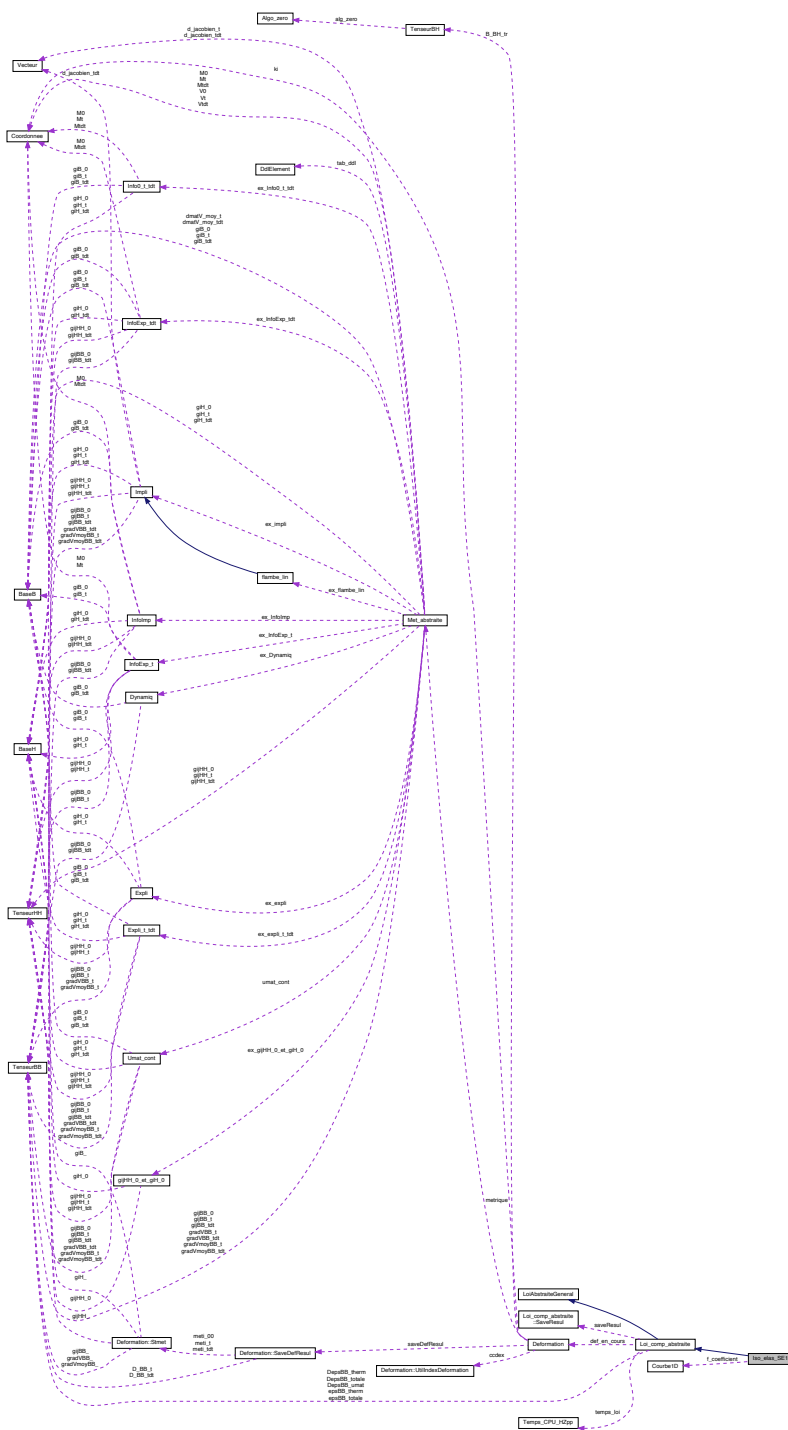
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_nonlinear/Iso\_elas\_expo3↵  
D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_nonlinear/iso\_elas\_expo3↵  
D.cc

## 6.419 Référence de la classe Iso\_elas\_SE1D

Graphe d'héritage de Iso\_elas\_SE1D:



Graphe de collaboration de Iso\_elas\_SE1D:



### Fonctions membres publiques

- Iso\_elas\_SE1D (const Iso\_elas\_SE1D &loi)
- void LectureDonneesParticulieres (UtilLecture \*, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)
- void Affiche () const
- int TestComple ()
- void Lecture\_base\_info\_loi (ifstream &ent, const int cas, LesReferences &lesRef, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)

- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure temps, const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gij←  
BB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_,  
[TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double  
&jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_←  
compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#)  
&giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#)  
&giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_eps←  
BB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* >  
&d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#)  
&d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const  
[EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_←  
abstraite::Impli](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const  
[ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const  
[Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const  
[TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- [Courbe1D](#) \* [f\\_coefficient](#)
- bool [symetrique](#)
- double [nu](#)

## Membres hérités additionnels

### 6.419.1 Documentation des fonctions membres

#### 6.419.1.1 Affiche()

void [Iso\\_elas\\_SE1D::Affiche](#) ( ) const [virtual]  
Implémente [LoiAbstraiteGeneral](#).

#### 6.419.1.2 Calcul\_DsigmaHH\_tdt()

```
void Iso\_elas\_SE1D::Calcul\_DsigmaHH\_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
```

```

TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.419.1.3 Calcul\_SigmaHH()

```

void Iso_elas_SE1D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.419.1.4 CalculGrandeurTravail()

```

virtual void Iso_elas_SE1D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],

```

[virtual]

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.419.1.5 Ecriture\_base\_info\_loi()

```
void Iso_elas_SE1D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.419.1.6 HsurH0()

```
virtual double Iso_elas_SE1D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.419.1.7 Info\_commande\_LoisDeComp()

```
void Iso_elas_SE1D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.419.1.8 Lecture\_base\_info\_loi()

```
void Iso_elas_SE1D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.419.1.9 LectureDonneesParticulieres()

```
void Iso_elas_SE1D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.419.1.10 Module\_young\_equivalent()

```
double Iso_elas_SE1D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.419.1.11 Nouvelle\_loi\_identique()

```
Loi\_comp\_abstraite * Iso_elas_SE1D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.419.1.12 TestComplet()

```
int Iso_elas_SE1D::TestComplet ( ) [virtual]
```

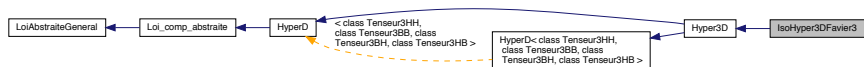
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

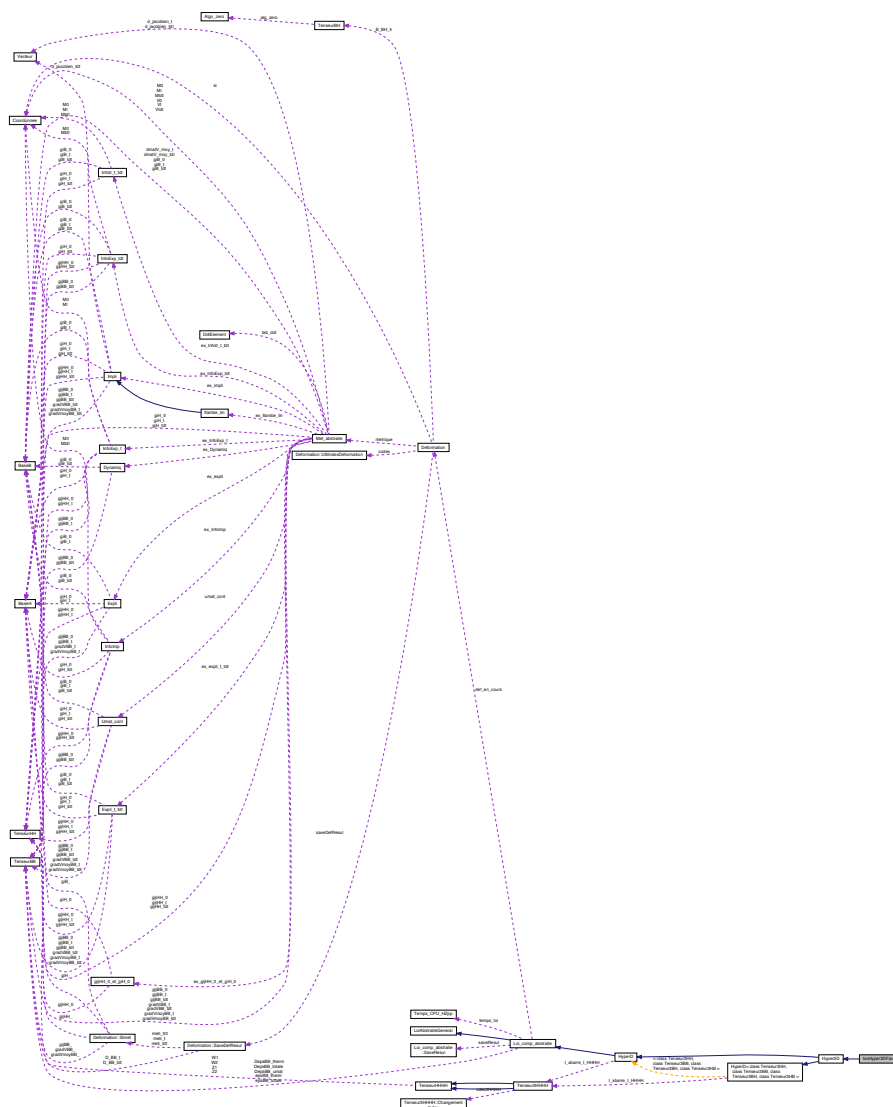
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_nonlinear/Iso\_elas\_SE1↔D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_nonlinear/Iso\_elas\_SE1↔D.cc

## 6.420 Référence de la classe IsoHyper3DFavier3

Graphe d'héritage de IsoHyper3DFavier3:



Graphe de collaboration de IsoHyper3DFavier3:



## Fonctions membres publiques

- **IsoHyper3DFavier3** (const [IsoHyper3DFavier3](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComplet](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#), const [Deformation](#) &, [SaveResul](#) \*)
- double [Module\\_compressibilite\\_equivalent](#) ([Enum\\_dure](#), const [Deformation](#) &def)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)
- double [PoGrenoble](#) (const double &Qeps, const [Invariant](#) &invariant)
  - les fonctions potentielles sont défini dans les classes dérivées*
- double [PoGrenoble](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenoble\\_V](#) [PoGrenoble\\_et\\_V](#) (const double &Qeps, const [Invariant](#) &invariant)
- [PoGrenoble\\_V](#) [PoGrenoble\\_et\\_V](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)



- [PoGrenoble\\_VV PoGrenoble\\_et\\_VV](#) (const double &Qeps, const [Invariant](#) &invariant)
- [PoGrenoble\\_VV PoGrenoble\\_et\\_VV](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseSansVar PoGrenoble](#) (const [InvariantQeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseSansVar PoGrenoblePhase](#) (const [InvariantQepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseAvecVar PoGrenoble\\_et\\_var](#) (const [Invariant2Qeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseAvecVar PoGrenoblePhase\\_et\\_var](#) (const [Invariant2QepsCosphi](#) &inv, const [Invariant](#) &invariant)

### Fonctions membres protégées

- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const [PoGrenobleSansPhaseAvecVar](#) &potret)  
*===== fonctions pour la vérification et la mise au point =====*
- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const double &cos3phi, const [PoGrenobleAvecPhaseAvecVar](#) &potret)

### Attributs protégés

- double [K](#)
- double [Qor](#)
- double [mur](#)
- double [mu\\_inf](#)
- double [nQor](#)
- double [gammaQor](#)
- double [n\\_mu\\_inf](#)
- double [gamma\\_mu\\_inf](#)

### Attributs protégés statiques

- static double [limite\\_co2](#) =700.
- static int [indic\\_Verif\\_PoGrenoble\\_et\\_var](#) = 0

### Membres hérités additionnels

#### 6.420.1 Documentation des fonctions membres

##### 6.420.1.1 Affiche()

void IsoHyper3DFavier3::Affiche ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

##### 6.420.1.2 Ecriture\_base\_info\_loi()

void IsoHyper3DFavier3::Ecriture\_base\_info\_loi (   
     ofstream & sort,   
     const int cas ) [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

##### 6.420.1.3 HsurH0()

virtual double IsoHyper3DFavier3::HsurH0 (   
     SaveResul \* saveResul ) const [inline], [virtual]  
 Implémente [Loi\\_comp\\_abstraite](#).

**6.420.1.4 Info\_commande\_LoisDeComp()**

```
void IsoHyper3DFavier3::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.420.1.5 Lecture\_base\_info\_loi()**

```
void IsoHyper3DFavier3::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.420.1.6 LectureDonneesParticulieres()**

```
void IsoHyper3DFavier3::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.420.1.7 Module\_young\_equivalent()**

```
double IsoHyper3DFavier3::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.420.1.8 Nouvelle\_loi\_identique()**

```
Loi\_comp\_abstraite * IsoHyper3DFavier3::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.420.1.9 PoGrenoble() [1/3]**

```
double IsoHyper3DFavier3::PoGrenoble (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

les fonctions potentielles sont défini dans les classes dérivées

Implémente [Hyper3D](#).

**6.420.1.10 PoGrenoble() [2/3]**

```
double IsoHyper3DFavier3::PoGrenoble (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.420.1.11 PoGrenoble()** [3/3]

```
Hyper3D::PoGrenobleSansPhaseSansVar IsoHyper3DFavier3::PoGrenoble (
    const InvariantQeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.420.1.12 PoGrenoble\_et\_V()** [1/2]

```
Hyper3D::PoGrenoble_V IsoHyper3DFavier3::PoGrenoble_et_V (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.420.1.13 PoGrenoble\_et\_V()** [2/2]

```
Hyper3D::PoGrenoble_V IsoHyper3DFavier3::PoGrenoble_et_V (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.420.1.14 PoGrenoble\_et\_var()**

```
Hyper3D::PoGrenobleSansPhaseAvecVar IsoHyper3DFavier3::PoGrenoble_et_var (
    const Invariant2Qeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.420.1.15 PoGrenoble\_et\_VV()** [1/2]

```
Hyper3D::PoGrenoble_VV IsoHyper3DFavier3::PoGrenoble_et_VV (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.420.1.16 PoGrenoble\_et\_VV()** [2/2]

```
Hyper3D::PoGrenoble_VV IsoHyper3DFavier3::PoGrenoble_et_VV (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.420.1.17 PoGrenoblePhase()**

```
Hyper3D::PoGrenobleAvecPhaseSansVar IsoHyper3DFavier3::PoGrenoblePhase (
    const InvariantQepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.420.1.18 PoGrenoblePhase\_et\_var()**

```
Hyper3D::PoGrenobleAvecPhaseAvecVar IsoHyper3DFavier3::PoGrenoblePhase_et_var (
    const Invariant2QepsCosphi & inv,
```

```
const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

### 6.420.1.19 TestComplet()

```
int IsoHyper3DFavier3::TestComplet ( ) [virtual]
```

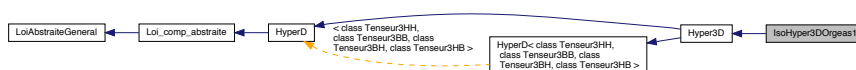
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

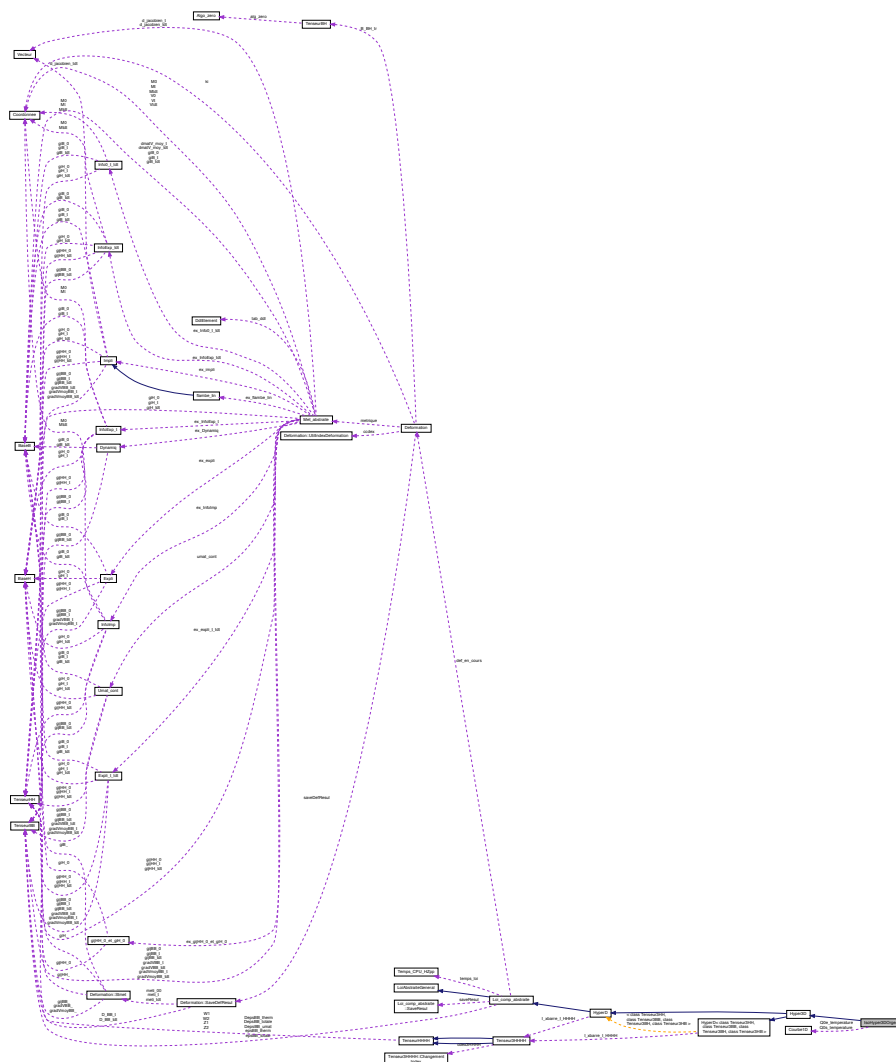
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyper3DFavier3.↵  
h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyper3DFavier3.↵  
cc

## 6.421 Référence de la classe IsoHyper3DOrgeas1

Graphes d'héritage de IsoHyper3DOrgeas1:



Graphe de collaboration de IsoHyper3DOrgeas1:



## Fonctions membres publiques

- **IsoHyper3DOrgeas1** (const [IsoHyper3DOrgeas1](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &, [SaveResul](#) \*)
- double [Module\\_compressibilite\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &def)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)
- double [PoGrenoble](#) (const double &Qeps, const [Invariant](#) &invariant)
  - les fonctions potentielles sont défini dans les classes dérivées*
  - double [PoGrenoble](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
  - [PoGrenoble\\_V](#) [PoGrenoble\\_et\\_V](#) (const double &Qeps, const [Invariant](#) &invariant)
  - [PoGrenoble\\_V](#) [PoGrenoble\\_et\\_V](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
  - [PoGrenoble\\_VV](#) [PoGrenoble\\_et\\_VV](#) (const double &Qeps, const [Invariant](#) &invariant)
  - [PoGrenoble\\_VV](#) [PoGrenoble\\_et\\_VV](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)

- [PoGrenobleSansPhaseSansVar](#) [PoGrenoble](#) (const [InvariantQeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseSansVar](#) [PoGrenoblePhase](#) (const [InvariantQepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseAvecVar](#) [PoGrenoble\\_et\\_var](#) (const [Invariant2Qeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseAvecVar](#) [PoGrenoblePhase\\_et\\_var](#) (const [Invariant2QepsCosphi](#) &inv, const [Invariant](#) &invariant)

### Fonctions membres protégées

- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const [PoGrenobleSansPhaseAvecVar](#) &potret)
- *===== fonctions pour la vérification et la mise au point =====*
- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const double &cos3phi, const [PoGrenobleAvecPhaseAvecVar](#) &potret)
- void [Verif\\_PoGrenoble\\_et\\_var\\_VV](#) (const double &Qeps, const [Invariant](#) &inv, const [PoGrenoble\\_VV](#) &potret)
- void [CompaFormel](#) ()

### Attributs protégés

- double [K](#)
- double [Q0s](#)
- double [mu01](#)
- double [mu02](#)
- double [mu03](#)
- double [alpha1](#)
- double [alpha3](#)
- double [Q0e](#)
- double [nQs](#)
- double [gammaQs](#)
- double [nQe](#)
- double [gammaQe](#)
- double [nMu1](#)
- double [gammaMu1](#)
- double [nMu2](#)
- double [gammaMu2](#)
- double [nMu3](#)
- double [gammaMu3](#)
- double [alpha1\\_2](#)
- double [alpha3\\_2](#)
- int [cas\\_Q0s\\_thermo\\_dependant](#)
- double [T0r](#)
- double [Gr](#)
- double [Qrmax](#)
- double [ar](#)
- int [cas\\_Q0e\\_thermo\\_dependant](#)
- double [T0rQe](#)
- double [Qe\\_T0rQe](#)
- double [h1](#)
- double [Qs\\_T0rQe](#)
- double [Tc](#)
- double [Ts](#)
- [Courbe1D](#) \* [Q0s\\_temperature](#)
- [Courbe1D](#) \* [Q0e\\_temperature](#)

### Attributs protégés statiques

- static int [indic\\_Verif\\_PoGrenobleorgeas1\\_et\\_var](#) = 0
- static int [indic\\_Verif\\_PoGrenobleorgeas1\\_et\\_var\\_VV](#) = 0
- static int [indic\\_Verif\\_CompaFormel](#) = 0

## Membres hérités additionnels

### 6.421.1 Documentation des fonctions membres

#### 6.421.1.1 Affiche()

```
void IsoHyper3DOrgeas1::Affiche ( ) const [virtual]  
Implémente LoiAbstraiteGeneral.
```

#### 6.421.1.2 Ecriture\_base\_info\_loi()

```
void IsoHyper3DOrgeas1::Ecriture_base_info_loi (  
    ofstream & sort,  
    const int cas ) [virtual]  
Implémente LoiAbstraiteGeneral.
```

#### 6.421.1.3 HsurH0()

```
virtual double IsoHyper3DOrgeas1::HsurH0 (  
    SaveResul * saveResul ) const [inline], [virtual]  
Implémente Loi\_comp\_abstraite.
```

#### 6.421.1.4 Info\_commande\_LoisDeComp()

```
void IsoHyper3DOrgeas1::Info_commande_LoisDeComp (  
    UtilLecture & lec ) [virtual]  
Implémente LoiAbstraiteGeneral.
```

#### 6.421.1.5 Lecture\_base\_info\_loi()

```
void IsoHyper3DOrgeas1::Lecture_base_info_loi (  
    ifstream & ent,  
    const int cas,  
    LesReferences & lesRef,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]  
Implémente LoiAbstraiteGeneral.
```

#### 6.421.1.6 LectureDonneesParticulieres()

```
void IsoHyper3DOrgeas1::LectureDonneesParticulieres (  
    UtilLecture * entreePrinc,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]  
Implémente LoiAbstraiteGeneral.
```

#### 6.421.1.7 Module\_young\_equivalent()

```
double IsoHyper3DOrgeas1::Module_young_equivalent (  
    Enum_dure temps,  
    const Deformation & ,  
    SaveResul * ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.421.1.8 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * IsoHyper3DOrgeas1::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.421.1.9 PoGrenoble() [1/3]

```
double IsoHyper3DOrgeas1::PoGrenoble (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

les fonctions potentielles sont défini dans les classes dérivées

Implémente [Hyper3D](#).

#### 6.421.1.10 PoGrenoble() [2/3]

```
double IsoHyper3DOrgeas1::PoGrenoble (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

#### 6.421.1.11 PoGrenoble() [3/3]

```
Hyper3D::PoGrenobleSansPhaseSansVar IsoHyper3DOrgeas1::PoGrenoble (
    const InvariantQeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

#### 6.421.1.12 PoGrenoble\_et\_V() [1/2]

```
Hyper3D::PoGrenoble_V IsoHyper3DOrgeas1::PoGrenoble_et_V (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

#### 6.421.1.13 PoGrenoble\_et\_V() [2/2]

```
Hyper3D::PoGrenoble_V IsoHyper3DOrgeas1::PoGrenoble_et_V (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

#### 6.421.1.14 PoGrenoble\_et\_var()

```
Hyper3D::PoGrenobleSansPhaseAvecVar IsoHyper3DOrgeas1::PoGrenoble_et_var (
    const Invariant2Qeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).



**6.421.1.15 PoGrenoble\_et\_VV()** [1/2]

```
Hyper3D::PoGrenoble_VV IsoHyper3DOrgeas1::PoGrenoble_et_VV (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.421.1.16 PoGrenoble\_et\_VV()** [2/2]

```
Hyper3D::PoGrenoble_VV IsoHyper3DOrgeas1::PoGrenoble_et_VV (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.421.1.17 PoGrenoblePhase()**

```
Hyper3D::PoGrenobleAvecPhaseSansVar IsoHyper3DOrgeas1::PoGrenoblePhase (
    const InvariantQepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.421.1.18 PoGrenoblePhase\_et\_var()**

```
Hyper3D::PoGrenobleAvecPhaseAvecVar IsoHyper3DOrgeas1::PoGrenoblePhase_et_var (
    const Invariant2QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.421.1.19 TestComplet()**

```
int IsoHyper3DOrgeas1::TestComplet ( ) [virtual]
```

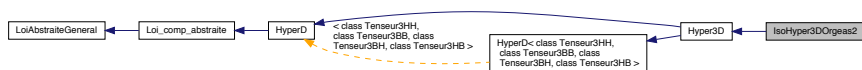
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

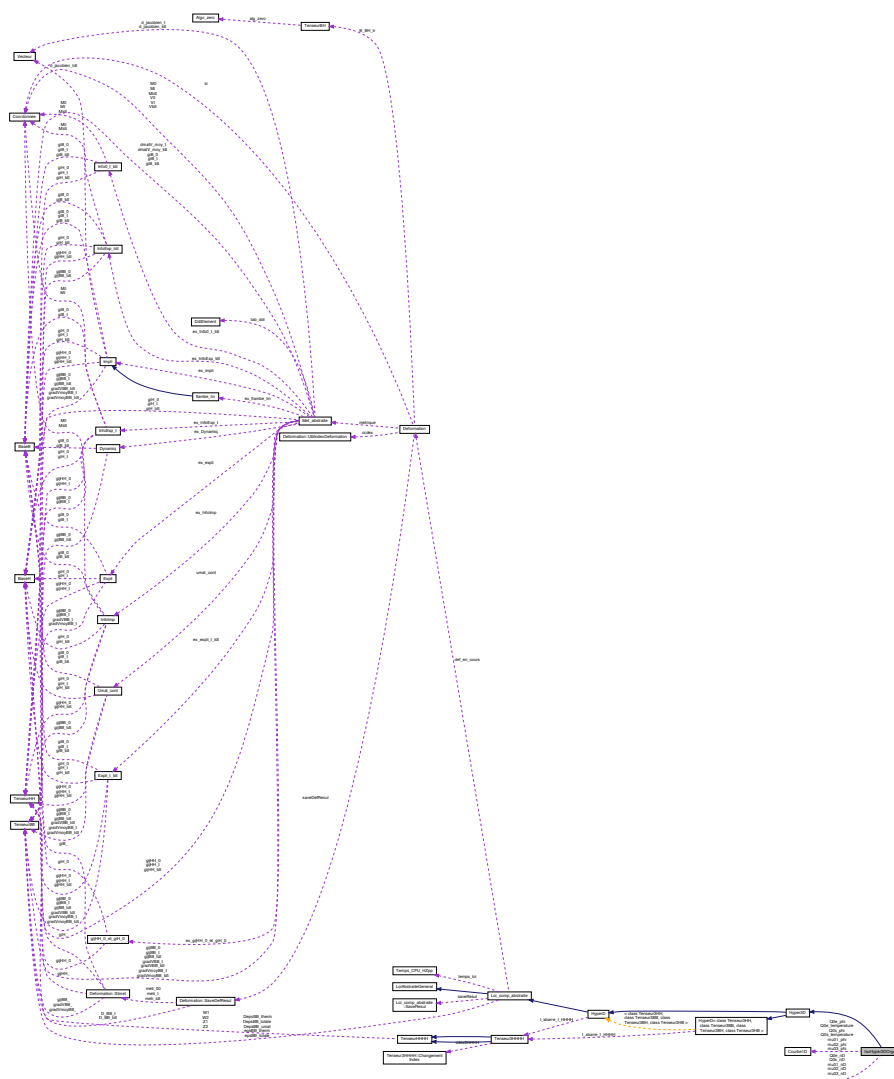
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyper3DOrgeas1.↔h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyper3DOrgeas1.↔cc

**6.422 Référence de la classe IsoHyper3DOrgeas2**

Graphes d'héritage de IsoHyper3DOrgeas2:



Graphe de collaboration de IsoHyper3DOrgeas2:



## Fonctions membres publiques

- **IsoHyper3DOrgeas2** (const [IsoHyper3DOrgeas2](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &, [SaveResul](#) \*saveResul)
- double [Module\\_compressibilite\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul](#) \*saveResul)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)
- double [PoGrenoble](#) (const double &Qeps, const [Invariant](#) &invariant)
  - les fonctions potentielles sont défini dans les classes dérivées*
- double [PoGrenoble](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenoble\\_V](#) [PoGrenoble\\_et\\_V](#) (const double &Qeps, const [Invariant](#) &invariant)
- [PoGrenoble\\_V](#) [PoGrenoble\\_et\\_V](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)

- [PoGrenoble\\_VV](#) [PoGrenoble\\_et\\_VV](#) (const double &Qeps, const [Invariant](#) &invariant)
- [PoGrenoble\\_VV](#) [PoGrenoble\\_et\\_VV](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseSansVar](#) [PoGrenoble](#) (const [InvariantQeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseSansVar](#) [PoGrenoblePhase](#) (const [InvariantQepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseAvecVar](#) [PoGrenoble\\_et\\_var](#) (const [Invariant2Qeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseAvecVar](#) [PoGrenoblePhase\\_et\\_var](#) (const [Invariant2QepsCosphi](#) &inv, const [Invariant](#) &invariant)

### Fonctions membres protégées

- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const [PoGrenobleSansPhaseAvecVar](#) &potret)  
*===== fonctions pour la vérification et la mise au point =====*
- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const double &cos3phi, const [PoGrenobleAvecPhaseAvecVar](#) &potret)
- void [Verif\\_PoGrenoble\\_et\\_var\\_VV](#) (const double &Qeps, const [Invariant](#) &inv, const [PoGrenoble\\_VV](#) &potret)
- void [CompaFormel](#) ()

### Attributs protégés

- double [K](#)
- double [Q0s](#)
- double [mu01](#)
- double [mu02](#)
- double [mu03](#)
- double [alpha1](#)
- double [alpha3](#)
- double [Q0e](#)
- [Courbe1D](#) \* [mu01\\_phi](#)
- [Courbe1D](#) \* [mu02\\_phi](#)
- [Courbe1D](#) \* [mu03\\_phi](#)
- [Courbe1D](#) \* [Q0s\\_phi](#)
- [Courbe1D](#) \* [Q0e\\_phi](#)
- [Fonction\\_nD](#) \* [mu01\\_nD](#)
- [Fonction\\_nD](#) \* [mu02\\_nD](#)
- [Fonction\\_nD](#) \* [mu03\\_nD](#)
- [Fonction\\_nD](#) \* [Q0s\\_nD](#)
- [Fonction\\_nD](#) \* [Q0e\\_nD](#)
- double [alpha1\\_2](#)
- double [alpha3\\_2](#)
- int [cas\\_Q0s\\_thermo\\_dependant](#)
- double [T0r](#)
- double [Gr](#)
- double [Qrmax](#)
- double [ar](#)
- int [cas\\_Q0e\\_thermo\\_dependant](#)
- double [T0rQe](#)
- double [Qe\\_T0rQe](#)
- double [h1](#)
- double [Qs\\_T0rQe](#)
- double [Tc](#)
- double [Ts](#)
- [Courbe1D](#) \* [Q0s\\_temperature](#)
- [Courbe1D](#) \* [Q0e\\_temperature](#)

### Attributs protégés statiques

- static int [indic\\_Verif\\_PoGrenobleorgeas2\\_et\\_var](#) = 0
- static int [indic\\_Verif\\_PoGrenobleorgeas2\\_et\\_var\\_VV](#) = 0
- static int [indic\\_Verif\\_CompaFormel](#) = 0

## Membres hérités additionnels

### 6.422.1 Documentation des fonctions membres

#### 6.422.1.1 Affiche()

```
void IsoHyper3DOrgeas2::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.422.1.2 Ecriture\_base\_info\_loi()

```
void IsoHyper3DOrgeas2::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.422.1.3 HsurH0()

```
virtual double IsoHyper3DOrgeas2::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.422.1.4 Info\_commande\_LoisDeComp()

```
void IsoHyper3DOrgeas2::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.422.1.5 Lecture\_base\_info\_loi()

```
void IsoHyper3DOrgeas2::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.422.1.6 LectureDonneesParticulieres()

```
void IsoHyper3DOrgeas2::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.422.1.7 Module\_compressibilite\_equivalent()

```
double IsoHyper3DOrgeas2::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.422.1.8 Module\_young\_equivalent()

```
double IsoHyper3DOrgeas2::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.422.1.9 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * IsoHyper3DOrgeas2::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.422.1.10 PoGrenoble() [1/3]

```
double IsoHyper3DOrgeas2::PoGrenoble (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

les fonctions potentielles sont défini dans les classes dérivées

Implémente [Hyper3D](#).

#### 6.422.1.11 PoGrenoble() [2/3]

```
double IsoHyper3DOrgeas2::PoGrenoble (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

#### 6.422.1.12 PoGrenoble() [3/3]

```
Hyper3D::PoGrenobleSansPhaseSansVar IsoHyper3DOrgeas2::PoGrenoble (
    const InvariantQeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

#### 6.422.1.13 PoGrenoble\_et\_V() [1/2]

```
Hyper3D::PoGrenoble_V IsoHyper3DOrgeas2::PoGrenoble_et_V (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

#### 6.422.1.14 PoGrenoble\_et\_V() [2/2]

```
Hyper3D::PoGrenoble_V IsoHyper3DOrgeas2::PoGrenoble_et_V (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.422.1.15 PoGrenoble\_et\_var()**

```
Hyper3D::PoGrenobleSansPhaseAvecVar IsoHyper3DOrgeas2::PoGrenoble_et_var (
    const Invariant2Qeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.422.1.16 PoGrenoble\_et\_VV() [1/2]**

```
Hyper3D::PoGrenoble_VV IsoHyper3DOrgeas2::PoGrenoble_et_VV (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.422.1.17 PoGrenoble\_et\_VV() [2/2]**

```
Hyper3D::PoGrenoble_VV IsoHyper3DOrgeas2::PoGrenoble_et_VV (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.422.1.18 PoGrenoblePhase()**

```
Hyper3D::PoGrenobleAvecPhaseSansVar IsoHyper3DOrgeas2::PoGrenoblePhase (
    const InvariantQepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.422.1.19 PoGrenoblePhase\_et\_var()**

```
Hyper3D::PoGrenobleAvecPhaseAvecVar IsoHyper3DOrgeas2::PoGrenoblePhase_et_var (
    const Invariant2QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.422.1.20 TestCompleet()**

```
int IsoHyper3DOrgeas2::TestCompleet ( ) [virtual]
```

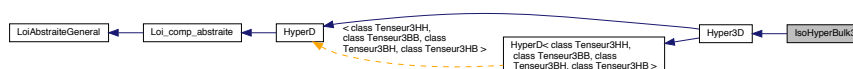
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

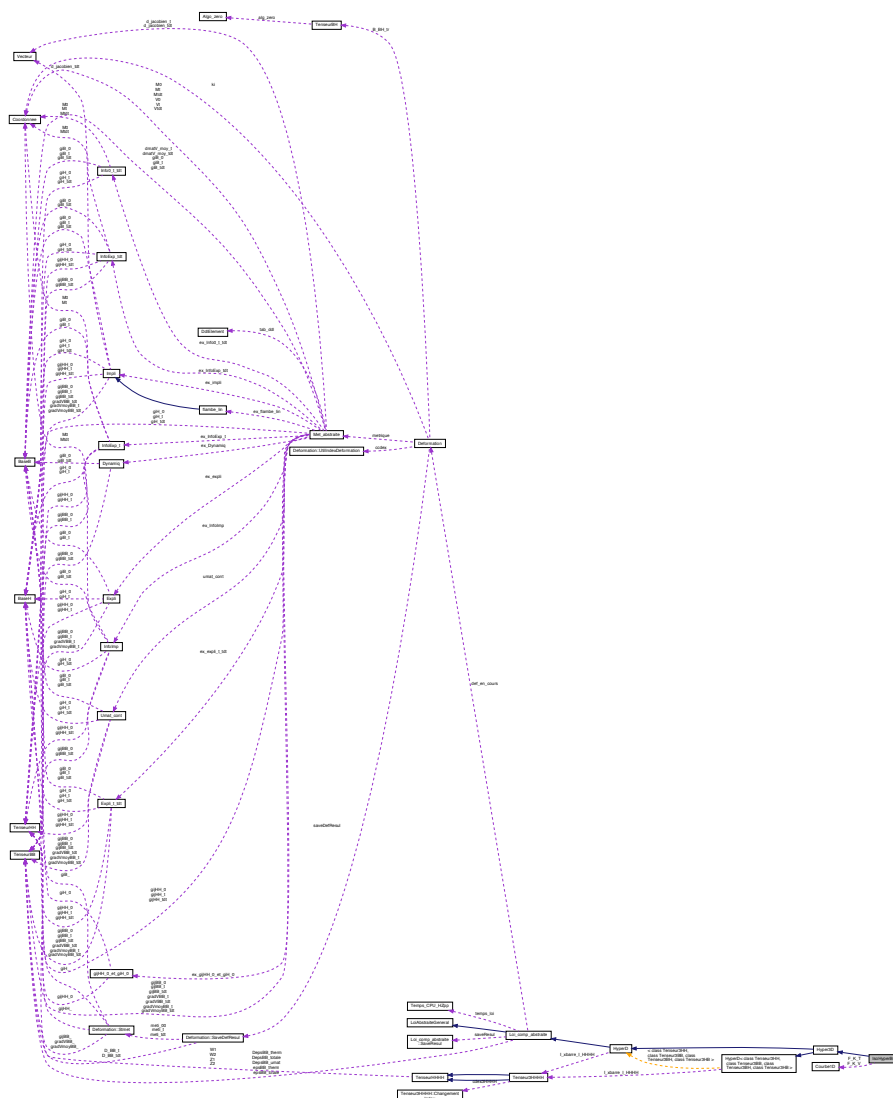
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyper3DOrgeas2.↵h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyper3DOrgeas2.↵cc

**6.423 Référence de la classe IsoHyperBulk3**

Grappe d'héritage de IsoHyperBulk3:



Graphe de collaboration de IsoHyperBulk3:



## Fonctions membres publiques

- **IsoHyperBulk3** (const **IsoHyperBulk3** &loi)
- void **LectureDonneesParticulieres** (**UtilLecture** \*, **LesCourbes1D** &lesCourbes1D, **LesFonctions\_nD** &lesFonctionsnD)
- void **Affiche** () const
- int **TestComple** ()
- void **Lecture\_base\_info\_loi** (ifstream &ent, const int cas, **LesReferences** &lesRef, **LesCourbes1D** &lesCourbes1D, **LesFonctions\_nD** &lesFonctionsnD)
- void **Ecriture\_base\_info\_loi** (ofstream &sort, const int cas)
- double **Module\_young\_equivalent** (**Enum\_dure** temps, const **Deformation** &, **SaveResul** \*saveResul)
- virtual double **HsurH0** (**SaveResul** \*saveResul) const
- double **Module\_compressibilite\_equivalent** (**Enum\_dure** temps, const **Deformation** &def, **SaveResul** \*saveResul)
- **Loi\_comp\_abstraite** \* **Nouvelle\_loi\_identique** () const
- void **Info\_commande\_LoisDeComp** (**UtilLecture** &lec)
- double **PoGrenoble** (const double &Qeps, const **Invariant** &invariant)  
*les fonctions potentielles sont défini dans les classes dérivées*
- double **PoGrenoble** (const **Invariant0QepsCosphi** &inv, const **Invariant** &invariant)
- **PoGrenoble\_V** **PoGrenoble\_et\_V** (const double &Qeps, const **Invariant** &invariant)

- [PoGrenoble\\_V](#) [PoGrenoble\\_et\\_V](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenoble\\_VV](#) [PoGrenoble\\_et\\_VV](#) (const double &Qeps, const [Invariant](#) &invariant)
- [PoGrenoble\\_VV](#) [PoGrenoble\\_et\\_VV](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseSansVar](#) [PoGrenoble](#) (const [InvariantQeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseSansVar](#) [PoGrenoblePhase](#) (const [InvariantQepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseAvecVar](#) [PoGrenoble\\_et\\_var](#) (const [Invariant2Qeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseAvecVar](#) [PoGrenoblePhase\\_et\\_var](#) (const [Invariant2QepsCosphi](#) &inv, const [Invariant](#) &invariant)

## Fonctions membres protégées

- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const [PoGrenobleSansPhaseAvecVar](#) &potret)  
     ===== fonctions pour la vérification et la mise au point =====
- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const double &cos3phi, const [PoGrenobleAvecPhaseAvecVar](#) &potret)

## Attributs protégés

- double [K](#)
- [Courbe1D](#) \* [F\\_K\\_V](#)
- [Courbe1D](#) \* [F\\_K\\_T](#)

## Attributs protégés statiques

- static int [indic\\_Verif\\_PoGrenoble\\_et\\_var](#) = 0

## Membres hérités additionnels

### 6.423.1 Documentation des fonctions membres

#### 6.423.1.1 Affiche()

```
void IsoHyperBulk3::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.423.1.2 Ecriture\_base\_info\_loi()

```
void IsoHyperBulk3::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.423.1.3 HsurH0()

```
virtual double IsoHyperBulk3::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.423.1.4 Info\_commande\_LoisDeComp()

```
void IsoHyperBulk3::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).



**6.423.1.5 Lecture\_base\_info\_loi()**

```
void IsoHyperBulk3::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.423.1.6 LectureDonneesParticulieres()**

```
void IsoHyperBulk3::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.423.1.7 Module\_compressibilite\_equivalent()**

```
double IsoHyperBulk3::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.423.1.8 Module\_young\_equivalent()**

```
double IsoHyperBulk3::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.423.1.9 Nouvelle\_loi\_identique()**

```
Loi\_comp\_abstraite * IsoHyperBulk3::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.423.1.10 PoGrenoble() [1/3]**

```
double IsoHyperBulk3::PoGrenoble (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

les fonctions potentielles sont défini dans les classes dérivées

Implémente [Hyper3D](#).

**6.423.1.11 PoGrenoble() [2/3]**

```
double IsoHyperBulk3::PoGrenoble (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.423.1.12 PoGrenoble()** [3/3]

```
Hyper3D::PoGrenobleSansPhaseSansVar IsoHyperBulk3::PoGrenoble (
    const InvariantQeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.423.1.13 PoGrenoble\_et\_V()** [1/2]

```
Hyper3D::PoGrenoble_V IsoHyperBulk3::PoGrenoble_et_V (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.423.1.14 PoGrenoble\_et\_V()** [2/2]

```
Hyper3D::PoGrenoble_V IsoHyperBulk3::PoGrenoble_et_V (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.423.1.15 PoGrenoble\_et\_var()**

```
Hyper3D::PoGrenobleSansPhaseAvecVar IsoHyperBulk3::PoGrenoble_et_var (
    const Invariant2Qeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.423.1.16 PoGrenoble\_et\_VV()** [1/2]

```
Hyper3D::PoGrenoble_VV IsoHyperBulk3::PoGrenoble_et_VV (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.423.1.17 PoGrenoble\_et\_VV()** [2/2]

```
Hyper3D::PoGrenoble_VV IsoHyperBulk3::PoGrenoble_et_VV (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.423.1.18 PoGrenoblePhase()**

```
Hyper3D::PoGrenobleAvecPhaseSansVar IsoHyperBulk3::PoGrenoblePhase (
    const InvariantQepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.423.1.19 PoGrenoblePhase\_et\_var()**

```
Hyper3D::PoGrenobleAvecPhaseAvecVar IsoHyperBulk3::PoGrenoblePhase_et_var (
    const Invariant2QepsCosphi & inv,
```

```
const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

### 6.423.1.20 TestComplet()

```
int IsoHyperBulk3::TestComplet ( ) [virtual]
```

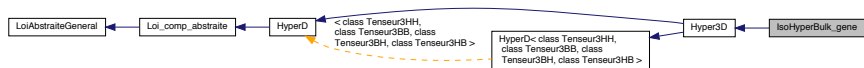
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

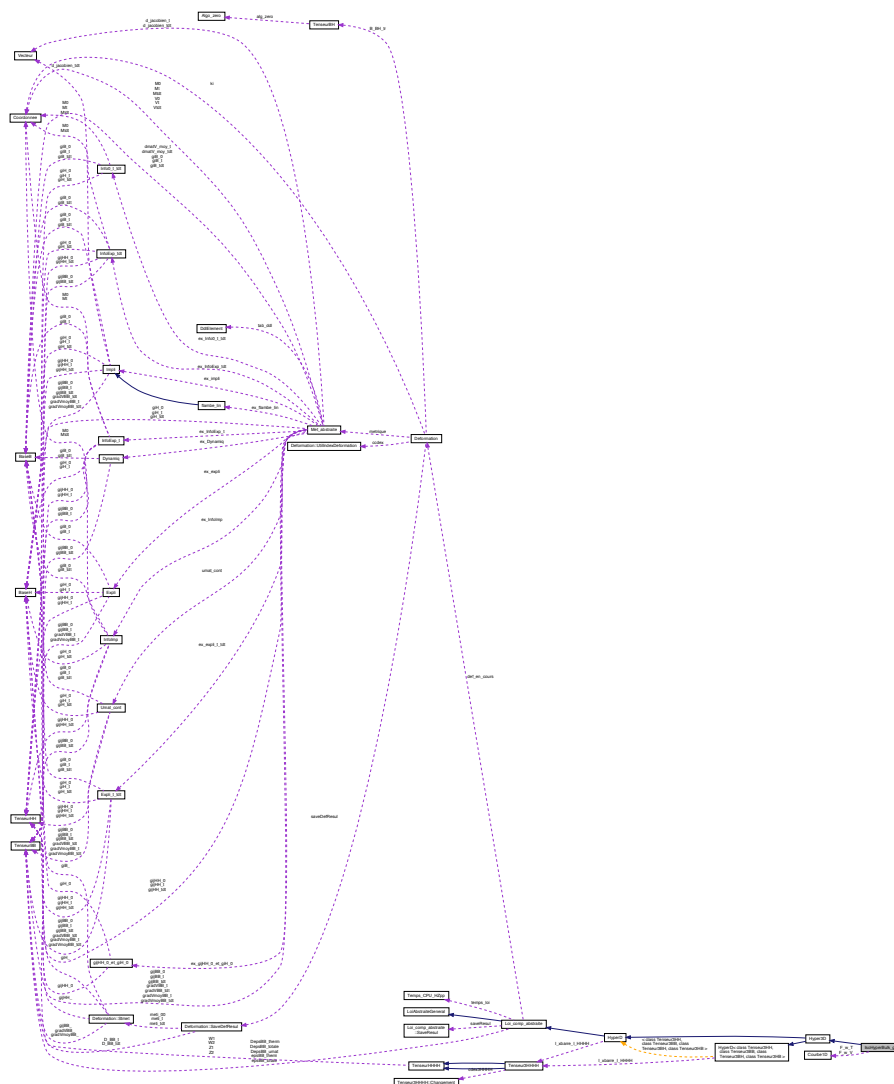
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyperBulk3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyperBulk3.cc

## 6.424 Référence de la classe IsoHyperBulk\_gene

Graphe d'héritage de IsoHyperBulk\_gene:



Graphe de collaboration de IsoHyperBulk\_gene:



## Fonctions membres publiques

- **IsoHyperBulk\_gene** (const **IsoHyperBulk\_gene** &loi)
- void **LectureDonneesParticulieres** (**UtilLecture** \*, **LesCourbes1D** &lesCourbes1D, **LesFonctions\_nD** &lesFonctionsnD)
- void **Affiche** () const
- int **TestComple** ()
- void **Lecture\_base\_info\_loi** (ifstream &ent, const int cas, **LesReferences** &lesRef, **LesCourbes1D** &lesCourbes1D, **LesFonctions\_nD** &lesFonctionsnD)
- void **Ecriture\_base\_info\_loi** (ofstream &sort, const int cas)
- double **Module\_young\_equivalent** (**Enum\_dure** temps, const **Deformation** &, **SaveResul** \*saveResul)
- double **Module\_compressibilite\_equivalent** (**Enum\_dure** temps, const **Deformation** &def, **SaveResul** \*saveResul)
- virtual double **HsurH0** (**SaveResul** \*saveResul) const
- **Loi\_comp\_abstraite** \* **Nouvelle\_loi\_identique** () const
- void **Info\_commande\_LoisDeComp** (**UtilLecture** &lec)
- double **PoGrenoble** (const double &Qeps, const **Invariant** &invariant)
- les fonctions potentielles sont défini dans les classes dérivées*
- double **PoGrenoble** (const **Invariant0QepsCosphi** &inv, const **Invariant** &invariant)
- **PoGrenoble\_V** **PoGrenoble\_et\_V** (const double &Qeps, const **Invariant** &invariant)
- **PoGrenoble\_V** **PoGrenoble\_et\_V** (const **Invariant0QepsCosphi** &inv, const **Invariant** &invariant)

- [PoGrenoble\\_VV PoGrenoble\\_et\\_VV](#) (const double &Qeps, const [Invariant](#) &invariant)
- [PoGrenoble\\_VV PoGrenoble\\_et\\_VV](#) (const [Invariant0QepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseSansVar PoGrenoble](#) (const [InvariantQeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseSansVar PoGrenoblePhase](#) (const [InvariantQepsCosphi](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleSansPhaseAvecVar PoGrenoble\\_et\\_var](#) (const [Invariant2Qeps](#) &inv, const [Invariant](#) &invariant)
- [PoGrenobleAvecPhaseAvecVar PoGrenoblePhase\\_et\\_var](#) (const [Invariant2QepsCosphi](#) &inv, const [Invariant](#) &invariant)

### Fonctions membres protégées

- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const [PoGrenobleSansPhaseAvecVar](#) &potret)  
*===== fonctions pour la vérification et la mise au point =====*
- void [Verif\\_PoGrenoble\\_et\\_var](#) (const double &Qeps, const [Invariant](#) &inv, const double &cos3phi, const [PoGrenobleAvecPhaseAvecVar](#) &potret)

### Attributs protégés

- [Courbe1D](#) \* [F\\_w\\_V](#)
- [Courbe1D](#) \* [F\\_w\\_T](#)

### Attributs protégés statiques

- static int [indic\\_Verif\\_PoGrenoble\\_et\\_var](#) = 0

### Membres hérités additionnels

#### 6.424.1 Documentation des fonctions membres

##### 6.424.1.1 Affiche()

```
void IsoHyperBulk_gene::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

##### 6.424.1.2 Ecriture\_base\_info\_loi()

```
void IsoHyperBulk_gene::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

##### 6.424.1.3 HsurH0()

```
virtual double IsoHyperBulk_gene::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

##### 6.424.1.4 Info\_commande\_LoisDeComp()

```
void IsoHyperBulk_gene::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.424.1.5 Lecture\_base\_info\_loi()**

```
void IsoHyperBulk_gene::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.424.1.6 LectureDonneesParticulieres()**

```
void IsoHyperBulk_gene::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.424.1.7 Module\_compressibilite\_equivalent()**

```
double IsoHyperBulk_gene::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.424.1.8 Module\_young\_equivalent()**

```
double IsoHyperBulk_gene::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.424.1.9 Nouvelle\_loi\_identique()**

```
Loi\_comp\_abstraite * IsoHyperBulk_gene::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.424.1.10 PoGrenoble() [1/3]**

```
double IsoHyperBulk_gene::PoGrenoble (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

les fonctions potentielles sont défini dans les classes dérivées

Implémente [Hyper3D](#).

**6.424.1.11 PoGrenoble() [2/3]**

```
double IsoHyperBulk_gene::PoGrenoble (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.424.1.12 PoGrenoble()** [3/3]

```
Hyper3D::PoGrenobleSansPhaseSansVar IsoHyperBulk_gene::PoGrenoble (
    const InvariantQeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.424.1.13 PoGrenoble\_et\_V()** [1/2]

```
Hyper3D::PoGrenoble_V IsoHyperBulk_gene::PoGrenoble_et_V (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.424.1.14 PoGrenoble\_et\_V()** [2/2]

```
Hyper3D::PoGrenoble_V IsoHyperBulk_gene::PoGrenoble_et_V (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.424.1.15 PoGrenoble\_et\_var()**

```
Hyper3D::PoGrenobleSansPhaseAvecVar IsoHyperBulk_gene::PoGrenoble_et_var (
    const Invariant2Qeps & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.424.1.16 PoGrenoble\_et\_VV()** [1/2]

```
Hyper3D::PoGrenoble_VV IsoHyperBulk_gene::PoGrenoble_et_VV (
    const double & Qeps,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.424.1.17 PoGrenoble\_et\_VV()** [2/2]

```
Hyper3D::PoGrenoble_VV IsoHyperBulk_gene::PoGrenoble_et_VV (
    const Invariant0QepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.424.1.18 PoGrenoblePhase()**

```
Hyper3D::PoGrenobleAvecPhaseSansVar IsoHyperBulk_gene::PoGrenoblePhase (
    const InvariantQepsCosphi & inv,
    const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

**6.424.1.19 PoGrenoblePhase\_et\_var()**

```
Hyper3D::PoGrenobleAvecPhaseAvecVar IsoHyperBulk_gene::PoGrenoblePhase_et_var (
    const Invariant2QepsCosphi & inv,
```

```
const Invariant & invariant ) [virtual]
```

Implémente [Hyper3D](#).

#### 6.424.1.20 TestComplet()

```
int IsoHyperBulk_gene::TestComplet ( ) [virtual]
```

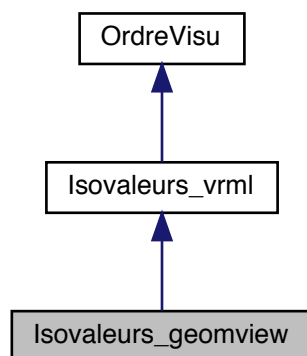
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyperBulk\_↔gene.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/IsoHyperBulk\_↔gene.cc

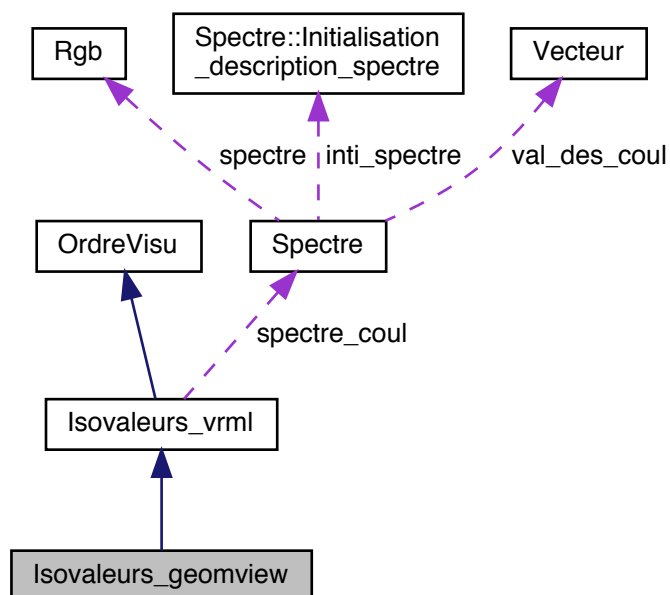
### 6.425 Référence de la classe Isovaleurs\_geomview

Graphe d'héritage de Isovaleurs\_geomview:





Graph de collaboration de Isovaleurs\_geomview:



### Fonctions membres publiques

- `Isovaleurs_geomview` (const `Isovaleurs_geomview` &algo)

### Fonctions membres protégées

- void `Sortie_dessin_legende` (`UtilLecture` &entreePrinc)

### Membres hérités additionnels

#### 6.425.1 Documentation des fonctions membres

##### 6.425.1.1 Sortie\_dessin\_legende()

```
void Isovaleurs_geomview::Sortie_dessin_legende (
    UtilLecture & entreePrinc ) [protected], [virtual]
```

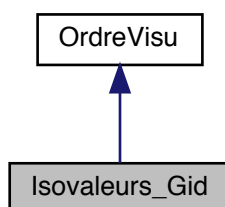
Réimplémentée à partir de `Isovaleurs_vrml`.

La documentation de cette classe a été générée à partir du fichier suivant :

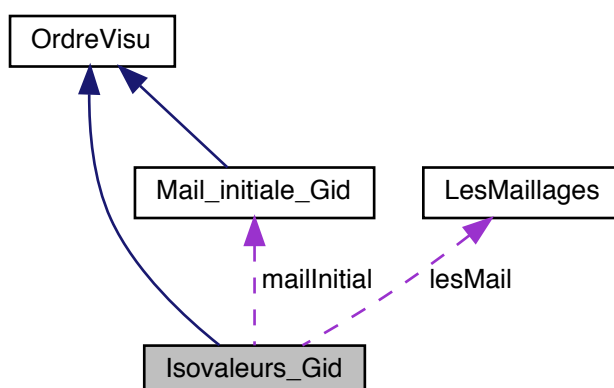
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Isovaleurs_geomview.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Isovaleurs_geomview.cc`

## 6.426 Référence de la classe Isovaleurs\_Gid

Graphe d'héritage de Isovaleurs\_Gid:



Graphe de collaboration de Isovaleurs\_Gid:



### Classes

- class [P\\_gauss](#)

### Fonctions membres publiques

- **Isovaleurs\_Gid** (const [Isovaleurs\\_Gid](#) &algo)
- void **Initialisation** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, EnumTypeIncre type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_jeu](#)< [TypeQuelconque](#) > &listeVec↔Glob, bool fil\_calcul)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu](#)::EnumTypeIncre type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_jeu](#)< [TypeQuelconque](#) > &listeVec↔Glob)
- void **ChoixOrdre** ()

- void `Init_liste_isovaleur` (`LesMaillages *lesMail`, `LesContacts *lesContacts`, `bool fil_calcul`)
- void `Jonction_MaillageInitiale` (`const Mail_initiale_Gid *mailIni`)
- void `Lecture_parametres_OrdreVisu` (`UtilLecture &entreePrinc`)
- void `Ecriture_parametres_OrdreVisu` (`UtilLecture &entreePrinc`)

## Fonctions membres protégées

- `Isovaleurs_Gid` (`const string &comment_som`, `const string &explication`, `const string &ordre`)
- void `ParametresGeneraux` (`const string &choix`)
- void `ChoixIsovaleur_noeud` (`const string &choix`)
- void `ChoixIsovaleur_ddl_etendu_noeud` (`const string &choix`)
- void `ChoixIsovaleur_evoluee_noeud` (`const string &choix`)
- void `ChoixIsovaleur_ddl_ptinteg` (`const string &choix`)
- void `ChoixIsovaleur_tensorielle_ptinteg` (`const string &choix`)
- void `ChoixIsovaleur_quelc_ptinteg` (`const string &choix`)
- void `TransfertGrandeursPtIntegAuNoeud` ()
- void `InitPremierIncrExecutionTransfertAuxNoeuds` ()
- void `ExecutionTransfert` ()
- void `VerificationTransfertPossible` ()
- void `EnteteSortielsovaleurs_G_Quelconque` (`ostream &sort`, `TypeQuelconque &typ`, `bool drap_noeud`)
- void `EnteteSortielsovaleurs_G_Quelconque_TYPE_SIMPLE` (`ostream &sort`, `const TypeQuelconque::Grandeur &grandeur`, `const string nom_typeQuelconque`, `int dima`)
- void `EnteteSortielsovaleursNoeud_G_Quelconque_TYPE_SIMPLE` (`ostream &sort`, `const TypeQuelconque::Grandeur &grandeur`, `const string nom_typeQuelconque`, `int dima`)
- void `SortieGrandeursQuelconque` (`ostream &sort`, `const TypeQuelconque &typ`)
- void `SortieGrandeursQuelconque_TYPE_SIMPLE` (`ostream &sort`, `const TypeQuelconque::Grandeur &grandeur`, `int dima`)
- void `GlobalisationListSurTousLesMaillagesPourLesNoeuds` ()
- void `ExeOrdrePremierIncrement` (`const Tableau< int > &tab_mail`, `LesMaillages *lesMail`, `ostream &sort`)
- void `EcritureAuxPtInteg` (`const Tableau< int > &tab_mail`, `LesMaillages *lesMail`, `ostream &sort`, `int incre`)
- void `EcritureAuxNoeuds` (`const Tableau< int > &tab_mail`, `LesMaillages *lesMail`, `ostream &sort`, `int incre`)

## Attributs protégés

- `bool absolue`
- `Tableau< List_io< Ddl_enum_etendu > > tabnoeud_type_ddl`
- `Tableau< List_io< Ddl_enum_etendu > > tabnoeud_type_ddl_retenu`
- `Tableau< List_io< bool > > choix_var_ddl`
- `Tableau< List_io< Ddl_enum_etendu > > tabnoeud_type_ddlEtendu`
- `Tableau< List_io< Ddl_enum_etendu > > tabnoeud_type_ddlEtendu_retenu`
- `Tableau< List_io< Ddl_enum_etendu > > a_accumuler_tabnoeud_type_ddlEtendu`
- `Tableau< List_io< Ddl_enum_etendu > > a_accumuler_tabnoeud_type_ddlEtendu_retenu`
- `Tableau< List_io< TypeQuelconque > > tabnoeud_evoluee`
- `Tableau< List_io< TypeQuelconque > > tabnoeud_evoluee_retenu`
- `Tableau< List_io< TypeQuelconque > > a_accumuler_tabnoeud_evoluee`
- `Tableau< List_io< TypeQuelconque > > a_accumuler_tabnoeud_evoluee_retenu`
- `List_io< TypeQuelconque > list_vect_globalPourNoeud`
- `List_io< TypeQuelconque > list_vect_globalPourNoeud_retenu`
- `Tableau< List_io< Ddl_enum_etendu > > tabellement_type_ddl`
- `Tableau< List_io< Ddl_enum_etendu > > tabellement_type_ddl_retenu`
- `Tableau< List_io< TypeQuelconque > > tabellement_evoluee`
- `Tableau< List_io< TypeQuelconque > > tabellement_evoluee_retenu`
- `Tableau< List_io< TypeQuelconque > > tabellement_typeParti`
- `Tableau< List_io< TypeQuelconque > > tabellement_typeParti_retenu`
- `bool transfert_au_noeud`
- `int cas_transfert`
- `const Mail_initiale_Gid * mailInitial`
- `LesMaillages * lesMail`
- `bool ddlSurY_1D`
- `list< P_gauss > li_P_gauss_total`
- `Tableau< Tableau< Tableau< P_gauss > > > tp_tp_tp_gauss_base`
- `map< string, List_io< Ddl_enum_etendu >, std::less< string > > map_gauss_base`
- `map< string, List_io< TypeQuelconque >, std::less< string > > map_gauss_baseEVol`
- `map< string, Tableau< List_io< TypeQuelconque > >, std::less< string > > map_gauss_tab_baseEVol`

- `map< string, List_io< TypeQuelconque >, std::less< string > > map_gauss_basePQ`
- `map< string, Tableau< List_io< TypeQuelconque > >, std::less< string > > map_gauss_tab_basePQ`
- `List_io< Ddl_enum_etendu > glob_noe_ddl_retenu`
- `Tableau< List_io< bool > > t_g_noeud_ddl_asortir`
- `List_io< Ddl_enum_etendu > glob_noeud_ddl_etendu_retenu`
- `Tableau< List_io< bool > > t_g_noeud_ddl_etendu_asortir`
- `List_io< TypeQuelconque > glob_noeud_evol_retenu`
- `Tableau< List_io< bool > > t_g_noeud_evoluee_asortir`
- `List_io< Ddl_enum_etendu > glob_elem_ddl_retenu`
- `Tableau< List_io< bool > > t_g_elem_ddl_asortir`
- `List_io< TypeQuelconque > glob_elem_evol_retenu`
- `Tableau< List_io< bool > > t_g_elem_evoluee_asortir`
- `List_io< TypeQuelconque > glob_elem_Partir_retenu`
- `Tableau< List_io< bool > > t_g_elem_typePartir_asortir`
- `Tableau< List_io< TypeQuelconque > * > tab_quelconque`
- `List_io< TypeQuelconque > li_glob_restreinte_TQ`

## Membres hérités additionnels

### 6.426.1 Documentation des fonctions membres

#### 6.426.1.1 ChoixOrdre()

```
void Isovaleurs_Gid::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.426.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Isovaleurs_Gid::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.426.1.3 ExeOrdre()

```
void Isovaleurs_Gid::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * lesContacts,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.426.1.4 Initialisation()

```
void Isovaleurs_Gid::Initialisation (
    ParaGlob * paraGlob,
    LesMaillages * lesmail,
    LesReferences * lesRef,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    EnumTypeIncre type_incre,
    int incre,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob,
    bool fil_calcul ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.426.1.5 Lecture\_parametres\_OrdreVisu()

```
void Isovaleurs_Gid::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

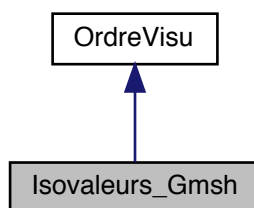
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

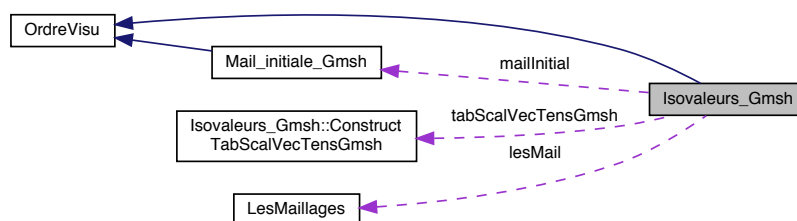
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Isovaleurs\_Gid.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Isovaleurs\_Gid.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Isovaleurs\_Gid2.cc

## 6.427 Référence de la classe Isovaleurs\_Gmsh

Grappe d'héritage de Isovaleurs\_Gmsh:



Graphe de collaboration de Isovaleurs\_Gmsh:



## Classes

- class [ConstructTabScalVecTensGmsh](#)
- class [P\\_gauss](#)

## Fonctions membres publiques

- **Isovaleurs\_Gmsh** (const [Isovaleurs\\_Gmsh](#) &algo)
- void **Initialisation** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, EnumTypeIncre type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob, bool fil\_calcul)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu](#)::EnumTypeIncre type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob)
- void **ChoixOrdre** ()
- void **Init\_liste\_isovaleur** ([LesMaillages](#) \*lesMail, [LesContacts](#) \*lesContacts, bool fil\_calcul)
- void **Jonction\_MaillagelInitiale** ([Mail\\_initiale\\_Gmsh](#) \*maillIni)
- bool **Use\_hold\_gmsh\_format** () const
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- const list< string > & **NomsGrandeurSortie** () const
- void **SortieDefCoordonnees\_old\_format** (ostream &sort, const int &incre, int nbmail, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail) const
- void **SortieUseUneCoordonnee** (int incre, ostream &sort, bool fin, int num\_N) const
- void **EnteteView** (ostream &sort, const int &incre, const string &nom, const int &numero) const
- void **EnteteNodeData** (ostream &sort, const int &incre, const double &temps, const string &nom, const int &numero) const
- const [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > & **Tabnoeud\_type\_ddl** () const

## Fonctions membres publiques statiques

- static const [ConstructTabScalVecTensGmsh](#) & **TabScalVecTensGmsh** ()

## Attributs publics statiques

- static [ConstructTabScalVecTensGmsh](#) **tabScalVecTensGmsh**

## Fonctions membres protégées

- **Isovaleurs\_Gmsh** (const string &comment\_som, const string &explication, const string &ordre)
- void **ParametresGeneraux** (const string &choix)
- void **ChoixIsovaleur\_noeud** (const string &choix)

- void **ChoixIsovaleur\_ddl\_etendu\_noeud** (const string &choix)
- void **ChoixIsovaleur\_evoluee\_noeud** (const string &choix)
- void **ChoixIsovaleur\_ddl\_ptinteg** (const string &choix)
- void **ChoixIsovaleur\_tensorielle\_ptinteg** (const string &choix)
- void **ChoixIsovaleur\_quelc\_ptinteg** (const string &choix)
- void **InitPremierIncrExecutionTransfertAuxNoeuds** ()
- void **ExecutionTransfert** ()
- void **VerificationTransfertPossible\_et\_doublon** ()
- void **Passage\_grandeursRetenuesAuxNoeuds\_elementsVersNoeuds** ()
- void **Passage\_grandeursDisponiblesAuxNoeuds\_elementsVersNoeuds** ()
- void **GlobalisationListSurTousLesMaillagesPourLesNoeuds** ()
- void **ExeOrdrePremierIncrement** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail)
- void **CalculAuxPtinteg** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail)
- void **CalculAuxElemAuxNoeuds** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail)
- void **EcritureAuxNoeuds\_ancien\_format** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail, int incre, [UtilLecture](#) &entreePrinc)
- void **EcritureAuxNoeuds** (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail, int incre, [UtilLecture](#) &entreePrinc)
- void **SortirUneViewTypeQuelconque\_old\_format** (ostream &sort, [TypeQuelconque\\_enum\\_etendu](#) enu, [EnumType2Niveau](#) enuStructure, int numero, [Tableau](#)< [List\\_io](#)< bool >::iterator > &t\_indic\_sortie, int nbmail, const [Tableau](#)< int > &tab\_mail, [EnumTypeGrandeur](#) enugrandeur, [LesMaillages](#) \*lesMail, const int &incre)
- void **SortirUneViewTypeQuelconque** (ostream &sort, [TypeQuelconque](#) &typ, [EnumType2Niveau](#) enu↔ Structure, int numero, [Tableau](#)< [List\\_io](#)< bool >::iterator > &t\_indic\_sortie, int nbmail, const [Tableau](#)< int > &tab\_mail, [EnumTypeGrandeur](#) enugrandeur, [LesMaillages](#) \*lesMail, const int &incre)
- void **SortieTenseurHH\_ancien\_format** (ostream &sort, const [TenseurHH](#) &t\_HH)
- void **SortieTenseurBB\_ancien\_format** (ostream &sort, const [TenseurBB](#) &t\_BB)
- void **SortieTenseurHH** (ostream &sort, const [TenseurHH](#) &t\_HH)
- void **SortieTenseurBB** (ostream &sort, const [TenseurBB](#) &t\_BB)
- void **SortieVecteur** (ostream &sort, const [Vecteur](#) &vec)
- void **CreatListNomsTousLesGrandeurs** ()

## Attributs protégés

- bool **use\_hold\_gmsh\_format**
- bool **absolue**
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **tabnoeud\_type\_ddl**
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **tabnoeud\_type\_ddl\_retenu**
- [Tableau](#)< [List\\_io](#)< bool > > **choix\_var\_ddl**
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **tabnoeud\_type\_ddlEtendu**
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **tabnoeud\_type\_ddlEtendu\_retenu**
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **a\_accumuler\_tabnoeud\_type\_ddlEtendu**
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **a\_accumuler\_tabnoeud\_type\_ddlEtendu\_retenu**
- [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > **tabnoeud\_evoluee**
- [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > **tabnoeud\_evoluee\_retenu**
- [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > **a\_accumuler\_tabnoeud\_evoluee**
- [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > **a\_accumuler\_tabnoeud\_evoluee\_retenu**
- [List\\_io](#)< [TypeQuelconque](#) > **list\_vect\_globalPourNoeud**
- [List\\_io](#)< [TypeQuelconque](#) > **list\_vect\_globalPourNoeud\_retenu**
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **tabelement\_type\_ddl**
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **tabelement\_type\_ddl\_retenu**
- [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > **tabelement\_evoluee**
- [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > **tabelement\_evoluee\_retenu**
- [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > **tabelement\_typeParti**
- [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > **tabelement\_typeParti\_retenu**
- int **cas\_transfert**
- [Mail\\_initiale\\_Gmsh](#) \* **maillInitial**
- [LesMaillages](#) \* **lesMail**
- [list](#)< [P\\_gauss](#) > **li\_P\_gauss\_total**
- [Tableau](#)< [Tableau](#)< [Tableau](#)< [P\\_gauss](#) > > > **tp\_tp\_tp\_gauss\_base**
- [map](#)< string, [List\\_io](#)< [Ddl\\_enum\\_etendu](#) >, [std::less](#)< string > > **map\_gauss\_base**
- [map](#)< string, [List\\_io](#)< [TypeQuelconque](#) >, [std::less](#)< string > > **map\_gauss\_baseEVol**
- [map](#)< string, [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > >, [std::less](#)< string > > **map\_gauss\_tab\_baseEVol**
- [map](#)< string, [List\\_io](#)< [TypeQuelconque](#) >, [std::less](#)< string > > **map\_gauss\_basePQ**

- `map< string, Tableau< List_io< TypeQuelconque > >, std::less< string > > map_gauss_tab_basePQ`
- `List_io< Ddl_enum_etendu > glob_noe_ddl_retenu`
- `Tableau< List_io< bool > > t_g_noeud_ddl_asortir`
- `List_io< Ddl_enum_etendu > glob_noeud_ddl_etendu_retenu`
- `Tableau< List_io< bool > > t_g_noeud_ddl_etendu_asortir`
- `List_io< TypeQuelconque > glob_noeud_evol_retenu`
- `Tableau< List_io< bool > > t_g_noeud_evoluee_asortir`
- `List_io< Ddl_enum_etendu > glob_elem_ddl_retenu`
- `Tableau< List_io< bool > > t_g_elem_ddl_asortir`
- `List_io< TypeQuelconque > glob_elem_evol_retenu`
- `Tableau< List_io< bool > > t_g_elem_evoluee_asortir`
- `List_io< TypeQuelconque > glob_elem_Partir_retenu`
- `Tableau< List_io< bool > > t_g_elem_typePartir_asortir`
- `Tableau< List_io< TypeQuelconque > * > tab_quelconque`
- `List_io< TypeQuelconque > li_glob_restreinte_TQ`
- `list< string > nomsGrandeursSortie`

## Membres hérités additionnels

### 6.427.1 Documentation des fonctions membres

#### 6.427.1.1 ChoixOrdre()

```
void Isovaleurs_Gmsh::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.427.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Isovaleurs_Gmsh::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.427.1.3 ExeOrdre()

```
void Isovaleurs_Gmsh::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * lesContacts,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).



#### 6.427.1.4 Initialisation()

```
void Isovaleurs_Gmsh::Initialisation (
    ParaGlob * paraGlob,
    LesMaillages * lesmail,
    LesReferences * lesRef,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    EnumTypeIncre type_incre,
    int incre,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob,
    bool fil_calcul ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.427.1.5 Lecture\_parametres\_OrdreVisu()

```
void Isovaleurs_Gmsh::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

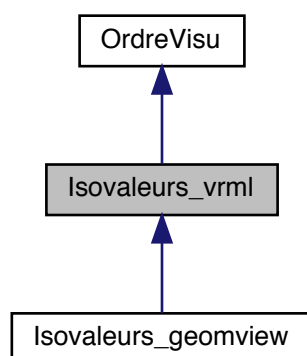
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

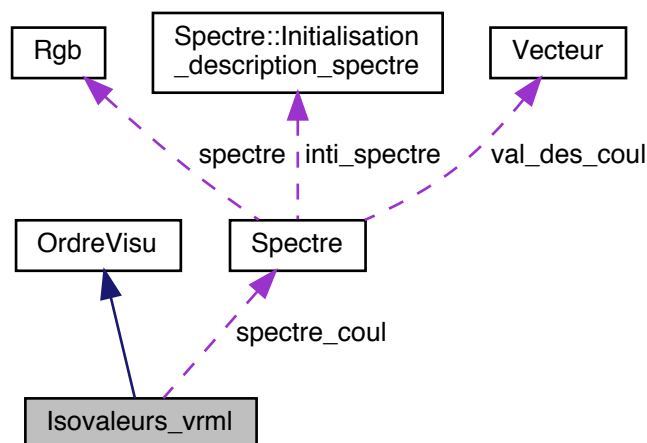
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Isovaleurs\_Gmsh.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Isovaleurs\_Gmsh.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Isovaleurs\_Gmsh2.cc

## 6.428 Référence de la classe Isovaleurs\_vrml

Grappe d'héritage de Isovaleurs\_vrml:



Graphe de collaboration de Isovaleurs\_vrml:



## Fonctions membres publiques

- **Isovaleurs\_vrml** (const [Isovaleurs\\_vrml](#) &algo)
- void **Initialisation** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, EnumTypeIncr type\_incr, int incr, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob, bool fil\_calcul)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incr, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu](#)::EnumTypeIncr type\_incr, int incr, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob)
- void **ChoixOrdre** ()
- void **Init\_liste\_ivoaleur** ([LesMaillages](#) \*lesMail, [LesContacts](#) \*lesContacts)
- const list< [Tableau](#)< [Rgb](#) > > & **List\_couleur** () const
- const list< [Deuxentiers](#) > & **List\_num\_mail\_incr** () const
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

## Fonctions membres protégées

- virtual void **Sortie\_dessin\_legende** ([UtilLecture](#) &entreePrinc)
- **Isovaleurs\_vrml** (const string &comment\_som, const string &explication, const string &ordre)
- void **ParametresGeneraux** ()
- void **ChoixIvoaleur** ()
- void **TransfertGrandeursPtIntegAuNoeud** ()

## Attributs protégés

- bool **choix\_peau**
- bool **choix\_legende**
- int **choix\_min\_max**
- double **valMini**
- double **valMaxi**
- double **pour\_legende\_min**
- double **pour\_legende\_max**

- int `nb_iso_val`
- bool `absolue`
- `Tableau< List_io< Ddl_enum_etendu > >` `tabnoeud_type_ddl`
- `Tableau< Ddl_enum_etendu >` `type_ddl_retenu`
- `Tableau< bool >` `choix_var_ddl`
- `Tableau< List_io< Ddl_enum_etendu > >` `tabelement_type_ddl`
- `list< Tableau< Rgb > >` `couleurs_noeud`
- `list< Tableau< double > >` `val_noeud`
- `list< Deuxentiers >` `num_mail_incr`
- `Spectre` `spectre_coul`

## Membres hérités additionnels

### 6.428.1 Documentation des fonctions membres

#### 6.428.1.1 ChoixOrdre()

`void Isovaleurs_vrml::ChoixOrdre ( ) [virtual]`  
 Réimplémentée à partir de [OrdreVisu](#).

#### 6.428.1.2 Ecriture\_parametres\_OrdreVisu()

`void Isovaleurs_vrml::Ecriture_parametres_OrdreVisu (`  
     `UtilLecture & entreePrinc ) [virtual]`  
 Réimplémentée à partir de [OrdreVisu](#).

#### 6.428.1.3 ExeOrdre()

```
void Isovaleurs_vrml::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.428.1.4 Initialisation()

```
void Isovaleurs_vrml::Initialisation (
    ParaGlob * paraGlob,
    LesMaillages * lesmail,
    LesReferences * lesRef,
    LesLoisDeComp * lesLoisDeComp,
```

```

DiversStockage * diversStockage,
Charge * charge,
LesCondLim * lesCondLim,
LesContacts * lesContacts,
Resultats * resultats,
EnumTypeIncre type_incre,
int incre,
const map< string, const double *, std::less< string > > & listeVarGlob,
const List_io< TypeQuelconque > & listeVecGlob,
bool fil_calcul ) [virtual]

```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.428.1.5 Lecture\_parametres\_OrdreVisu()

```

void Isovaleurs_vrml::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]

```

Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

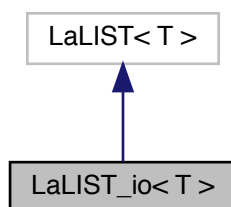
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Isovaleurs\_vrml.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Isovaleurs\_vrml.cc

## 6.429 Référence du modèle de la classe LaLIST\_io< T >

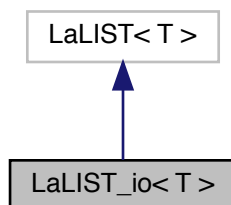
[LaLIST\\_io](#) classe template identique à [List\\_io](#): existe pour des raisons historiques et en prévision de changements futurs éventuelles (?)

```
#include <List_io.h>
```

Graphe d'héritage de LaLIST\_io< T >:



Grphe de collaboration de LaLIST\_io< T >:



## Amis

- `istream & operator>>` (`istream &entree`, [LaLIST\\_io< T >](#) &)
- `ostream & operator<<` (`ostream &sort`, `const LaLIST_io< T > &a`)
- `istream & operator>>` (`istream &entree`, `typename LaLIST< T >::iterator &`)
- `ostream & operator<<` (`ostream &sort`, `const typename LaLIST< T >::iterator`)
- `istream & operator>>` (`istream &entree`, `typename LaLIST< T >::const_iterator &`)
- `ostream & operator<<` (`ostream &sort`, `const typename LaLIST< T >::const_iterator &`)

### 6.429.1 Description détaillée

```
template<class T>
class LaLIST_io< T >
```

[LaLIST\\_io](#) classe template identique à [List\\_io](#): existe pour des raisons historiques et en prévision de changements futurs éventuelles (?)

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

24/8/2003

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/List_io.h`

## 6.430 Référence de la classe LectBloc

### Fonctions membres publiques

- `void Lecture` ([UtilLecture](#) &entreePrinc, [LesReferences](#) &lesRef, `string motcle`, `string message`, [Tableau< T >](#) &tab, [Tableau< string >](#) &TsousMot=tab\_Zero\_string, `bool saut=true`)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/LectBloc_T.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/LectBloc_T.cc`

## 6.431 Référence de la classe LectBlocmot

### Fonctions membres publiques

- [BlocDdlLim](#)< [BlocGen](#) > [Lecture](#) ([UtilLecture](#) &entreePrinc, [LesReferences](#) &lesRef, string message, [Tableau](#)< string > &tabMot, [Tableau](#)< string > &TsousMot)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/LectBlocMot.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/LectBlocMot.cc

## 6.432 Référence de la classe LesChargeExtSurElement

### Fonctions membres publiques

- [LesChargeExtSurElement](#) (int nbpti, int dimtens)
- [LesChargeExtSurElement](#) (const [LesChargeExtSurElement](#) &lespti)
- [LesChargeExtSurElement](#) & [operator=](#) (const [LesChargeExtSurElement](#) &lespti)
- void [Zero](#) ()
- [Tableau](#)< [Tableau](#)< [Pression\\_appliquee](#) > > \* [LesPressionsExternes](#) ()
- void [LesPressionsExternes\\_Change\\_taille](#) (int n)
- [Tableau](#)< [Coordonnee](#) > \* [Force\\_volume](#) ()
- void [Force\\_volume\\_Change\\_taille](#) (int n)
- [Tableau](#)< [Tableau](#)< [Coordonnee](#) > > \* [LesEffortsDirFixe](#) ()
- void [LesEffortsDirFixe\\_Change\\_taille](#) (int n)
- [Tableau](#)< [Tableau](#)< [Coordonnee](#) > > \* [LesPressDir](#) ()
- void [LesPressDir\\_Change\\_taille](#) (int n)
- [Tableau](#)< [Tableau](#)< [Force\\_hydroDyna](#) > > \* [LesHydroDyna](#) ()
- void [LesHydroDyna\\_Change\\_taille](#) (int n)
- [Tableau](#)< [Tableau](#)< [Coordonnee](#) > > \* [LesLineique](#) ()
- void [LesLineique\\_Change\\_taille](#) (int n)
- [Tableau](#)< [Tableau](#)< [Coordonnee](#) > > \* [LesLineiqueSuiveuse](#) ()
- void [LesLineiqueSuiveuse\\_Change\\_taille](#) (int n)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)

### Attributs protégés

- [Tableau](#)< [Tableau](#)< [Pression\\_appliquee](#) > > \* [lesPressionsExternes](#)
- [Tableau](#)< [Coordonnee](#) > \* [force\\_volume](#)
- [Tableau](#)< [Tableau](#)< [Coordonnee](#) > > \* [lesEffortsDirFixe](#)
- [Tableau](#)< [Tableau](#)< [Coordonnee](#) > > \* [lesPressDir](#)
- [Tableau](#)< [Tableau](#)< [Force\\_hydroDyna](#) > > \* [lesHydroDyna](#)
- [Tableau](#)< [Tableau](#)< [Coordonnee](#) > > \* [lesLineique](#)
- [Tableau](#)< [Tableau](#)< [Coordonnee](#) > > \* [lesLineiqueSuiveuse](#)

### Amis

- istream & [operator](#)>> (istream &, [LesChargeExtSurElement](#) &)
- ostream & [operator](#)<< (ostream &, const [LesChargeExtSurElement](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/LesChargeExtSurElement.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/LesChargeExtSurElement.cc

## 6.433 Référence de la classe LesCondLim

### Classes

- class [Gene\\_asso](#)
- class [ReactStoc](#)
- class [TorseurReac](#)

## Fonctions membres publiques

- void **Lecture1** ([UtilLecture](#) &entreePrinc, [LesReferences](#) &lesRef)
- void **Lecture2** ([UtilLecture](#) &entreePrinc, [LesReferences](#) &lesRef)
- void **Lecture3** ([UtilLecture](#) &entreePrinc, [LesReferences](#) &lesRef)
- void **Affiche** () const
- void **Affiche1** () const
- void **Affiche2** () const
- void **Affiche3** () const
- void **IntroductionDonnees** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- void **Initial** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD, bool choix, int cas)
- void **InitNombreCasAssemblage** (int nb\_cas)
- void **MiseAJour\_tdt** (const double &mult\_gene, [LesMaillages](#) \*lesMail, const double &deltat, const [LesReferences](#) \*lesRef, const double &temps, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD, const double &coef, bool &ch\_statut, int cas, Enum\_ddl en\_ddl=NU\_DDL)
- void **MiseAJour\_condilinaire\_tdt** (const double &mult\_gene, [LesMaillages](#) \*lesMail, const double &deltat, [LesReferences](#) \*lesRef, const double &temps, [LesCourbes1D](#) \*lesCourbes1D, [LesFonctions\\_nD](#) \*lesFonctionsnD, const double &coef, bool &ch\_statut, int cas, Enum\_ddl en\_ddl=NU\_DDL)
- void **Validation\_blocage** ([LesReferences](#) \*lesRef, const double &temps)
- bool **Changement\_statut** (const [LesMaillages](#) \*lesMail, const [LesReferences](#) \*lesRef, [LesFonctions\\_nD](#) \*lesFonctionsnD, const double &temps, const Enum\_ddl en\_ddl=NU\_DDL)
- list< [Gene\\_asso](#) > **Tableau\_indice** (const [LesMaillages](#) \*lesMail, const [Tableau](#)< [Nb\\_assemb](#) > &t\_assemb, const [LesReferences](#) \*lesRef, const double &temps, int cas)
- void **ImposeConLimtdt** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [Mat\\_abstraite](#) &matglob, [Vecteur](#) &vecglob, const [Nb\\_assemb](#) &nb\_casAssemb, int cas, [Vecteur](#) \*vec2)
- void **ImposeConLimtdt** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [Vecteur](#) &vecglob, const [Nb\\_assemb](#) &nb\_casAssemb, int cas, [Vecteur](#) \*vec2)
- void **ImpConLimtdt2Mat** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [Mat\\_abstraite](#) &matglob, [Mat\\_abstraite](#) &matgeom, const [Nb\\_assemb](#) &nb\_casAssemb, int cas)
- double **MaxEffort** (int &ili, const [Nb\\_assemb](#) &nb\_casAssemb)
- void **CalculReaction** ([LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, const [Nb\\_assemb](#) &nb\_casAssemb, int cas)
- void **ReacAvantCHrepere** ([Vecteur](#) &residu, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, const [Nb\\_assemb](#) &nb\_casAssemb, int cas)
- void **ReacApresCHrepere** ([Vecteur](#) &residu, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, const [Nb\\_assemb](#) &nb\_casAssemb, int cas)
- void **Affiche\_reaction** (ofstream &sort, const [LesMaillages](#) \*lesMail) const
- bool **CoLinCHrepere\_ext** ([Mat\\_abstraite](#) &matglob, [Vecteur](#) &vecglob, const [Tableau](#)< [Condilinaire](#) > &tabCondLine, const [Nb\\_assemb](#) &nb\_casAssemb, [Vecteur](#) \*vec2)
- void **CoLinBlocage** ([Mat\\_abstraite](#) &matglob, [Vecteur](#) &vecglob, const [Nb\\_assemb](#) &nb\_casAssemb, [Vecteur](#) \*vec2)
- void **Replinitiaux** ([Vecteur](#) &sol, const [Nb\\_assemb](#) &nb\_casAssemb)
- bool **CoLinUneOpe\_ext** ([Mat\\_abstraite](#) &matglob, [Vecteur](#) &vecglob, const [Tableau](#)< [Condilinaire](#) > &tabCondLine, const [Nb\\_assemb](#) &nb\_casAssemb, [Vecteur](#) \*vec2)
- bool **CoLinCHrepere\_int** ([Mat\\_abstraite](#) &matglob, [Vecteur](#) &vecglob, const [Nb\\_assemb](#) &nb\_casAssemb, [Vecteur](#) \*vec2)
- void **EffMarque** ()
- bool **Largeur\_Bande** (int &demi, int &total, const [Nb\\_assemb](#) &casAssemb, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, int &cumule)
- [Tableau](#)< [Tableau](#)< [Condilinaire](#) > > **ConnectionCLL** (const [LesMaillages](#) \*lesMail, const [LesReferences](#) \*lesRef) const
- list< [Condilinaire](#) > **ConnectionCLL\_stricte\_egalite** (const [LesMaillages](#) \*lesMail, const [LesReferences](#) \*lesRef) const
- const [Tableau](#)< [Tableau](#)< [Condilinaire](#) > > & **Tab\_CLinApplique** () const
- bool **ExisteCondiLimite** () const
- void **InitSauve** (const [Nb\\_assemb](#) &nb\_casAssemb)
- void **LectureDonneesExternes** ([UtilLecture](#) &, [LesReferences](#) &, const int, const string &)
- void **Info\_commande\_LesCondLim1** ([UtilLecture](#) &entreePrinc)
- void **Info\_commande\_LesCondLim2** ([UtilLecture](#) &entreePrinc)
- void **Info\_commande\_LesCondLim3** ([UtilLecture](#) &entreePrinc)

- void **Lecture\_base\_info** (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &les←  
Courbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- const [Temps\\_CPU\\_HZpp](#) & [Temps\\_cpu\\_CL](#) () const
- const [Temps\\_CPU\\_HZpp](#) & [Temps\\_cpu\\_CLL](#) () const
- [Tableau](#)< [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > > **Les\_type\_de\_ddl\_en\_reaction** () const
- [Tableau](#)< [List\\_io](#)< [String\\_et\\_entier](#) > > **TabListTorseurReaction** (const [LesMaillages](#) &lesmail) const
- const [TorseurReac](#) & [Torseur\\_de\\_reaction](#) (int i) const

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesCondLim.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesCondLim.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesCondLim2.cc

## 6.434 Référence de la classe LesContacts

### Classes

- class [ReactCont](#)

### Fonctions membres publiques

- **LesContacts** (const [LesContacts](#) &a)
- [LaLIST](#)< [EIContact](#) > & **LesElementsDeContact** ()
- void **Init\_contact** ([LesMaillages](#) &lesMail, const [LesReferences](#) &lesRef, [LesFonctions\\_nD](#) \*les←  
FonctionsnD)
- void **Mise\_a\_jour\_indice** (const [Tableau](#)< const [Tableau](#)< [List\\_io](#)< [Element](#) \* > > \* > ind)
- bool **Init\_contact\_pas\_effectue** () const
- void **Verification** ()
- bool **DefElemCont** (double dep\_max)
- bool **Nouveau** (double dep\_max)
- bool **SuppressionDefinitiveElemlnactif** ()
- bool **RelachementNoeudcolle** ()
- const [Tableau](#)< [Condilinaire](#) > & **ConditionLin** (const [Nb\\_assemb](#) &casAssemb)
- const [Tableau](#)< [Condilinaire](#) > & **ConnectionCLL** ()
- void **EffMarque** ()
- bool **Maitres\_avec\_deplacement\_imposer** () const
- bool **Largeur\_Bande** (int &demi, int &total, const [Nb\\_assemb](#) &casAssemb, int &cumule)
- bool **Actualisation** ()
- void **Liste\_noeuds\_position\_changer** (list< [Noeud](#) \* > &li\_noe)
- void **CalculReaction** ([Vecteur](#) &residu, bool &decol, const [Nb\\_assemb](#) &casAssemb, bool affiche)
- [DeuxDoubles Forces\\_contact\\_maxi](#) (bool affiche) const
- [DeuxDoubles Gap\\_contact\\_maxi](#) (bool affiche) const
- double **Pas\_de\_temps\_ideal** () const
- void **Affiche** (ofstream &sort) const
- void **Affiche** () const
- void **Info\_commande\_LesContacts** ([UtilLecture](#) &entreePrinc)
- void **Lecture\_zone\_contact** ([UtilLecture](#) &entreePrinc, const [LesReferences](#) &lesRef)
- void **TdtversT** ()
- void **TversTdt** ()
- const [Temps\\_CPU\\_HZpp](#) & [Temps\\_cpu\\_Contact](#) () const
- void **Ecri\_base\_info\_LesContacts** (ofstream &sort)
- template<class T >  
void **Lec\_base\_info\_LesContacts** (ifstream &ent, T &instance, [Noeud](#) &(T::\*RecupNoeud)(int i, int j) const,  
[Element](#) &(T::\*RecupElement\_LesMaille)(int i, int j) const)
- void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &, [Loi\\_comp\\_abstraite::SaveResul](#)  
\*, list< int > &decal) const
- void **Mise\_a\_jour\_Pour\_Grandeur\_particuliere** ([List\\_io](#)< [TypeQuelconque](#) > &li\_restreinte\_TQ)
- [List\\_io](#)< [TypeQuelconque](#) > **ListeGrandeurs\_particulieres** (bool absolue) const
- void **List\_reduite\_aux\_contact** (const [List\\_io](#)< [TypeQuelconque](#) > &liTQ, [List\\_io](#)< [TypeQuelconque](#) >  
&li\_restreinte\_TQ)
- void **Init\_Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &li1)



## 6.434.1 Documentation des fonctions membres

### 6.434.1.1 CalculReaction()

```
void LesContacts::CalculReaction (
    Vecteur & residu,
    bool & decol,
    const Nb_assemb & casAssemb,
    bool affiche )
```

!!!! il faudra prévoir un cas ou on ne sauvegarde pas les maitres

La documentation de cette classe a été générée à partir du fichier suivant :

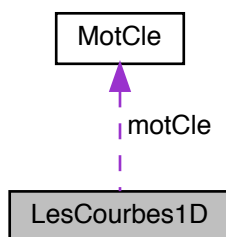
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/LesContacts.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/LesContacts.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/LesContacts\_3.cc

## 6.435 Référence de la classe LesCourbes1D

Classe de gestion des différentes courbes 1D enregistrées.

```
#include <LesCourbes1D.h>
```

Graphe de collaboration de LesCourbes1D:



### Fonctions membres publiques

- void **Lecture** (UtilLecture &entreePrinc)
- void **Info\_commande\_lesCourbes1D** (UtilLecture &entreePrinc)
- void **Affiche** () const
- bool **Existe** (const string &st1) const
- Courbe1D \* **Trouve** (const string &st1) const
- bool **Complet** ()
- Courbe1D \* **Lecture\_pour\_base\_info** (ifstream &ent, const int cas, Courbe1D \*ptcourbe)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **SchemaXML\_lesCourbes1D** (ofstream &sort, const Enum\_IO\_XML enu)

### Fonctions membres publiques statiques

- static void **Ecriture\_pour\_base\_info** (ofstream &sort, const int cas, Courbe1D \*ptcourbe)

### Attributs protégés

- map< string, Courbe1D \*, std::less< string > > **listeDeCourbe1D**

## Attributs protégés statiques

- static [MotCle](#) motCle

### 6.435.1 Description détaillée

Classe de gestion des différentes courbes 1D enregistrées.

Auteur

Gérard Rio

Version

1.0

Date

19/01/2001

La documentation de cette classe a été générée à partir du fichier suivant :

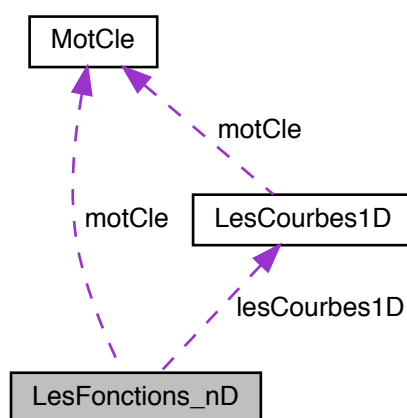
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/LesCourbes1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/LesCourbes1D.cc

## 6.436 Référence de la classe [LesFonctions\\_nD](#)

gestion des différentes fonctions multivariées enregistrées.

```
#include <LesFonctions_nD.h>
```

Graphe de collaboration de [LesFonctions\\_nD](#):



## Fonctions membres publiques

- [LesFonctions\\_nD](#) (const [LesCourbes1D](#) \*lesC1=NULL)
- void [Attribut\\_pointeur\\_lesCourbes1D](#) (const [LesCourbes1D](#) \*lesC)
- void [Lecture](#) ([UtilLecture](#) &entreePrinc)
- void [Info\\_commande\\_lesFonctions\\_nD](#) ([UtilLecture](#) &entreePrinc)
- void [Affiche](#) () const
- bool [Existe](#) (const string &st1) const
- [Fonction\\_nD](#) \* [Trouve](#) (const string &st1) const
- bool [Comple](#) ()
- [Fonction\\_nD](#) \* [Lecture\\_pour\\_base\\_info](#) (ifstream &ent, const int cas, [Fonction\\_nD](#) \*ptcourbe)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [SchemaXML\\_lesFonctions\\_nD](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)

## Fonctions membres publiques statiques

- static void **Ecriture\_pour\_base\_info** (ofstream &sort, const int cas, [Fonction\\_nD](#) \*ptcourbe)

## Attributs protégés

- map< string, [Fonction\\_nD](#) \*, std::less< string > > **listeDeFonction\_nD**
- const [LesCourbes1D](#) \* **lesCourbes1D**

## Attributs protégés statiques

- static [MotCle](#) **motCle**

### 6.436.1 Description détaillée

gestion des différentes fonctions multivariables enregistrées.

Auteur

Gérard Rio

Version

1.0

Date

01/06/2016

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/LesFonctions\_nD.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/LesFonctions\_nD.cc

## 6.437 Référence de la classe LesLoisDeComp

### Classes

- class [Loi](#)
- class [RefLoi](#)

### Fonctions membres publiques

- void **LectureLesLoisDeComp** ([UtilLecture](#) &lec, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void **Affiche** () const
- void **Info\_commande\_lesLoisDeComp** ([UtilLecture](#) &lec, [LesReferences](#) &lesRef)
- const [Tableau](#)< [RefLoi](#) > & **TabRefLoi** () const
- const [Tableau](#)< [Loi](#) > & **TabLoi** () const
- [LoiAbstraiteGeneral](#) \* **PtLoi\_abstraite** (const string &st) const
- void **Loi\_simplifie** (bool ordre)
- bool **Test\_loi\_simplife** ()
- void **LectureDonneesExternes** ([UtilLecture](#) &, [LesReferences](#) &, const int, const string &)
- void **MiseAJour\_umat\_nbiter** (const int &num\_iter)
- void **MiseAJour\_umat\_nbincr** (const int &num\_incr)
- void **Lecture\_base\_info** (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **Sortie\_temps\_cpu** ([UtilLecture](#) &lec)

### Fonctions membres publiques statiques

- static [LoiAbstraiteGeneral](#) \* **Def\_loi** (string &nom)

## Amis

- class **Loi**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LesLoisDeComp.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LesLoisDeComp.cc

## 6.438 Référence de la classe LesMaillages

**LesMaillages**: l'ensemble des maillages.

```
#include <LesMaillages.h>
```

### Fonctions membres publiques

- **LesMaillages** ([UtilLecture](#) \*, [ParaGlob](#) \*, [LesReferences](#) \*lesRef)
- **LesMaillages** (const [LesMaillages](#) &lesmail)
- void **LectureLesMaillages** ()
- int **Creation\_nouveau\_maillage** (list< [Noeud](#) \* > &li\_noeud, list< [Element](#) \* > &list\_elem, const string &nom\_maillage)
- void **Suppression\_maillage** (const string &nom\_maillage, const bool sans\_conservation\_noeuds\_↔ elements=true)
- void **Ajout\_de\_Noeuds** (const list< [Noeud](#) \* > &taN, int numMail)
- void **Ajout\_elements\_et\_noeuds** (const list< [Noeud](#) \* > &taN, const list< [Element](#) \* > &taE, list< const [Reference](#) \* > \*lref, [LesReferences](#) \*lesRef, int numMail)
- void **Info\_commande\_lesMaillages** ()
- void **Affiche** () const
- void **Affiche** (char \*nom\_fichier) const
- bool **Complet** ()
- void **Completer** ([DiversStockage](#) \*divers, [LesLoisDeComp](#) \*lesLois, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- void **Mise\_a\_jour\_repere\_anisotropie** ([DiversStockage](#) \*divers, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- int **NbMaillage** () const
- int **Nombre\_element** (int i) const
- int **Nombre\_noeud** (int i) const
- [Element](#) & **Element\_LesMaille** (int i, int j) const
- const [Element](#) & **Element\_LesMaille\_const** (int i, int j) const
- [Noeud](#) & **Noeud\_LesMaille** (int i, int j) const
- [Tableau](#)< [Noeud](#) \* > & **Tab\_noeud** (int i)
- string **NomMaillage** (int i) const
- int **NumMaillage** (const string &nom) const
- void **SuppressionNoeudNonReferencer** ([LesReferences](#) &lesRef)
- void **AffichageNoeudNonReferencer** ()
- bool **Renumerotation** ([LesReferences](#) &lesRef, const [Tableau](#)< [Tableau](#)< [Condilinaire](#) > > &cond↔ CLL, [TroisEntiers](#) &nouvelles\_largeur\_en\_ddl, const [Nb\\_assemb](#) \*nb\_casAssemb=NULL, bool sans↔ changement\_num\_noeud=false)
- bool **Renumerotation** ([LesReferences](#) &lesRef)
- void **CreationRefNoeudNonReferencer** ([LesReferences](#) &lesRef)
- int **Noeud\_le\_plus\_proche\_0** (const [Coordonnee](#) &M, int i=1)
- int **Noeud\_le\_plus\_proche\_t** (const [Coordonnee](#) &M, int i=1)
- int **Noeud\_le\_plus\_proche\_tdt** (const [Coordonnee](#) &M, int i=1)
- [Maillage::NBelemEtpInteg](#) **Element\_le\_plus\_proche** ([Enum\\_dure](#) enu\_temps, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &list\_enu, const [Coordonnee](#) &M, int i=1)
- const list< [Enum\\_ddl](#) > & **Ddl\_representatifs\_des\_physiques** () const
- const list< [EnumElemTypeProblem](#) > & **Types\_de\_problemes** () const
- int **NbTotalDdlActifs** () const
- int **NbTotalDdlActifs** ([Enum\\_ddl](#) enum\_ddl) const
- int **NbTotalPtlInteg** ([Enum\\_ddl](#) enum\_ddl) const
- int **NbTotalGrandeursGeneratrices** ([Enum\\_ddl](#) enu) const
- [Vecteur](#) & **Vect\_loc\_vers\_glob** ([Enum\\_dure](#) duree, [Enum\\_ddl](#) enum\_actif, [Vecteur](#) &vect, [Enum\\_ddl](#) enum↔\_ddl)
- void **Vect\_glob\_vers\_local** ([Enum\\_dure](#) duree, [Enum\\_ddl](#) enum\_actif, const [Vecteur](#) &vect, [Enum\\_ddl](#) enum\_ddl)

- **Vecteur** & **Vect\_loc\_vers\_glob** (**Enum\_dure** duree, const **Tableau**< **Enum\_ddl** > &tab\_enum\_actif, **Vecteur** &vect, const **Tableau**< **Enum\_ddl** > &tab\_enum\_ddl)
- void **Vect\_glob\_vers\_local** (**Enum\_dure** duree, const **Tableau**< **Enum\_ddl** > &tab\_enum\_actif, const **Vecteur** &vect, const **Tableau**< **Enum\_ddl** > &tab\_enum\_ddl)
- **Vecteur** & **Scalaire\_loc\_vers\_glob** (**Enum\_dure** duree, **Enum\_ddl** enum\_actif, **Vecteur** &vect, **Enum\_ddl** enum\_ddl)
- void **Scalaire\_glob\_vers\_local** (**Enum\_dure** duree, **Enum\_ddl** enum\_actif, **Vecteur** &vect, **Enum\_ddl** enum\_ddl)
- **Vecteur** & **Scalaire\_loc\_vers\_glob** (**Enum\_ddl** enum\_actif, **Vecteur** &vect, const **Ddl\_enum\_etendu** &enum\_↵ddl\_etendu)
- void **Scalaire\_glob\_vers\_local** (**Enum\_ddl** enum\_actif, **Vecteur** &vect, const **Ddl\_enum\_etendu** &enum\_↵ddl\_etendu)
- **Vecteur** & **Quelconque\_loc\_vers\_glob** (**Enum\_ddl** enum\_actif, **Vecteur** &vect, const **TypeQuelconque** &type\_generique)
- void **Quelconque\_glob\_vers\_local** (**Enum\_ddl** enum\_actif, **Vecteur** &vect, const **TypeQuelconque** &type\_↵generique)
- **Tableau**< **List\_io**< **Ddl\_enum\_etendu** > > **Les\_type\_de\_ddl\_par\_noeud** (bool absolue)
- **Tableau**< **List\_io**< **Ddl\_enum\_etendu** > > **Les\_type\_de\_ddl\_etendu\_par\_noeud** (bool absolue)
- **Tableau**< **List\_io**< **TypeQuelconque** > > **Les\_type\_de\_TypeQuelconque\_par\_noeud** (bool absolue)
- void **Init\_par\_defaut\_conteneurs** (**List\_io**< **TypeQuelconque** > &li\_restreinte\_TQ)
- void **Init\_par\_defaut\_conteneurs** (**TypeQuelconque\_enum\_etendu** enuTypeQuelconque)
- void **Intro\_Conteneurs\_internes\_noeud\_relier\_auto\_autres\_grandeur** (const **List\_io**< **TypeQuelconque** > &glob\_noeud\_evol\_retenu)
- void **Intro\_Conteneurs\_internes\_noeud\_relier\_auto\_autres\_grandeur** (const **Tableau**< **List\_io**< **TypeQuelconque** > > &tab\_noeud\_evol\_retenu)
- **Tableau**< **List\_io**< **Ddl\_enum\_etendu** > > **Les\_type\_de\_ddl\_par\_element** (bool absolue)
- **Tableau**< **List\_io**< **TypeQuelconque** > > **Les\_type\_de\_donnees\_evolues\_internes\_par\_element** (bool absolue)
- **Tableau**< **List\_io**< **TypeQuelconque** > > **Les\_type\_de\_donnees\_particulieres\_par\_element** (bool absolue)
- **Tableau**< **List\_io**< **TypeQuelconque** > > **Les\_type\_de\_donnees\_evolues\_internes\_par\_face\_element** (bool absolue)
- **Tableau**< **List\_io**< **TypeQuelconque** > > **Les\_type\_de\_donnees\_evolues\_internes\_par\_arete\_element** (bool absolue)
- void **PassageInterneDansNoeud\_composantes\_vers\_vectorielles** ()
- void **AjoutConteneurAuNoeud** (int num\_maillage, const **List\_io**< **Ddl\_enum\_etendu** > &lienu, const **Tableau**< **List\_io**< **TypeQuelconque** > \* > &tabQ)
- void **AjoutConteneurAuNoeud** (const **List\_io**< **Ddl\_enum\_etendu** > &lienu, const **Tableau**< **List\_io**< **TypeQuelconque** > \* > &tabQ)
- void **InitUpdateAuNoeud** (const **List\_io**< **Ddl\_enum\_etendu** > &lienu, const **Tableau**< **List\_io**< **TypeQuelconque** > \* > &tabQ, int cas)
- void **InitUpdateAuNoeud** (int numMail, const **List\_io**< **Ddl\_enum\_etendu** > &lienu, const **Tableau**< **List\_io**< **TypeQuelconque** > \* > &tabQ, int cas)
- void **TransfertPtIntegAuNoeud** (int numMail, **Element** &ele, const **List\_io**< **Ddl\_enum\_etendu** > &lietendu, const **Tableau**< **Tableau**< double > > &tab\_val, int cas)
- void **TransfertPtIntegAuNoeud** (int numMail, **Element** &ele, const **Tableau**< **List\_io**< **TypeQuelconque** > > &tab\_liQ, **List\_io**< **TypeQuelconque** > &liQ\_travail, int cas)
- void **FinTransfertPtIntegAuNoeud** (const **List\_io**< **Ddl\_enum\_etendu** > &lienu, const **Tableau**< **List\_io**< **TypeQuelconque** > \* > &tabQ, int cas)
- void **FinTransfertPtIntegAuNoeud** (int numMail, const **List\_io**< **Ddl\_enum\_etendu** > &lienu, const **Tableau**< **List\_io**< **TypeQuelconque** > \* > &tabQ, int cas)
- void **Accumul\_aux\_noeuds** (int numMail, const **List\_io**< **Ddl\_enum\_etendu** > &lietendu, **List\_io**< **TypeQuelconque** > &liQ, int cas)
- void **AjoutConteneurAuNoeud** (**TypeQuelconque** &tQ)
- void **AjoutConteneurAuNoeud** (const **List\_io**< **Ddl\_enum\_etendu** > &lienu)
- void **InitUpdateAuNoeud** (const **List\_io**< **Ddl\_enum\_etendu** > &lienu)
- void **InitUpdateAuNoeud** (const **Ddl\_enum\_etendu** &enu)
- void **MoyenneCompteurAuNoeud** (const **Ddl\_enum\_etendu** &enu)
- **Nb\_assemb** **InitNouveauCasAssemblage** (int nb\_cas)
- int **Nb\_total\_en\_cours\_de\_cas\_Assemblage** () const
- void **MiseAJourPointeurAssemblage** (const **Nb\_assemb** &nb\_casAssemb)
- void **ZeroDdl** (bool indic, bool cas=true)

- void **Force\_Ddl\_aux\_noeuds\_a\_une\_valeur** (Enum\_ddl enu, const double &val, [Enum\\_dure](#) temps, bool fonction\_de\_la\_dimension)
- void **Force\_Ddl\_etendu\_aux\_noeuds\_a\_zero** (const [Tableau](#)< [Ddl\\_enum\\_etendu](#) > &tab\_enu)
- void **Largeur\_Bande** (int &demi, int &total, const [Nb\\_assemb](#) &nb\_casAssemb)
- void **Table\_connexion** ([Tableau](#)< [Tableau](#)< int > > &petites\_matricespetites\_matrices, const [Nb\\_assemb](#) &nb\_casAssemb) const
- void **TdtversT** ()
- void **TversTdt** ()
- void **PlusDelta\_tdt** ([Vecteur](#) &sol, const [Nb\\_assemb](#) &nb\_casAssemb)
- void **PlusDelta\_t** ([Vecteur](#) &sol, const [Nb\\_assemb](#) &nb\_casAssemb)
- [Vecteur](#) & **RecupDepde\_tatdt** ([Vecteur](#) &sol, const [Nb\\_assemb](#) &nb\_casAssemb)
- void **ChangeDdla\_tdt** ([Vecteur](#) &sol, const [Nb\\_assemb](#) &nb\_casAssemb)
- **Ddl NoeudIndice** (int inSol, int &nbNoeud, int &nbMaillage, const [Nb\\_assemb](#) &nb\_casAssemb)
- **Ddl NoeudIndice** (int inSol, int &nbNoeud, int &nbMaillage, double &val0, const [Nb\\_assemb](#) &nb\_casAssemb)
- void **InitNormaleAuxNoeuds** ()
- void **MiseAJourNormaleAuxNoeuds** ()
- void **MiseAJourNormaleAuxNoeuds\_de\_tdt\_vers\_T** ()
- int **CreeElemFront** ()
- int **NbEsclave** ()
- [Tableau](#)< [LaLIST](#)< [Front](#) > \* > & **ListFrontiere** ()
- [Tableau](#)< [Tableau](#)< [Noeud](#) \* > \* > **Tab\_noeud\_frontiere\_esclave** ()
- const [Tableau](#)< [Tableau](#)< [Noeud](#) \* > \* > & **Tab\_noeud\_frontiere** ()
- const [Tableau](#)< [Noeud](#) \* > **Esclave** ()
- void **Mise\_a\_jour\_boite\_encroisement\_elem\_front** ([Enum\\_dure](#) temps)
- const [Tableau](#)< const [Tableau](#)< [List\\_io](#)< [Element](#) \* > \* > & **Indice** ()
- void **Inactive\_ddl\_et\_donnees** ()
- void **Inactive\_ddl** ()
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **Plus\_ddl\_Erreur** ()
- void **Inactive\_ddl\_Erreur** ()
- void **Active\_ddl\_Erreur** ()
- void **Active\_un\_type\_ddl\_particulier** (Enum\_ddl en)
- void **Active\_un\_type\_ddl\_particulier** ([Tableau](#)< Enum\_ddl > &tab\_en)
- void **Active\_un\_type\_ddl\_particulier** (const list< Enum\_ddl > &list\_en)
- void **Inactive\_un\_type\_ddl\_particulier** (Enum\_ddl en)
- void **Inactive\_un\_type\_ddl\_particulier** ([Tableau](#)< Enum\_ddl > &tab\_en)
- void **Inactive\_un\_type\_ddl\_particulier** (const list< Enum\_ddl > &list\_en)
- void **ChangeToutesLesConditions** (const [Tableau](#)< [Ddl](#) > &ta)
- void **ChangeStatut** (int cas, Enum\_ddl enuta)
- void **ChangeStatut** (int cas, [Enum\\_boolddl](#) enubold)
- void **Libere\_Ddl\_representatifs\_des\_physiques** ([Enum\\_boolddl](#) enubold)
- [Tableau](#)< [Tableau](#)< double > > **ErreurSurChaqueElement** (int type)
- void **LectureDonneesExternes** (const int type, const string &nomMaillage)
- void **Travail\_t** ()
- void **Travail\_t** (int num)
- void **Insert\_coord1** ()
- void **Travail\_tdt** ()
- void **Init\_Xi\_t\_et\_tdt\_de\_0** ()
- double **Max\_var\_dep\_t\_a\_tdt** () const
- double **Min\_dist2Noeud\_des\_elements** ([Enum\\_dure](#) temps) const
- void **Drapeau\_preparation\_calcul\_precis** (int precision)
- void **Plus\_Les\_ddl\_Vitesse** ([Enum\\_boolddl](#) val\_fixe)
- void **Plus\_Les\_ddl\_Acceleration** ([Enum\\_boolddl](#) val\_fixe)
- double **Longueur\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- void **Init\_bulk\_viscosity** (int choix, const [DeuxDoubles](#) &coef)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **SchemaXML\_LesMaillages** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const
- [Tableau](#)< [Vecteur](#) > **Taille\_boite** (int nbmail)

- [Tableau](#)< [Vecteur](#) > [Taille\\_boite](#) ()
- void [CreeMaillagesQuadratiques\\_a\\_partir\\_des\\_lineaires](#) ([LesReferences](#) \*lesRef)
- void [CreeMaillagesQuadratiquesComplets\\_a\\_partir\\_des\\_incomplets](#) ([LesReferences](#) \*lesRef)
- void [CreeMaillageExtrusion2D3D](#) ([LesReferences](#) \*lesRef)
- void [CreationInteractiveListesRef](#) ([LesReferences](#) \*lesRef)
- void [Modif\\_orientation\\_element](#) (int cas\_orientation, [LesReferences](#) \*lesRef)
- void [Collapse\\_noeuds\\_proches](#) (double rayon, [LesReferences](#) \*lesRef)
- void [Collapse\\_element\\_supperpose](#) ([LesReferences](#) \*lesRef)
- void [Fusion\\_maillages](#) ([List\\_io](#)< string > &nom\_mails\_a\_fusionner, const string &new\_mail, [LesReferences](#) \*lesRef)
- void [Cree\\_sous\\_maillage](#) (int num\_mail, [LesReferences](#) \*lesRef, string nom\_ref, const string &new\_mail)
- void [CreationMaillageSFE](#) ()
- void [Affiche\\_maillage\\_dans\\_her\\_lis](#) ([Enum\\_dure](#) temps, [LesReferences](#) &lesRef, int imail=-1)
- void [RelocPtMilieuMailleQuadra](#) ()
- void [Integration](#) ()
- void [CalStatistique](#) ()

### 6.438.1 Description détaillée

[LesMaillages](#): l'ensemble des maillages.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesMaillages.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesMaillages.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesMaillages2.cc

## 6.439 Référence de la classe LesMaillonsBB

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires  
`#include <Tenseur.h>`

### Fonctions membres publiques statiques

- static void [Libere](#) ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void [NouveauMaillon](#) (const [TenseurBB](#) \*)  
*enregistrement de l'ajout d'un tenseur*

### Attributs publics statiques

- static [PtTenseurBB](#) \* [maille](#) = NULL  
*dernier maillon*

### 6.439.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

### 6.439.2 Documentation des données membres

### 6.439.2.1 maille

```
PtTenseurBB * LesMaillonsBB::maille = NULL [static]
dernier maillon
```

initialisation de la variable static pour la gestion des tenseurs intermediaires du second ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.440 Référence de la classe LesMaillonsBBBB

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <TenseurQ.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurBBBB](#) \*)  
*enregistrement de l'ajout d'un tenseur*

### Attributs publics statiques

- static PtTenseurBBBB \* [maille](#) = NULL  
*dernier maillon*

### 6.440.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

### 6.440.2 Documentation des données membres

#### 6.440.2.1 maille

```
PtTenseurBBBB * LesMaillonsBBBB::maille = NULL [static]
dernier maillon
```

initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

## 6.441 Référence de la classe LesMaillonsBBHH

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <TenseurQ.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurBBHH](#) \*)  
*enregistrement de l'ajout d'un tenseur*



## Attributs publics statiques

- static PtTenseurBBHH \* [maille](#) = NULL  
*dernier maillon*

### 6.441.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

### 6.441.2 Documentation des données membres

#### 6.441.2.1 maille

```
PtTenseurBBHH * LesMaillonsBBHH::maille = NULL [static]
```

dernier maillon

initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

## 6.442 Référence de la classe LesMaillonsBH

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <Tenseur.h>
```

## Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurBH](#) \*)  
*enregistrement de l'ajout d'un tenseur*

## Attributs publics statiques

- static PtTenseurBH \* [maille](#) = NULL  
*dernier maillon*

### 6.442.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

### 6.442.2 Documentation des données membres

#### 6.442.2.1 maille

```
PtTenseurBH * LesMaillonsBH::maille = NULL [static]
```

dernier maillon

initialisation de la variable static pour la gestion des tenseurs intermediaires du second ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[Tenseur.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

## 6.443 Référence de la classe LesMaillonsBHBH

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <TenseurQ.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurBHBH](#) \*)  
*enregistrement de l'ajout d'un tenseur*

### Attributs publics statiques

- static PtTenseurBHBH \* **maille** = NULL  
*dernier maillon*

#### 6.443.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

#### 6.443.2 Documentation des données membres

##### 6.443.2.1 maille

```
PtTenseurBHBH * LesMaillonsBHBH::maille = NULL [static]
```

dernier maillon

initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

## 6.444 Référence de la classe LesMaillonsBHHB

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <TenseurQ.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurBHHB](#) \*)  
*enregistrement de l'ajout d'un tenseur*

### Attributs publics statiques

- static PtTenseurBHHB \* **maille** = NULL  
*dernier maillon*

#### 6.444.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

#### 6.444.2 Documentation des données membres

### 6.444.2.1 maille

```
PtTenseurBHHB * LesMaillonsBHHB::maille = NULL [static]
dernier maillon
```

initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.445 Référence de la classe LesMaillonsHB

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <Tenseur.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurHB](#) \*)  
*enregistrement de l'ajout d'un tenseur*

### Attributs publics statiques

- static PtTenseurHB \* [maille](#) = NULL  
*dernier maillon*

### 6.445.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

### 6.445.2 Documentation des données membres

#### 6.445.2.1 maille

```
PtTenseurHB * LesMaillonsHB::maille = NULL [static]
dernier maillon
```

initialisation de la variable static pour la gestion des tenseurs intermediaires du second ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.446 Référence de la classe LesMaillonsHBBH

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <TenseurQ.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurHBBH](#) \*)  
*enregistrement de l'ajout d'un tenseur*

## Attributs publics statiques

- static PtTenseurHBBH \* [maille](#) = NULL  
*dernier maillon*

### 6.446.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

### 6.446.2 Documentation des données membres

#### 6.446.2.1 maille

```
PtTenseurHBBH * LesMaillonsHBBH::maille = NULL [static]
```

dernier maillon

initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

## 6.447 Référence de la classe LesMaillonsHBHB

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <TenseurQ.h>
```

## Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurHBHB](#) \*)  
*enregistrement de l'ajout d'un tenseur*

## Attributs publics statiques

- static PtTenseurHBHB \* [maille](#) = NULL  
*dernier maillon*

### 6.447.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

### 6.447.2 Documentation des données membres

#### 6.447.2.1 maille

```
PtTenseurHBHB * LesMaillonsHBHB::maille = NULL [static]
```

dernier maillon

initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

## 6.448 Référence de la classe LesMaillonsHH

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <Tenseur.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurHH](#) \*)  
*enregistrement de l'ajout d'un tenseur*

### Attributs publics statiques

- static PtTenseurHH \* **maille** = NULL  
*dernier maillon*

#### 6.448.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

#### 6.448.2 Documentation des données membres

##### 6.448.2.1 maille

```
PtTenseurHH * LesMaillonsHH::maille = NULL [static]
```

dernier maillon

initialisation de la variable static pour la gestion des tenseurs intermediaires du second ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

## 6.449 Référence de la classe LesMaillonsHHBB

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <TenseurQ.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurHHBB](#) \*)  
*enregistrement de l'ajout d'un tenseur*

### Attributs publics statiques

- static PtTenseurHHBB \* **maille** = NULL  
*dernier maillon*

#### 6.449.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

#### 6.449.2 Documentation des données membres

### 6.449.2.1 maille

```
PtTenseurHHBB * LesMaillonsHHBB::maille = NULL [static]
dernier maillon
```

initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.450 Référence de la classe LesMaillonsHHHH

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

```
#include <TenseurQ.h>
```

### Fonctions membres publiques statiques

- static void **Libere** ()  
*liberation de la place occupee par des tenseurs crees pour les operations intermediaires*
- static void **NouveauMaillon** (const [TenseurHHHH](#) \*)  
*enregistrement de l'ajout d'un tenseur*

### Attributs publics statiques

- static [PtTenseurHHHH](#) \* **maille** = NULL  
*dernier maillon*

### 6.450.1 Description détaillée

def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

### 6.450.2 Documentation des données membres

#### 6.450.2.1 maille

```
PtTenseurHHHH * LesMaillonsHHHH::maille = NULL [static]
dernier maillon
```

initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.451 Référence de la classe LesPtIntegMecalInterne

### Fonctions membres publiques

- **LesPtIntegMecalInterne** (int nbpti, int dimtens)
- **LesPtIntegMecalInterne** (const [LesPtIntegMecalInterne](#) &lespti)
- **LesPtIntegMecalInterne** & **operator=** (const [LesPtIntegMecalInterne](#) &lespti)
- **Tableau**< [PtIntegMecalInterne](#) > & **TabPtIntMeca** ()
- **PtIntegMecalInterne** & **operator()** (int i)
- **Tableau**< [TenseurHH](#) \* > & **TabSigHH** ()
- **Tableau**< [TenseurHH](#) \* > & **TabSigHH\_t** ()
- int **NbPti** () const
- void **Change\_taille\_PtIntegMeca** (int nbpti, int dimtens)
- void **Change\_taille\_PtIntegMeca** (int nbpti)
- int **DimTens** () const

- void **TdtversT** ()
- void **TversTdt** ()
- double **CompressibiliteMoyenne** () const
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

### Attributs protégés

- [Tableau](#)< [PtIntegMecalInterne](#) > **tabPtInt**
- [Tableau](#)< [TenseurHH](#) \* > **tabSigHH**
- [Tableau](#)< [TenseurHH](#) \* > **tabSigHH\_t**

### Amis

- ifstream & **operator**>> (ifstream &, [LesPtIntegMecalInterne](#) &)
- ostream & **operator**<< (ostream &, const [LesPtIntegMecalInterne](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/LesPtIntegMecalInterne.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/LesPtIntegMecalInterne.cc

## 6.452 Référence de la classe LesPtIntegThermilInterne

### Fonctions membres publiques

- [LesPtIntegThermilInterne](#) (int nbpti, int dimcoor)
- [LesPtIntegThermilInterne](#) (const [LesPtIntegThermilInterne](#) &lespti)
- [LesPtIntegThermilInterne](#) & **operator=** (const [LesPtIntegThermilInterne](#) &lespti)
- [Tableau](#)< [PtIntegThermilInterne](#) > & **TabPtIntThermi** ()
- [PtIntegThermilInterne](#) & **operator()** (int i)
- [Tableau](#)< [CoordonneeH](#) \* > & **TabfluxH** ()
- [Tableau](#)< [CoordonneeH](#) \* > & **TabfluxH\_t** ()
- int **NbPti** () const
- void **Change\_taille\_PtIntegThermi** (int nbpti, int dimtens)
- void **Change\_taille\_PtIntegThermi** (int nbpti)
- int **DimCoord** () const
- void **TdtversT** ()
- void **TversTdt** ()
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

### Attributs protégés

- [Tableau](#)< [PtIntegThermilInterne](#) > **tabPtInt**
- [Tableau](#)< [CoordonneeH](#) \* > **tabfluxH**
- [Tableau](#)< [CoordonneeH](#) \* > **tabfluxH\_t**

### Amis

- ifstream & **operator**>> (ifstream &, [LesPtIntegThermilInterne](#) &)
- ostream & **operator**<< (ostream &, const [LesPtIntegThermilInterne](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

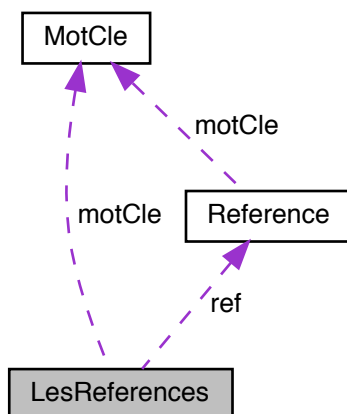
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/LesPtIntegThermilInterne.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/LesPtIntegThermilInterne.cc

## 6.453 Référence de la classe LesReferences

Gestion des listes de references.

```
#include <LesReferences.h>
```

Graphe de collaboration de LesReferences:



### Fonctions membres publiques

- void **NbMaille** (int nb)
- void **Indic** (string type)
- void **Lecture** ([UtilLecture](#) &entreePrinc)
- void **Ajout\_reference** ([Reference](#) \*refi)
- void **SupprimeReference** (const string &st1, int num\_mail)
- void **Supprime\_tour\_lesRef\_un\_maillage** (int num\_mail, bool avec\_diminution\_num\_maillage)
- void **Affiche** (int niveau=0) const
- void **Affiche** (int indic, int niveau) const
- void **Affiche\_dans\_lis** (const string &nom\_maillage, int imail)
- void **Info\_commande\_lesRef** (int nbMaxiNoeud, int nbMaxiElem, [UtilLecture](#) \*entreePrinc, int cas)
- const [Reference](#) \* **Init\_et\_Premiere** ()
- const [Reference](#) \* **Reference\_suivante** ()
- bool **Existe** (const string &st1, int num\_mail) const
- bool **Existe** (const string &st1, const string \*nom\_mail) const
- const [Reference](#) & **Trouve** (const string &st1, int num\_mail) const
- const [Reference](#) & **Trouve** (const string &st1, const string \*nom\_mail) const
- void **MiseAJourMap** (const map< string, int, std::less< string > > &listNomMail)
- void **Mise\_a\_jour\_ref\_noeud** ([Tableau](#)< int > &nv\_tab, int num\_mail, [Tableau](#)< bool > &non\_referencer)
- void **Mise\_a\_jour\_ref\_noeud** ([Tableau](#)< int > &nv\_tab, int num\_mail)
- void **Mise\_a\_jour\_ref\_element** ([Tableau](#)< int > &nv\_tab, int num\_mail, [Tableau](#)< bool > &non\_referencer)
- void **Mise\_a\_jour\_num\_maillage\_ref** (const string nom\_ref, int old\_num\_maill, int new\_num\_maill)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

### Fonctions membres protégées

- bool **LectureReference** ([UtilLecture](#) &entreePrinc)
- [Reference](#) & **Trouve\_interne** (const string &st1, int num\_mail) const
- [Reference](#) & **Trouve\_interne** (const string &st1, const string \*nom\_mail) const



## Attributs protégés

- int **nbMaille**
- int **indic**
- [Reference](#) \* **ref**
- [Tableau](#)< map< string, [Reference](#) \*, std::less< string > > > **t\_mapDeRef**
- const map< string, int, std::less< string > > \* **listeNomMail**
- map< string, [Reference](#) \*, std::less< string > >::const\_iterator **iref**
- int **num\_mail\_presuivant**

## Attributs protégés statiques

- static [MotCle](#) **motCle**

### 6.453.1 Description détaillée

Gestion des listes de references.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/LesReferences.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/LesReferences.cc

## 6.454 Référence de la classe LesSuiteReel

gestion des différentes Suites de réels enregistrées.

```
#include <LesSuitesReel.h>
```

## Fonctions membres publiques statiques

- static [SuiteReel](#) \* **Lecture\_uneSuiteReel** (double amplitude=0., int n=0)

## Attributs protégés statiques

- static list< [SuiteReel](#) \* > **listeDeSuites**

### 6.454.1 Description détaillée

gestion des différentes Suites de réels enregistrées.

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/LesSuitesReel.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/LesSuitesReel.cc

## 6.455 Référence de la classe LesValVecPropres

### Fonctions membres publiques

- **LesValVecPropres** ([Tableau](#)< [VeurPropre](#) > &VP)
- **LesValVecPropres** (const [LesValVecPropres](#) &a)
- **LesValVecPropres** & **operator=** (const [LesValVecPropres](#) &a)
- void **Affiche** (ofstream &sort) const

La documentation de cette classe a été générée à partir du fichier suivant :

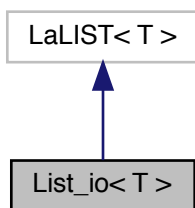
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Flambage/LesValVecPropres.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Flambage/LesValVecPropres.cc

## 6.456 Référence du modèle de la classe `List_io< T >`

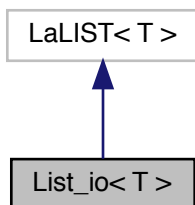
`List_io` classe template identique à `list` de `stl`, avec en plus une lecture et écriture.

```
#include <List_io.h>
```

Graphe d'héritage de `List_io< T >` :



Graphe de collaboration de `List_io< T >` :



### Amis

- `istream & operator>>` (`istream &entree`, `List_io< T > &`)
- `ostream & operator<<` (`ostream &sort`, `const List_io< T > &a`)
- `istream & operator>>` (`istream &entree`, `typename LaLIST< T >::iterator &`)
- `ostream & operator<<` (`ostream &sort`, `const typename LaLIST< T >::iterator`)
- `istream & operator>>` (`istream &entree`, `typename LaLIST< T >::const_iterator &`)
- `ostream & operator<<` (`ostream &sort`, `const typename LaLIST< T >::const_iterator &`)

### 6.456.1 Description détaillée

```
template<class T>
class List_io< T >
```

`List_io` classe template identique à `list` de `stl`, avec en plus une lecture et écriture.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/List\_io.h

## 6.457 Référence de la classe Algori::ListDeuxString

**Attributs publics**

— [List\\_io](#)< [DeuxString](#) > **infos**

**Amis**

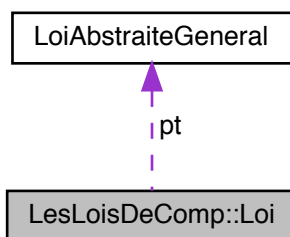
— `istream & operator>>` (`istream &`, [Algori::ListDeuxString &](#))  
 — `ostream & operator<<` (`ostream &`, `const Algori::ListDeuxString &`)

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Algo/AlgoRef/Algori.h

## 6.458 Référence de la classe LesLoisDeComp::Loi

Graphes de collaboration de LesLoisDeComp::Loi:

**Fonctions membres publiques**

— **Loi** (`string &s1`, [LoiAbstraiteGeneral \\*p](#))  
 — **Loi** (`const Loi &a`)  
 — **Loi & operator=** (`const Loi &a`)  
 — `void Lecture_base_info` (`ifstream &ent`, `const int cas`, [LesReferences &lesRef](#), [LesCourbes1D &lesCourbes1D](#), [LesFonctions\\_nD &lesFonctionsnD](#))  
 — `void Ecriture_base_info` (`ofstream &sort`, `const int cas`)

**Attributs publics**

— `string st1`  
 — [LoiAbstraiteGeneral \\* pt](#)

### Amis

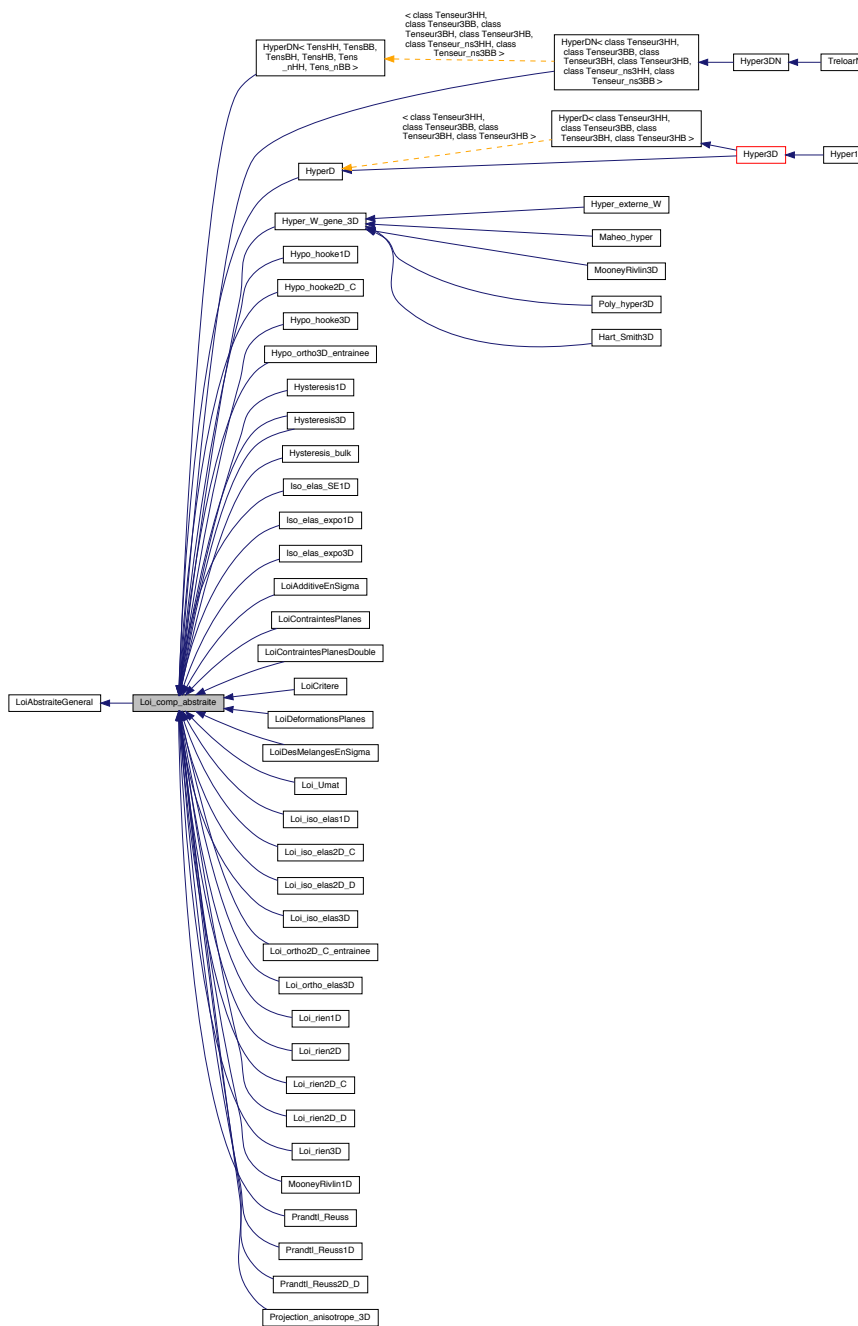
- istream & **operator**>> (istream &, LesLoisDeComp::Loi &)
- ostream & **operator**<< (ostream &, const LesLoisDeComp::Loi &)

La documentation de cette classe a été générée à partir du fichier suivant :

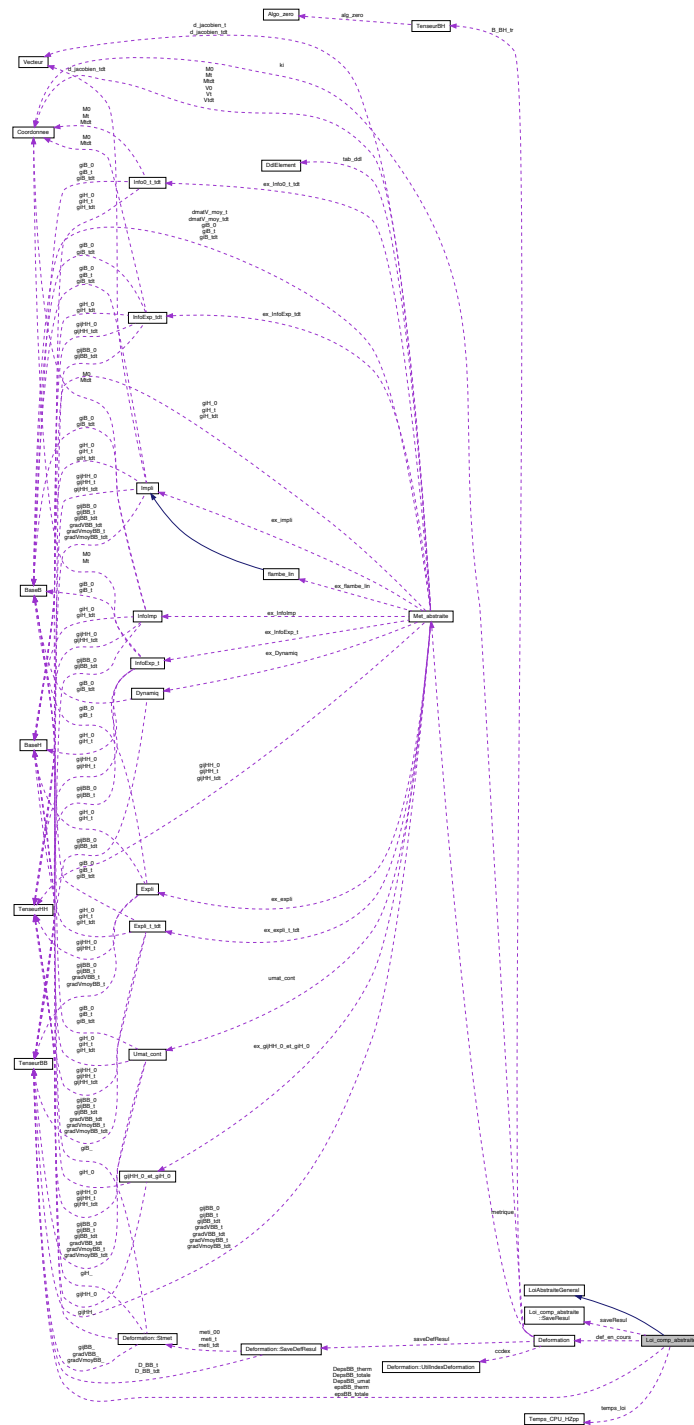
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LesLoisDeComp.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LesLoisDeComp.cc

## 6.459 Référence de la classe Loi\_comp\_abstraite

Graphe d'héritage de Loi\_comp\_abstraite:



Graphe de collaboration de `Loi_comp_abstraite`:



## Classes

- class `SaveResul`
- class `SaveResul_C`

## Fonctions membres publiques

- `Loi_comp_abstraite` (`Enum_comp` id\_comp, `Enum_categorie_loi_comp` categorie\_comp, int dimension, bool vit\_def=false)

- **Loi\_comp\_abstraite** (char \*nom, [Enum\\_categorie\\_loi\\_comp](#) categorie\_comp, int dimension, bool vit\_↵ def=false)
- **Loi\_comp\_abstraite** (const [Loi\\_comp\\_abstraite](#) &a)
- virtual [SaveResul](#) \* **New\_et\_Initialise** ()
- virtual void **AfficheDataSpecif** (ofstream &, [SaveResul](#) \*) const
- void **Def\_type\_deformation** ([Deformation](#) &def)
- virtual const [Met\\_abstraite::Expli](#) & **Cal\_explicit\_t** ([Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [Deformation](#) &def, [DdlElement](#) &tab\_ddl, [PtIntegMecalInterne](#) &ptintmeca, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, double &Jacobien, [CompThermoPhysiqueAbstraite::SaveResul](#) \*saveTP, [CompThermoPhysiqueAbstraite](#) \*loiTP, bool dilatation, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, bool premier\_calcul)
- virtual const [Met\\_abstraite::Expli\\_t\\_tdt](#) & **Cal\_explicit\_tdt** ([Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [Deformation](#) &def, [DdlElement](#) &tab\_ddl, [PtIntegMecalInterne](#) &ptintmeca, [Tableau](#)< [TenseurBB](#) \* > &d\_↵ epsBB, double &Jacobien, [CompThermoPhysiqueAbstraite::SaveResul](#) \*saveTP, [CompThermoPhysiqueAbstraite](#) \*loiTP, bool dilatation, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, bool premier\_calcul)
- virtual const [Met\\_abstraite::Impli](#) & **Cal\_implicit** ([Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [Deformation](#) &def, [DdlElement](#) &tab\_ddl, [PtIntegMecalInterne](#) &ptintmeca, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB\_tdt, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, const [ParaAlgoControle](#) &pa, [CompThermoPhysiqueAbstraite::SaveResul](#) \*saveTP, [CompThermoPhysiqueAbstraite](#) \*loiTP, bool dilatation, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, bool premier\_calcul)
- virtual void **Cal\_flamb\_lin** ([Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [Deformation](#) &def, [DdlElement](#) &tab\_ddl, [PtIntegMecalInterne](#) &ptintmeca, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB\_tdt, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, const [ParaAlgoControle](#) &pa, [CompThermoPhysiqueAbstraite::SaveResul](#) \*saveTP, [CompThermoPhysiqueAbstraite](#) \*loiTP, bool dilatation, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, bool premier\_calcul)
- virtual void **ComportementUmat** ([Loi\\_comp\\_abstraite::SaveResul](#) \*saveDon, [Deformation](#) &def, [PtIntegMecalInterne](#) &ptintmeca, [ParaAlgoControle](#) &pa, [CompThermoPhysiqueAbstraite::SaveResul](#) \*saveTP, [CompThermoPhysiqueAbstraite](#) \*loiTP, bool dilatation, [UmatAbaqus](#) &umatAbaqusqus, bool premier\_calcul)
- virtual void **Activation\_donnees** ([Tableau](#)< [Noeud](#) \* > &tabnoeud, bool dilatation, [LesPtIntegMecalInterne](#) &lesPtMecalInt)
- virtual void **Besoin\_de\_grandeurs\_particuliere** (list< [EnumTypeQuelconque](#) > &listEnuQuelc) const
- virtual void **Activation\_stockage\_grandeurs\_quelconques** (list< [EnumTypeQuelconque](#) > &listEnu↵ Quelc)
- bool **Existe\_stockage\_grandeurs\_quelconques** ([EnumTypeQuelconque](#) enuQuelc) const
- virtual void **Insertion\_conteneur\_dans\_save\_result** ([SaveResul](#) \*saveResul)
- const list< [EnumTypeQuelconque](#) > & **ListQuelc\_mis\_en\_acces\_localement** () const
- const list< [EnumTypeQuelconque](#) > & **ListdeTouslesQuelc\_dispo\_localement** () const
- void **Modif\_comp\_tangent\_simplifie** (bool modif)
- bool **Test\_loi\_simplifie** ()
- virtual double **Module\_young\_equivalent** ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul](#) \*save↵ Resul)
- virtual double **Module\_compressibilite\_equivalent** ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul](#) \*saveResul)
- virtual double **Eps33BH** ([SaveResul](#) \*saveResul) const
- virtual double **Eps22BH** ([SaveResul](#) \*saveResul) const
- virtual double **HsurH0** ([SaveResul](#) \*saveResul) const =0
- virtual double **d\_HsurH0** ([SaveResul](#) \*saveResul, [Vecteur](#) &d\_hsurh0) const
- virtual double **BsurB0** ([SaveResul](#) \*saveResul) const
- virtual double **d\_BsurB0** ([SaveResul](#) \*saveResul, [Vecteur](#) &d\_bsurb0) const
- virtual bool **Contraintes\_planes\_de\_3D** () const
- bool **ThermoDependante** () const
- void **Mise\_a\_jour\_temperature** ([Enum\\_dure](#) temps, [Deformation](#) &def)
- virtual void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &, [Loi\\_comp\\_abstraite::SaveResul](#) \*, list< int > &) const
- virtual void **ListeGrandeurs\_particulieres** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &) const
- virtual [Loi\\_comp\\_abstraite](#) \* **Nouvelle\_loi\_identique** () const =0
- const [PtIntegMecalInterne](#) \* **Ptintmeca\_en\_cours** () const
- void **Signature\_pti\_encours** (ostream &sort) const
- const [Temps\\_CPU\\_HZpp](#) & **Temp\_CPU\_loi** () const

## Fonctions membres publiques statiques

- static void **Affichage\_grandeurs\_Valeur\_multi\_interpoler\_ou\_calculer** ()
- static void **Affichage\_grandeurs\_Valeurs\_Tensorielles\_interpoler\_ou\_calculer** ()
- static void **Affichage\_grandeurs\_Valeurs\_quelconque\_interpoler\_ou\_calculer** ()

## Fonctions membres protégées

- virtual void **Calcul\_SigmaHH** (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB, `TenseurHH` &gijHH, `Tableau`< `TenseurBB` \* > &d\_gijBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)=0
- virtual void **Calcul\_DsigmaHH\_tdt** (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)=0
- virtual void **Calcul\_dsigma\_deps** (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)
- void **Lecture\_type\_deformation\_et\_niveau\_commentaire** (`UtilLecture` &lec, `LesFonctions_nD` &lesFonctionsnD, bool avec\_passage\_nouvelle\_donnee=true)
- void **Affiche\_don\_classe\_abstraite** () const
- void **Info\_commande\_don LoisDeComp** (`UtilLecture` &entreePrinc) const
- void **Lecture\_don\_base\_info** (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void **Ecriture\_don\_base\_info** (ofstream &sort, const int cas) const
- void **Activ\_donnees** (`Tableau`< `Noeud` \* > &tabnoeud, bool dilatation, `LesPtIntegMecalInterne` &lesPtMecalInt)
- virtual void **CalculGrandeurTravail** (const `PtIntegMecalInterne` &ptintmeca, const `Deformation` &Enum\_dure, const `ThermoDonnee` &, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)=0
- void **IndiqueSaveResult** (`SaveResul` \*saveR)
- virtual void **IndiquePtIntegMecalInterne** (const `PtIntegMecalInterne` \*ptintmeca)
- void **IndiqueDef\_en\_cours** (`Deformation` \*def\_en\_cours\_)
- `Tableau`< double > **Valeur\_multi\_interpoler\_ou\_calculer** (bool absolue, `Enum_dure` temps, const `List_io`< `Ddl_enum_etendu` > &enu, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_enum_etendu` > \*exclure\_dd\_etend)
- void **Valeurs\_Tensorielles\_interpoler\_ou\_calculer** (bool absolue, `Enum_dure` temps, `List_io`< `TypeQuelconque` > &enu, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `EnumTypeQuelconque` > \*exclure\_Q)
- void **Valeurs\_quelconque\_interpoler\_ou\_calculer** (bool absolue, `Enum_dure` temps, const `Tableau`< `EnumTypeQuelconque` > &tqi, `List_io`< `TypeQuelconque` > &li\_quelc, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `EnumTypeQuelconque` > \*exclure\_Q)
- `Tableau`< double > & **Loi\_comp\_Valeur\_FnD\_Evoluee** (`Fonction_nD` \*fct, int nb\_retour, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*deja\_calculer\_etend=NULL, const `List_io`< const `TypeQuelconque` \* > \*deja\_calculer\_Q=NULL, list< `SaveResul` \* > \*list\_save=NULL)
- int **Permet\_affichage** ()
- void **Lecture\_permet\_affichage** (`UtilLecture` \*entreePrinc, `LesFonctions_nD` &lesFonctionsnD)
- void **Affiche\_niveau\_affichage** () const
- void **Affiche\_niveau\_affichage** (ofstream &sort, const int cas)
- void **Lecture\_permet\_affichage** (ifstream &ent, const int cas, `LesFonctions_nD` &lesFonctionsnD)
- virtual void **RepercuteChangeTemperature** (`Enum_dure`)
- void **CalculInvariants\_contraintes** (`PtIntegMecalInterne` &ptIntegMeca, `TenseurBB` &gijBB, `TenseurHH` &gijHH)
- void **CalculInvariants\_cinematique** (`PtIntegMecalInterne` &ptIntegMeca, `TenseurBB` &gijBB, `TenseurHH` &gijHH)
- list< `EnumTypeQuelconque` > & **ListdeTouslesQuelc\_dispo\_localement** ()
- list< `EnumTypeQuelconque` > & **ListQuelc\_mis\_en\_acces\_localement** ()
- int **Cal\_permet\_affichage** ()

## Attributs protégés

- [SaveResul](#) \* **saveResul**
- bool **comp\_tangent\_simplifie**
- bool **utilise\_vitesse\_deformation**
- [Enum\\_type\\_deformation](#) **type\_de\_deformation**
- bool **thermo\_dependant**
- double **temperature\_0**
- double **temperature\_t**
- double **temperature\_tdt**
- double \* **temperature**
- bool **dilatation**
- [Deformation](#) \* **def\_en\_cours**
- [Temps\\_CPU\\_HZpp](#) **temps\_loi**
- [TenseurBB](#) \* **epsBB\_totale**
- [TenseurBB](#) \* **epsBB\_therm**
- [TenseurBB](#) \* **DepsBB\_totale**
- [TenseurBB](#) \* **DepsBB\_therm**
- [TenseurBB](#) \* **DepsBB\_umat**

## Amis

- class **LoiAdditiveEnSigma**
- class **LoiDesMelangesEnSigma**
- class **Loi\_Umat**
- class **LoiContraintesPlanes**
- class **LoiContraintesPlanesDouble**
- class **LoiDeformationsPlanes**
- class **LoiCritere**
- class **Projection\_anisotrope\_3D**
- class **ElemMeca**

## 6.459.1 Documentation des fonctions membres

### 6.459.1.1 Calcul\_dsigma\_deps()

```
void Loi_comp_abstraite::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée dans [Hart\\_Smith3D](#), [Hyper\\_externe\\_W](#), [Maheo\\_hyper](#), [MooneyRivlin3D](#), et [Poly\\_hyper3D](#).

### 6.459.1.2 Calcul\_DsigmaHH\_tdt()

```
virtual void Loi_comp_abstraite::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
```



```

DdlElement & tab_ddl,
BaseB & giB_t,
TenseurBB & gijBB_t,
TenseurHH & gijHH_t,
BaseB & giB_tdt,
Tableau< BaseB > & d_giB_tdt,
BaseH & giH_tdt,
Tableau< BaseH > & d_giH_tdt,
TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [pure virtual]

```

Implémenté dans [Prandtl\\_Reuss](#).

### 6.459.1.3 `Modif_comp_tangent_simplifie()`

```

void Loi_comp_abstraite::Modif_comp_tangent_simplifie (
    bool modif ) [inline], [virtual]

```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

### 6.459.1.4 `Test_loi_simplifie()`

```

bool Loi_comp_abstraite::Test_loi_simplifie ( ) [inline], [virtual]

```

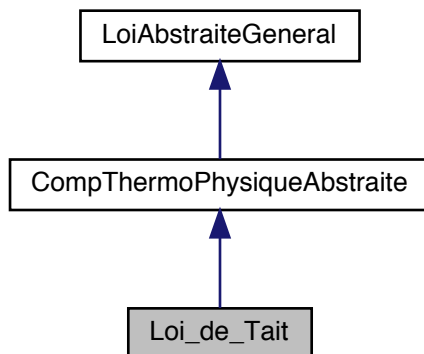
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

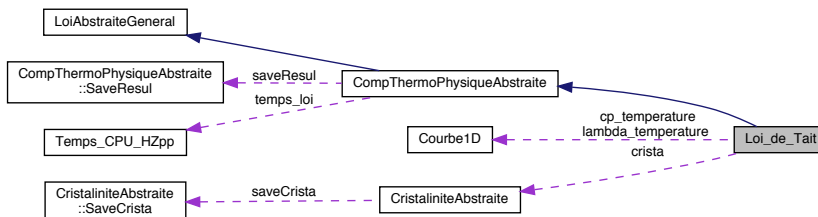
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Loi_comp_abstraite.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Loi_comp_abstraite.cc`

## 6.460 Référence de la classe Loi\_de\_Tait

Graphe d'héritage de Loi\_de\_Tait:



Graphe de collaboration de Loi\_de\_Tait:



### Classes

— class [SaveResul\\_Loi\\_de\\_Tait](#)

### Fonctions membres publiques

- **Loi\_de\_Tait** (const [Loi\\_de\\_Tait](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &les↔  
FonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &les↔  
Courbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- [CompThermoPhysiqueAbstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()
- virtual void [AfficheDataSpecif](#) (ofstream &sort, [SaveResul](#) \*a) const
- virtual void [Grandeur\\_particuliere](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &, [CompThermoPhysiqueAbstraite::SaveResul](#)  
\*, [list](#)< int > &)
- virtual void [ListeGrandeurs\\_particulieres](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &)
- void [Cal\\_donnees\\_thermiques](#) (const double &P\_t, [CompThermoPhysiqueAbstraite::SaveResul](#) \*saveTP,  
const [Deformation](#) &def, const double &P, [Enum\\_dure](#) temps, [ThermoDonnee](#) &donneeThermique)

## Fonctions membres protégées

- void **Calcul\_diff\_valeurs** (const double &pression, double &temp\_trans, double &vol\_spec)
- virtual void **Calcul\_DfluxH\_tdt** (const double &P\_t, [PtIntegThermiInterne](#) &ptIntegThermi, const double &P, [DdlElement](#) &tab\_ddl, const [Deformation](#) &def, [Tableau](#)< [CoordonneeB](#) > &d\_gradTB, [Tableau](#)< [CoordonneeH](#) > &d\_flux, [ThermoDonnee](#) &dTP, [EnergieThermi](#) &energ, const [EnergieThermi](#) &energ\_← t, const [Met\\_abstraite::Impli](#) &ex)

## Attributs protégés

- double **alphaT**
- double **compressibilite**
- double **lambda**
- [Courbe1D](#) \* **lambda\_temperature**
- double **cp**
- [Courbe1D](#) \* **cp\_temperature**
- double **b1s**
- double **b2s**
- double **b3s**
- double **b4s**
- double **b1m**
- double **b2m**
- double **b3m**
- double **b4m**
- double **b5**
- double **b6**
- double **b7**
- double **b8**
- double **b9**
- int **type\_de\_calcul**
- [CristaliniteAbstraite](#) \* **crista**
- bool **sortie\_post**

## 6.460.1 Documentation des fonctions membres

### 6.460.1.1 Affiche()

```
void Loi_de_Tait::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.460.1.2 AfficheDataSpecif()

```
virtual void Loi_de_Tait::AfficheDataSpecif (
    ofstream & sort,
    SaveResul * a ) const [inline], [virtual]
```

Réimplémentée à partir de [CompThermoPhysiqueAbstraite](#).

### 6.460.1.3 Cal\_donnees\_thermiques()

```
void Loi_de_Tait::Cal_donnees_thermiques (
    const double & P_t,
    CompThermoPhysiqueAbstraite::SaveResul * saveTP,
    const Deformation & def,
    const double & P,
    Enum\_dure temps,
    ThermoDonnee & donneeThermique ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite](#).

**6.460.1.4 Calcul\_DfluxH\_tdt()**

```
void Loi_de_Tait::Calcul_DfluxH_tdt (
    const double & P_t,
    PtIntegThermiInterne & ptIntegThermi,
    const double & P,
    DdlElement & tab_ddl,
    const Deformation & def,
    Tableau< CoordonneeB > & d_gradTB,
    Tableau< CoordonneeH > & d_flux,
    ThermoDonnee & dTP,
    EnergieThermi & energ,
    const EnergieThermi & energ_t,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [CompThermoPhysiqueAbstraite](#).

**6.460.1.5 Ecriture\_base\_info\_loi()**

```
void Loi_de_Tait::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.460.1.6 Grandeur\_particuliere()**

```
void Loi_de_Tait::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & litQ,
    CompThermoPhysiqueAbstraite::SaveResul * saveDon,
    list< int > & ) [virtual]
```

Réimplémentée à partir de [CompThermoPhysiqueAbstraite](#).

**6.460.1.7 Info\_commande\_LoisDeComp()**

```
void Loi_de_Tait::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.460.1.8 Lecture\_base\_info\_loi()**

```
void Loi_de_Tait::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.460.1.9 LectureDonneesParticulieres()**

```
void Loi_de_Tait::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.460.1.10 `ListeGrandeurs_particulieres()`

```
void Loi_de_Tait::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) [virtual]
```

Réimplémentée à partir de [CompThermoPhysiqueAbstraite](#).

#### 6.460.1.11 `New_et_Initialise()`

```
Loi_de_Tait::SaveResul * Loi_de_Tait::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [CompThermoPhysiqueAbstraite](#).

#### 6.460.1.12 `Nouvelle_loi_identique()`

```
CompThermoPhysiqueAbstraite * Loi_de_Tait::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [CompThermoPhysiqueAbstraite](#).

#### 6.460.1.13 `TestComplet()`

```
int Loi_de_Tait::TestComplet ( ) [virtual]
```

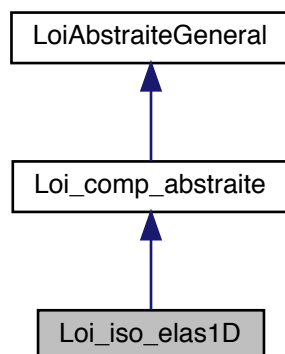
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

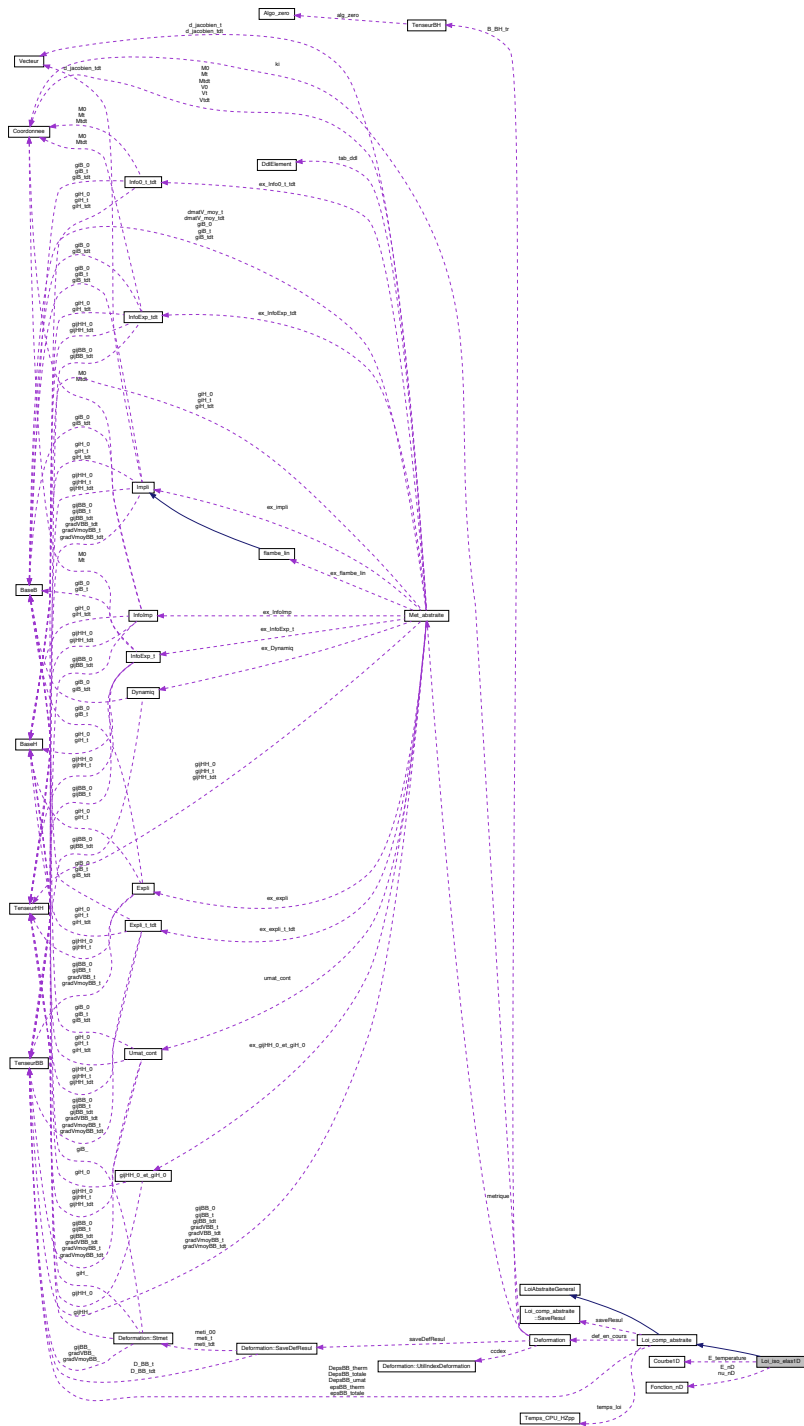
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi\_de\_Tait.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi\_de\_Tait.cc

## 6.461 Référence de la classe `Loi_iso_elas1D`

Grappe d'héritage de `Loi_iso_elas1D`:



Graphe de collaboration de Loi\_iso\_elas1D:



**Classes**

- class [SaveResulLoi\\_iso\\_elas1D](#)

**Fonctions membres publiques**

- **Loi\_iso\_elas1D** (const double &EE, const double &nu)
- **Loi\_iso\_elas1D** (const [Loi\\_iso\\_elas1D](#) &loi)
- **SaveResul \* New\_et\_Initialise** ()

- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &, `Loi_comp_abstraite::SaveResul` \*, list< int > &decal) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &) const
- virtual void `Insertion_conteneur_dans_save_result` (`SaveResul` \*saveResul)
- virtual void `Activation_stockage_grandeurs_quelconques` (list< `EnumTypeQuelconque` > &listEnuQuelc)

## Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &, const `Deformation` &, `Enum_dure`, const `ThermoDonnee` &, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)

## Attributs protégés

- double `E`
- double `nu`
- `Courbe1D` \* `E_temperature`
- `Fonction_nD` \* `E_nD`
- `Fonction_nD` \* `nu_nD`

## Amis

- class `SaveResulLoi_iso_elas1D`

## Membres hérités additionnels

### 6.461.1 Documentation des fonctions membres

**6.461.1.1 Activation\_stockage\_grandeurs\_quelconques()**

```
void Loi_iso_elas1D::Activation_stockage_grandeurs_quelconques (
    list< EnumTypeQuelconque > & listEnuQuelc ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.461.1.2 Affiche()**

```
void Loi_iso_elas1D::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.461.1.3 Calcul\_DsigmaHH\_tdt()**

```
void Loi_iso_elas1D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.461.1.4 Calcul\_SigmaHH()**

```
void Loi_iso_elas1D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
```



```

TenseurHH & gijHH_,
Tableau< TenseurBB * > & d_gijBB_,
double & jacobien_0,
double & jacobien,
TenseurHH & sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.461.1.5 CalculGrandeurTravail()

```

virtual void Loi_iso_elas1D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.461.1.6 Ecriture\_base\_info\_loi()

```

void Loi_iso_elas1D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.461.1.7 Grandeur\_particuliere()

```

void Loi_iso_elas1D::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.461.1.8 HsurH0()

```

virtual double Loi_iso_elas1D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.461.1.9 Info\_commande\_LoisDeComp()

```

void Loi_iso_elas1D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.461.1.10 Insertion\_conteneur\_dans\_save\_result()

```
void Loi_iso_elas1D::Insertion_conteneur_dans_save_result (
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.461.1.11 Lecture\_base\_info\_loi()

```
void Loi_iso_elas1D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.461.1.12 LectureDonneesParticulieres()

```
void Loi_iso_elas1D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.461.1.13 ListeGrandeurs\_particulieres()

```
void Loi_iso_elas1D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.461.1.14 Module\_compressibilite\_equivalent()

```
double Loi_iso_elas1D::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.461.1.15 Module\_young\_equivalent()

```
double Loi_iso_elas1D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.461.1.16 New\_et\_Initialise()

```
SaveResul * Loi_iso_elas1D::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.461.1.17 `Nouvelle_loi_identique()`

```
Loi_comp_abstraite * Loi_iso_elas1D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.461.1.18 `TestComplet()`

```
int Loi_iso_elas1D::TestComplet ( ) [virtual]
```

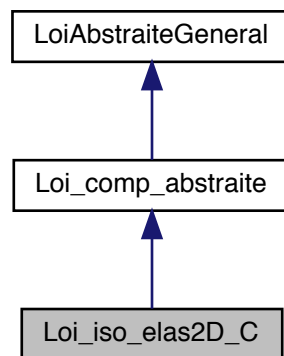
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_hooke/Loi_iso_elas1D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_hooke/Loi_iso_elas1D.cc`

## 6.462 Référence de la classe `Loi_iso_elas2D_C`

Graphe d'héritage de `Loi_iso_elas2D_C`:





- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &`lesCourbes1D`, `LesFonctions_nD` &`lesFonctionsnD`)
- void `Affiche` () const
- int `TestComple` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &, `SaveResul` \*saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ, `Loi_comp_abstraite::SaveResul` \*saveDon, list< int > &decal) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ) const
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &`lesCourbes1D`, `LesFonctions_nD` &`lesFonctionsnD`)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande LoisDeComp` (`UtilLecture` &lec)
- virtual void `Insertion_conteneur_dans_save_result` (`SaveResul` \*saveResul)
- virtual void `Activation_stockage_grandeurs_quelconques` (list< `EnumTypeQuelconque` > &listEnuQuelc)
- virtual double `Eps33BH` (`SaveResul` \*saveResul) const
- virtual bool `Contraintes_planes_de_3D` () const

### Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB\_tdt, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &, const `Deformation` &, `Enum_dure`, const `ThermoDonnee` &, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)

### Attributs protégés

- double `E`
- double `nu`
- `Courbe1D` \* `E_temperature`
- `Fonction_nD` \* `E_nD`
- `Fonction_nD` \* `nu_nD`
- short int `cas_calcul`

### Amis

- class `SaveResul_Loi_iso_elas2D_C`

### Membres hérités additionnels

#### 6.462.1 Documentation des fonctions membres

**6.462.1.1 Activation\_stockage\_grandeurs\_quelconques()**

```
void Loi_iso_elas2D_C::Activation_stockage_grandeurs_quelconques (
    list< EnumTypeQuelconque > & listEnuQuelc ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.462.1.2 Affiche()**

```
void Loi_iso_elas2D_C::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.462.1.3 Calcul\_DsigmaHH\_tdt()**

```
void Loi_iso_elas2D_C::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.462.1.4 Calcul\_SigmaHH()**

```
void Loi_iso_elas2D_C::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
```

```

TenseurHH & gijHH_,
Tableau< TenseurBB * > & d_gijBB_,
double & jacobien_0,
double & jacobien,
TenseurHH & sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.462.1.5 CalculGrandeurTravail()

```

virtual void Loi_iso_elas2D_C::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_imple,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.462.1.6 Contraintes\_planes\_de\_3D()

```

virtual bool Loi_iso_elas2D_C::Contraintes_planes_de_3D ( ) const [inline], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.462.1.7 Ecriture\_base\_info\_loi()

```

void Loi_iso_elas2D_C::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.462.1.8 Eps33BH()

```

virtual double Loi_iso_elas2D_C::Eps33BH (
    SaveResul * saveResul ) const [inline], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.462.1.9 Grandeur\_particuliere()

```

void Loi_iso_elas2D_C::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.462.1.10 HsurH0()

```
double Loi_iso_elas2D_C::HsurH0 (
    SaveResul * saveResul ) const [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.462.1.11 Info\_commande\_LoisDeComp()

```
void Loi_iso_elas2D_C::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.462.1.12 Insertion\_conteneur\_dans\_save\_result()

```
void Loi_iso_elas2D_C::Insertion_conteneur_dans_save_result (
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.462.1.13 Lecture\_base\_info\_loi()

```
void Loi_iso_elas2D_C::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.462.1.14 LectureDonneesParticulieres()

```
void Loi_iso_elas2D_C::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.462.1.15 ListeGrandeurs\_particulieres()

```
void Loi_iso_elas2D_C::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.462.1.16 Module\_compressibilite\_equivalent()

```
double Loi_iso_elas2D_C::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).



### 6.462.1.17 Module\_young\_equivalent()

```
double Loi_iso_elas2D_C::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.462.1.18 New\_et\_Initialise()

```
SaveResul * Loi_iso_elas2D_C::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.462.1.19 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Loi_iso_elas2D_C::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.462.1.20 TestComplet()

```
int Loi_iso_elas2D_C::TestComplet ( ) [virtual]
```

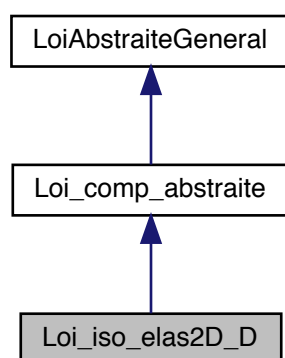
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas2D\_C.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas2D\_C.cc

## 6.463 Référence de la classe Loi\_iso\_elas2D\_D

Graphe d'héritage de Loi\_iso\_elas2D\_D:





- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &, `SaveResul` \*saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &, `Loi_comp_abstraite::SaveResul` \*, `list`< int > &decal) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &) const
- virtual void `Insertion_conteneur_dans_save_result` (`SaveResul` \*saveResul)
- virtual void `Activation_stockage_grandeurs_quelconques` (`list`< `EnumTypeQuelconque` > &listEnuQuelc)
- double `Sig33BH` (`TenseurBB` &epsBB, `TenseurHH` &gijHH)

## Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &, const `Deformation` &, `Enum_dure`, const `ThermoDonnee` &, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)

## Attributs protégés

- double `E`
- double `nu`
- `Courbe1D` \* `E_temperature`
- `Fonction_nD` \* `E_nD`
- `Fonction_nD` \* `nu_nD`
- short int `cas_calcul`

## Amis

- class `SaveResulLoi_iso_elas2D_D`

## Membres hérités additionnels

### 6.463.1 Documentation des fonctions membres

**6.463.1.1 Activation\_stockage\_grandeurs\_quelconques()**

```
void Loi_iso_elas2D_D::Activation_stockage_grandeurs_quelconques (
    list< EnumTypeQuelconque > & listEnuQuelc ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.463.1.2 Affiche()**

```
void Loi_iso_elas2D_D::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.463.1.3 Calcul\_DsigmaHH\_tdt()**

```
void Loi_iso_elas2D_D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.463.1.4 Calcul\_SigmaHH()**

```
void Loi_iso_elas2D_D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
```

```

TenseurHH & gijHH_,
Tableau< TenseurBB * > & d_gijBB_,
double & jacobien_0,
double & jacobien,
TenseurHH & sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.463.1.5 CalculGrandeurTravail()

```

virtual void Loi_iso_elas2D_D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.463.1.6 Ecriture\_base\_info\_loi()

```

void Loi_iso_elas2D_D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.463.1.7 Grandeur\_particuliere()

```

void Loi_iso_elas2D_D::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.463.1.8 HsurH0()

```

virtual double Loi_iso_elas2D_D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.463.1.9 Info\_commande\_LoisDeComp()

```

void Loi_iso_elas2D_D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.463.1.10 Insertion\_conteneur\_dans\_save\_result()

```
void Loi_iso_elas2D_D::Insertion_conteneur_dans_save_result (
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.463.1.11 Lecture\_base\_info\_loi()

```
void Loi_iso_elas2D_D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.463.1.12 LectureDonneesParticulieres()

```
void Loi_iso_elas2D_D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.463.1.13 ListeGrandeurs\_particulieres()

```
void Loi_iso_elas2D_D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.463.1.14 Module\_compressibilite\_equivalent()

```
double Loi_iso_elas2D_D::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.463.1.15 Module\_young\_equivalent()

```
double Loi_iso_elas2D_D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.463.1.16 New\_et\_Initialise()

```
SaveResul * Loi_iso_elas2D_D::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.463.1.17 Nouvelle\_loi\_identique()

`Loi_comp_abstraite * Loi_iso_elas2D_D::Nouvelle_loi_identique ( ) const [inline], [virtual]`  
Implémente [Loi\\_comp\\_abstraite](#).

### 6.463.1.18 TestComplet()

`int Loi_iso_elas2D_D::TestComplet ( ) [virtual]`

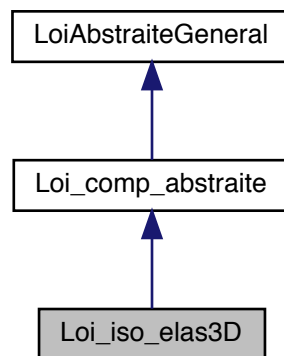
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

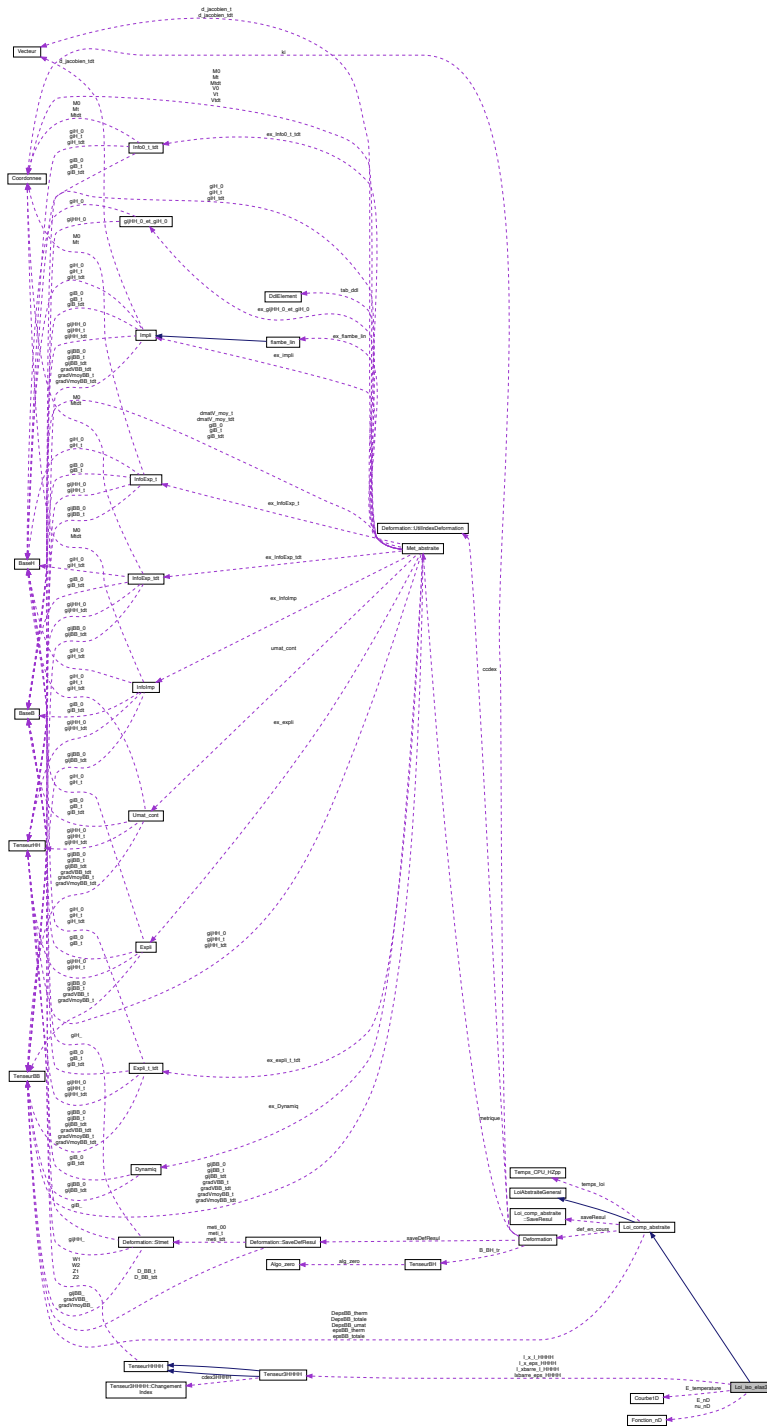
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas2D\_D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas2D\_D.cc

## 6.464 Référence de la classe Loi\_iso\_elas3D

Graphe d'héritage de Loi\_iso\_elas3D:



Graphe de collaboration de Loi\_iso\_elas3D:



### Classes

- class [SaveResulLoi\\_iso\\_elas3D](#)

### Fonctions membres publiques

- [Loi\\_iso\\_elas3D](#) (const double &EE, const double &nu)
- [Loi\\_iso\\_elas3D](#) (const [Loi\\_iso\\_elas3D](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()



- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &, `SaveResul` \*saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &, `Loi_comp_abstraite::SaveResul` \*, list< int > &decal) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &) const
- virtual void `Insertion_conteneur_dans_save_result` (`SaveResul` \*saveResul)
- virtual void `Activation_stockage_grandeurs_quelconques` (list< `EnumTypeQuelconque` > &listEnuQuelc)

### Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &, const `Deformation` &, `Enum_dure`, const `ThermoDonnee` &, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)

### Attributs protégés

- double `E`
- double `nu`
- `Courbe1D` \* `E_temperature`
- `Fonction_nD` \* `E_nD`
- `Fonction_nD` \* `nu_nD`
- short int `cas_calcul`
- `Tenseur3HHHH` `I_x_I_HHHH`
- `Tenseur3HHHH` `I_xbarre_I_HHHH`
- `Tenseur3HHHH` `I_x_eps_HHHH`
- `Tenseur3HHHH` `Ixbarre_eps_HHHH`

### Amis

- class `SaveResulLoi_iso_elas3D`

## Membres hérités additionnels

### 6.464.1 Documentation des fonctions membres

#### 6.464.1.1 Activation\_stockage\_grandeurs\_quelconques()

```
void Loi_iso_elas3D::Activation_stockage_grandeurs_quelconques (
    list< EnumTypeQuelconque > & listEnuQuelc ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.464.1.2 Affiche()

```
void Loi_iso_elas3D::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.464.1.3 Calcul\_dsigma\_deps()

```
void Loi_iso_elas3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.464.1.4 Calcul\_DsigmaHH\_tdt()

```
void Loi_iso_elas3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
```

```

double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.464.1.5 Calcul\_SigmaHH()

```

void Loi_iso_elas3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.464.1.6 CalculGrandeurTravail()

```

virtual void Loi_iso_elas3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.464.1.7 Ecriture\_base\_info\_loi()

```

void Loi_iso_elas3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.464.1.8 Grandeur\_particuliere()

```
void Loi_iso_elas3D::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.464.1.9 HsurH0()

```
virtual double Loi_iso_elas3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.464.1.10 Info\_commande\_LoisDeComp()

```
void Loi_iso_elas3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.464.1.11 Insertion\_conteneur\_dans\_save\_result()

```
void Loi_iso_elas3D::Insertion_conteneur_dans_save_result (
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.464.1.12 Lecture\_base\_info\_loi()

```
void Loi_iso_elas3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.464.1.13 LectureDonneesParticulieres()

```
void Loi_iso_elas3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.464.1.14 ListeGrandeurs\_particulieres()

```
void Loi_iso_elas3D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.464.1.15 `Module_compressibilite_equivalent()`

```
double Loi_iso_elas3D::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.464.1.16 `Module_young_equivalent()`

```
double Loi_iso_elas3D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.464.1.17 `New_et_Initialise()`

```
SaveResul * Loi_iso_elas3D::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.464.1.18 `Nouvelle_loi_identique()`

```
Loi_comp_abstraite * Loi_iso_elas3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.464.1.19 `TestComplet()`

```
int Loi_iso_elas3D::TestComplet ( ) [virtual]
```

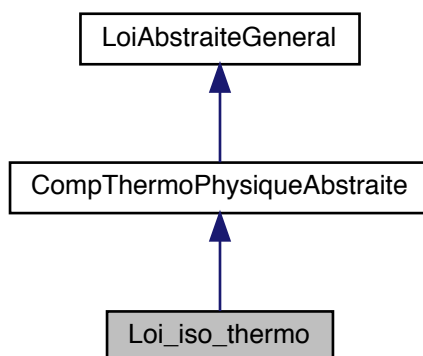
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

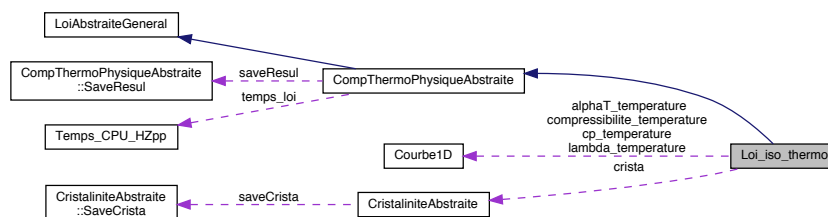
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_hooke/Loi_iso_elas3D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso_elas_hooke/Loi_iso_elas3D.cc`

## 6.465 Référence de la classe Loi\_iso\_thermo

Graphe d'héritage de Loi\_iso\_thermo:



Graphe de collaboration de Loi\_iso\_thermo:



### Classes

- class [SaveResul\\_Loi\\_iso\\_thermo](#)

### Fonctions membres publiques

- **Loi\_iso\_thermo** (const [Loi\\_iso\\_thermo](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- [CompThermoPhysiqueAbstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()
- virtual void [AfficheDataSpecif](#) (ofstream &sort, [SaveResul](#) \*a) const
- virtual void [Grandeur\\_particuliere](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &, [CompThermoPhysiqueAbstraite::SaveResul](#) \*, [list](#)< int > &)
- virtual void [ListeGrandeurs\\_particulieres](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &)
- void [Cal\\_donnees\\_thermiques](#) (const double &P\_t, [CompThermoPhysiqueAbstraite::SaveResul](#) \*saveTP, const [Deformation](#) &def, const double &P, [Enum\\_dure](#) temps, [ThermoDonnee](#) &donneeThermique)

## Fonctions membres protégées

- virtual void `Calcul_DfluxH_tdt` (const double &P\_t, `PtIntegThermiInterne` &ptIntegThermi, const double &P, `DdlElement` &tab\_ddl, const `Deformation` &def, `Tableau`< `CoordonneeB` > &d\_gradTB, `Tableau`< `CoordonneeH` > &d\_flux, `ThermoDonnee` &dTP, `EnergieThermi` &energ, const `EnergieThermi` &energ\_↔ t, const `Met_abstraite::Impli` &ex)

## Attributs protégés

- double `alphaT`
- `Courbe1D` \* `alphaT_temperature`
- double `compressibilite`
- `Courbe1D` \* `compressibilite_temperature`
- double `lambda`
- `Courbe1D` \* `lambda_temperature`
- double `cp`
- `Courbe1D` \* `cp_temperature`
- int `type_de_calcul`
- `CristaliniteAbstraite` \* `crista`
- bool `sortie_post`

## 6.465.1 Documentation des fonctions membres

### 6.465.1.1 Affiche()

void `Loi_iso_thermo::Affiche` ( ) const [virtual]  
 Implémente `LoiAbstraiteGeneral`.

### 6.465.1.2 AfficheDataSpecif()

```
virtual void Loi_iso_thermo::AfficheDataSpecif (
    ofstream & sort,
    SaveResul * a ) const [inline], [virtual]
```

Réimplémentée à partir de `CompThermoPhysiqueAbstraite`.

### 6.465.1.3 Cal\_donnees\_thermiques()

```
void Loi_iso_thermo::Cal_donnees_thermiques (
    const double & P_t,
    CompThermoPhysiqueAbstraite::SaveResul * saveTP,
    const Deformation & def,
    const double & P,
    Enum_dure temps,
    ThermoDonnee & donneeThermique ) [virtual]
```

Implémente `CompThermoPhysiqueAbstraite`.

### 6.465.1.4 Calcul\_DfluxH\_tdt()

```
void Loi_iso_thermo::Calcul_DfluxH_tdt (
    const double & P_t,
    PtIntegThermiInterne & ptIntegThermi,
    const double & P,
    DdlElement & tab_ddl,
    const Deformation & def,
    Tableau< CoordonneeB > & d_gradTB,
```

```

    Tableau< CoordonneeH > & d_flux,
    ThermoDonnee & dTP,
    EnergieThermi & energ,
    const EnergieThermi & energ_t,
    const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [CompThermoPhysiqueAbstraite](#).

#### 6.465.1.5 Ecriture\_base\_info\_loi()

```

void Loi_iso_thermo::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.465.1.6 Grandeur\_particuliere()

```

void Loi_iso_thermo::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    CompThermoPhysiqueAbstraite::SaveResul * saveDon,
    list< int > & ) [virtual]

```

Réimplémentée à partir de [CompThermoPhysiqueAbstraite](#).

#### 6.465.1.7 Info\_commande\_LoisDeComp()

```

void Loi_iso_thermo::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.465.1.8 Lecture\_base\_info\_loi()

```

void Loi_iso_thermo::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.465.1.9 LectureDonneesParticulieres()

```

void Loi_iso_thermo::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.465.1.10 ListeGrandeurs\_particulieres()

```

void Loi_iso_thermo::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) [virtual]

```

Réimplémentée à partir de [CompThermoPhysiqueAbstraite](#).



**6.465.1.11 `New_et_Initialise()`**

```
Loi_iso_thermo::SaveResul * Loi_iso_thermo::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [CompThermoPhysiqueAbstraite](#).

**6.465.1.12 `Nouvelle_loi_identique()`**

```
CompThermoPhysiqueAbstraite * Loi_iso_thermo::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [CompThermoPhysiqueAbstraite](#).

**6.465.1.13 `TestComplet()`**

```
int Loi_iso_thermo::TestComplet ( ) [virtual]
```

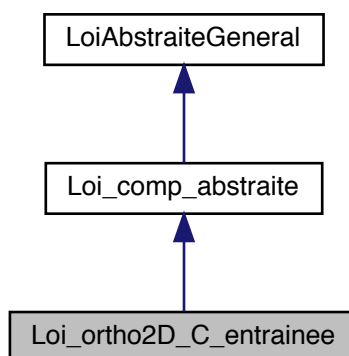
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

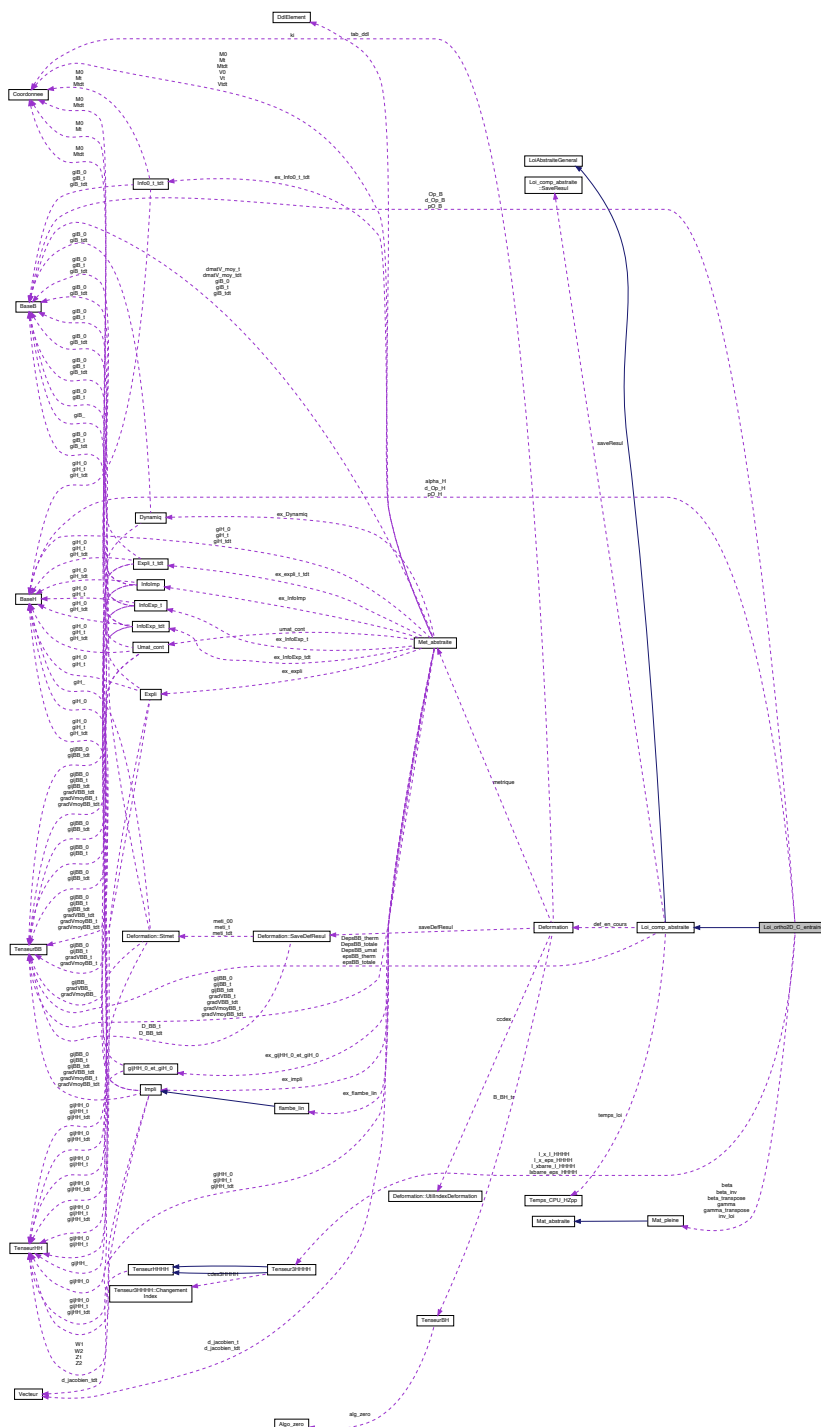
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi_iso_thermo.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi_iso_thermo.cc`

**6.466 Référence de la classe `Loi_ortho2D_C_entrainee`**

Grappe d'héritage de `Loi_ortho2D_C_entrainee`:



Graphe de collaboration de Loi\_ortho2D\_C\_entrainee:



**Classes**

- class [SaveResulLoi\\_ortho2D\\_C\\_entrainee](#)

**Fonctions membres publiques**

- **Loi\_ortho2D\_C\_entrainee** (const double &EE1, const double &EE2, const double &EE3, const double &nu12, const double &nu13, const double &nu23, const double &GG12, const string &nom\_rep)
- **Loi\_ortho2D\_C\_entrainee** (const [Loi\\_ortho2D\\_C\\_entrainee](#) &loi)

- `SaveResul * New_et_Initialise ()`
- `void LectureDonneesParticulieres (UtilLecture *, LesCourbes1D &lesCourbes1D, LesFonctions_nD &lesFonctionsnD)`
- `void Affiche () const`
- `int TestComplet ()`
- `double Module_young_equivalent (Enum_dure temps, const Deformation &, SaveResul *saveResul)`
- `double Module_compressibilite_equivalent (Enum_dure temps, const Deformation &def, SaveResul *saveResul)`
- `virtual double HsurH0 (SaveResul *saveResul) const`
- `Loi_comp_abstraite * Nouvelle_loi_identique () const`
- `void Lecture_base_info_loi (ifstream &ent, const int cas, LesReferences &lesRef, LesCourbes1D &lesCourbes1D, LesFonctions_nD &lesFonctionsnD)`
- `void Ecriture_base_info_loi (ofstream &sort, const int cas)`
- `void Info_commande_LoisDeComp (UtilLecture &iec)`
- `virtual double Eps33BH (SaveResul *saveResul) const`
- `virtual bool Contraintes_planes_de_3D () const`
- `virtual void Grandeur_particuliere (bool absolue, List_io< TypeQuelconque > &, Loi_comp_abstraite::SaveResul *, list< int > &decal) const`
- `virtual void ListeGrandeurs_particulieres (bool absolue, List_io< TypeQuelconque > &) const`
- `const string & NomRepere () const`
- `const int & Type_transport () const`

### Fonctions membres protégées

- `void Calcul_SigmaHH (TenseurHH &sigHH_t, TenseurBB &DepsBB, DdlElement &tab_ddl, TenseurBB &gijBB_t, TenseurHH &gijHH_t, BaseB &giB, BaseH &gi_H, TenseurBB &epsBB_, TenseurBB &delta_epsBB_, TenseurBB &gijBB_, TenseurHH &gijHH_, Tableau< TenseurBB * > &d_gijBB_, double &jacobien_0, double &jacobien, TenseurHH &sigHH, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Expli_t_tdt &ex)`
- `void Calcul_DsigmaHH_tdt (TenseurHH &sigHH_t, TenseurBB &DepsBB, DdlElement &tab_ddl, BaseB &giB_t, TenseurBB &gijBB_t, TenseurHH &gijHH_t, BaseB &giB_tdt, Tableau< BaseB > &d_giB_tdt, BaseH &giH_tdt, Tableau< BaseH > &d_giH_tdt, TenseurBB &epsBB_tdt, Tableau< TenseurBB * > &d_epsBB_, TenseurBB &delta_epsBB, TenseurBB &gijBB_tdt, TenseurHH &gijHH_tdt, Tableau< TenseurBB * > &d_gijBB_tdt, Tableau< TenseurHH * > &d_gijHH_tdt, double &jacobien_0, double &jacobien, Vecteur &d_jacobien_tdt, TenseurHH &sigHH, Tableau< TenseurHH * > &d_sigHH, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Impli &ex)`
- `void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH &sigHH_t, TenseurBB &DepsBB, TenseurBB &epsBB_tdt, TenseurBB &delta_epsBB, double &jacobien_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d_sigma_deps, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Umat_cont &ex)`
- `bool Verif_convexite ()`
- `virtual void CalculGrandeurTravail (const PtIntegMecalInterne &, const Deformation &, Enum_dure, const ThermoDonnee &, const Met_abstraite::Impli *ex_impli, const Met_abstraite::Expli_t_tdt *ex_expli_tdt, const Met_abstraite::Umat_cont *ex_umat, const List_io< Ddl_etendu > *exclure_dd_etend, const List_io< const TypeQuelconque * > *exclure_Q)`

### Attributs protégés

- `double E1`
- `double E2`
- `double E3`
- `double nu12`
- `double nu13`
- `double nu23`
- `double G12`
- `Tableau< Fonction_nD * > fct_para`
- `bool null_fct_para`
- `string nom_repere`
- `int cas_calcul`
- `int verification_convexite`
- `Mat_pleine inv_loi`

- int `type_transport`
- `BaseB` `Op_B`
- `BaseB` `d_Op_B`
- `BaseB` `pO_B`
- `BaseH` `d_Op_H`
- `BaseH` `pO_H`
- `BaseH` `alpha_H`
- `Mat_pleine` `beta`
- `Mat_pleine` `gamma`
- `Mat_pleine` `beta_transpose`
- `Mat_pleine` `gamma_transpose`
- `Mat_pleine` `beta_inv`
- int `sortie_post`
- `Tenseur3HHHH` `I_x_I_HHHH`
- `Tenseur3HHHH` `I_xbarre_I_HHHH`
- `Tenseur3HHHH` `I_x_eps_HHHH`
- `Tenseur3HHHH` `Ixbarre_eps_HHHH`

## Amis

- class `SaveResulLoi_ortho2D_C_entrainee`

## Membres hérités additionnels

### 6.466.1 Documentation des fonctions membres

#### 6.466.1.1 Affiche()

`void Loi_ortho2D_C_entrainee::Affiche ( ) const [virtual]`  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.466.1.2 Calcul\_dsigma\_deps()

```
void Loi_ortho2D_C_entrainee::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.466.1.3 Calcul\_DsigmaHH\_tdt()

```
void Loi_ortho2D_C_entrainee::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
```

```

TenseurBB & gijBB_t,
TenseurHH & gijHH_t,
BaseB & giB_tdt,
Tableau< BaseB > & d_giB_tdt,
BaseH & giH_tdt,
Tableau< BaseH > & d_giH_tdt,
TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.466.1.4 Calcul\_SigmaHH()

```

void Loi_ortho2D_C_entraine::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.466.1.5 CalculGrandeurTravail()

```

virtual void Loi_ortho2D_C_entraine::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,

```

```

const Met_abstraite::Impli * ex_impli,
const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
const Met_abstraite::Umat_cont * ex_umat,
const List_io< Ddl_etendu > * exclure_dd_etend,
const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],

```

[virtual]

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.466.1.6 Contraintes\_planes\_de\_3D()

```
virtual bool Loi_ortho2D_C_entraine::Contraintes_planes_de_3D ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.466.1.7 Ecriture\_base\_info\_loi()

```
void Loi_ortho2D_C_entraine::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.466.1.8 Eps33BH()

```
virtual double Loi_ortho2D_C_entraine::Eps33BH (
    SaveResul * saveResul ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.466.1.9 Grandeur\_particuliere()

```
void Loi_ortho2D_C_entraine::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.466.1.10 HsurH0()

```
virtual double Loi_ortho2D_C_entraine::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.466.1.11 Info\_commande\_LoisDeComp()

```
void Loi_ortho2D_C_entraine::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.466.1.12 Lecture\_base\_info\_loi()

```
void Loi_ortho2D_C_entraine::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
```

```
    LesReferences & lesRef,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.466.1.13 `LectureDonneesParticulieres()`

```
void Loi_ortho2D_C_entrainee::LectureDonneesParticulieres (  
    UtilLecture * entreePrinc,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.466.1.14 `ListeGrandeurs_particulieres()`

```
void Loi_ortho2D_C_entrainee::ListeGrandeurs_particulieres (  
    bool absolue,  
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.466.1.15 `Module_compressibilite_equivalent()`

```
double Loi_ortho2D_C_entrainee::Module_compressibilite_equivalent (  
    Enum_dure temps,  
    const Deformation & def,  
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.466.1.16 `Module_young_equivalent()`

```
double Loi_ortho2D_C_entrainee::Module_young_equivalent (  
    Enum_dure temps,  
    const Deformation & def,  
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.466.1.17 `New_et_Initialise()`

```
SaveResul * Loi_ortho2D_C_entrainee::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.466.1.18 `Nouvelle_loi_identique()`

```
Loi_comp_abstraite * Loi_ortho2D_C_entrainee::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.466.1.19 `TestComplet()`

```
int Loi_ortho2D_C_entrainee::TestComplet ( ) [virtual]
```

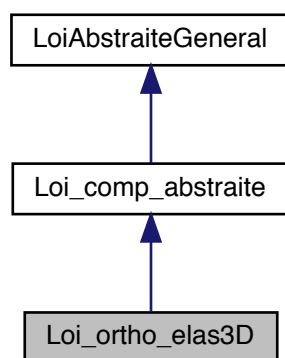
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi\_ortho2D\_C\_↔  
entrainee.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi\_ortho2D\_C\_↔  
entrainee.cc

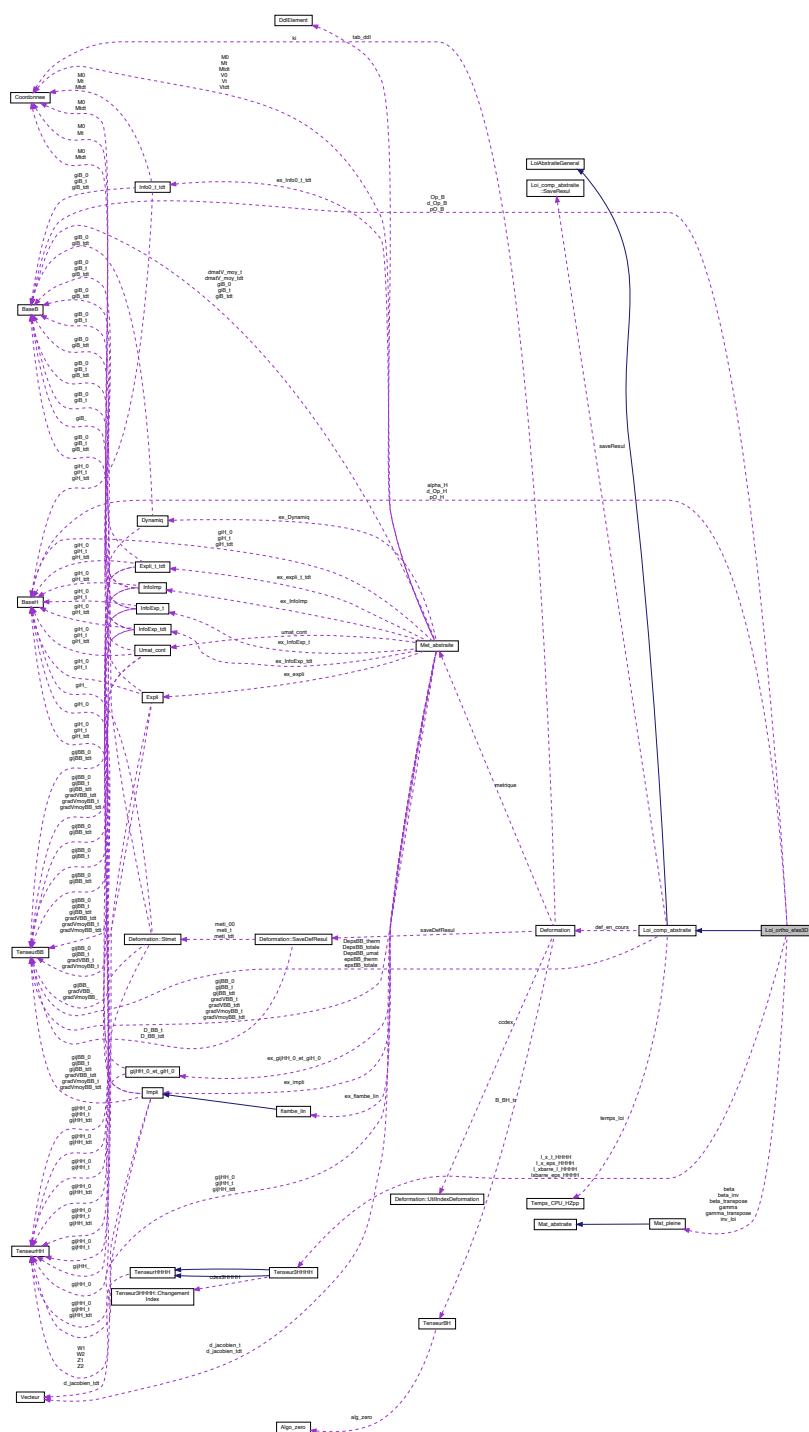
## 6.467 Référence de la classe Loi\_ortho\_elas3D

Grphe d'héritage de Loi\_ortho\_elas3D:





Graphe de collaboration de Loi\_ortho\_elas3D:



## Classes

- class [SaveResultLoi\\_ortho\\_elas3D](#)

## Fonctions membres publiques

- **Loi\_ortho\_elas3D** (const double &EE1, const double &EE2, const double &EE3, const double &nu12, const double &nu13, const double &nu23, const double &GG12, const double &GG13, const double &GG23, const string &nom\_rep)

- **Loi\_ortho\_elas3D** (const **Loi\_ortho\_elas3D** &loi)
- **SaveResul \* New\_et\_Initialise** ()
- void **LectureDonneesParticulieres** (**UtilLecture \***, **LesCourbes1D** &lesCourbes1D, **LesFonctions\_nD** &lesFonctionsnD)
- void **Affiche** () const
- int **TestComple** ()
- double **Module\_young\_equivalent** (**Enum\_dure** temps, const **Deformation** &, **SaveResul \*saveResul**)
- double **Module\_compressibilite\_equivalent** (**Enum\_dure** temps, const **Deformation** &def, **SaveResul \*saveResul**)
- virtual double **HsurH0** (**SaveResul \*saveResul**) const
- **Loi\_comp\_abstraite \* Nouvelle\_loi\_identique** () const
- void **Lecture\_base\_info\_loi** (ifstream &ent, const int cas, **LesReferences** &lesRef, **LesCourbes1D** &lesCourbes1D, **LesFonctions\_nD** &lesFonctionsnD)
- void **Ecriture\_base\_info\_loi** (ofstream &sort, const int cas)
- void **Info\_commande\_LoisDeComp** (**UtilLecture** &lec)
- virtual void **Grandeur\_particuliere** (bool absolue, **List\_io**< **TypeQuelconque** > &, **Loi\_comp\_abstraite::SaveResul \***, list< int > &decal) const
- virtual void **ListeGrandeurs\_particulieres** (bool absolue, **List\_io**< **TypeQuelconque** > &) const
- const string & **NomRepere** () const
- const int & **Type\_transport** () const

### Fonctions membres protégées

- void **Calcul\_SigmaHH** (**TenseurHH** &sigHH\_t, **TenseurBB** &DepsBB, **DdlElement** &tab\_ddl, **TenseurBB** &gijBB\_t, **TenseurHH** &gijHH\_t, **BaseB** &giB, **BaseH** &gi\_H, **TenseurBB** &epsBB\_, **TenseurBB** &delta\_epsBB\_, **TenseurBB** &gijBB\_, **TenseurHH** &gijHH\_, **Tableau**< **TenseurBB \*** > &d\_gijBB\_, double &jacobien\_0, double &jacobien, **TenseurHH** &sigHH, **EnergieMeca** &energ, const **EnergieMeca** &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const **Met\_abstraite::Expli\_t\_tdt** &ex)
- void **Calcul\_DsigmaHH\_tdt** (**TenseurHH** &sigHH\_t, **TenseurBB** &DepsBB, **DdlElement** &tab\_ddl, **BaseB** &giB\_t, **TenseurBB** &gijBB\_t, **TenseurHH** &gijHH\_t, **BaseB** &giB\_tdt, **Tableau**< **BaseB** > &d\_giB\_tdt, **BaseH** &giH\_tdt, **Tableau**< **BaseH** > &d\_giH\_tdt, **TenseurBB** &epsBB\_tdt, **Tableau**< **TenseurBB \*** > &d\_epsBB, **TenseurBB** &delta\_epsBB, **TenseurBB** &gijBB\_tdt, **TenseurHH** &gijHH\_tdt, **Tableau**< **TenseurBB \*** > &d\_gijBB\_tdt, **Tableau**< **TenseurHH \*** > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, **Vecteur** &d\_jacobien\_tdt, **TenseurHH** &sigHH, **Tableau**< **TenseurHH \*** > &d\_sigHH, **EnergieMeca** &energ, const **EnergieMeca** &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const **Met\_abstraite::Impli** &ex)
- void **Calcul\_dsigma\_deps** (bool en\_base\_orthonormee, **TenseurHH** &sigHH\_t, **TenseurBB** &DepsBB, **TenseurBB** &epsBB\_tdt, **TenseurBB** &delta\_epsBB, double &jacobien\_0, double &jacobien, **TenseurHH** &sigHH, **TenseurHHHH** &d\_sigma\_deps, **EnergieMeca** &energ, const **EnergieMeca** &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const **Met\_abstraite::Umat\_cont** &ex)
- bool **Verif\_convexite** ()
- virtual void **CalculGrandeurTravail** (const **PtIntegMecalInterne** &, const **Deformation** &, **Enum\_dure**, const **ThermoDonnee** &, const **Met\_abstraite::Impli \*ex\_impli**, const **Met\_abstraite::Expli\_t\_tdt \*ex\_expli\_tdt**, const **Met\_abstraite::Umat\_cont \*ex\_umat**, const **List\_io**< **Ddl\_etendu** > \*exclure\_dd\_etend, const **List\_io**< const **TypeQuelconque \*** > \*exclure\_Q)

### Attributs protégés

- double **E1**
- double **E2**
- double **E3**
- double **nu12**
- double **nu13**
- double **nu23**
- double **G12**
- double **G13**
- double **G23**
- **Tableau**< **Fonction\_nD \*** > **fct\_para**
- bool **null\_fct\_para**
- string **nom\_repere**
- int **cas\_calcul**
- double **ratio\_inf\_module\_compressibilite**

- int `verification_convexite`
- `Mat_pleine` `inv_loi`
- int `type_transport`
- `BaseB` `Op_B`
- `BaseB` `d_Op_B`
- `BaseB` `pO_B`
- `BaseH` `d_Op_H`
- `BaseH` `pO_H`
- `BaseH` `alpha_H`
- `Mat_pleine` `beta`
- `Mat_pleine` `gamma`
- `Mat_pleine` `beta_transpose`
- `Mat_pleine` `gamma_transpose`
- `Mat_pleine` `beta_inv`
- int `sortie_post`
- `Tenseur3HHHH` `I_x_I_HHHH`
- `Tenseur3HHHH` `I_xbarre_I_HHHH`
- `Tenseur3HHHH` `I_x_eps_HHHH`
- `Tenseur3HHHH` `Ixbarre_eps_HHHH`

## Amis

- class `SaveResulLoi_ortho_elas3D`

## Membres hérités additionnels

### 6.467.1 Documentation des fonctions membres

#### 6.467.1.1 Affiche()

```
void Loi_ortho_elas3D::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.467.1.2 Calcul\_dsigma\_deps()

```
void Loi_ortho_elas3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.467.1.3 Calcul\_DsigmaHH\_tdt()

```
void Loi_ortho_elas3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
```

```

DdlElement & tab_ddl,
BaseB & giB_t,
TenseurBB & gijBB_t,
TenseurHH & gijHH_t,
BaseB & giB_tdt,
Tableau< BaseB > & d_giB_tdt,
BaseH & giH_tdt,
Tableau< BaseH > & d_giH_tdt,
TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.467.1.4 Calcul\_SigmaHH()

```

void Loi_ortho_elas3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.467.1.5 CalculGrandeurTravail()

```

virtual void Loi_ortho_elas3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,

```

```

Enum_dure ,
const ThermoDonnee & ,
const Met_abstraite::Impli * ex_impli,
const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
const Met_abstraite::Umat_cont * ex_umat,
const List_io< Ddl_etendu > * exclure_dd_etend,
const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],

```

[virtual]

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.467.1.6 Ecriture\_base\_info\_loi()

```

void Loi_ortho_elas3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.467.1.7 Grandeur\_particuliere()

```

void Loi_ortho_elas3D::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.467.1.8 HsurH0()

```

virtual double Loi_ortho_elas3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.467.1.9 Info\_commande\_LoisDeComp()

```

void Loi_ortho_elas3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.467.1.10 Lecture\_base\_info\_loi()

```

void Loi_ortho_elas3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.467.1.11 LectureDonneesParticulieres()

```

void Loi_ortho_elas3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,

```

```

    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.467.1.12 ListeGrandeurs\_particulieres()

```

void Loi_ortho_elas3D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.467.1.13 Module\_compressibilite\_equivalent()

```

double Loi_ortho_elas3D::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.467.1.14 Module\_young\_equivalent()

```

double Loi_ortho_elas3D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.467.1.15 New\_et\_Initialise()

```

SaveResul * Loi_ortho_elas3D::New_et_Initialise ( ) [inline], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.467.1.16 Nouvelle\_loi\_identique()

```

Loi_comp_abstraite * Loi_ortho_elas3D::Nouvelle_loi_identique ( ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.467.1.17 TestComplet()

```

int Loi_ortho_elas3D::TestComplet ( ) [virtual]

```

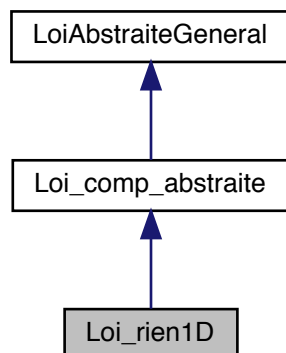
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

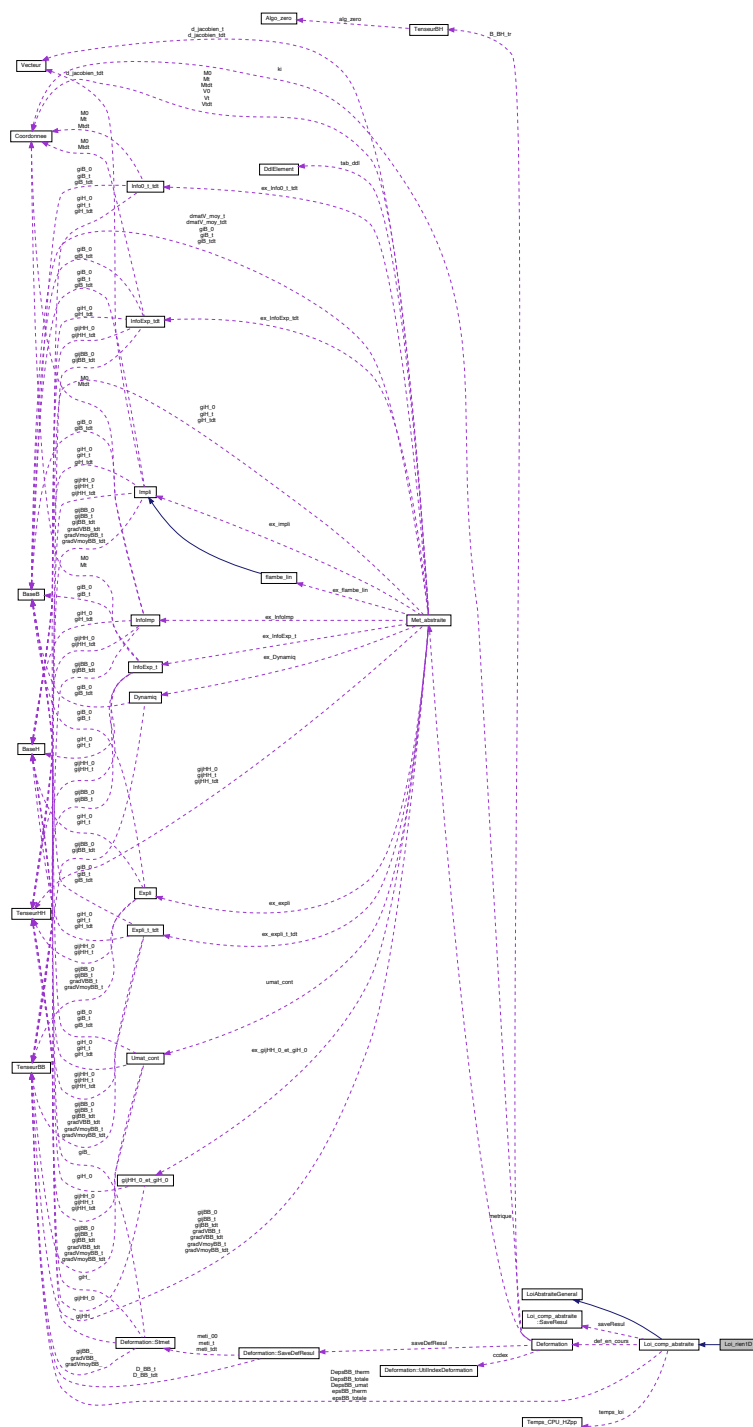
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi\_ortho3D\_entrainee.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi\_ortho3D\_entrainee.cc

## 6.468 Référence de la classe Loi\_rien1D

Graphe d'héritage de Loi\_rien1D:



Graphe de collaboration de Loi\_rien1D:



## Fonctions membres publiques

- **Loi\_rien1D** (const [Loi\\_rien1D](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)



- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure, const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [BaseH](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, double &, double &, [TenseurHH](#) &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) &)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [BaseB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [Tableau](#)< [BaseB](#) > &, [BaseH](#) &, [Tableau](#)< [BaseH](#) > &, [TenseurBB](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [Tableau](#)< [TenseurHH](#) \* > &, double &, double &, [Vecteur](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurHH](#) \* > &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Impli](#) &)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite](#)::[Impli](#) \*ex\_impli, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite](#)::[Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)

## Membres hérités additionnels

### 6.468.1 Documentation des fonctions membres

#### 6.468.1.1 Affiche()

void [Loi\\_rien1D::Affiche](#) ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.468.1.2 Calcul\_DsigmaHH\_tdt()

```
void Loi\_rien1D::Calcul\_DsigmaHH\_tdt (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    BaseB & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    Tableau< BaseB > & ,
    BaseH & ,
    Tableau< BaseH > & ,
    TenseurBB & ,
    Tableau< TenseurBB * > & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    Tableau< TenseurHH * > & ,
    double & ,
    double & ,
    Vecteur & ,
    TenseurHH & ,
    Tableau< TenseurHH * > & ,
    EnergieMeca & ,
```

```

    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Impli & ) [inline], [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.468.1.3 Calcul\_SigmaHH()

```

void Loi_rien1D::Calcul_SigmaHH (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    BaseH & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    double & ,
    double & ,
    TenseurHH & ,
    EnergieMeca & ,
    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Expli_t_tdt & ) [inline], [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.468.1.4 CalculGrandeurTravail()

```

virtual void Loi_rien1D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.468.1.5 Ecriture\_base\_info\_loi()

```

void Loi_rien1D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

### 6.468.1.6 `HsurH0()`

```
virtual double Loi_rien1D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.468.1.7 `Info_commande_LoisDeComp()`

```
void Loi_rien1D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.468.1.8 `Lecture_base_info_loi()`

```
void Loi_rien1D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.468.1.9 `LectureDonneesParticulieres()`

```
void Loi_rien1D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.468.1.10 `Module_young_equivalent()`

```
double Loi_rien1D::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.468.1.11 `Nouvelle_loi_identique()`

```
Loi_comp_abstraite * Loi_rien1D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.468.1.12 `TestComplet()`

```
int Loi_rien1D::TestComplet ( ) [virtual]
```

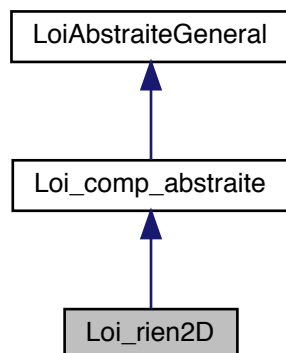
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

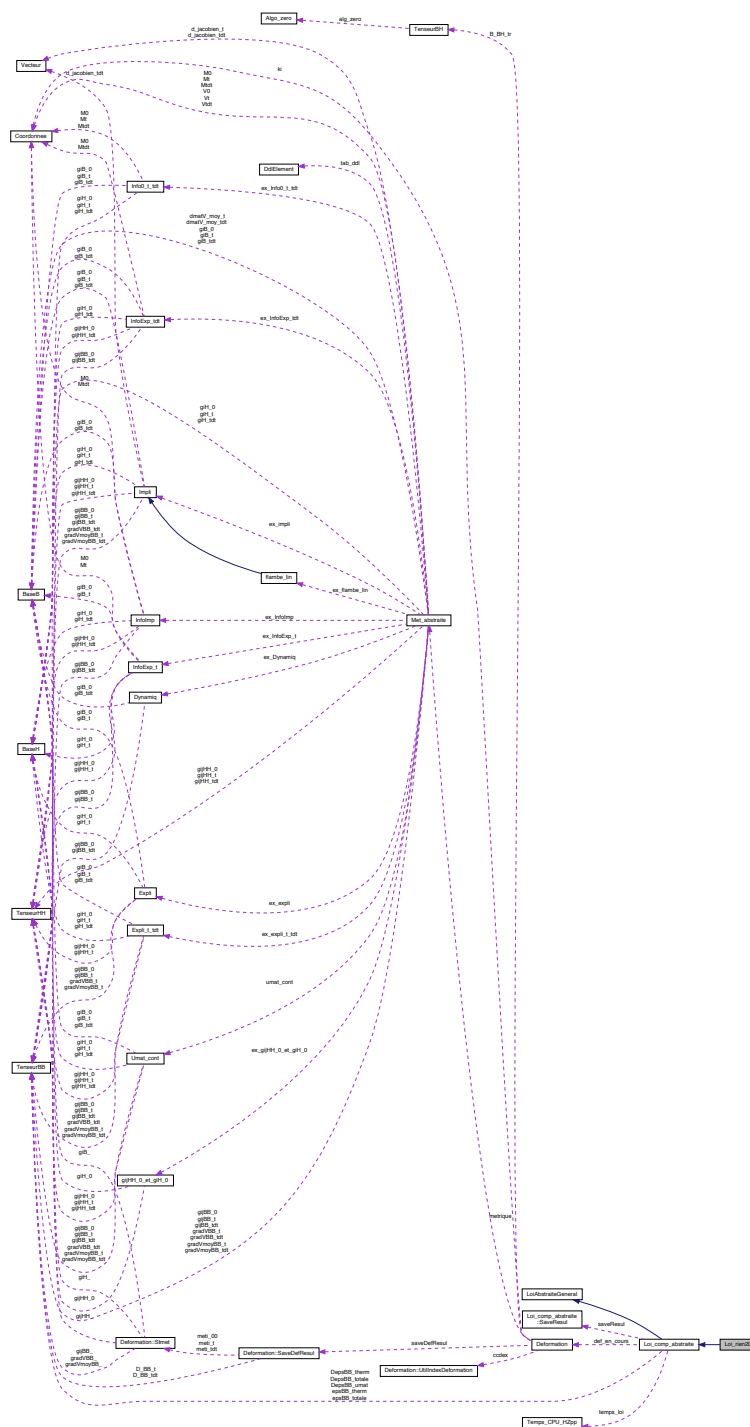
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien1D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien1D.cc`

## 6.469 Référence de la classe Loi\_rien2D

Graphe d'héritage de Loi\_rien2D:



Graphe de collaboration de `Loi_rien2D`:



## Fonctions membres publiques

- `Loi_rien2D` (const `Loi_rien2D` &loi)
- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D)
- void `Affiche` () const
- int `TestComple` ()
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &les←  
Courbes1D)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)

- double **Module\_young\_equivalent** (const [Deformation](#) &)
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [BaseH](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, double &, double &, [TenseurHH](#) &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [BaseB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [Tableau](#)< [BaseB](#) > &, [BaseH](#) &, [Tableau](#)< [BaseH](#) > &, [TenseurBB](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [Tableau](#)< [TenseurHH](#) \* > &, double &, double &, [Vecteur](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurHH](#) \* > &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite::Impli](#) &)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &)

## Membres hérités additionnels

### 6.469.1 Documentation des fonctions membres

#### 6.469.1.1 Affiche()

void [Loi\\_rien2D::Affiche](#) ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.469.1.2 Calcul\_DsigmaHH\_tdt()

```
void Loi\_rien2D::Calcul\_DsigmaHH\_tdt (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    BaseB & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    Tableau< BaseB > & ,
    BaseH & ,
    Tableau< BaseH > & ,
    TenseurBB & ,
    Tableau< TenseurBB * > & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    Tableau< TenseurHH * > & ,
    double & ,
    double & ,
    Vecteur & ,
    TenseurHH & ,
    Tableau< TenseurHH * > & ,
    EnergieMeca & ,
    const EnergieMeca & ,
    double & ,
```

```
double & ,
const Met_abstraite::Impli & ) [inline], [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.469.1.3 Calcul\_SigmaHH()

```
void Loi_rien2D::Calcul_SigmaHH (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    BaseH & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    double & ,
    double & ,
    TenseurHH & ,
    EnergieMeca & ,
    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Expli_t_tdt & ) [inline], [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.469.1.4 Ecriture\_base\_info\_loi()

```
void Loi_rien2D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.469.1.5 Info\_commande\_LoisDeComp()

```
void Loi_rien2D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.469.1.6 Nouvelle\_loi\_identique()

```
Loi\_comp\_abstraite * Loi_rien2D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.469.1.7 TestComplet()

```
int Loi_rien2D::TestComplet ( ) [virtual]
```

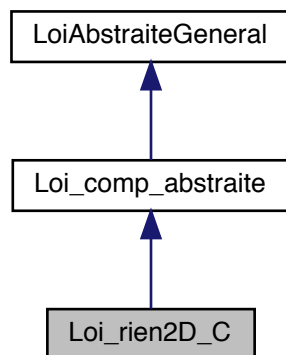
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_speciales/Loi\_rien2D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_speciales/Loi\_rien2D.cc

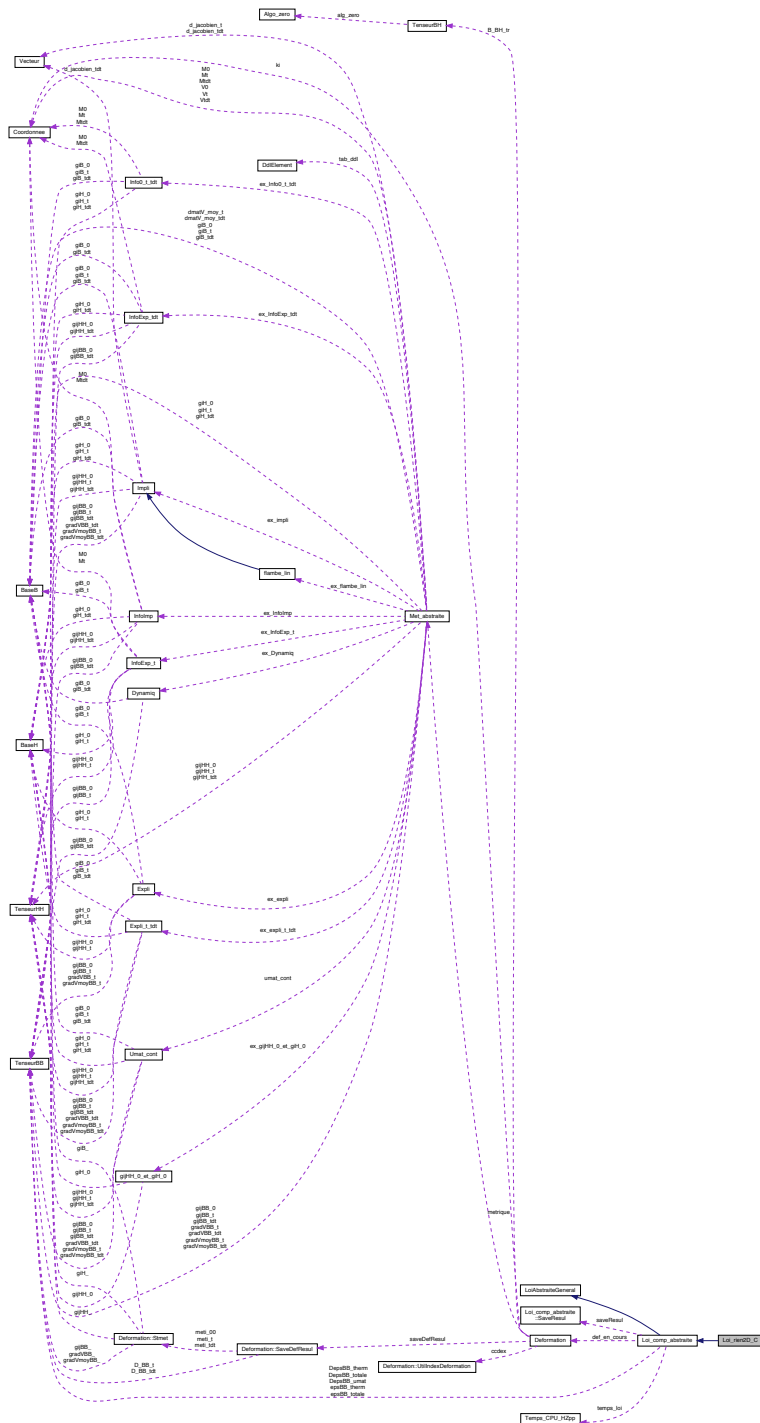
## 6.470 Référence de la classe Loi\_rien2D\_C

Graphe d'héritage de Loi\_rien2D\_C:





Graphe de collaboration de `Loi_rien2D_C`:



### Fonctions membres publiques

- `Loi_rien2D_C` (const `Loi_rien2D_C &loi`)
- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D &lesCourbes1D`, `LesFonctions_nD &lesFonctionsnD`)
- void `Affiche` () const
- int `TestComple` ()
- void `Lecture_base_info_loi` (`ifstream &ent`, const int cas, `LesReferences &lesRef`, `LesCourbes1D &lesCourbes1D`, `LesFonctions_nD &lesFonctionsnD`)

- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure, const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [BaseH](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, double &, double &, [TenseurHH](#) &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) &)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [BaseB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [Tableau](#)< [BaseB](#) > &, [BaseH](#) &, [Tableau](#)< [BaseH](#) > &, [TenseurBB](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [Tableau](#)< [TenseurHH](#) \* > &, double &, double &, [Vecteur](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurHH](#) \* > &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Impli](#) &)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite](#)::[Impli](#) \*ex\_impli, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite](#)::[Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)

## Membres hérités additionnels

### 6.470.1 Documentation des fonctions membres

#### 6.470.1.1 Affiche()

void [Loi\\_rien2D\\_C::Affiche](#) ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.470.1.2 Calcul\_DsigmaHH\_tdt()

```
void Loi\_rien2D\_C::Calcul\_DsigmaHH\_tdt (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    BaseB & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    Tableau< BaseB > & ,
    BaseH & ,
    Tableau< BaseH > & ,
    TenseurBB & ,
    Tableau< TenseurBB * > & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    Tableau< TenseurHH * > & ,
    double & ,
    double & ,
    Vecteur & ,
    TenseurHH & ,
    Tableau< TenseurHH * > & ,
    EnergieMeca & ,
```

```

    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Impli & ) [inline], [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.470.1.3 Calcul\_SigmaHH()

```

void Loi_rien2D_C::Calcul_SigmaHH (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    BaseH & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    double & ,
    double & ,
    TenseurHH & ,
    EnergieMeca & ,
    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Expli_t_tdt & ) [inline], [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.470.1.4 CalculGrandeurTravail()

```

virtual void Loi_rien2D_C::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.470.1.5 Ecriture\_base\_info\_loi()

```

void Loi_rien2D_C::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

### 6.470.1.6 HsurH0()

```
virtual double Loi_rien2D_C::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.470.1.7 Info\_commande\_LoisDeComp()

```
void Loi_rien2D_C::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.470.1.8 Lecture\_base\_info\_loi()

```
void Loi_rien2D_C::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.470.1.9 LectureDonneesParticulieres()

```
void Loi_rien2D_C::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.470.1.10 Module\_young\_equivalent()

```
double Loi_rien2D_C::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.470.1.11 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Loi_rien2D_C::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.470.1.12 TestComplet()

```
int Loi_rien2D_C::TestComplet ( ) [virtual]
```

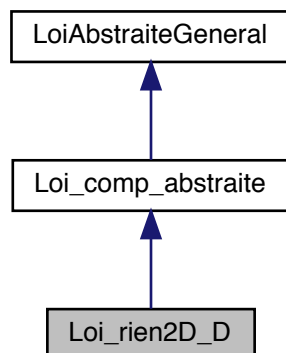
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

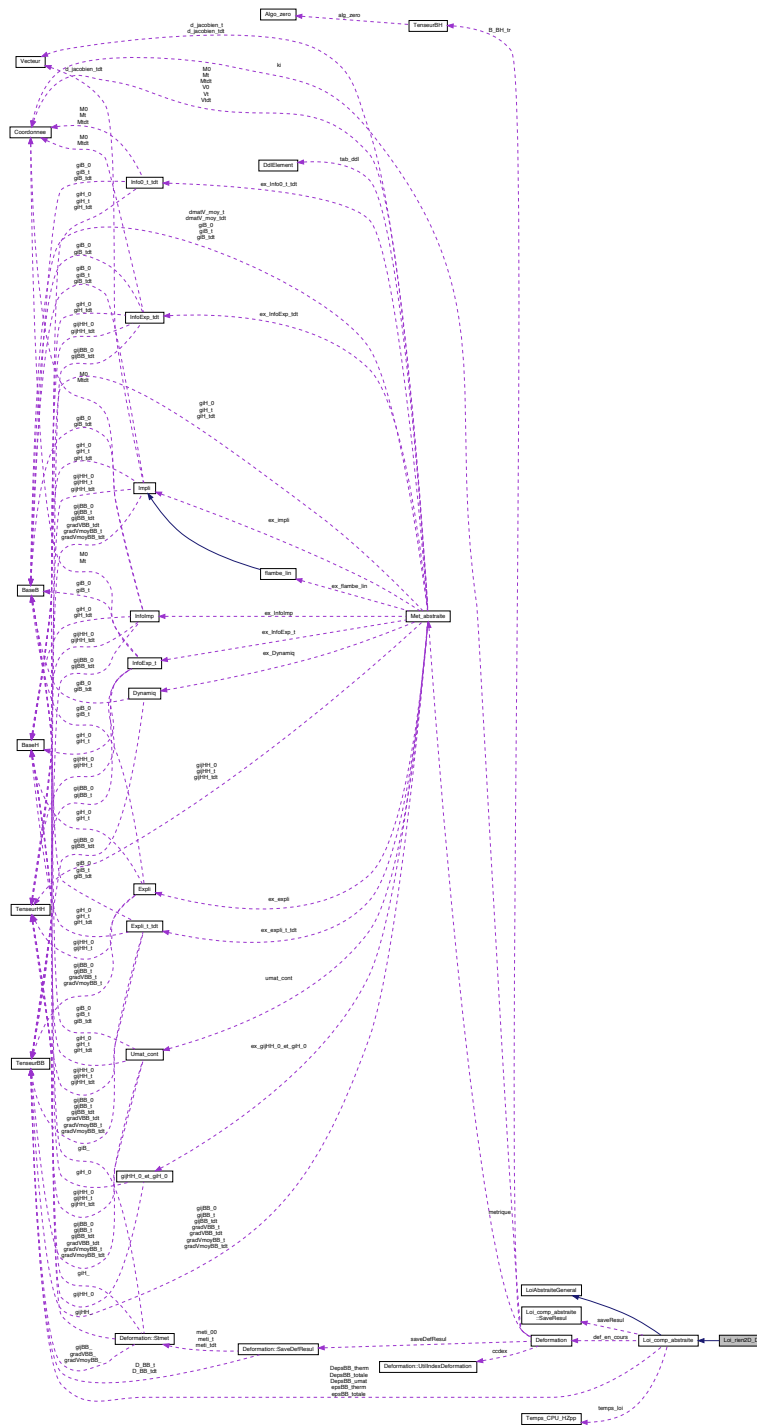
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_speciales/Loi\_rien2D\_C.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_speciales/Loi\_rien2D\_C.cc

## 6.471 Référence de la classe Loi\_rien2D\_D

Graphe d'héritage de Loi\_rien2D\_D:



Graphe de collaboration de Loi\_rien2D\_D:



### Fonctions membres publiques

- Loi\_rien2D\_D (const Loi\_rien2D\_D &loi)
- void LectureDonneesParticulieres (UtilLecture \*, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)
- void Affiche () const
- int TestComple ()
- void Lecture\_base\_info\_loi (ifstream &ent, const int cas, LesReferences &lesRef, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)

- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure, const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [BaseH](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, double &, double &, [TenseurHH](#) &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) &)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [BaseB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [Tableau](#)< [BaseB](#) > &, [BaseH](#) &, [Tableau](#)< [BaseH](#) > &, [TenseurBB](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [Tableau](#)< [TenseurHH](#) \* > &, double &, double &, [Vecteur](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurHH](#) \* > &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Impli](#) &)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite](#)::[Impli](#) \*ex\_impli, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite](#)::[Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)

## Membres hérités additionnels

### 6.471.1 Documentation des fonctions membres

#### 6.471.1.1 Affiche()

void [Loi\\_rien2D\\_D::Affiche](#) ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.471.1.2 Calcul\_DsigmaHH\_tdt()

```
void Loi\_rien2D\_D::Calcul\_DsigmaHH\_tdt (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    BaseB & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    Tableau< BaseB > & ,
    BaseH & ,
    Tableau< BaseH > & ,
    TenseurBB & ,
    Tableau< TenseurBB * > & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    Tableau< TenseurHH * > & ,
    double & ,
    double & ,
    Vecteur & ,
    TenseurHH & ,
    Tableau< TenseurHH * > & ,
    EnergieMeca & ,
```

```

    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Impli & ) [inline], [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.471.1.3 Calcul\_SigmaHH()

```

void Loi_rien2D_D::Calcul_SigmaHH (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    BaseH & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    double & ,
    double & ,
    TenseurHH & ,
    EnergieMeca & ,
    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Expli_t_tdt & ) [inline], [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.471.1.4 CalculGrandeurTravail()

```

virtual void Loi_rien2D_D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.471.1.5 Ecriture\_base\_info\_loi()

```

void Loi_rien2D_D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).



### 6.471.1.6 `HsurH0()`

```
virtual double Loi_rien2D_D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.471.1.7 `Info_commande_LoisDeComp()`

```
void Loi_rien2D_D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.471.1.8 `Lecture_base_info_loi()`

```
void Loi_rien2D_D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.471.1.9 `LectureDonneesParticulieres()`

```
void Loi_rien2D_D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.471.1.10 `Module_young_equivalent()`

```
double Loi_rien2D_D::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.471.1.11 `Nouvelle_loi_identique()`

```
Loi_comp_abstraite * Loi_rien2D_D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.471.1.12 `TestComplet()`

```
int Loi_rien2D_D::TestComplet ( ) [virtual]
```

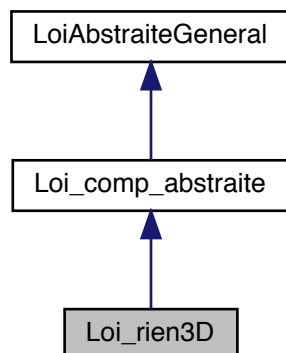
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

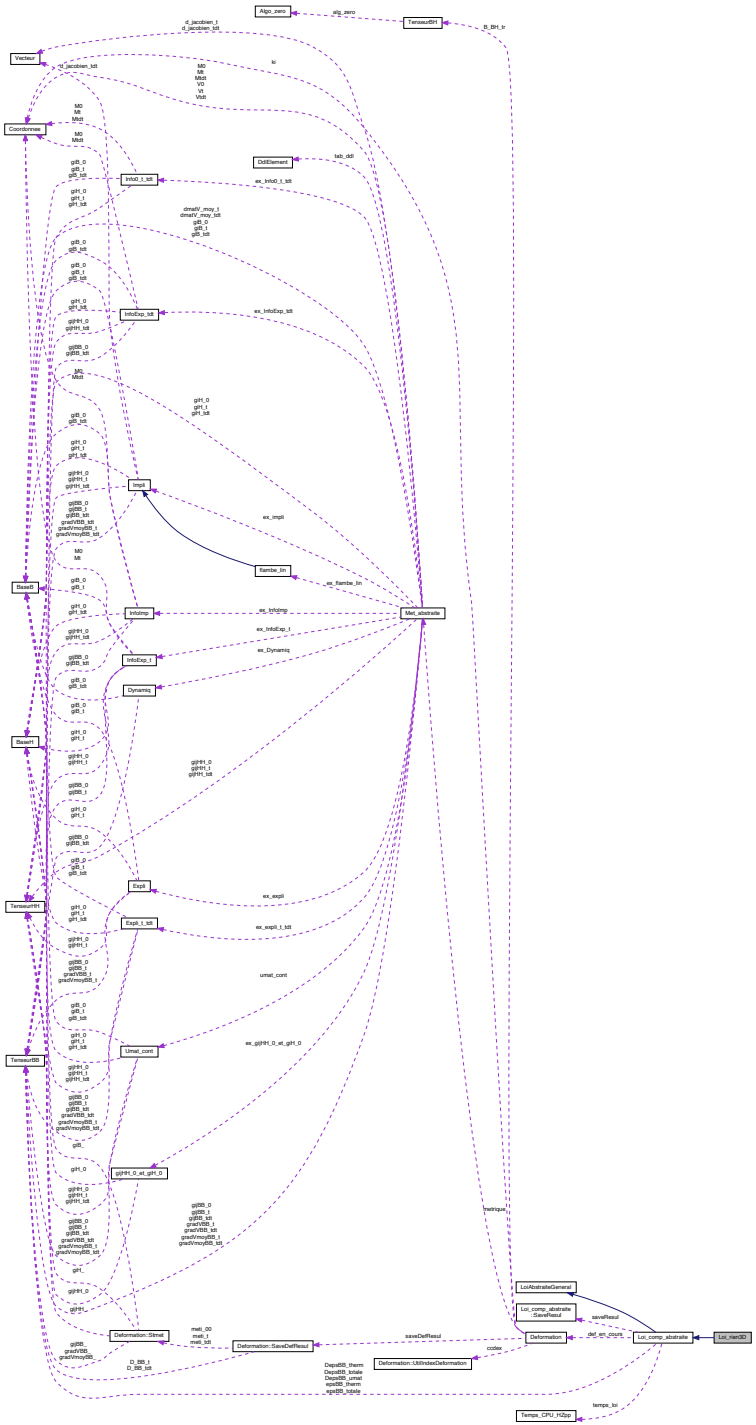
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien2D_D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois_speciales/Loi_rien2D_D.cc`

## 6.472 Référence de la classe Loi\_rien3D

Graphe d'héritage de Loi\_rien3D:



Graphe de collaboration de Loi\_rien3D:



**Fonctions membres publiques**

- Loi\_rien3D (const Loi\_rien3D &loi)
- void LectureDonneesParticulières (UtilLecture \*, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)
- void Affiche () const
- int TestComple ( )
- void Lecture\_base\_info\_loi (ifstream &ent, const int cas, LesReferences &lesRef, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)

- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure, const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [BaseH](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, double &, double &, [TenseurHH](#) &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) &)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &, [TenseurBB](#) &, [DdlElement](#) &, [BaseB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [BaseB](#) &, [Tableau](#)< [BaseB](#) > &, [BaseH](#) &, [Tableau](#)< [BaseH](#) > &, [TenseurBB](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurBB](#) \* > &, [Tableau](#)< [TenseurHH](#) \* > &, double &, double &, [Vecteur](#) &, [TenseurHH](#) &, [Tableau](#)< [TenseurHH](#) \* > &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Impli](#) &)
- virtual void [Calcul\\_dsigma\\_deps](#) (bool en\_base\_orthonormee, [TenseurHH](#) &, [TenseurBB](#) &, [TenseurBB](#) &, [TenseurBB](#) &, double &, double &, [TenseurHH](#) &, [TenseurHHHH](#) &, [EnergieMeca](#) &, const [EnergieMeca](#) &, double &, double &, const [Met\\_abstraite](#)::[Umat\\_cont](#) &)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite](#)::[Impli](#) \*ex\_impli, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite](#)::[Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)

## Membres hérités additionnels

### 6.472.1 Documentation des fonctions membres

#### 6.472.1.1 Affiche()

```
void Loi_rien3D::Affiche ( ) const [virtual]
Implémente LoiAbstraiteGeneral.
```

#### 6.472.1.2 Calcul\_dsigma\_deps()

```
virtual void Loi_rien3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurBB & ,
    double & ,
    double & ,
    TenseurHH & ,
    TenseurHHHH & ,
    EnergieMeca & ,
    const EnergieMeca & ,
    double & ,
    double & ,
    const Met\_abstraite::Umat\_cont & ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.472.1.3 Calcul\_DsigmaHH\_tdt()**

```

void Loi_rien3D::Calcul_DsigmaHH_tdt (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    BaseB & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    Tableau< BaseB > & ,
    BaseH & ,
    Tableau< BaseH > & ,
    TenseurBB & ,
    Tableau< TenseurBB * > & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    Tableau< TenseurHH * > & ,
    double & ,
    double & ,
    Vecteur & ,
    TenseurHH & ,
    Tableau< TenseurHH * > & ,
    EnergieMeca & ,
    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Impli & ) [inline], [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

**6.472.1.4 Calcul\_SigmaHH()**

```

void Loi_rien3D::Calcul_SigmaHH (
    TenseurHH & ,
    TenseurBB & ,
    DdlElement & ,
    TenseurBB & ,
    TenseurHH & ,
    BaseB & ,
    BaseH & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurBB & ,
    TenseurHH & ,
    Tableau< TenseurBB * > & ,
    double & ,
    double & ,
    TenseurHH & ,
    EnergieMeca & ,
    const EnergieMeca & ,
    double & ,
    double & ,
    const Met_abstraite::Expli_t_tdt & ) [inline], [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

**6.472.1.5 CalculGrandeurTravail()**

```
virtual void Loi_rien3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_imple,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.472.1.6 Ecriture\_base\_info\_loi()**

```
void Loi_rien3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.472.1.7 HsurH0()**

```
virtual double Loi_rien3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.472.1.8 Info\_commande\_LoisDeComp()**

```
void Loi_rien3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.472.1.9 Lecture\_base\_info\_loi()**

```
void Loi_rien3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.472.1.10 LectureDonneesParticulieres()**

```
void Loi_rien3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.472.1.11 Module\_young\_equivalent()

```
double Loi_rien3D::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.472.1.12 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Loi_rien3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.472.1.13 TestComplet()

```
int Loi_rien3D::TestComplet ( ) [virtual]
```

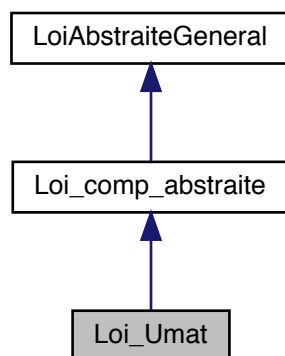
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

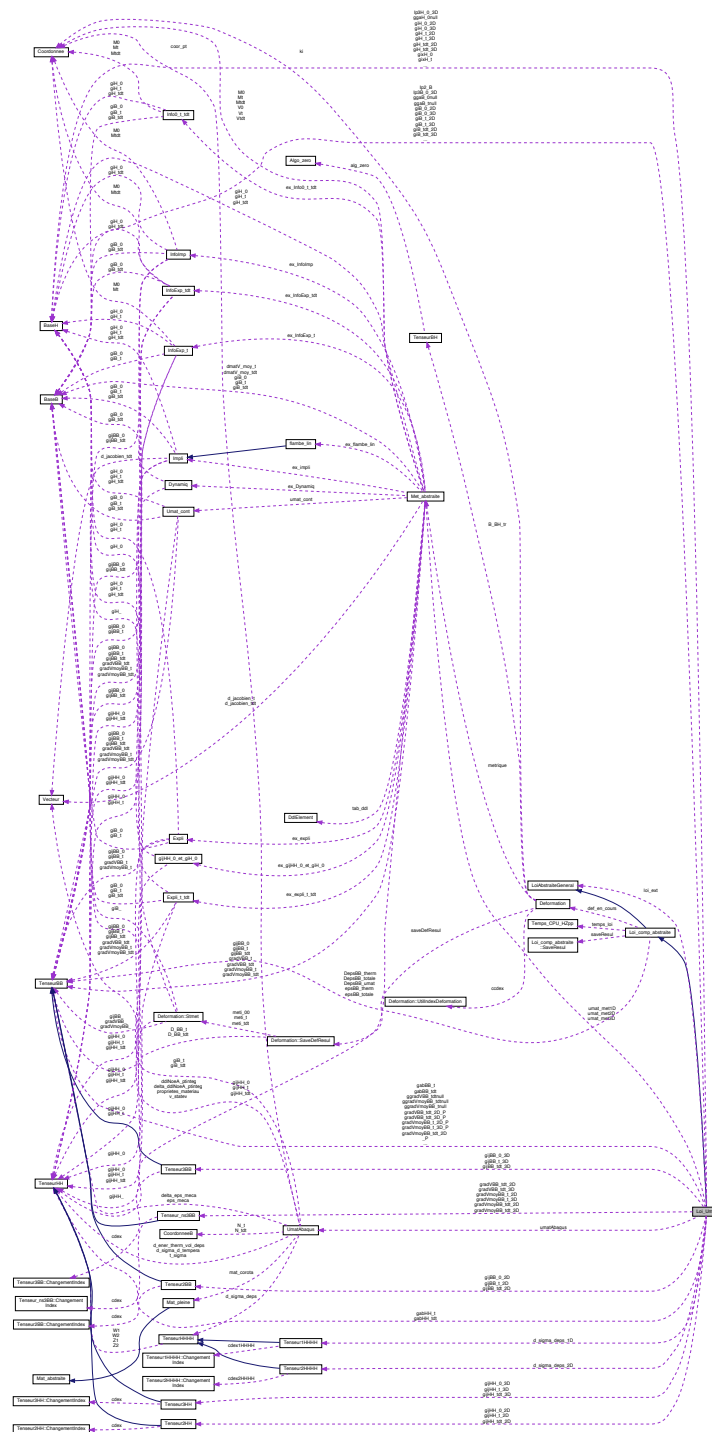
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_speciales/Loi\_rien3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_speciales/Loi\_rien3D.cc

## 6.473 Référence de la classe Loi\_Umat

Graphe d'héritage de Loi\_Umat:



Graphe de collaboration de Loi\_Umat:



## Classes

— class [SaveResul\\_Loi\\_Umat](#)

## Fonctions membres publiques

- [Loi\\_Umat](#) ([Enum\\_comp](#) enu=LOI\_VIA\_UMAT)
- [Loi\\_Umat](#) (const [Loi\\_Umat](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()



- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- void `Mise_a_jour_num_increment` (int num)
- void `Mise_a_jour_num_iteration` (int num)
- void `Mise_a_jour_nbe_nbptinteg` (int nbe, int nbptinteg)
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)
- const string & `NomDeLaLoi` () const
- void `DefLoiExt` (`LoiAbstraiteGeneral` \*lext)
- bool `Utilise_une_umat_interne` () const

### Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB\_tdt, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &, const `Deformation` &, `Enum_dure`, const `ThermoDonnee` &, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)
- void `RepercuteChangeTemperature` (`Enum_dure` temps)
- void `Passage_metriquer_ordre2_vers_3` (const `Met_abstraite::Umat_cont` &ex)
- void `Calcul_compressibilite_cisaillement` (const `Met_abstraite::Umat_cont` &ex, double &module\_compressibilite, double &module\_cisaillement)

### Fonctions membres protégées statiques

- static int `Choix_dim` (`Enum_comp` enu)

### Attributs protégés

- string `nom_de_la_loi`
- bool `utilisation_umat_interne`
- `UmatAbaqus` `umatAbaqus`
- `Met_abstraite` `umat_met3D`
- `Met_abstraite` `umat_met2D`
- `Met_abstraite` `umat_met1D`
- `Tenseur2HHHH` `d_sigma_deps_2D`
- `Tenseur1HHHH` `d_sigma_deps_1D`
- `BaseB` `gixB_0`

- [BaseB](#) [gixB\\_t](#)
- [BaseB](#) [gixB\\_tdt](#)
- [BaseH](#) [gixH\\_0](#)
- [BaseH](#) [gixH\\_t](#)
- [BaseH](#) [gixH\\_tdt](#)
- [TenseurBB](#) \* [gabBB\\_tdt](#)
- [TenseurHH](#) \* [gabHH\\_tdt](#)
- [TenseurBB](#) \* [gabBB\\_t](#)
- [TenseurHH](#) \* [gabHH\\_t](#)
- [BaseB](#) \* [ggaB\\_0null](#)
- [BaseB](#) \* [ggaB\\_tnull](#)
- [BaseH](#) \* [ggaH\\_0null](#)
- [TenseurBB](#) \* [ggradVmoyBB\\_tnull](#)
- [TenseurBB](#) \* [ggradVmoyBB\\_tdtnull](#)
- [TenseurBB](#) \* [ggradVBB\\_tdtnull](#)
- [Met\\_abstraite::Umat\\_cont](#) \* [umat\\_cont\\_3D](#)
- [Met\\_abstraite::Umat\\_cont](#) \* [umat\\_cont\\_2D](#)
- [BaseB](#) [lp2\\_B](#)
- [BaseB](#) [giB\\_0\\_3D](#)
- [BaseH](#) [giH\\_0\\_3D](#)
- [BaseB](#) [giB\\_t\\_3D](#)
- [BaseH](#) [giH\\_t\\_3D](#)
- [BaseB](#) [giB\\_tdt\\_3D](#)
- [BaseH](#) [giH\\_tdt\\_3D](#)
- [Tenseur3BB](#) [gijBB\\_0\\_3D](#)
- [Tenseur3HH](#) [gijHH\\_0\\_3D](#)
- [Tenseur3BB](#) [gijBB\\_t\\_3D](#)
- [Tenseur3HH](#) [gijHH\\_t\\_3D](#)
- [Tenseur3BB](#) [gijBB\\_tdt\\_3D](#)
- [Tenseur3HH](#) [gijHH\\_tdt\\_3D](#)
- [TenseurBB](#) \* [gradVmoyBB\\_t\\_3D\\_P](#)
- [Tenseur\\_ns3BB](#) [gradVmoyBB\\_t\\_3D](#)
- [TenseurBB](#) \* [gradVmoyBB\\_tdt\\_3D\\_P](#)
- [Tenseur\\_ns3BB](#) [gradVmoyBB\\_tdt\\_3D](#)
- [TenseurBB](#) \* [gradVBB\\_tdt\\_3D\\_P](#)
- [Tenseur\\_ns3BB](#) [gradVBB\\_tdt\\_3D](#)
- [double](#) [jacobien\\_tdt\\_3D](#)
- [double](#) [jacobien\\_t\\_3D](#)
- [double](#) [jacobien\\_0\\_3D](#)
- [BaseB](#) [lp3B\\_0\\_3D](#)
- [BaseH](#) [lp3H\\_0\\_3D](#)
- [BaseB](#) [giB\\_0\\_2D](#)
- [BaseH](#) [giH\\_0\\_2D](#)
- [BaseB](#) [giB\\_t\\_2D](#)
- [BaseH](#) [giH\\_t\\_2D](#)
- [BaseB](#) [giB\\_tdt\\_2D](#)
- [BaseH](#) [giH\\_tdt\\_2D](#)
- [Tenseur2BB](#) [gijBB\\_0\\_2D](#)
- [Tenseur2HH](#) [gijHH\\_0\\_2D](#)
- [Tenseur2BB](#) [gijBB\\_t\\_2D](#)
- [Tenseur2HH](#) [gijHH\\_t\\_2D](#)
- [Tenseur2BB](#) [gijBB\\_tdt\\_2D](#)
- [Tenseur2HH](#) [gijHH\\_tdt\\_2D](#)
- [TenseurBB](#) \* [gradVmoyBB\\_t\\_2D\\_P](#)
- [Tenseur\\_ns3BB](#) [gradVmoyBB\\_t\\_2D](#)
- [TenseurBB](#) \* [gradVmoyBB\\_tdt\\_2D\\_P](#)
- [Tenseur\\_ns3BB](#) [gradVmoyBB\\_tdt\\_2D](#)
- [TenseurBB](#) \* [gradVBB\\_tdt\\_2D\\_P](#)
- [Tenseur\\_ns3BB](#) [gradVBB\\_tdt\\_2D](#)
- [double](#) [jacobien\\_tdt\\_2D](#)
- [double](#) [jacobien\\_t\\_2D](#)
- [double](#) [jacobien\\_0\\_2D](#)
- [LoiAbstraiteGeneral](#) \* [loi\\_ext](#)

## Attributs protégés statiques

- static int **compteur\_simili\_num\_elemEtPtInteg**
- static int **max\_simili\_num\_ptinteg**

## Membres hérités additionnels

### 6.473.1 Documentation des fonctions membres

#### 6.473.1.1 Affiche()

```
void Loi_Umat::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.473.1.2 Calcul\_dsigma\_deps()

```
void Loi_Umat::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.473.1.3 Calcul\_DsigmaHH\_tdt()

```
void Loi_Umat::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
```

```

TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.473.1.4 Calcul\_SigmaHH()

```

void Loi_Umat::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.473.1.5 CalculGrandeurTravail()

```

virtual void Loi_Umat::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.473.1.6 Ecriture\_base\_info\_loi()

```

void Loi_Umat::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

**6.473.1.7 HsurH0()**

```
virtual double Loi_Umat::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.473.1.8 Info\_commande\_LoisDeComp()**

```
void Loi_Umat::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.473.1.9 Lecture\_base\_info\_loi()**

```
void Loi_Umat::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.473.1.10 LectureDonneesParticulieres()**

```
void Loi_Umat::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.473.1.11 Module\_young\_equivalent()**

```
double Loi_Umat::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.473.1.12 New\_et\_Initialise()**

```
Loi\_comp\_abstraite::SaveResul * Loi_Umat::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.473.1.13 Nouvelle\_loi\_identique()**

```
Loi\_comp\_abstraite * Loi_Umat::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.473.1.14 RepercuteChangeTemperature()**

```
void Loi_Umat::RepercuteChangeTemperature (
    Enum_dure temps ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.473.1.15 TestComplet()**

```
int Loi_Umat::TestComplet ( ) [virtual]
```

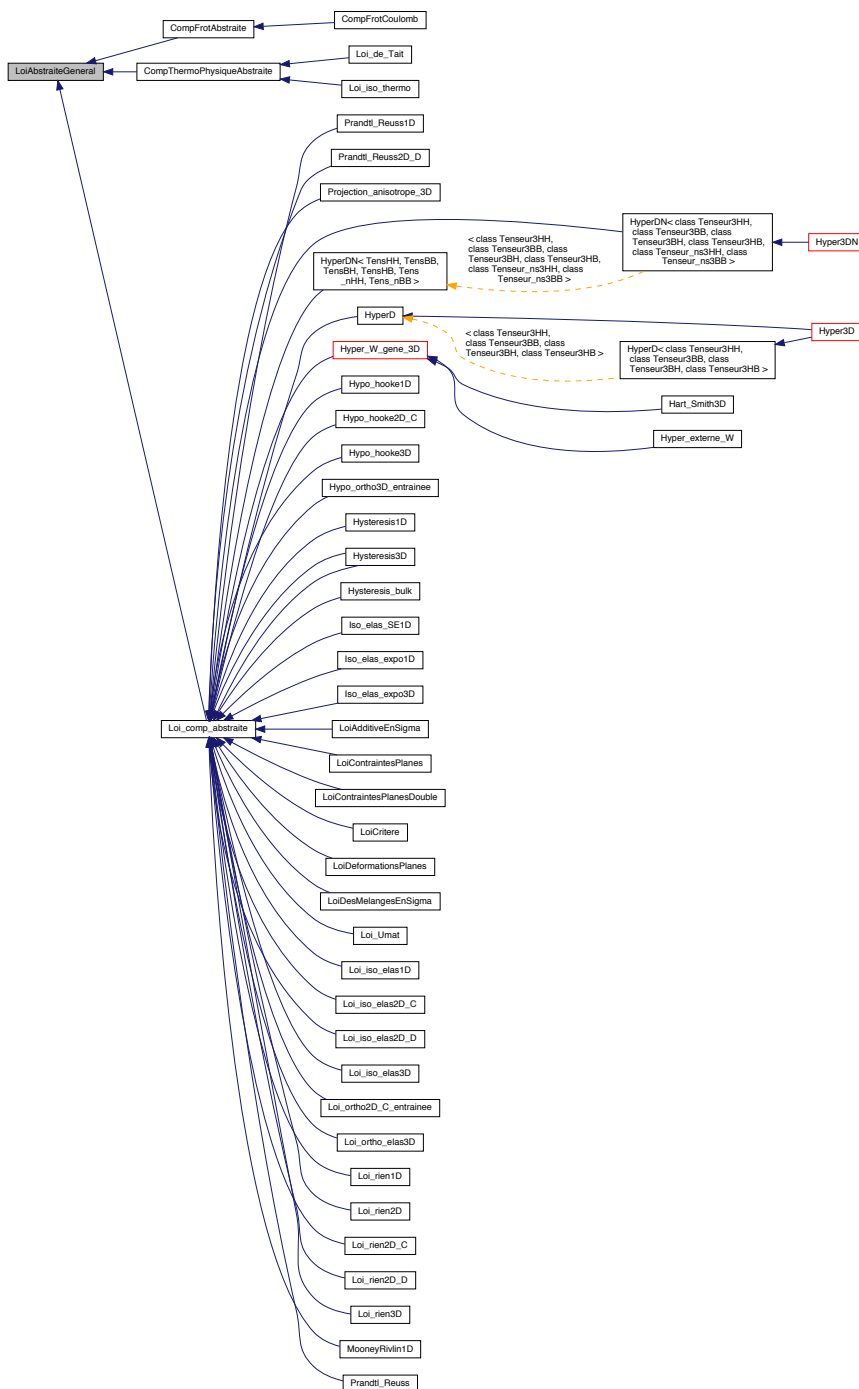
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loi\_Umat/Loi\_Umat.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loi\_Umat/Loi\_Umat.cc

## 6.474 Référence de la classe LoiAbstraiteGeneral

Grphe d'héritage de LoiAbstraiteGeneral:



### Fonctions membres publiques

- **LoiAbstraiteGeneral** ([Enum\\_comp](#) id\_comp, int dimension, [Enum\\_categorie\\_loi\\_comp](#) id\_categorie, Enum\_ddl ddl=SIG11)
- **LoiAbstraiteGeneral** (string nom, int dimension, [Enum\\_categorie\\_loi\\_comp](#) id\_categorie, Enum\_ddl ddl=SIG11)
- **LoiAbstraiteGeneral** (const [LoiAbstraiteGeneral](#) &a)

- void **Change\_comport** (string nouveau\_nom)
- void **Change\_comport** (Enum\_comp nouveau\_id)
- Enum\_comp **Id\_comport** () const
- string **Nom\_comport** () const
- Enum\_categorie\_loi\_comp **Id\_categorie** ()
- Enum\_ddl **EnumDdlTypePt** () const
- void **Change\_EnumDdlTypePt** (Enum\_ddl enu)
- virtual void **LectureDonneesParticulieres** (UtilLecture \*, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)=0
- virtual void **Affiche** () const =0
- virtual void **Lecture\_base\_info\_loi** (ifstream &ent, const int cas, LesReferences &lesRef, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)=0
- virtual void **Ecriture\_base\_info\_loi** (ofstream &sort, const int cas)=0
- virtual void **Info\_commande\_LoisDeComp** (UtilLecture &lec)=0
- virtual int **TestComple** ()
- virtual int **Dimension\_loi** () const
- virtual void **Modif\_comp\_tangent\_simplifie** (bool)
- virtual bool **Test\_loi\_simplifie** ()
- virtual Enum\_comp\_3D\_CP\_DP\_1D **Comportement\_3D\_CP\_DP\_1D** ()

### Fonctions membres protégées

- void **Lect\_base\_info\_loi** (ifstream &ent, const int cas, LesReferences &lesRef, LesCourbes1D &lesCourbes1D, LesFonctions\_nD &lesFonctionsnD)
- void **Ecrit\_base\_info\_loi** (ofstream &sort, const int cas) const
- void **ChangeCategorie** (Enum\_categorie\_loi\_comp new\_categorie)
- void **Change\_dimension** (int dime)
- Courbe1D \* **Lecture\_courbe** (UtilLecture \*entreePrinc, LesCourbes1D &lesCourbes1D)

### Attributs protégés

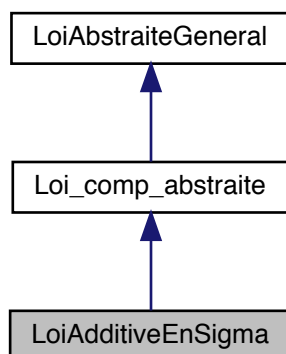
- Enum\_comp **id\_comp**
- int **dim**
- Enum\_categorie\_loi\_comp **categorie\_loi\_comp**
- Enum\_ddl **enu\_ddl\_type\_pt**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LoiAbstraiteGeneral.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LoiAbstraiteGeneral.cc

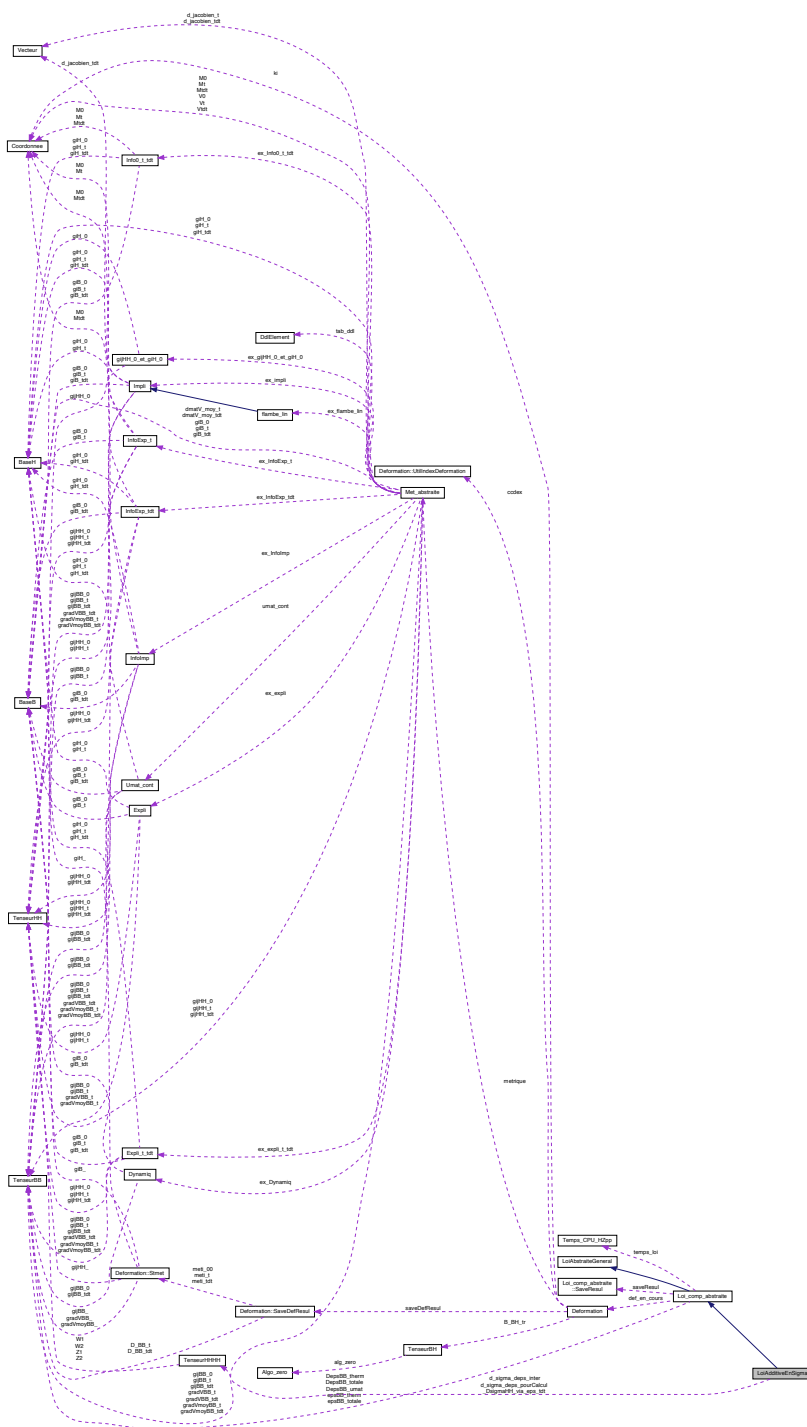
## 6.475 Référence de la classe LoiAdditiveEnSigma

Grappes d'héritage de LoiAdditiveEnSigma:





Graphe de collaboration de LoiAdditiveEnSigma:



## Classes

- class [SaveResul\\_LoiAdditiveEnSigma](#)

## Fonctions membres publiques

- [LoiAdditiveEnSigma](#) (const [LoiAdditiveEnSigma](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()

- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComplet` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- virtual void `Activation_donnees` (`Tableau`< `Noeud *` > &tabnoeud, bool dilatation, `LesPtIntegMecalInterne` &lesPtMecalInt)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ, `Loi_comp_abstraite::SaveResul` \*saveDon, list< int > &) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- virtual `Enum_comp_3D_CP_DP_1D` `Comportement_3D_CP_DP_1D` ()
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)

## Types protégés

- enum `Enumcompletudecalcul` { `CONTRAINTE_ET_TANGENT` =0 , `CONTRAINTE_UNIQUEMENT` , `TANGENT_UNIQUEMENT` }

## Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_t, `TenseurBB` &delta\_epsBB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `Tableau`< `TenseurBB *` > &d\_gijBB\_t, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB *` > &d\_epsBB\_tdt, `TenseurBB` &delta\_epsBB\_tdt, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB *` > &d\_gijBB\_tdt, `Tableau`< `TenseurHH *` > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH *` > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_DsigmaHH_via_eps_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB *` > &d\_epsBB\_tdt, `TenseurBB` &delta\_epsBB\_tdt, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB *` > &d\_gijBB\_tdt, `Tableau`< `TenseurHH *` > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH *` > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &ptintmeca, const `Deformation` &def, `Enum_dure` temps, const `ThermoDonnee` &dTP, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque *` > \*exclure\_Q)
- virtual void `IndiquePtIntegMecalInterne` (const `PtIntegMecalInterne` \*ptintmeca)
- void `RepercuteChangeTemperature` (`Enum_dure` temps)
- void `Verif_et_preparation_acces_grandeurs_locale` ()

## Attributs protégés

- list< [Loi\\_comp\\_abstraite](#) \* > **lois\_internes**
- list< Enumcompletudecalcul > **list\_completude\_calcul**
- int **type\_calcul**
- int **tangent\_ddl\_via\_eps**
- bool **avec\_ponderation**
- list< [Ponderation](#) > **list\_ponderation**
- list< [Ponderation\\_TypeQuelconque](#) \* > **list\_ponderation\_nD\_quelconque**
- [Tableau](#)< [TenseurHH](#) \* > **d\_sigtotalHH**
- [TenseurHHHH](#) \* **d\_sigma\_deps\_inter**
- [TenseurHHHH](#) \* **d\_sigma\_deps\_pourCalcul\_DsigmaHH\_via\_eps\_tdt**

## Amis

- class **SaveResul\_LoiAdditiveEnSigma**

## Membres hérités additionnels

### 6.475.1 Documentation des fonctions membres

#### 6.475.1.1 Activation\_donnees()

```
void LoiAdditiveEnSigma::Activation_donnees (
    Tableau< Noeud * > & tabnoeud,
    bool dilatation,
    LesPtIntegMecaInterne & lesPtMecaInt ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.475.1.2 Affiche()

```
void LoiAdditiveEnSigma::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.475.1.3 Calcul\_dsigma\_deps()

```
void LoiAdditiveEnSigma::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met\_abstraite::Umat\_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.475.1.4 Calcul\_DsigmaHH\_tdt()

```
void LoiAdditiveEnSigma::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.475.1.5 Calcul\_SigmaHH()

```
void LoiAdditiveEnSigma::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.475.1.6 CalculGrandeurTravail()**

```
void LoiAdditiveEnSigma::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,
    const ThermoDonnee & dTP,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.475.1.7 Comportement\_3D\_CP\_DP\_1D()**

```
Enum_comp_3D_CP_DP_1D LoiAdditiveEnSigma::Comportement_3D_CP_DP_1D ( ) [virtual]
```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

**6.475.1.8 Ecriture\_base\_info\_loi()**

```
void LoiAdditiveEnSigma::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.475.1.9 Grandeur\_particuliere()**

```
void LoiAdditiveEnSigma::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.475.1.10 HsurH0()**

```
virtual double LoiAdditiveEnSigma::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.475.1.11 IndiquePtIntegMecalInterne()**

```
virtual void LoiAdditiveEnSigma::IndiquePtIntegMecaInterne (
    const PtIntegMecaInterne * ptintmeca ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.475.1.12 Info\_commande\_LoisDeComp()**

```
void LoiAdditiveEnSigma::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.475.1.13 Lecture\_base\_info\_loi()

```
void LoiAdditiveEnSigma::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.475.1.14 LectureDonneesParticulieres()

```
void LoiAdditiveEnSigma::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.475.1.15 ListeGrandeurs\_particulieres()

```
void LoiAdditiveEnSigma::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.475.1.16 Module\_compressibilite\_equivalent()

```
double LoiAdditiveEnSigma::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.475.1.17 Module\_young\_equivalent()

```
double LoiAdditiveEnSigma::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.475.1.18 New\_et\_Initialise()

```
SaveResul * LoiAdditiveEnSigma::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.475.1.19 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * LoiAdditiveEnSigma::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.475.1.20 RepercuteChangeTemperature()

```
void LoiAdditiveEnSigma::RepercuteChangeTemperature (
    Enum_dure temps ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.475.1.21 TestComplet()

```
int LoiAdditiveEnSigma::TestComplet ( ) [virtual]
```

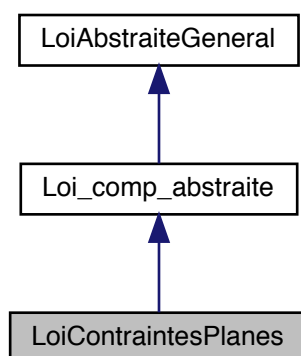
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

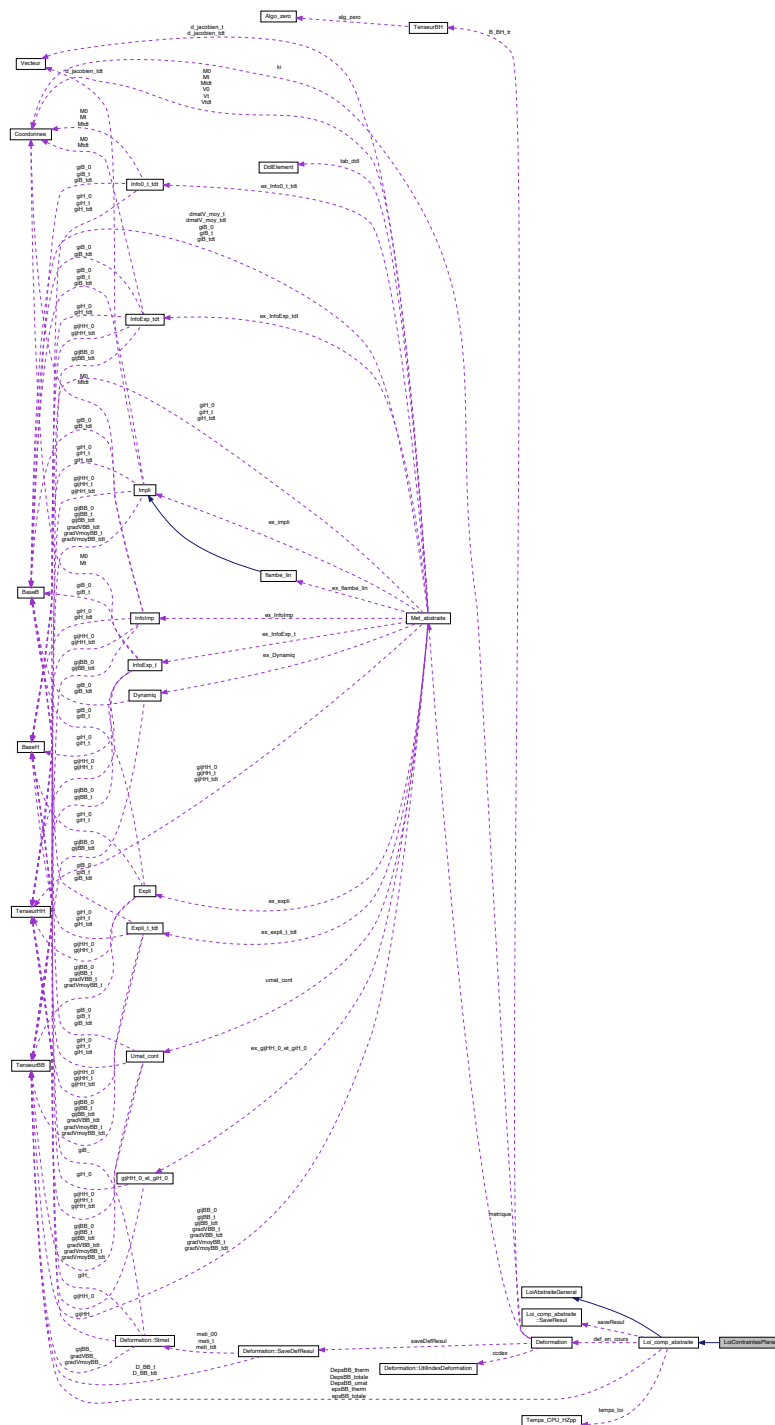
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiAdditiveEn↔Sigma.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiAdditiveEn↔Sigma.cc

## 6.476 Référence de la classe LoiContraintesPlanes

Graphe d'héritage de LoiContraintesPlanes:



Graphe de collaboration de LoiContraintesPlanes:



### Classes

- class [SaveResul\\_LoiContraintesPlanes](#)

### Fonctions membres publiques

- **LoiContraintesPlanes** (const [LoiContraintesPlanes](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()



- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `Eps33BH` (`SaveResul` \*saveResul) const
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- virtual bool `Contraintes_planes_de_3D` () const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- virtual void `Activation_donnees` (`Tableau`< `Noeud` \* > &tabnoeud, bool dilatation, `LesPtIntegMecalInterne` &lesPtMecalInt)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ, `Loi_comp_abstraite::SaveResul` \*saveDon, list< int > &decal) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ) const
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)
- `Vecteur` & `Residu_constitutif` (const double &alpha, const `Vecteur` &x, int &test)
- `Mat_abstraite` & `Mat_tangente_constitutif` (const double &alpha, const `Vecteur` &x, `Vecteur` &resi, int &test)

## Fonctions membres protégées

- virtual void `IndiquePtIntegMecalInterne` (const `PtIntegMecalInterne` \*ptintmeca)
- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &ptintmeca, const `Deformation` &def, `Enum_dure` temps, const `ThermoDonnee` &dTP, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)
- void `RepercuteChangeTemperature` (`Enum_dure` temps)
- const `Met_abstraite::Impli` \* `Passage_metrique_ordre2_vers_3` (const `Met_abstraite::Impli` &ex)
- const `Met_abstraite::Expli_t_tdt` \* `Passage_metrique_ordre2_vers_3` (const `Met_abstraite::Expli_t_tdt` &ex)
- const `Met_abstraite::Umat_cont` \* `Passage_metrique_ordre2_vers_3` (const `Met_abstraite::Umat_cont` &ex)
- void `Passage_deformation_volume_ordre2_vers_3` (const `TenseurBB` &DepsBB, const `TenseurBB` &epsBB\_tdt, const `TenseurBB` &delta\_epsBB, const `TenseurBB` &Deps\_BB\_3D, const `TenseurBB` &eps\_BB\_3D, const `TenseurBB` &delta\_eps\_BB\_3D)

## Amis

- class `LoiContraintesPlanesDouble`
- class `LoiCritere`
- class `SaveResul_LoiContraintesPlanes`

## Membres hérités additionnels

### 6.476.1 Documentation des fonctions membres

#### 6.476.1.1 Activation\_donnees()

```
void LoiContraintesPlanes::Activation_donnees (
    Tableau< Noeud * > & tabnoeud,
    bool dilatation,
    LesPtIntegMecaInterne & lesPtMecaInt ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.2 Affiche()

```
void LoiContraintesPlanes::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.476.1.3 Calcul\_dsigma\_deps()

```
void LoiContraintesPlanes::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.4 Calcul\_DsigmaHH\_tdt()

```
void LoiContraintesPlanes::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
```

```

Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.476.1.5 Calcul\_SigmaHH()

```

void LoiContraintesPlanes::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.476.1.6 CalculGrandeurTravail()

```

void LoiContraintesPlanes::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,
    const ThermoDonnee & dTP,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.476.1.7 Contraintes\_planes\_de\_3D()

```

virtual bool LoiContraintesPlanes::Contraintes_planes_de_3D ( ) const [inline], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.8 Ecriture\_base\_info\_loi()

```
void LoiContraintesPlanes::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.476.1.9 Eps33BH()

```
double LoiContraintesPlanes::Eps33BH (
    SaveResul * saveResul ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.10 Grandeur\_particuliere()

```
void LoiContraintesPlanes::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.11 HsurH0()

```
double LoiContraintesPlanes::HsurH0 (
    SaveResul * saveResul ) const [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.476.1.12 IndiquePtIntegMecaInterne()

```
virtual void LoiContraintesPlanes::IndiquePtIntegMecaInterne (
    const PtIntegMecaInterne * ptintmeca ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.13 Info\_commande\_LoisDeComp()

```
void LoiContraintesPlanes::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.476.1.14 Lecture\_base\_info\_loi()

```
void LoiContraintesPlanes::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.476.1.15 LectureDonneesParticulieres()

```
void LoiContraintesPlanes::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.476.1.16 ListeGrandeurs\_particulieres()

```
void LoiContraintesPlanes::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.17 Module\_compressibilite\_equivalent()

```
double LoiContraintesPlanes::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.18 Module\_young\_equivalent()

```
double LoiContraintesPlanes::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.19 New\_et\_Initialise()

```
LoiContraintesPlanes::SaveResul * LoiContraintesPlanes::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.20 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * LoiContraintesPlanes::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.476.1.21 RepercuteChangeTemperature()

```
void LoiContraintesPlanes::RepercuteChangeTemperature (
    Enum_dure temps ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.476.1.22 TestComplet()

```
int LoiContraintesPlanes::TestComplet ( ) [virtual]
```

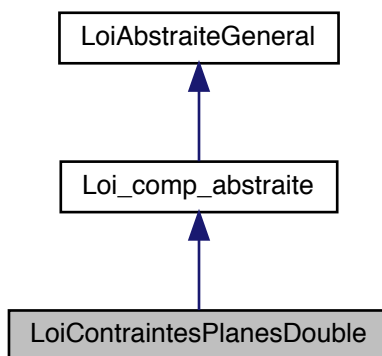
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

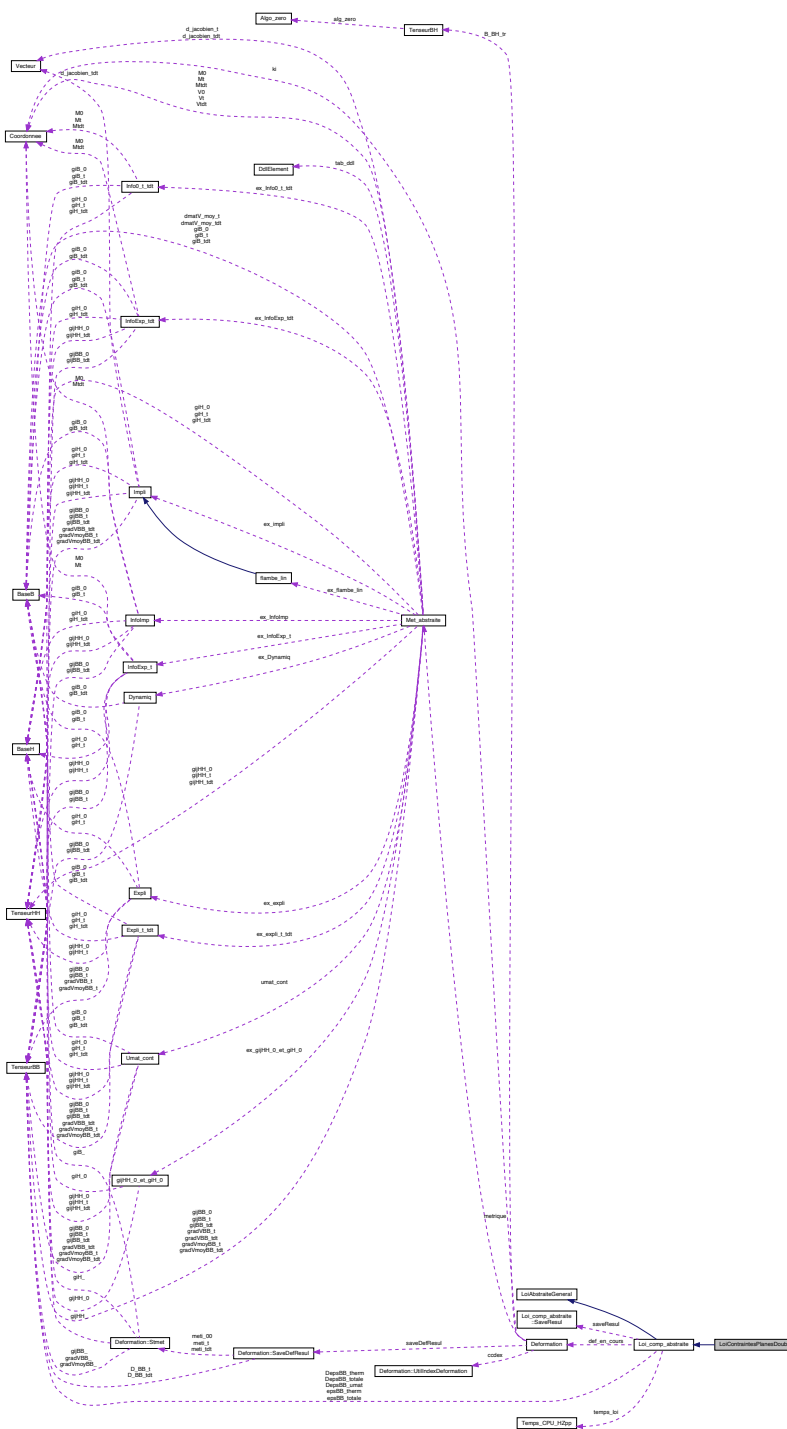
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔Planes.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔Planes.cc

## 6.477 Référence de la classe LoiContraintesPlanesDouble

Grphe d'héritage de LoiContraintesPlanesDouble:



Graphe de collaboration de LoiContraintesPlanesDouble:



## Classes

- class [SaveResul\\_LoiContraintesPlanesDouble](#)

## Fonctions membres publiques

- **LoiContraintesPlanesDouble** ([LoiContraintesPlanes](#) &loiCP\_de3D, bool calcul\_en\_3D\_via\_direction\_←quelconque\_)
- **LoiContraintesPlanesDouble** (const [LoiContraintesPlanesDouble](#) &loi)

- `SaveResul * New_et_Initialise ()`
- `void LectureDonneesParticulieres (UtilLecture *, LesCourbes1D &lesCourbes1D, LesFonctions_nD &lesFonctionsnD)`
- `void LectureParametres_controles (UtilLecture *, LesCourbes1D &lesCourbes1D, LesFonctions_nD &lesFonctionsnD)`
- `void Affiche () const`
- `int TestCompleet ()`
- `double Module_young_equivalent (Enum_dure temps, const Deformation &def, SaveResul *saveResul)`
- `double Module_compressibilite_equivalent (Enum_dure temps, const Deformation &def, SaveResul *saveResul)`
- `virtual double Eps33BH (SaveResul *saveResul) const`
- `virtual double Eps22BH (SaveResul *saveResul) const`
- `virtual double HsurH0 (SaveResul *saveResul) const`
- `virtual double d_HsurH0 (SaveResul *saveResul, Vecteur &d_hsurh0) const`
- `virtual double BsurB0 (SaveResul *saveResul) const`
- `virtual double d_BsurB0 (SaveResul *saveResul, Vecteur &d_bsurb0) const`
- `Loi_comp_abstraite * Nouvelle_loi_identique () const`
- `virtual void Activation_donnees (Tableau< Noeud * > &tabnoeud, bool dilatation, LesPtIntegMecaInterne &lesPtMecaInt)`
- `virtual void Grandeur_particuliere (bool absolue, List_io< TypeQuelconque > &liTQ, Loi_comp_abstraite::SaveResul *saveDon, list< int > &decal) const`
- `virtual void ListeGrandeurs_particulieres (bool absolue, List_io< TypeQuelconque > &liTQ) const`
- `void Lecture_base_info_loi (ifstream &ent, const int cas, LesReferences &lesRef, LesCourbes1D &lesCourbes1D, LesFonctions_nD &lesFonctionsnD)`
- `void Ecriture_base_info_loi (ofstream &sort, const int cas)`
- `void Info_commande_LoisDeComp (UtilLecture &lec)`
- `Vecteur & Residu_constitutif (const double &alpha, const Vecteur &x, int &test)`
- `Mat_abstraite & Mat_tangente_constitutif (const double &alpha, const Vecteur &x, Vecteur &resi, int &test)`
- `Vecteur & Residu_constitutif_3D (const double &alpha, const Vecteur &x, int &test)`
- `Mat_abstraite & Mat_tangente_constitutif_3D (const double &alpha, const Vecteur &x, Vecteur &resi, int &test)`

## Fonctions membres protégées

- `void Calcul_SigmaHH (TenseurHH &sigHH_t, TenseurBB &DepsBB, DdlElement &tab_ddl, TenseurBB &gijBB_t, TenseurHH &gijHH_t, BaseB &giB, BaseH &gi_H, TenseurBB &epsBB_t, TenseurBB &delta_epsBB_t, TenseurBB &gijBB_t, TenseurHH &gijHH_t, Tableau< TenseurBB * > &d_gijBB_t, double &jacobien_0, double &jacobien, TenseurHH &sigHH, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Expli_t_tdt &ex)`
- `void Calcul_DsigmaHH_tdt (TenseurHH &sigHH_t, TenseurBB &DepsBB, DdlElement &tab_ddl, BaseB &giB_t, TenseurBB &gijBB_t, TenseurHH &gijHH_t, BaseB &giB_tdt, Tableau< BaseB > &d_giB_tdt, BaseH &giH_tdt, Tableau< BaseH > &d_giH_tdt, TenseurBB &epsBB_tdt, Tableau< TenseurBB * > &d_epsBB_tdt, TenseurBB &delta_epsBB_t, TenseurBB &gijBB_tdt, TenseurHH &gijHH_tdt, Tableau< TenseurBB * > &d_gijBB_tdt, Tableau< TenseurHH * > &d_gijHH_tdt, double &jacobien_0, double &jacobien, Vecteur &d_jacobien_tdt, TenseurHH &sigHH, Tableau< TenseurHH * > &d_sigHH, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Impli &ex)`
- `void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH &sigHH_t, TenseurBB &DepsBB, TenseurBB &epsBB_tdt, TenseurBB &delta_epsBB, double &jacobien_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d_sigma_deps, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Umat_cont &ex)`
- `void Calcul_dsigma_deps (bool en_base_orthonormee, const BaseB *ViB, const BaseH *ViH, const TenseurHH &sigHH_t, const TenseurBB &DepsBB, const TenseurBB &epsBB_tdt, const TenseurBB &delta_epsBB, double &jacobien_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d_sigma_deps, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Umat_cont &ex)`
- `virtual void CalculGrandeurTravail (const PtIntegMecaInterne &ptintmeca, const Deformation &def, Enum_dure temps, const ThermoDonnee &dTP, const Met_abstraite::Impli *ex_impli, const Met_abstraite::Expli_t_tdt *ex_expli_tdt, const Met_abstraite::Umat_cont *ex_umat, const List_io< Ddl_etendu > *exclure_dd_etend, const List_io< const TypeQuelconque * > *exclure_Q)`
- `virtual void IndiquePtIntegMecaInterne (const PtIntegMecaInterne *ptintmeca)`



- void [RepercuteChangeTemperature](#) (Enum\_dure temps)
- [Enum\\_contrainte\\_mathematique Type\\_de\\_contrainte](#) () const
- void [Change\\_calcul\\_en\\_3D\\_via\\_direction\\_quelconque](#) (bool calcul\_en\_3D\_via\_direction\_quelconque↔  
\_)

## Amis

- class [LoiContraintesPlanes](#)
- class [LoiCritere](#)
- class [SaveResul\\_LoiContraintesPlanesDouble](#)

## Membres hérités additionnels

### 6.477.1 Documentation des fonctions membres

#### 6.477.1.1 Activation\_donnees()

```
void LoiContraintesPlanesDouble::Activation_donnees (
    Tableau< Noeud * > & tabnoeud,
    bool dilatation,
    LesPtIntegMecaInterne & lesPtMecaInt ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.477.1.2 Affiche()

```
void LoiContraintesPlanesDouble::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.477.1.3 BsurB0()

```
virtual double LoiContraintesPlanesDouble::BsurB0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.477.1.4 Calcul\_dsigma\_deps()

```
void LoiContraintesPlanesDouble::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.477.1.5 Calcul\_DsigmaHH\_tdt()**

```
void LoiContraintesPlanesDouble::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.477.1.6 Calcul\_SigmaHH()**

```
void LoiContraintesPlanesDouble::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.477.1.7 CalculGrandeurTravail()**

```
void LoiContraintesPlanesDouble::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,
    const ThermoDonnee & dTP,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.477.1.8 d\_BsurB0()**

```
virtual double LoiContraintesPlanesDouble::d_BsurB0 (
    SaveResul * saveResul,
    Vecteur & d_bsurb0 ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.477.1.9 d\_HsurH0()**

```
virtual double LoiContraintesPlanesDouble::d_HsurH0 (
    SaveResul * saveResul,
    Vecteur & d_hsurh0 ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.477.1.10 Ecriture\_base\_info\_loi()**

```
void LoiContraintesPlanesDouble::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.477.1.11 Eps22BH()**

```
virtual double LoiContraintesPlanesDouble::Eps22BH (
    SaveResul * saveResul ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.477.1.12 Eps33BH()**

```
virtual double LoiContraintesPlanesDouble::Eps33BH (
    SaveResul * saveResul ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.477.1.13 Grandeur\_particuliere()**

```
void LoiContraintesPlanesDouble::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.477.1.14 HsurH0()

```
virtual double LoiContraintesPlanesDouble::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.477.1.15 IndiquePtIntegMecaInterne()

```
virtual void LoiContraintesPlanesDouble::IndiquePtIntegMecaInterne (
    const PtIntegMecaInterne * ptintmeca ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.477.1.16 Info\_commande\_LoisDeComp()

```
void LoiContraintesPlanesDouble::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.477.1.17 Lecture\_base\_info\_loi()

```
void LoiContraintesPlanesDouble::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.477.1.18 LectureDonneesParticulieres()

```
void LoiContraintesPlanesDouble::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.477.1.19 ListeGrandeurs\_particulieres()

```
void LoiContraintesPlanesDouble::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.477.1.20 Module\_compressibilite\_equivalent()

```
double LoiContraintesPlanesDouble::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.477.1.21 Module\_young\_equivalent()

```
double LoiContraintesPlanesDouble::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.477.1.22 New\_et\_Initialise()

```
LoiContraintesPlanesDouble::SaveResul * LoiContraintesPlanesDouble::New_et_Initialise ( )
[virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.477.1.23 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * LoiContraintesPlanesDouble::Nouvelle_loi_identique ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.477.1.24 RepercuteChangeTemperature()

```
void LoiContraintesPlanesDouble::RepercuteChangeTemperature (
    Enum_dure temps ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.477.1.25 TestComplet()

```
int LoiContraintesPlanesDouble::TestComplet ( ) [virtual]
```

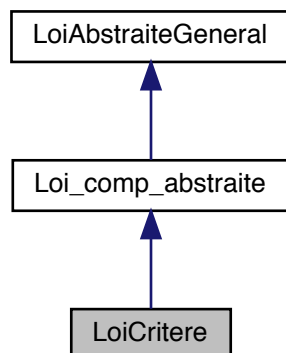
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

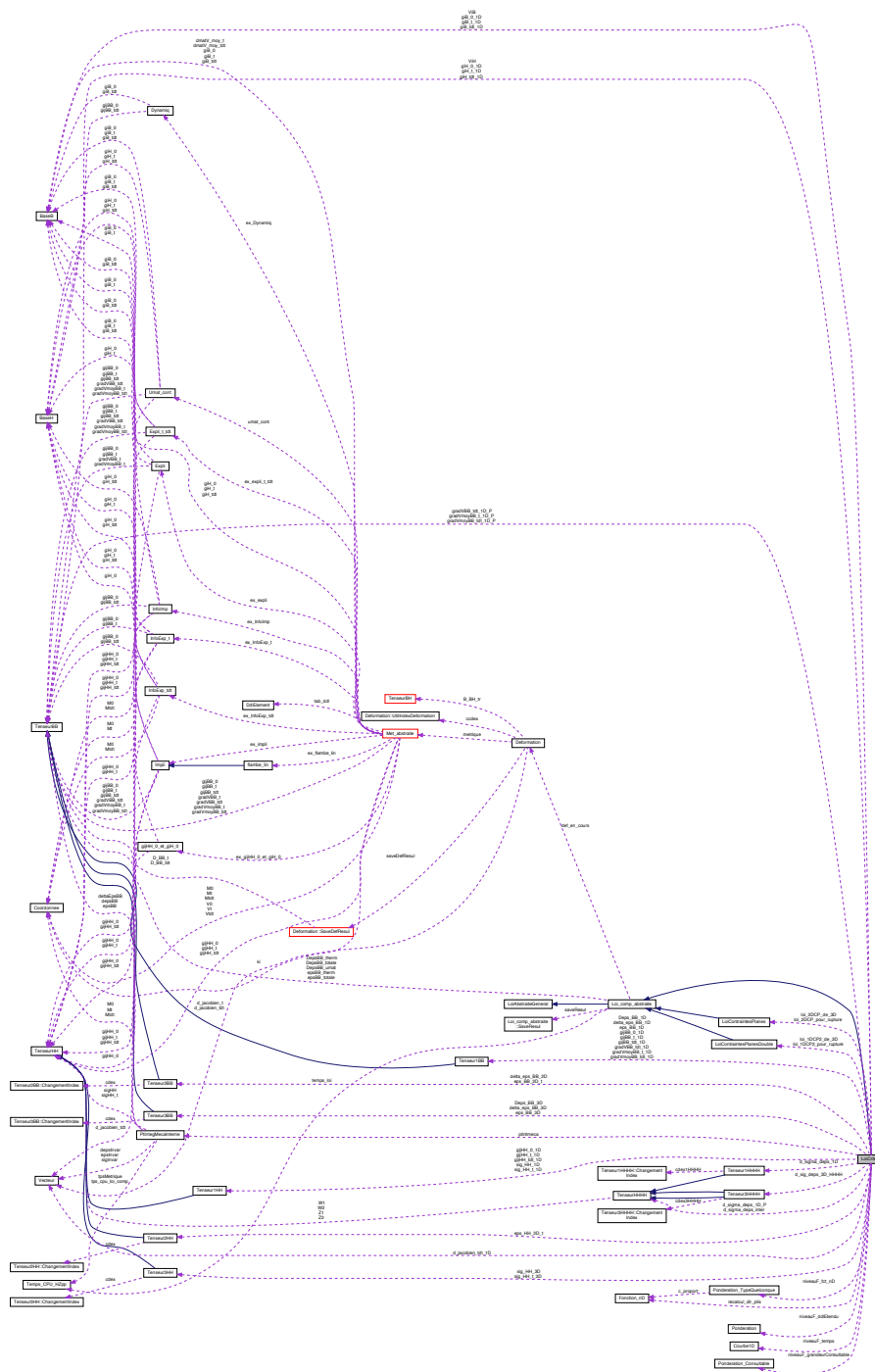
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔  
PlanesDouble.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔  
PlanesDouble.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔  
PlanesDouble\_2.cc

## 6.478 Référence de la classe LoiCritere

Graphe d'héritage de LoiCritere:



Graphe de collaboration de LoiCritere:



### Classes

- class [SaveResul\\_LoiCritere](#)

### Fonctions membres publiques

- **LoiCritere** (const [LoiCritere](#) &loi)
- **SaveResul** \* [New\\_et\\_Initialise](#) ()
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)

- void `Affiche ()` const
- int `TestCompleter ()`
- double `Module_young_equivalent (Enum_dure temps, const Deformation &def, SaveResul *saveResul)`
- double `Module_compressibilite_equivalent (Enum_dure temps, const Deformation &def, SaveResul *saveResul)`
- virtual double `HsurH0 (SaveResul *saveResul) const`
- virtual void `Activation_donnees (Tableau< Noeud * > &tabnoeud, bool dilatation, LesPtIntegMecalInterne &lesPtMecalInt)`
- virtual void `Grandeur_particuliere (bool absolue, List_io< TypeQuelconque > &litQ, Loi_comp_abstraite::SaveResul *saveDon, list< int > &) const`
- virtual void `ListeGrandeurs_particulieres (bool absolue, List_io< TypeQuelconque > &litQ) const`
- `Loi_comp_abstraite * Nouvelle_loi_identique ()` const
- virtual `Enum_comp_3D_CP_DP_1D Comportement_3D_CP_DP_1D ()`
- void `Lecture_base_info_loi (ifstream &ent, const int cas, LesReferences &lesRef, LesCourbes1D &lesCourbes1D, LesFonctions_nD &lesFonctionsnD)`
- void `Ecriture_base_info_loi (ofstream &sort, const int cas)`
- void `Info_commande LoisDeComp (UtilLecture &lec)`
- virtual void `Activation_stockage_grandeurs_quelconques (list< EnumTypeQuelconque > &listEnuQuelc)`
- virtual void `Insertion_conteneur_dans_save_result (SaveResul *saveResul)`

## Types protégés

- enum `Enumcompletudecalcul { CONTRAINTE_ET_TANGENT =0 , CONTRAINTE_UNIQUEMENT , TANGENT_UNIQUEMENT }`

## Fonctions membres protégées

- void `Calcul_SigmaHH (TenseurHH &sigHH_t, TenseurBB &DepsBB, DdlElement &tab_ddl, TenseurBB &gijBB_t, TenseurHH &gijHH_t, BaseB &giB, BaseH &gi_H, TenseurBB &epsBB_, TenseurBB &delta_epsBB_, TenseurBB &gijBB_, TenseurHH &gijHH_, Tableau< TenseurBB * > &d_gijBB_, double &jacobien_0, double &jacobien, TenseurHH &sigHH, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Expli_t_tdt &ex)`
- void `Calcul_DsigmaHH_tdt (TenseurHH &sigHH_t, TenseurBB &DepsBB, DdlElement &tab_ddl, BaseB &giB_t, TenseurBB &gijBB_t, TenseurHH &gijHH_t, BaseB &giB_tdt, Tableau< BaseB > &d_giB_tdt, BaseH &giH_tdt, Tableau< BaseH > &d_giH_tdt, TenseurBB &epsBB_tdt, Tableau< TenseurBB * > &d_epsBB_tdt, TenseurBB &delta_epsBB, TenseurBB &gijBB_tdt, TenseurHH &gijHH_tdt, Tableau< TenseurBB * > &d_gijBB_tdt, Tableau< TenseurHH * > &d_gijHH_tdt, double &jacobien_0, double &jacobien, Vecteur &d_jacobien_tdt, TenseurHH &sigHH, Tableau< TenseurHH * > &d_sigHH, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Impli &ex)`
- void `Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH &sigHH_t, TenseurBB &DepsBB, TenseurBB &epsBB_tdt, TenseurBB &delta_epsBB, double &jacobien_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d_sigma_deps, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, const Met_abstraite::Umat_cont &ex)`
- virtual void `CalculGrandeurTravail (const PtIntegMecalInterne &ptintmeca, const Deformation &def, Enum_dure temps, const ThermoDonnee &dTP, const Met_abstraite::Impli *ex_impli, const Met_abstraite::Expli_t_tdt *ex_expli_tdt, const Met_abstraite::Umat_cont *ex_umat, const List_io< Ddl_etendu > *exclure_dd_etend, const List_io< const TypeQuelconque * > *exclure_Q)`
- virtual void `IndiquePtIntegMecalInterne (const PtIntegMecalInterne *ptintmeca)`
- void `RepercuteChangeTemperature (Enum_dure temps)`
- int `Critere (bool en_base_orthonormee, TenseurHH &sigHH_t, TenseurBB &DepsBB, TenseurBB &epsBB_tdt, TenseurBB &delta_epsBB, double &jacobien_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d_sigma_deps_inter, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, bool implicit, const Met_abstraite::Umat_cont &ex)`
- int `Critere_plis_membrane (bool en_base_orthonormee, TenseurHH &sigHH_t, TenseurBB &DepsBB, TenseurBB &epsBB_tdt, TenseurBB &delta_epsBB, double &jacobien_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d_sigma_deps_inter, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, bool implicit, const Met_abstraite::Umat_cont &ex)`
- int `Critere_plis_biel (bool en_base_orthonormee, TenseurHH &sigHH_t, TenseurBB &DepsBB, TenseurBB &epsBB_tdt, TenseurBB &delta_epsBB, double &jacobien_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d_sigma_deps_inter, EnergieMeca &energ, const EnergieMeca &energ_t, double &module_compressibilite, double &module_cisaillement, bool implicit, const Met_abstraite::Umat_cont &ex)`



- void **Creation\_metrique\_a\_partir\_vecteurs\_propres\_pour\_Umat1D** (const Met\_abstraite::Umat\_cont &ex, const Mat\_pleine &gamma)
- void **Passage\_metrique\_ordre\_3\_2\_vers\_1** (const Met\_abstraite::Umat\_cont &ex, const Mat\_pleine &gamma)
- void **Passage\_3ou2\_vers\_1** (const Mat\_pleine &gamma, const TenseurHH &sigHH\_t, const Mat\_pleine &beta, const TenseurBB &DepsBB, const TenseurBB &epsBB\_tdt, const TenseurBB &delta\_epsBB, const bool &deux\_plis, Coordonnee2 &eps\_pli)
- void **Passage\_1\_vers\_3ou2** (const Mat\_pleine &gamma, TenseurHH &sigHH, const TenseurHHHH &d\_sigma\_deps\_1D, const Mat\_pleine &beta, TenseurHHHH &d\_sigma\_deps\_inter, const Met\_abstraite::Umat\_cont &ex, const Tableau< Coordonnee2H > &V\_Pr\_H, const Tableau< Coordonnee > &V\_P\_sig)
- int **Calcul\_directions\_plis** (const TenseurBB &epsBB\_tdt, const TenseurHH &sigHH, Coordonnee2 &valPropreEps, Tableau< Coordonnee2H > &V\_Pr\_H, const Met\_abstraite::Umat\_cont &ex, Coordonnee2 &valPropreSig, bool force=true)
- void **Premie\_type\_calcul\_en\_un\_pli** (const TenseurBB &epsBB\_tdt, const TenseurBB &delta\_epsBB, const TenseurHH &sigHH\_t, const double &jacobien\_0, const double &jacobien, EnergieMeca &energ, const EnergieMeca &energ\_t, const TenseurBB &DepsBB, double &module\_compressibilite, double &module\_cisaillement, const Tableau< Coordonnee2H > &V\_Pr\_H, TenseurHH &sigHH, TenseurHHHH &d\_sigma\_deps\_inter, const Coordonnee2 &valPropreSig, const Met\_abstraite::Umat\_cont &ex)
- void **Deuxieme\_type\_calcul\_en\_un\_pli** (const TenseurBB &epsBB\_tdt, const TenseurBB &delta\_epsBB, const TenseurHH &sigHH\_t, double &jacobien\_0, double &jacobien, EnergieMeca &energ, const EnergieMeca &energ\_t, const TenseurBB &DepsBB, double &module\_compressibilite, double &module\_cisaillement, const Tableau< Coordonnee2H > &V\_Pr\_H, TenseurHH &sigHH, TenseurHHHH &d\_sigma\_deps\_inter, const Coordonnee2 &valPropreSig, const Met\_abstraite::Umat\_cont &ex)
- int **Pre\_Critere** (const TenseurBB &DepsBB, TenseurBB &epsBB\_tdt, TenseurBB &delta\_epsBB, double &jacobien\_0, double &jacobien, const Met\_abstraite::Expli\_t\_tdt \*ex\_expli, const Met\_abstraite::Impli \*ex\_impli, const Met\_abstraite::Umat\_cont \*ex\_umat)
- void **Pre\_Critere\_plis\_membrane** (TenseurBB &epsBB\_tdt, TenseurBB &delta\_epsBB, const Met\_abstraite::Expli\_t\_tdt \*ex\_expli, const Met\_abstraite::Impli \*ex\_impli, const Met\_abstraite::Umat\_cont \*ex\_umat)
- void **Passage\_deformation\_contrainte\_ordre2\_vers\_3** (const TenseurBB &DepsBB, const TenseurBB &epsBB\_tdt, const TenseurBB &delta\_epsBB, const TenseurHH &sig\_HH\_t)
- void **Passage\_contrainte\_et\_operateur\_tangent\_ordre2\_vers\_3** (TenseurHH &sig\_HH\_tdt, TenseurHHHH &d\_sigma\_deps\_inter, const bool &deux\_plis, Coordonnee2 &eps\_pli)

## Attributs protégés

- Enum\_Critere\_Loi type\_critere
- int choix\_methode\_cal\_plis\_memb
- Tableau< int > ordre\_criteres
- LoiContraintesPlanes \* loi\_2DCP\_de\_3D
- LoiContraintesPlanesDouble \* loi\_1DCP2\_de\_3D
- double niveau\_declenchement\_critere
- bool avecNiveauSigma\_mini\_pour\_plis
- Ponderation\_TypeQuelconque \* niveauF\_fct\_nD
- Ponderation \* niveauF\_ddlEtendu
- Courbe1D \* niveauF\_temps
- Ponderation\_Consultable \* niveauF\_grandeurConsultable
- int choix\_calcul\_epaisseur\_si\_relachement\_complet
- Fonction\_nD \* recalcul\_dir\_plis
- LoiContraintesPlanes \* loi\_2DCP\_pour\_rupture
- LoiContraintesPlanesDouble \* loi\_1DCP2\_pour\_rupture
- bool avec\_ponderation\_grandeur\_globale
- list< Ponderation\_GGGlobal > list\_ponderation\_GGlob
- list< double > fonc\_ponder\_GGlob
- Met\_abstraite::Expli\_t\_tdt \* expli\_1D
- Met\_abstraite::Impli \* impli\_1D
- Met\_abstraite::Umat\_cont \* umat\_cont\_1D
- BaseB giB\_0\_1D
- BaseH giH\_0\_1D
- BaseB giB\_t\_1D
- BaseH giH\_t\_1D
- BaseB giB\_tdt\_1D

- BaseH **giH\_tdt\_1D**
- Tenseur1BB **gijBB\_0\_1D**
- Tenseur1HH **gijHH\_0\_1D**
- Tenseur1BB **gijBB\_t\_1D**
- Tenseur1HH **gijHH\_t\_1D**
- Tenseur1BB **gijBB\_tdt\_1D**
- Tenseur1HH **gijHH\_tdt\_1D**
- TenseurBB \* **gradVmoyBB\_t\_1D\_P**
- Tenseur1BB **gradVmoyBB\_t\_1D**
- TenseurBB \* **gradVmoyBB\_tdt\_1D\_P**
- Tenseur1BB **gradVmoyBB\_tdt\_1D**
- TenseurBB \* **gradVBB\_tdt\_1D\_P**
- Tenseur1BB **gradVBB\_tdt\_1D**
- double **jacobien\_tdt\_1D**
- double **jacobien\_t\_1D**
- double **jacobien\_0\_1D**
- Vecteur **d\_jacobien\_tdt\_1D**
- Tableau< BaseB > **d\_giB\_tdt\_1D**
- Tableau< BaseH > **d\_giH\_tdt\_1D**
- Tableau< TenseurBB \* > **d\_gijBB\_tdt\_1D\_P**
- Tableau< Tenseur1BB > **d\_gijBB\_tdt\_1D**
- Tableau2< TenseurBB \* > \* **d2\_gijBB\_tdt\_1D\_P**
- Tableau2< Tenseur1BB > **d2\_gijBB\_tdt\_1D**
- Tableau< TenseurHH \* > **d\_gijHH\_tdt\_1D\_P**
- Tableau< Tenseur1HH > **d\_gijHH\_tdt\_1D**
- Tableau< TenseurBB \* > \* **d\_gradVmoyBB\_t\_1D\_P**
- Tableau< Tenseur1BB > **d\_gradVmoyBB\_t\_1D**
- Tableau< TenseurBB \* > \* **d\_gradVmoyBB\_tdt\_1D\_P**
- Tableau< Tenseur1BB > **d\_gradVmoyBB\_tdt\_1D**
- Tableau< TenseurBB \* > \* **d\_gradVBB\_t\_1D\_P**
- Tableau< Tenseur1BB > **d\_gradVBB\_t\_1D**
- Tableau< TenseurBB \* > \* **d\_gradVBB\_tdt\_1D\_P**
- Tableau< Tenseur1BB > **d\_gradVBB\_tdt\_1D**
- Tenseur1HH **sig\_HH\_t\_1D**
- Tenseur1HH **sig\_HH\_1D**
- Tenseur1BB **Deps\_BB\_1D**
- Tenseur1BB **eps\_BB\_1D**
- Tenseur1BB **delta\_eps\_BB\_1D**
- Tableau< TenseurBB \* > **d\_eps\_BB\_1D\_P**
- Tableau< Tenseur1BB > **d\_eps\_BB\_1D**
- Tableau< TenseurHH \* > **d\_sig\_HH\_1D\_P**
- Tableau< Tenseur1HH > **d\_sig\_HH\_1D**
- TenseurHHHH \* **d\_sigma\_deps\_1D\_P**
- Tenseur1HHHH **d\_sigma\_deps\_1D**
- Tenseur2BB **eps\_BB\_2D\_t**
- Tenseur2BB **delta\_eps\_BB\_2D**
- Tenseur2HH **eps\_HH\_2D\_t**
- BaseB **ViB**
- BaseH **ViH**
- Tenseur3HH **sig\_HH\_t\_3D**
- Tenseur3HH **sig\_HH\_3D**
- Tenseur3BB **Deps\_BB\_3D**
- Tenseur3BB **eps\_BB\_3D**
- Tenseur3BB **delta\_eps\_BB\_3D**
- PtIntegMecalInterne **ptintmeca**
- list< Loi\_comp\_abstraite \* > **lois\_internes**
- list< Enumcompletudecalcul > **list\_completude\_calcul**
- int **type\_calcul**
- bool **avec\_ponderation**
- list< Ponderation > **list\_ponderation**
- list< double > **fonc\_ponder**
- Tableau< TenseurHH \* > **d\_sigtotalHH**
- TenseurHHHH \* **d\_sigma\_deps\_inter**
- Tenseur3HHHH **d\_sig\_deps\_3D\_HHHH**

## Amis

- class `LoiContraintesPlanes`
- class `LoiContraintesPlanesDouble`
- class `SaveResul_LoiCritere`

## Membres hérités additionnels

### 6.478.1 Documentation des fonctions membres

#### 6.478.1.1 Activation\_donnees()

```
void LoiCritere::Activation_donnees (
    Tableau< Noeud * > & tabnoeud,
    bool dilatation,
    LesPtIntegMecaInterne & lesPtMecaInt ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.2 Activation\_stockage\_grandeurs\_quelconques()

```
void LoiCritere::Activation_stockage_grandeurs_quelconques (
    list< EnumTypeQuelconque > & listEnuQuelc ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.3 Affiche()

```
void LoiCritere::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.478.1.4 Calcul\_dsigma\_deps()

```
void LoiCritere::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.5 Calcul\_DsigmaHH\_tdt()

```
void LoiCritere::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
```

```

BaseB & giB_t,
TenseurBB & gijBB_t,
TenseurHH & gijHH_t,
BaseB & giB_tdt,
Tableau< BaseB > & d_giB_tdt,
BaseH & giH_tdt,
Tableau< BaseH > & d_giH_tdt,
TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.478.1.6 Calcul\_SigmaHH()

```

void LoiCritere::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.478.1.7 CalculGrandeurTravail()

```

void LoiCritere::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,

```

```
const ThermoDonnee & dTP,  
const Met_abstraite::Impli * ex_impli,  
const Met_abstraite::Expli_t_tdt * ex_expli_tdt,  
const Met_abstraite::Umat_cont * ex_umat,  
const List_io< Ddl_etendu > * exclure_dd_etend,  
const List_io< const TypeQuelconque * > * exclure_Q ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.478.1.8 Comportement\_3D\_CP\_DP\_1D()

```
Enum_comp_3D_CP_DP_1D LoiCritere::Comportement_3D_CP_DP_1D ( ) [virtual]
```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

#### 6.478.1.9 Ecriture\_base\_info\_loi()

```
void LoiCritere::Ecriture_base_info_loi (   
    ofstream & sort,  
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.478.1.10 Grandeur\_particuliere()

```
void LoiCritere::Grandeur_particuliere (   
    bool absolue,  
    List_io< TypeQuelconque > & liTQ,  
    Loi_comp_abstraite::SaveResul * saveDon,  
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.11 HsurH0()

```
double LoiCritere::HsurH0 (   
    SaveResul * saveResul ) const [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.478.1.12 IndiquePtIntegMecaInterne()

```
virtual void LoiCritere::IndiquePtIntegMecaInterne (   
    const PtIntegMecaInterne * ptintmeca ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.13 Info\_commande\_LoisDeComp()

```
void LoiCritere::Info_commande_LoisDeComp (   
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.478.1.14 Insertion\_conteneur\_dans\_save\_result()

```
void LoiCritere::Insertion_conteneur_dans_save_result (   
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.15 Lecture\_base\_info\_loi()

```
void LoiCritere::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.478.1.16 LectureDonneesParticulieres()

```
void LoiCritere::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.478.1.17 ListeGrandeurs\_particulieres()

```
void LoiCritere::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.18 Module\_compressibilite\_equivalent()

```
double LoiCritere::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.19 Module\_young\_equivalent()

```
double LoiCritere::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.20 New\_et\_Initialise()

```
SaveResul * LoiCritere::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.478.1.21 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * LoiCritere::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.478.1.22 RepercuteChangeTemperature()

```
void LoiCritere::RepercuteChangeTemperature (
    Enum_dure temps ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.478.1.23 TestComplet()

```
int LoiCritere::TestComplet ( ) [virtual]
```

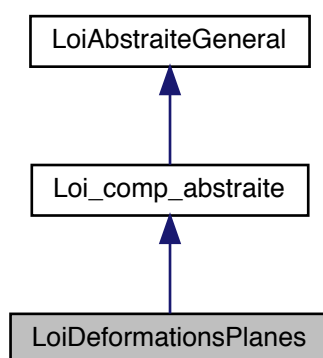
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

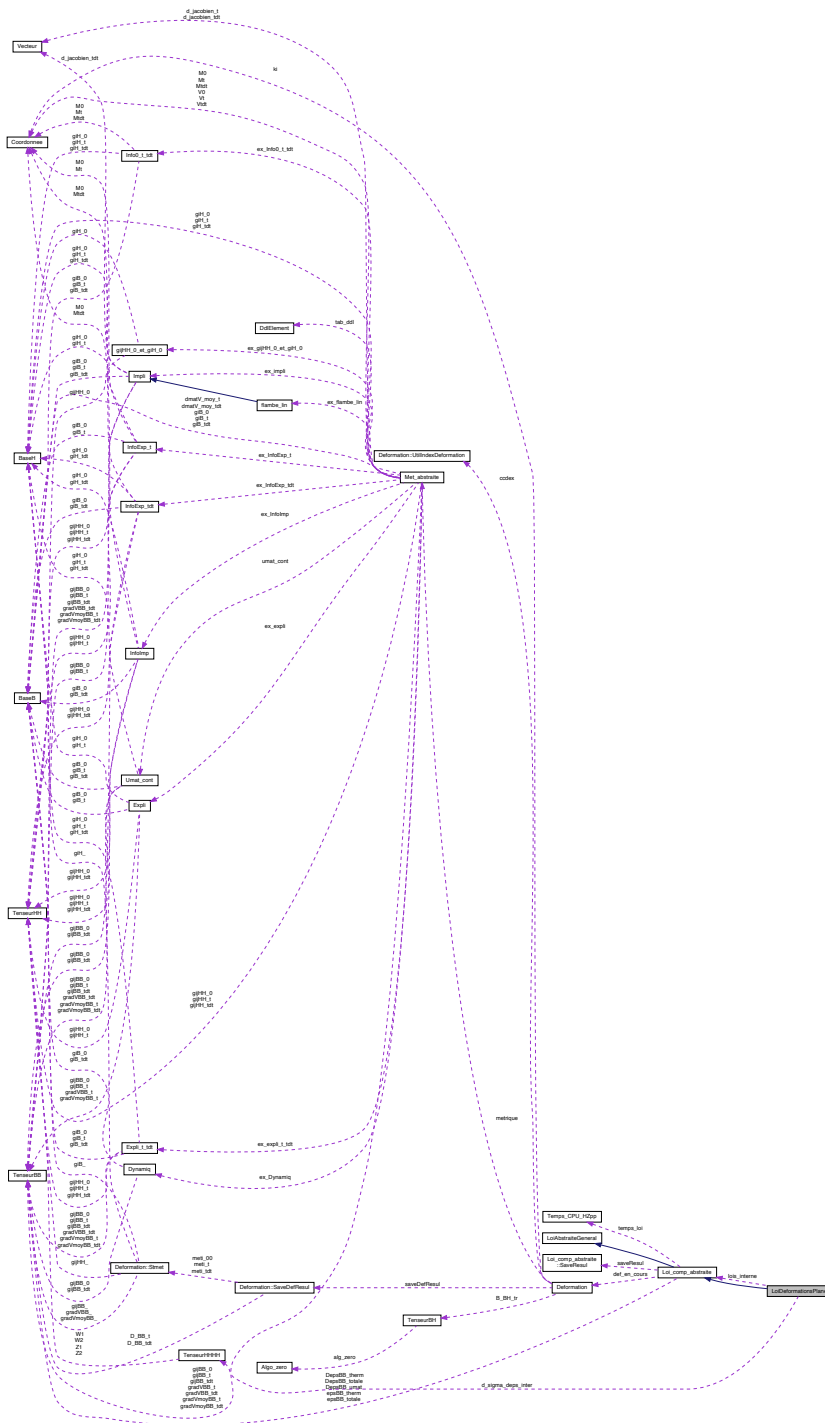
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiCritere.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiCritere.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiCritere2.cc

## 6.479 Référence de la classe LoiDeformationsPlanes

Graphe d'héritage de LoiDeformationsPlanes:



Graphe de collaboration de LoiDeformationsPlanes:



**Classes**

- class [SaveResul\\_LoiDeformationsPlanes](#)

**Fonctions membres publiques**

- **LoiDeformationsPlanes** (const [LoiDeformationsPlanes](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()



- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComplet](#) ()
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul](#) \*saveResul)
- double [Module\\_compressibilite\\_equivalent](#) ([Enum\\_dure](#) temps, const [Deformation](#) &def, [SaveResul](#) \*saveResul)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- virtual void [Activation\\_donnees](#) ([Tableau](#)< [Noeud](#) \* > &tabnoeud, bool dilatation, [LesPtIntegMecalInterne](#) &lesPtMecalInt)
- virtual void [Grandeur\\_particuliere](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ, [Loi\\_comp\\_abstraite](#)::[SaveResul](#) \*saveDon, list< int > &decal) const
- virtual void [ListeGrandeurs\\_particulieres](#) (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &liTQ) const
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_, [TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#) &giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#) &giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite](#)::[Impli](#) &ex)
- void [Calcul\\_dsigma\\_deps](#) (bool en\_base\_orthonormee, [TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [TenseurHHHH](#) &d\_sigma\_deps, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite](#)::[Umat\\_cont](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &ptintmeca, const [Deformation](#) &def, [Enum\\_dure](#) temps, const [ThermoDonnee](#) &dTP, const [Met\\_abstraite](#)::[Impli](#) \*ex\_impli, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite](#)::[Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)
- virtual void [IndiquePtIntegMecalInterne](#) (const [PtIntegMecalInterne](#) \*ptintmeca)
- void [RepercuteChangeTemperature](#) ([Enum\\_dure](#) temps)

## Attributs protégés

- [Loi\\_comp\\_abstraite](#) \* [lois\\_interne](#)
- [TenseurHHHH](#) \* [d\\_sigma\\_deps\\_inter](#)

## Amis

- class [SaveResul\\_LoiDeformationsPlanes](#)

## Membres hérités additionnels

### 6.479.1 Documentation des fonctions membres

**6.479.1.1 Activation\_donnees()**

```
void LoiDeformationsPlanes::Activation_donnees (
    Tableau< Noeud * > & tabnoeud,
    bool dilatation,
    LesPtIntegMecaInterne & lesPtMecaInt ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.479.1.2 Affiche()**

```
void LoiDeformationsPlanes::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.479.1.3 Calcul\_dsigma\_deps()**

```
void LoiDeformationsPlanes::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.479.1.4 Calcul\_DsigmaHH\_tdt()**

```
void LoiDeformationsPlanes::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
```

```

    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.479.1.5 Calcul\_SigmaHH()

```

void LoiDeformationsPlanes::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.479.1.6 CalculGrandeurTravail()

```

void LoiDeformationsPlanes::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,
    const ThermoDonnee & dTP,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.479.1.7 Ecriture\_base\_info\_loi()

```

void LoiDeformationsPlanes::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

### 6.479.1.8 Grandeur\_particuliere()

```
void LoiDeformationsPlanes::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.479.1.9 HsurH0()

```
virtual double LoiDeformationsPlanes::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.479.1.10 IndiquePtIntegMecaInterne()

```
virtual void LoiDeformationsPlanes::IndiquePtIntegMecaInterne (
    const PtIntegMecaInterne * ptintmeca ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.479.1.11 Info\_commande\_LoisDeComp()

```
void LoiDeformationsPlanes::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.479.1.12 Lecture\_base\_info\_loi()

```
void LoiDeformationsPlanes::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.479.1.13 LectureDonneesParticulieres()

```
void LoiDeformationsPlanes::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.479.1.14 ListeGrandeurs\_particulieres()

```
void LoiDeformationsPlanes::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.479.1.15 Module\_compressibilite\_equivalent()

```
double LoiDeformationsPlanes::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.479.1.16 Module\_young\_equivalent()

```
double LoiDeformationsPlanes::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.479.1.17 New\_et\_Initialise()

```
LoiDeformationsPlanes::SaveResul * LoiDeformationsPlanes::New_et_Initialise ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.479.1.18 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * LoiDeformationsPlanes::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.479.1.19 RepercuteChangeTemperature()

```
void LoiDeformationsPlanes::RepercuteChangeTemperature (
    Enum_dure temps ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.479.1.20 TestComplet()

```
int LoiDeformationsPlanes::TestComplet ( ) [virtual]
```

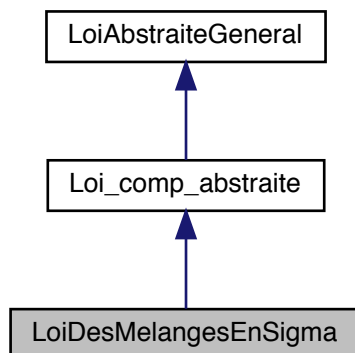
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

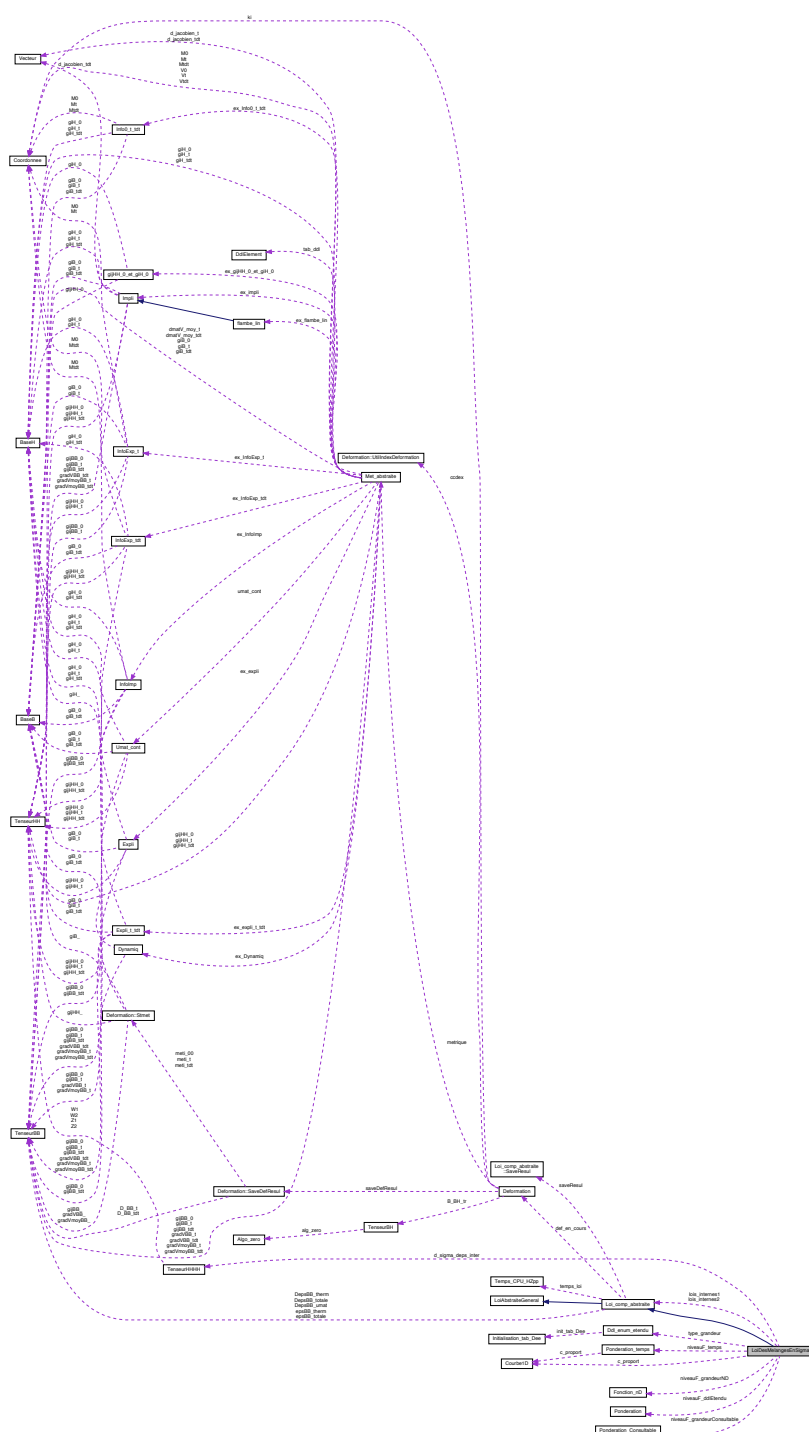
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiDeformations↵  
Planes.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiDeformations↵  
Planes.cc

## 6.480 Référence de la classe LoiDesMelangesEnSigma

Graphe d'héritage de LoiDesMelangesEnSigma:



Graphe de collaboration de LoiDesMelangesEnSigma:



## Classes

- class [SaveResul\\_LoiDesMelangesEnSigma](#)

## Fonctions membres publiques

- `LoiDesMelangesEnSigma` (const `LoiDesMelangesEnSigma` &loi)
- `SaveResul * New_et_Initialise` ()

- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- virtual void `Activation_donnees` (`Tableau`< `Noeud` \* > &tabnoeud, bool dilatation, `LesPtIntegMecaInterne` &lesPtMecaInt)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ, `Loi_comp_abstraite::SaveResul` \*saveDon, list< int > &decal) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ) const
- virtual `Enum_comp_3D_CP_DP_1D` `Comportement_3D_CP_DP_1D` ()
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)

### Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB` \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB` \* > &d\_epsBB\_tdt, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB` \* > &d\_gijBB\_tdt, `Tableau`< `TenseurHH` \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH` \* > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecaInterne` &ptintmeca, const `Deformation` &def, `Enum_dure` temps, const `ThermoDonnee` &dTP, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque` \* > \*exclure\_Q)
- virtual void `IndiquePtIntegMecaInterne` (const `PtIntegMecaInterne` \*ptintmeca)
- void `RepercuteChangeTemperature` (`Enum_dure` temps)

### Attributs protégés

- `Loi_comp_abstraite` \* `lois_internes1`
- `Loi_comp_abstraite` \* `lois_internes2`
- `Tableau`< int > `calcule_si_prop_non_nulle`
- `Tableau`< `TenseurHH` \* > `d_sigtotalHH`
- `TenseurHHHH` \* `d_sigma_deps_inter`
- `Ddl_enum_etendu` `type_grandeur`
- bool `valeur_aux_noeuds`
- double `proportion`
- double `aA`
- `Courbe1D` \* `c_proport`
- `Fonction_nD` \* `niveauF_grandeurND`
- `Ponderation` \* `niveauF_ddlEtendu`
- `Ponderation_temps` \* `niveauF_temps`
- `Ponderation_Constable` \* `niveauF_grandeurConsultable`
- int `type_melange`



## Amis

— class [SaveResul\\_LoiDesMelangesEnSigma](#)

## Membres hérités additionnels

### 6.480.1 Documentation des fonctions membres

#### 6.480.1.1 Activation\_donnees()

```
void LoiDesMelangesEnSigma::Activation_donnees (
    Tableau< Noeud * > & tabnoeud,
    bool dilatation,
    LesPtIntegMecaInterne & lesPtMecaInt ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.480.1.2 Affiche()

```
void LoiDesMelangesEnSigma::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.480.1.3 Calcul\_dsigma\_deps()

```
void LoiDesMelangesEnSigma::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.480.1.4 Calcul\_DsigmaHH\_tdt()

```
void LoiDesMelangesEnSigma::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
```

```

TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.480.1.5 Calcul\_SigmaHH()

```

void LoiDesMelangesEnSigma::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.480.1.6 CalculGrandeurTravail()

```

void LoiDesMelangesEnSigma::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,
    const ThermoDonnee & dTP,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.480.1.7 Comportement\_3D\_CP\_DP\_1D()

`Enum_comp_3D_CP_DP_1D` LoiDesMelangesEnSigma::Comportement\_3D\_CP\_DP\_1D ( ) [virtual]

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

### 6.480.1.8 Ecriture\_base\_info\_loi()

```
void LoiDesMelangesEnSigma::Ecriture_base_info_loi (
    ostream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.480.1.9 Grandeur\_particuliere()

```
void LoiDesMelangesEnSigma::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & liTQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.480.1.10 HsurH0()

```
virtual double LoiDesMelangesEnSigma::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.480.1.11 IndiquePtIntegMecaInterne()

```
virtual void LoiDesMelangesEnSigma::IndiquePtIntegMecaInterne (
    const PtIntegMecaInterne * ptintmeca ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.480.1.12 Info\_commande\_LoisDeComp()

```
void LoiDesMelangesEnSigma::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.480.1.13 Lecture\_base\_info\_loi()

```
void LoiDesMelangesEnSigma::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.480.1.14 LectureDonneesParticulieres()

```
void LoiDesMelangesEnSigma::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
```

```

    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.480.1.15 ListeGrandeurs\_particulieres()

```

void LoiDesMelangesEnSigma::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.480.1.16 Module\_compressibilite\_equivalent()

```

double LoiDesMelangesEnSigma::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.480.1.17 Module\_young\_equivalent()

```

double LoiDesMelangesEnSigma::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.480.1.18 New\_et\_Initialise()

```

LoiDesMelangesEnSigma::SaveResul * LoiDesMelangesEnSigma::New_et_Initialise ( ) [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.480.1.19 Nouvelle\_loi\_identique()

```

Loi_comp_abstraite * LoiDesMelangesEnSigma::Nouvelle_loi_identique ( ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.480.1.20 RepercuteChangeTemperature()

```

void LoiDesMelangesEnSigma::RepercuteChangeTemperature (
    Enum_dure temps ) [protected], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.480.1.21 TestComplet()

```

int LoiDesMelangesEnSigma::TestComplet ( ) [virtual]

```

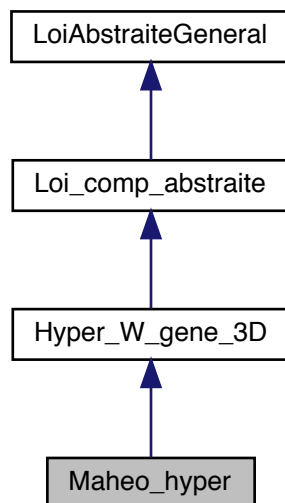
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

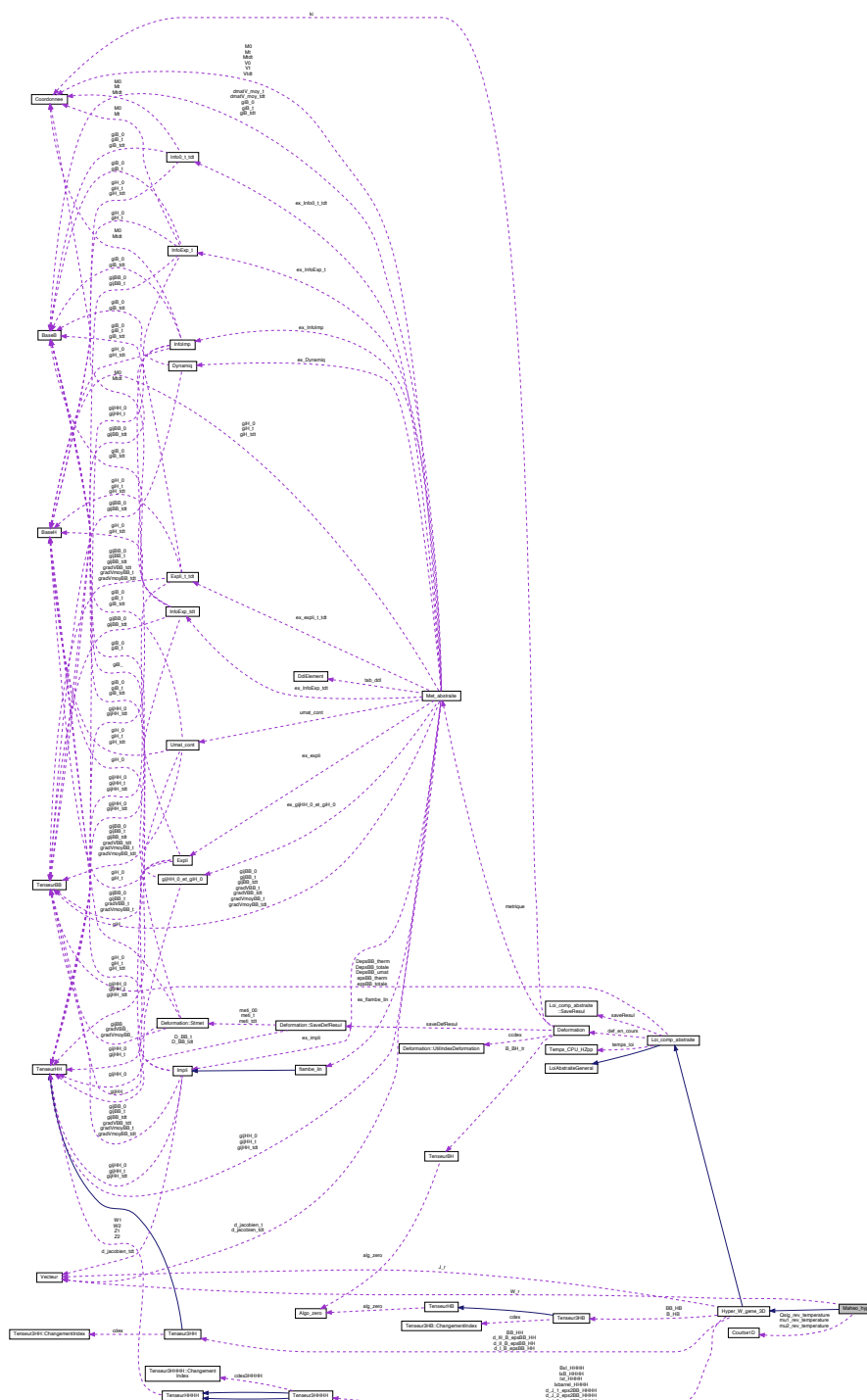
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiDesMelanges↔EnSigma.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiDesMelanges↔EnSigma.cc

## 6.481 Référence de la classe Maheo\_hyper

Graphe d'héritage de Maheo\_hyper:



Graphe de collaboration de Maheo\_hyper:



## Fonctions membres publiques

- **Maheo\_hyper** (const [Maheo\\_hyper](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)

- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure temps, const [Deformation](#) &, [SaveResul](#) \*saveResul)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gij←  
BB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_,  
[TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double  
&jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module←  
compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#)  
&giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#)  
&giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_eps←  
BB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* >  
&d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#)  
&d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const  
[EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met](#)←  
[abstraite::Impli](#) &ex)
- void [Calcul\\_dsigma\\_deps](#) (bool en\_base\_orthonormee, [TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB,  
[TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, double &jacobien\_0, double &jacobien, [TenseurHH](#)  
&sigHH, [TenseurHHHH](#) &d\_sigma\_deps, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double  
&module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Umat\\_cont](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const  
[ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const  
[Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const  
[TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- double [Qsig\\_rev](#)
- double [mu1\\_rev](#)
- double [mu2\\_rev](#)
- [Courbe1D](#) \* [Qsig\\_rev\\_temperature](#)
- [Courbe1D](#) \* [mu1\\_rev\\_temperature](#)
- [Courbe1D](#) \* [mu2\\_rev\\_temperature](#)
- double [fact\\_regularisation](#)
- double [W\\_d](#)
- double [W\\_v](#)
- [Vecteur](#) [W\\_r](#)
- double [W\\_d\\_J1](#)
- double [W\\_d\\_J2](#)
- double [W\\_d\\_J1\\_2](#)
- double [W\\_d\\_J1\\_J2](#)
- double [W\\_d\\_J2\\_2](#)
- double [W\\_v\\_J3](#)
- double [W\\_v\\_J3J3](#)
- [Tableau2](#)< double > [W\\_rs](#)

## Membres hérités additionnels

### 6.481.1 Documentation des fonctions membres

#### 6.481.1.1 Affiche()

void [Maheo\\_hyper::Affiche](#) ( ) const [virtual]  
Implémente [LoiAbstraiteGeneral](#).

### 6.481.1.2 Calcul\_dsigma\_deps()

```
void Maheo_hyper::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
Tenseur3HHHH d_sigma_depsHHHH; d_sigma_depsHHHH.TransfertDunTenseurGeneral(dSigdepsHHHH.↵
Symetrise1et2_3et4());
Réimplémentée à partir de Loi\_comp\_abstraite.
```

### 6.481.1.3 Calcul\_DsigmaHH\_tdt()

```
void Maheo_hyper::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
Implémente Loi\_comp\_abstraite.
```

### 6.481.1.4 Calcul\_SigmaHH()

```
void Maheo_hyper::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
```



```

TenseurBB & DepsBB,
DdlElement & tab_ddl,
TenseurBB & gijBB_t,
TenseurHH & gijHH_t,
BaseB & giB,
BaseH & gi_H,
TenseurBB & epsBB_,
TenseurBB & delta_epsBB_,
TenseurBB & gijBB_,
TenseurHH & gijHH_,
Tableau< TenseurBB * > & d_gijBB_,
double & jacobien_0,
double & jacobien,
TenseurHH & sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.481.1.5 CalculGrandeurTravail()

```

virtual void Maheo_hyper::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.481.1.6 Ecriture\_base\_info\_loi()

```

void Maheo_hyper::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.481.1.7 HsurH0()

```

virtual double Maheo_hyper::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.481.1.8 Info\_commande\_LoisDeComp()

```

void Maheo_hyper::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

### 6.481.1.9 Lecture\_base\_info\_loi()

```
void Maheo_hyper::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.481.1.10 LectureDonneesParticulieres()

```
void Maheo_hyper::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.481.1.11 Module\_young\_equivalent()

```
double Maheo_hyper::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.481.1.12 Nouvelle\_loi\_identique()

```
Loi\_comp\_abstraite * Maheo_hyper::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.481.1.13 TestComplet()

```
int Maheo_hyper::TestComplet ( ) [virtual]
```

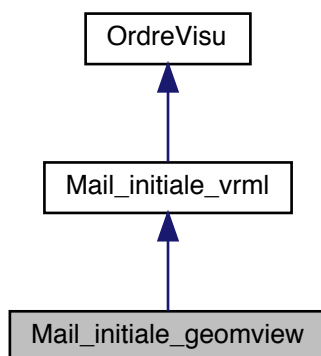
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

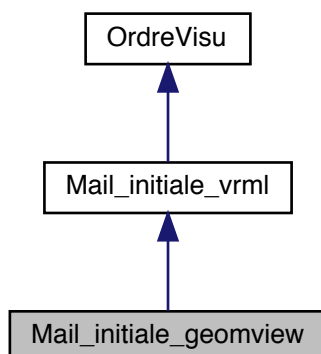
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Maheo\_hyper.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Maheo\_hyper.cc

## 6.482 Référence de la classe Mail\_initiale\_geomview

Graphe d'héritage de Mail\_initiale\_geomview:



Graphe de collaboration de Mail\_initiale\_geomview:



### Fonctions membres publiques

- **Mail\_initiale\_geomview** (const [Mail\\_initiale\\_geomview](#) &algo)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔Glob)

### Membres hérités additionnels

#### 6.482.1 Documentation des fonctions membres

### 6.482.1.1 ExeOrdre()

```
void Mail_initiale_geomview::ExeOrdre (
    ParaGlob * paraGlob,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

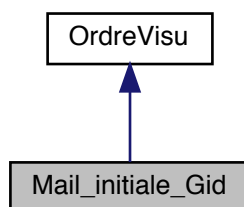
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

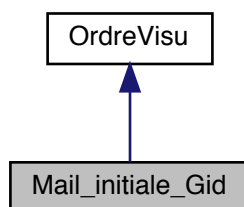
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Mail\_initiale\_geomview.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Mail\_initiale\_geomview.cc

## 6.483 Référence de la classe Mail\_initiale\_Gid

Graphe d'héritage de Mail\_initiale\_Gid:



Graphe de collaboration de Mail\_initiale\_Gid:



## Fonctions membres publiques

- **Mail\_initiale\_Gid** (const [Mail\\_initiale\\_Gid](#) &algo)
- void [Initialisation](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, EnumTypeIncre type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec←Glob, bool fil\_calcul)
- void [ExeOrdre](#) ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, [OrdreVisu](#)::EnumTypeIncre type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec←Glob)
- void [ChoixOrdre](#) ()
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- const [Tableau](#)< [Tableau](#)< [List\\_io](#)< Element \* > > > & [Tableau\\_de\\_sous\\_maillage](#) () const
- const [Tableau](#)< [List\\_io](#)< Element::Signature > > & [Tab\\_liste\\_type\\_element](#) () const
- const [Tableau](#)< [List\\_io](#)< string > > & [Tab\\_listes\\_nom\\_sousMaillage](#) () const
- int [DecalNumNoeud](#) (int nmail) const
- int [DecalNumElement](#) (int nmail) const

## Membres hérités additionnels

### 6.483.1 Documentation des fonctions membres

#### 6.483.1.1 ChoixOrdre()

```
void Mail_initiale_Gid::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.483.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Mail_initiale_Gid::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.483.1.3 ExeOrdre()

```
void Mail_initiale_Gid::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * lesRef,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.483.1.4 Initialisation()

```
void Mail_initiale_Gid::Initialisation (
    ParaGlob * paraGlob,
    LesMaillages * lesmail,
    LesReferences * lesRef,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
    EnumTypeIncre type_incre,
    int incre,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List_io< TypeQuelconque > & listeVecGlob,
    bool fil_calcul ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.483.1.5 Lecture\_parametres\_OrdreVisu()

```
void Mail_initiale_Gid::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

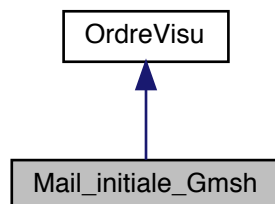
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

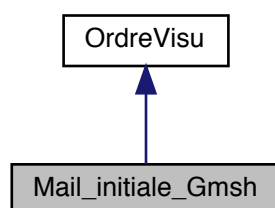
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Mail\_initiale\_Gid.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Mail\_initiale\_Gid.cc

## 6.484 Référence de la classe Mail\_initiale\_Gmsh

Graphe d'héritage de Mail\_initiale\_Gmsh:



Graphe de collaboration de Mail\_initiale\_Gmsh:



### Fonctions membres publiques

- **Mail\_initiale\_Gmsh** (const [Mail\\_initiale\\_Gmsh](#) &algo)
- void [Initialisation](#) ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, EnumTypeIncre type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔ Glob, bool fil\_calcul)
- void [ExeOrdre](#) ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, OrdreVisu::EnumTypeIncre type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec↔ Glob)
- void [ChoixOrdre](#) ()
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- const [Tableau](#)< string > \* **NomsGrandeurSortie** () const
- const [Tableau](#)< [Tableau](#)< [List\\_io](#)< Element \* > > > & **Tableau\_de\_sous\_maillage** () const
- const [Tableau](#)< [List\\_io](#)< Element::Signature > > & **Tab\_liste\_type\_element** () const
- const [Tableau](#)< [List\\_io](#)< string > > & **Tab\_listes\_nom\_sousMaillage** () const
- int **DecalNumNoeud** (int nmail) const
- int **DecalNumElement** (int nmail) const
- const [Tableau](#)< [Tableau](#)< int > > & **TabEgmsh** () const
- bool **Considerer\_homothetie** () const
- const [Tableau](#)< bool > & **T\_homothetie** () const

- const [Tableau](#)< [Coordonnee](#) > & [T\\_orig](#) () const
- const [Tableau](#)< [Coordonnee](#) > & [T\\_fact\\_mult](#) () const
- void [sortie\\_maillage\\_initial](#) (const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*lesMail, ostream &sort)
- int [Nb\\_total\\_noeud](#) () const
- int [Nb\\_total\\_element](#) () const

## Membres hérités additionnels

### 6.484.1 Documentation des fonctions membres

#### 6.484.1.1 ChoixOrdre()

```
void Mail_initiale_Gmsh::ChoixOrdre ( ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.484.1.2 Ecriture\_parametres\_OrdreVisu()

```
void Mail_initiale_Gmsh::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.484.1.3 ExeOrdre()

```
void Mail_initiale_Gmsh::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * lesRef,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & entreePrinc,
    OrdreVisu::EnumTypeIncre type_incre,
    int incre,
    bool animation,
    const map< string, const double *, std::less< string > > & listeVarGlob,
    const List\_io< TypeQuelconque > & listeVecGlob ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.484.1.4 Initialisation()

```
void Mail_initiale_Gmsh::Initialisation (
    ParaGlob * paraGlob,
    LesMaillages * lesmail,
    LesReferences * lesRef,
    LesLoisDeComp * lesLoisDeComp,
    DiversStockage * diversStockage,
    Charge * charge,
    LesCondLim * lesCondLim,
    LesContacts * lesContacts,
    Resultats * resultats,
```



```
EnumTypeIncre type_incre,  
int incre,  
const map< string, const double *, std::less< string > > & listeVarGlob,  
const List_io< TypeQuelconque > & listeVecGlob,  
bool fil_calcul ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.484.1.5 Lecture\_parametres\_OrdreVisu()

```
void Mail_initiale_Gmsh::Lecture_parametres_OrdreVisu (  
    UtilLecture & entreePrinc ) [virtual]
```

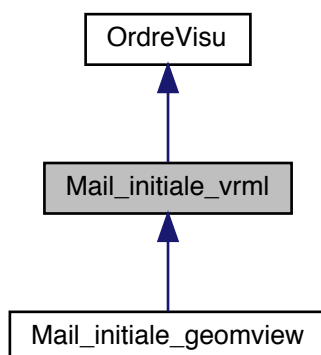
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

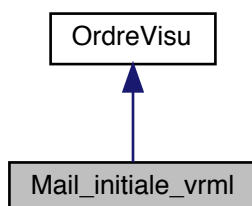
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Mail\_initiale\_Gmsh.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Mail\_initiale\_Gmsh.cc

## 6.485 Référence de la classe Mail\_initiale\_vrml

Grappe d'héritage de Mail\_initiale\_vrml:



Grappe de collaboration de Mail\_initiale\_vrml:



## Fonctions membres publiques

- `Mail_initiale_vrml` (const `Mail_initiale_vrml` &algo)
- void `ExeOrdre` (`ParaGlob` \*, const `Tableau`< int > &tab\_mail, `LesMaillages` \*, bool unseul\_incre, `LesReferences` \*, `LesLoisDeComp` \*, `DiversStockage` \*, `Charge` \*, `LesCondLim` \*, `LesContacts` \*, `Resultats` \*, `UtilLecture` &entreePrinc, `OrdreVisu::EnumTypeIncre` type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const `List_io`< `TypeQuelconque` > &listeVec←Glob)
- void `ChoixOrdre` ()
- void `Lecture_parametres_OrdreVisu` (`UtilLecture` &entreePrinc)
- void `Ecriture_parametres_OrdreVisu` (`UtilLecture` &entreePrinc)

## Attributs protégés

- bool `filaire`
- bool `surface`
- bool `numero`
- double `Rcoull`
- double `Gcoull`
- double `Bcoull`
- double `Rcoulf`
- double `Gcoulf`
- double `Bcoulf`
- double `Rcouln`
- double `Gcouln`
- double `Bcouln`
- double `taille_numero`

## Membres hérités additionnels

### 6.485.1 Documentation des fonctions membres

#### 6.485.1.1 ChoixOrdre()

void `Mail_initiale_vrml::ChoixOrdre` () [virtual]  
Réimplémentée à partir de `OrdreVisu`.

#### 6.485.1.2 Ecriture\_parametres\_OrdreVisu()

void `Mail_initiale_vrml::Ecriture_parametres_OrdreVisu` (  
    `UtilLecture` & `entreePrinc` ) [virtual]  
Réimplémentée à partir de `OrdreVisu`.

#### 6.485.1.3 ExeOrdre()

```
void Mail_initiale_vrml::ExeOrdre (
    ParaGlob * paraGlob,
    const Tableau< int > & tab_mail,
    LesMaillages * lesMail,
    bool unseul_incre,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
```

```

UtilLecture & entreePrinc,
OrdreVisu::EnumTypeIncre type_incre,
int incre,
bool animation,
const map< string, const double *, std::less< string > > & listeVarGlob,
const List_io< TypeQuelconque > & listeVecGlob ) [virtual]

```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.485.1.4 Lecture\_parametres\_OrdreVisu()

```

void Mail_initiale_vrml::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]

```

Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

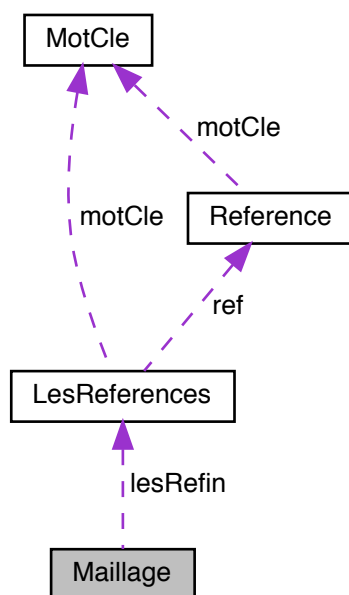
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Mail\_initiale\_vrml.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Mail\_initiale\_vrml.cc

## 6.486 Référence de la classe Maillage

[Maillage](#): un maillage particulier.

```
#include <Maillage.h>
```

Graphe de collaboration de Maillage:



### Classes

- class [NBelemEtArete](#)
- class [NBelemEtFace](#)
- class [NBelemEtptInteg](#)
- class [NBelemFAEtptInteg](#)
- class [Noeud\\_degre](#)
- class [PosiEtNoeud](#)

## Fonctions membres publiques

- **Maillage** (map< string, int, std::less< string > > &lisNomMail, int nmail=1, int dim=3, const string &nom↵\_maillage=".")
- **Maillage** (map< string, int, std::less< string > > &lisNomMail, int dim, int n\_noeud, int n\_elt, int nmail, const string &nom\_maillage=".")
- **Maillage** (map< string, int, std::less< string > > &lisNomMail, int dim, list< [Noeud](#) \* > &li\_noeud, list< Element \* > li\_element, int nmail, const string &nom\_maillage=".")
- **Maillage** (const [Maillage](#) &mail)
- **Maillage** (map< string, int, std::less< string > > &lisNomMail, int nmail, const string &nomDuMaillage, const [Maillage](#) &mail)
- void **LectureMaillage** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) &lesRef)
- void **LectureEtApplicationAffinage** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) &lesRef)
- void **Ajout\_de\_Noeuds** (const list< [Noeud](#) \* > &taN, list< const [Reference](#) \* > \*lref=NULL, [LesReferences](#) \*lesRef=NULL)
- void **Ajout\_elements\_et\_noeuds** (const list< [Noeud](#) \* > &taN, const list< Element \* > &taE, list< const [Reference](#) \* > \*lref, [LesReferences](#) \*lesRef)
- void **Info\_commande\_Maillages** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) &lesRef, int cas)
- void **Affiche** () const
- void **Affiche\_dans\_her\_lis** ([LesReferences](#) &lesRef, [Enum\\_dure](#) temps)
- void **Restreint\_sous\_maillage** ([LesReferences](#) \*lesRef, string nom\_ref)
- [Maillage](#) & operator= ([Maillage](#) &mail)
- int **Dimension** () const
- const [Tableau](#)< [Enum\\_ddl](#) > & **Ddl\_representatifs\_des\_physiques** () const
- const [Tableau](#)< [EnumElemTypeProblem](#) > & **Types\_de\_problemes** () const
- int **Nombre\_noeud\_elt** (int i) const
- int **Nombre\_noeud** () const
- int **Nombre\_element** () const
- [Tableau](#)< [Noeud](#) \* > & **Tab\_noeud** ()
- [Tableau](#)< Element \* > & **Tab\_element** ()
- [Noeud](#) & **Noeud\_mail** (int i)
- Element & **Element\_mail** (int i)
- const Element & **Element\_mail\_const** (int i)
- bool **Complet** ()
- void **Largeur\_Bande** (int &demi, int &total, const [Nb\\_assemb](#) &nb\_casAssemb)
- int **Tous\_Xi\_fixes** (const [Nb\\_assemb](#) &casAss) const
- void **InitNormaleAuxNoeuds** ()
- void **MiseAJourNormaleAuxNoeuds** ()
- void **MiseAJourNormaleAuxNoeuds\_de\_tdt\_vers\_T** ()
- void **CreeElemFront** ()
- LaLIST< [Front](#) > \* **ListFront** ()
- [Tableau](#)< [Noeud](#) \* > & **Tab\_noeud\_front** ()
- string **NomDuMaillage** ()
- void **ChangeNomNumeroMaillage** (const string &nom, int num)
- int **Noeud\_le\_plus\_proche\_0** (const [Coordonnee](#) &M)
- int **Noeud\_le\_plus\_proche\_t** (const [Coordonnee](#) &M)
- int **Noeud\_le\_plus\_proche\_tdt** (const [Coordonnee](#) &M)
- double **Max\_var\_dep\_t\_a\_tdt** () const
- double **Min\_dist2Noeud\_des\_elements** ([Enum\\_dure](#) temps) const
- void **AjoutConteneurAuNoeud** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lienu, const [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > \* > &tabQ)
- void **InitUpdateAuNoeud** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lienu, const [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > \* > &tabQ, int cas)
- void **TransfertPtIntegAuNoeud** (Element &ele, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lietendu, const [Tableau](#)< [Tableau](#)< double > > &tab\_val, int cas)
- void **TransfertPtIntegAuNoeud** (Element &ele, const [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > > &tab\_liQ, [List\\_io](#)< [TypeQuelconque](#) > &liQ\_travail, int cas)
- void **FinTransfertPtIntegAuNoeud** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lienu, const [Tableau](#)< [List\\_io](#)< [TypeQuelconque](#) > \* > &tabQ, int cas)
- void **Accumul\_aux\_noeuds** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lietendu, [List\\_io](#)< [TypeQuelconque](#) > &liQ, int cas)
- void **AjoutConteneurAuNoeud** ([TypeQuelconque](#) &tQ)
- void **AjoutConteneurAuNoeud** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lienu)
- void **InitUpdateAuNoeud** (const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &lienu)

- void **InitUpdateAuNoeud** (const [Ddl\\_enum\\_etendu](#) &enu)
- void **MoyenneCompteurAuNoeud** (const [Ddl\\_enum\\_etendu](#) &enu)
- void **Init\_Xi\_t\_et\_tdt\_de\_0** ()
- **NBelemEtpthnteg Element\_le\_plus\_proche** ([Enum\\_dure](#) enu\_temps, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &list\_enu, const [Coordonnee](#) &M)
- const Element \* **Centre\_de\_Gravite\_Element\_le\_plus\_proche** ([Enum\\_dure](#) enu\_temps, const [Coordonnee](#) &M)
- const [Tableau](#)< [List\\_io](#)< Element \* > > & **Indice** ()
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- [Tableau](#)< [Vecteur](#) > **Taille\_boiteMail** ()
- bool **Contient\_lineaire** ()
- bool **Contient\_quadratique\_incomplet** ()
- void **Transfo\_lin\_quadraIncomp** ([LesReferences](#) &lesRef)
- void **Transfo\_quadraIncomp\_quadraComp** ([LesReferences](#) &lesRef)
- void **RelocPtMilieuMailleQuadra** ()
- void **CreationInteractiveListesRef** ([LesReferences](#) \*lesRef)
- void **Modif\_orientation\_element** (int cas\_orientation, [LesReferences](#) \*lesRef)
- void **LectureEtCollapse\_element\_superpose** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) \*lesRef)
- void **Collapse\_element\_superpose** ([LesReferences](#) \*lesRef)
- void **LectureEtCollapse\_noeuds\_proches** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) \*lesRef)
- void **Collapse\_noeuds\_proches** (double rayon, [LesReferences](#) \*lesRef)
- void **LectureEtSup\_Elem\_noeudsConfondu** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) \*lesRef)
- void **Sup\_Elem\_noeudsConfondu** (double rayon, [LesReferences](#) \*lesRef)
- void **CreationMaillageSFE** ()
- bool **OKPourTransSfe** ()
- void **LectureEtSuppressionNoeudNonReferencer** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) &lesRef)
- void **SuppressionNoeudNonReferencer** ([LesReferences](#) &lesRef)
- void **AffichageNoeudNonReferencer** ()
- void **LectureEtCreationRefNoeudNonReferencer** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) &lesRef)
- void **CreationRefNoeudNonReferencer** ([LesReferences](#) &lesRef)
- void **VerifReference** ([LesReferences](#) &lesRef)
- void **LectureEtRenumerotation** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) &lesRef)
- bool **Renumerotation** ([LesReferences](#) &lesRef, const [Tableau](#)< [Tableau](#)< [Condilinaire](#) > > &condCLL)
- void **CreationRefFrontiere** ([UtilLecture](#) \*entreePrinc, [LesReferences](#) &lesRef)
- void **CreationRefFrontiere** ([LesReferences](#) &lesRef)
- void **Force\_Ddl\_aux\_noeuds\_a\_une\_valeur** ([Enum\\_ddl](#) enu, const double &val, [Enum\\_dure](#) temps, bool fonction\_de\_la\_dimension)
- void **Force\_Ddl\_etendu\_aux\_noeuds\_a\_zero** (const [Tableau](#)< [Ddl\\_enum\\_etendu](#) > &tab\_enu)

## Fonctions membres publiques statiques

- static void **SchemaXML\_Maillages** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)

## Fonctions membres protégées

- void **Preparation\_destruction\_avec\_conservation\_noeuds\_elements** ()
- void **Change\_numero\_maillage** (int new\_num)
- void **MitoyenFront** ()
- void **Calcul\_indice** ()
- void **Calcul\_tous\_les\_front\_et\_leurs\_mitoyens** ()
- void **OrdonancementDesLigne** ()
- void **Lecture\_info\_1element** ([UtilLecture](#) \*entreePrinc, int &num\_elt, [Enum\\_geom](#) &id\_geom, [Enum\\_interpol](#) &id\_interpol, [EnumElemTypeProblem](#) &id\_typeProb, string &discriminant)
- void **Lecture\_des\_mouvements\_solides** ([UtilLecture](#) \*entreePrinc)
- void **Affectation\_noeud** ([Noeud](#) &noeud)
- void **Affectation\_element** ([Element](#) &element)
- void **Change\_nb\_noeud** (int nouveau\_nb)
- void **Change\_nb\_element** (int nouveau\_nb)
- list< int > **Orientation\_elements\_mitoyens\_recuratif** (bool recursif, string &nom\_ref, int num\_elem, [LesReferences](#) &lesRef, double &angle\_maxi, bool inverse)
- void **Init\_Orientation\_elements\_mitoyens\_recuratif** ()
- void **Orientation\_via\_rayon** (const [Tableau](#)< bool > &indic\_G, const [Coordonnee](#) &A, const string &zone←\_a\_traiter, [LesReferences](#) &lesRef, bool inverse)

- void **CalculListRef\_1D** (list< string > &list\_nomReference, [LesReferences](#) \*lesRef, list< string > &list\_↵ methode, const Enum\_ddl &enu\_ddl)
- void **CalculListRef\_2D** (list< string > &list\_nomReference, [LesReferences](#) \*lesRef, list< string > &list\_↵ methode, const Enum\_ddl &enu\_ddl)
- void **CalculListRef\_3D** (list< string > &list\_nomReference, [LesReferences](#) \*lesRef, list< string > &list\_↵ methode, const Enum\_ddl &enu\_ddl)
- void **PresDe** (const list< string > &list\_nomReference, list< [Noeud](#) \* > &list\_noeud\_restant, list< [Element](#) \* > &list\_element\_restant, const Enum\_ddl &enu\_ddl, list< [NBelemEptInteg](#) > &list\_elemPtin\_restant, bool &premLpti, list< [NBelemEtFace](#) > &list\_elemFace\_restant, bool &premLface, list< [NBelemEtArete](#) > &list\_↵\_elemArrete\_restant, bool &premLarrete)
- void **ToutDedans** (const list< string > &list\_nomReference, list< [Noeud](#) \* > &list\_noeud\_restant, list< [Element](#) \* > &list\_element\_restant, const Enum\_ddl &enu\_ddl, list< [NBelemEptInteg](#) > &list\_elemPtin\_restant, bool &premLpti, list< [NBelemEtFace](#) > &list\_elemFace\_restant, bool &premLface, list< [NBelemEtArete](#) > &list\_↵\_elemArrete\_restant, bool &premLarrete)
- void **EnregRef** (const list< string > &list\_nomReference, list< [Noeud](#) \* > &list\_noeud\_restant, list< [NBelemEptInteg](#) > &list\_elemPtin\_restant, list< [Element](#) \* > &list\_element\_restant, list< [NBelemEtFace](#) > &list\_elemFace\_restant, list< [NBelemEtArete](#) > &list\_elemArrete\_restant, [LesReferences](#) \*lesRef)
- void **InitPresPoint** (double &dist)
- bool **ExePresPoint** (const double &dist, const [Coordonnee](#) &M) const
- void **InitCotePoint** ()
- bool **ExeCotePoint** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntrePoint** ()
- bool **ExeEntrePoint** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntrePoint\_avec\_distance** ()
- bool **ExeEntrePoint\_avec\_distance** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitPresLigne** (double &dist)
- bool **ExePresLigne** (const double &dist, const [Coordonnee](#) &M) const
- void **InitOrdonneLigne** ()
- void **OrdonneLigne** (list< [Noeud](#) \* > &list\_noeud\_restant, list< [NBelemEptInteg](#) > &list\_elemPtin\_restant, const Enum\_ddl &enu\_ddl)
- void **InitPresCercle** (double &dist)
- bool **ExePresCercle** (const double &dist, const [Coordonnee](#) &M) const
- void **InitPresPlan** (double &dist)
- bool **ExePresPlan** (const double &dist, const [Coordonnee](#) &M) const
- void **InitPresCylindre** (double &dist)
- bool **ExePresCylindre** (const double &dist, const [Coordonnee](#) &M) const
- void **InitPresSphere** (double &dist)
- bool **ExePresSphere** (const double &dist, const [Coordonnee](#) &M) const
- void **InitCotePlan** ()
- bool **ExeCotePlan** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntrePlan** ()
- bool **ExeEntrePlan** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntrePlan\_avec\_distance** ()
- bool **ExeEntrePlan\_avec\_distance** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitDansCylindre** ()
- bool **ExeDansCylindre** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- bool **ExeOutCylindre** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntreCylindre** ()
- bool **ExeEntreCylindre** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitDansSphere** ()
- bool **ExeDansSphere** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- bool **ExeOutSpheres** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntreSpheres** ()
- bool **ExeEntreSpheres** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitCoteDroite** ()
- bool **ExeCoteDroite** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntreDroite** ()
- bool **ExeEntreDroite** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntreDroite\_avec\_distance** ()
- bool **ExeEntreDroite\_avec\_distance** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitDansCercle** ()
- bool **ExeDansCercle** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- bool **ExeOutCercle** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **InitEntreCercles** ()

- bool **ExeEntreCercles** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- [Plan](#) **Acquisition\_interactive\_plan** ()
- [Coordonnee](#) **Acquisition\_interactive\_point** ()
- [Droite](#) **Acquisition\_interactive\_droite** ()
- [Coordonnee](#) **Acquisition\_interactive\_vecteur** ()
- void **InitInRef** ()
- void **InitOutRef** ()
- bool **Exe\_In\_out\_avecRefExistantes** (const [Tableau](#)< [Coordonnee](#) > &t\_M) const
- void **Exe\_In\_out\_avecRefExistantes\_N** (list< [Noeud](#) \* > &list\_noeud\_restant)
- void **Exe\_In\_out\_avecRefExistantes\_E** (list< [Element](#) \* > &list\_element\_restant)
- void **Exe\_In\_out\_avecRefExistantes\_G** (list< [NBelemEptInteg](#) > &list\_elemPtin\_restant)
- void **Exe\_In\_out\_avecRefExistantes\_F** (list< [NBelemEtFace](#) > &list\_elemFace\_restant)
- void **Exe\_In\_out\_avecRefExistantes\_A** (list< [NBelemEtArete](#) > &list\_elemArrete\_restant)
- void **Mise\_a\_jour\_type\_pb\_type\_associe\_ddl** ()

### Fonctions membres protégées statiques

- static [Noeud](#) \* **Point\_de\_depart** (const [Tableau](#)< [Element](#) \* > &tab\_elem, const [Tableau](#)< [Noeud](#) \* > &tab\_noe, const [Tableau](#)< [Tableau](#)< [Noeud](#) \* > \* > tt\_noeud\_front, [Tableau](#)< [LaLIST\\_io](#)< [Noeud](#) \* > > &t\_voisin, list< list< [Noeud\\_degre](#) > > &lis\_descent, const [Tableau](#)< [Tableau](#)< [Condilinaire](#) > > &cond←  
CLL, bool &calcul\_ok)
- static [Tableau](#)< [LaLIST\\_io](#)< [Noeud](#) \* > > & **Voisins** (const [Tableau](#)< [Element](#) \* > &tab\_elem, const [Tableau](#)< [Noeud](#) \* > &tab\_noe, [Tableau](#)< [LaLIST\\_io](#)< [Noeud](#) \* > > &t\_voisin, const [Tableau](#)< [Tableau](#)< [Condilinaire](#) > > &t\_t\_condCLL, bool &calcul\_ok)
- static list< list< [Noeud\\_degre](#) > > & **Descendance** (const int &taille\_tabnoeud, [Noeud](#) \*noeu, [Tableau](#)< [LaLIST\\_io](#)< [Noeud](#) \* > > &t\_voisin, list< list< [Noeud\\_degre](#) > > &lient, bool &calcul\_ok)
- static [Tableau](#)< [Noeud](#) \* > **Cuthill\_Mac\_Kee** (const int &taille\_tabnoeud, [Noeud](#) \*noeu, [Tableau](#)< [LaLIST\\_io](#)< [Noeud](#) \* > > &t\_voisin, list< list< [Noeud\\_degre](#) > > &lis\_descent)
- static int **LargeurBandeEnNoeuds** (const [Tableau](#)< [Element](#) \* > &tab\_elem)

### Attributs protégés

- int **idmail**
- string **nomDuMaillage**
- map< string, int, std::less< string > > & **listeNomMail**
- int **dimension**
- [Tableau](#)< [Noeud](#) \* > **tab\_noeud**
- [Tableau](#)< [Element](#) \* > **tab\_element**
- [LaLIST](#)< [Front](#) > **listFrontiere**
- [Tableau](#)< [Noeud](#) \* > **tab\_noeud\_front**
- [Tableau](#)< [List\\_io](#)< [Element](#) \* > > **indice**
- [Tableau](#)< [Tableau](#)< [Front](#) > > **mitoyen\_de\_chaque\_element**
- [Tableau](#)< [Enum\\_ddl](#) > **ddl\_representatifs\_des\_physiques**
- [Tableau](#)< [EnumElemTypeProblem](#) > **types\_de\_problemes**
- bool **destruire\_les\_noeuds\_et\_elements**
- [Tableau](#)< double > **tab\_sens\_element**
- [Tableau](#)< bool > **ind\_elem**
- [Tableau](#)< [Coordonnee](#) > **t\_poi**
- [Tableau](#)< [Droite](#) > **t\_droit**
- [Tableau](#)< double > **t\_para**
- [Tableau](#)< [Plan](#) > **t\_plan**
- [Tableau](#)< [Sphere](#) > **t\_sphere**
- [Tableau](#)< [Cylindre](#) > **t\_cylindre**
- [Tableau](#)< [Cercle](#) > **t\_cercle**
- list< const [Reference](#) \* > **list\_refl**
- list< const [Reference](#) \* > **list\_refOut**
- [LesReferences](#) \* **lesRefin**
- void(Maillage::\* **initConditionPresDe** )(double &dist)
- bool(Maillage::\* **ExeConditionPresDe** )(const double &dist, const [Coordonnee](#) &M) const
- void(Maillage::\* **initConditionToutDedans** )()
- bool(Maillage::\* **ExeConditionToutDedans** )(const [Tableau](#)< [Coordonnee](#) > &tab\_M) const

## Amis

— class **LesMaillages**

### 6.486.1 Description détaillée

**Maillage**: un maillage particulier.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

### 6.486.2 Documentation des fonctions membres

#### 6.486.2.1 Calcul\_indice()

```
void Maillage::Calcul_indice ( ) [protected]
les 3/ sont pour la version vector qui marche bien aussi
vector <vector <Element*> > indice(tab_noeud.Taille());
indice[(tabn(ij)->Num_noeud()-1)].push_back(elem1);
```

#### 6.486.2.2 CreeElemFront()

```
void Maillage::CreeElemFront ( )
vector<Element*>::iterator ina,inf; vector<Element*>& intertab = indice[numnoeud-1]; // pour alléger l'écriture
```

#### 6.486.2.3 Lecture\_info\_1element()

```
void Maillage::Lecture_info_1element (
    UtilLecture * entreePrinc,
    int & num_elt,
    Enum_geom & id_geom,
    Enum_interpol & id_interpol,
    EnumElemTypeProblem & id_typeProb,
    string & discriminant ) [protected]
```

int cr=(entreePrinc->entree)->peek(); ne marche pas !!!

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage3.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/maillage4.cc

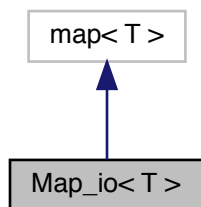
## 6.487 Référence du modèle de la classe Map\_io< T >

**Map\_io** classe template de type map STL avec une lecture et écriture.

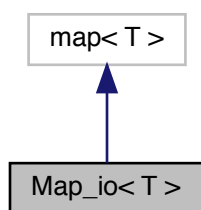
```
#include <Map_io.h>
```



Graphe d'héritage de Map\_io< T > :



Graphe de collaboration de Map\_io< T > :



## Amis

- istream & **operator**>> (istream &entree, [Map\\_io< T >](#) &)
- ostream & **operator**<< (ostream &sort, const [Map\\_io< T >](#) &a)
- istream & **operator**>> (istream &entree, typename map< T >::iterator &)
- ostream & **operator**<< (ostream &sort, const typename map< T >::iterator)
- istream & **operator**>> (istream &entree, typename map< T >::const\_iterator &)
- ostream & **operator**<< (ostream &sort, const typename map< T >::const\_iterator &)

### 6.487.1 Description détaillée

```

template<class T>
class Map_io< T >

```

[Map\\_io](#) classe template de type map STL avec une lecture et écriture.

Auteur

Gérard Rio

Version

1.0

Date

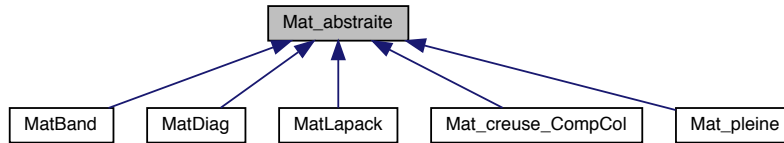
04/01/2007

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Map\_io.h

## 6.488 Référence de la classe Mat\_abstraite

Graphe d'héritage de Mat\_abstraite:



### Fonctions membres publiques

- **Mat\_abstraite** ([Enum\\_matrice](#) type\_mat=RIEN\_MATRICE, [Enum\\_type\\_resolution\\_matri](#) type\_↔ resol=RIEN\_TYPE\_RESOLUTION\_MATRI, [Enum\\_preconditionnement](#) type\_precondi=RIEN\_PRECONDITIONNEMENT)
- **Mat\_abstraite** (const [Mat\\_abstraite](#) &a)
- virtual [Mat\\_abstraite](#) & **operator=** (const [Mat\\_abstraite](#) &)=0
- virtual void **Transfert\_vers\_mat** ([Mat\\_abstraite](#) &A)=0
- virtual void **operator+=** (const [Mat\\_abstraite](#) &mat\_pl)=0
- virtual void **operator-=** (const [Mat\\_abstraite](#) &mat\_pl)=0
- virtual void **operator\*=** (const double r)=0
- virtual int **operator==** (const [Mat\\_abstraite](#) &mat\_pl) const =0
- int **operator!=** (const [Mat\\_abstraite](#) &mat\_pl) const
- virtual [Mat\\_abstraite](#) \* **NouvelElement** () const =0
- virtual void **Affiche** () const =0
- virtual void **Initialise** (double val\_init)=0
- virtual void **Libere** ()=0
- virtual int **Nb\_ligne** () const =0
- virtual int **Nb\_colonne** () const =0
- virtual int **Symetrie** () const =0
- virtual double & **operator()** (int i, int j)=0
- virtual bool **Existe** (int i, int j) const =0
- virtual double **operator()** (int i, int j) const =0
- virtual [Vecteur](#) & **Ligne\_set** (int i)=0
- virtual [Vecteur](#) **Ligne** (int i) const =0
- virtual [Vecteur](#) **LigneSpe** (int i) const =0
- virtual void **RemplaceLigneSpe** (int i, const [Vecteur](#) &v)=0
- virtual void **MetValLigne** (int i, double val)=0
- virtual [Vecteur](#) **Colonne** (int j) const =0
- virtual [Vecteur](#) **ColonneSpe** (int j) const =0
- virtual void **RemplaceColonneSpe** (int j, const [Vecteur](#) &v)=0
- virtual void **MetValColonne** (int j, double val)=0
- virtual [Vecteur](#) **Resol\_syst** (const [Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)=0
- virtual [Vecteur](#) & **Resol\_systID** ([Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)=0
- virtual [Tableau](#)< [Vecteur](#) > **Resol\_syst** (const [Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)=0
- virtual [Tableau](#)< [Vecteur](#) > & **Resol\_systID** ([Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)=0
- virtual [Vecteur](#) & **Resol\_systID\_2** (const [Vecteur](#) &b, [Vecteur](#) &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)=0
- virtual void **Preparation\_resol** ()=0
- virtual [Vecteur](#) **Simple\_Resol\_syst** (const [Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_↔\_default, const int restart=restart\_default) const =0
- virtual [Vecteur](#) & **Simple\_Resol\_systID** ([Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_↔\_default, const int restart=restart\_default) const =0
- virtual [Tableau](#)< [Vecteur](#) > **Simple\_Resol\_syst** (const [Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_↔\_default, const int maxit=maxit\_default, const int restart=restart\_default) const =0

- virtual [Tableau](#)< [Vecteur](#) > & **Simple\_Resol\_systID** ([Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_defaut, const int maxit=maxit\_defaut, const int restart=restart\_defaut) const =0
- virtual [Vecteur](#) & **Simple\_Resol\_systID\_2** (const [Vecteur](#) &b, [Vecteur](#) &vortie, const double &tol=tol\_defaut, const int maxit=maxit\_defaut, const int restart=restart\_defaut) const =0
- virtual [Vecteur](#) **Prod\_vec\_mat** (const [Vecteur](#) &vec) const =0
- virtual [Vecteur](#) & **Prod\_vec\_mat** (const [Vecteur](#) &vec, [Vecteur](#) &resul) const =0
- virtual [Vecteur](#) **Prod\_mat\_vec** (const [Vecteur](#) &vec) const =0
- virtual [Vecteur](#) & **Prod\_mat\_vec** (const [Vecteur](#) &vec, [Vecteur](#) &resul) const =0
- virtual double **Prod\_Ligne\_vec** (int iligne, const [Vecteur](#) &vec) const =0
- virtual double **Prod\_vec\_col** (int icol, const [Vecteur](#) &vec) const =0
- virtual double **vectT\_mat\_vec** (const [Vecteur](#) &vec1, const [Vecteur](#) &vec2) const =0
- virtual int **Place** () const =0
- virtual void **Affiche1** (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j) const
- virtual void **Affiche2** (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j, int nd) const
- virtual void **Change\_Choix\_resolution** ([Enum\\_type\\_resolution\\_matri](#) type\_resol, [Enum\\_preconditionnement](#) type\_precondi)
- const [Enum\\_matrice](#) **Type\_matrice** () const
- void **Affichage\_ecran** (string entete) const
- [Coordonnee3](#) **MinMaxMoy** (bool affiche) const
- bool **Limitation\_min\_diag** (double seuil\_bas, double val\_a\_imposer)
- [MV\\_Vector](#)< double > **operator\*** (const [MV\\_Vector](#)< double > &vec) const
- virtual [MV\\_Vector](#)< double > **trans\_mult** (const [MV\\_Vector](#)< double > &x) const
- [Tableau](#)< [VeurPropre](#) > \* **V\_Propres** ([Mat\\_abstraite](#) &KG, [Tableau](#)< [VeurPropre](#) > &VP)

### Fonctions membres protégées

- void **Resolution\_syst** (const [Vecteur](#) &b, [Vecteur](#) &sol, const double &tol, const int maxit=300, const int restart=32) const
- void **Resolution\_syst** (const [Tableau](#)< [Vecteur](#) > &b, [Tableau](#)< [Vecteur](#) > &sol, const double &tol, const int maxit=300, const int restart=32) const

### Attributs protégés

- [Enum\\_matrice](#) **type\_matrice**
- [Enum\\_type\\_resolution\\_matri](#) **type\_resolution**
- [Enum\\_preconditionnement](#) **type\_preconditionnement**

### Attributs protégés statiques

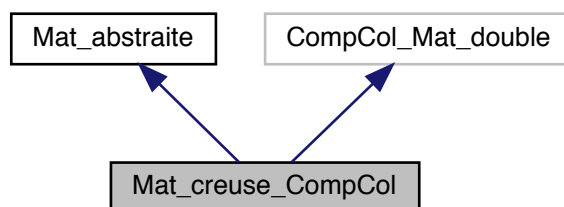
- static const double **tol\_defaut** = 1.E-7
- static const int **maxit\_defaut** = 150
- static const int **restart\_defaut** = 32

La documentation de cette classe a été générée à partir du fichier suivant :

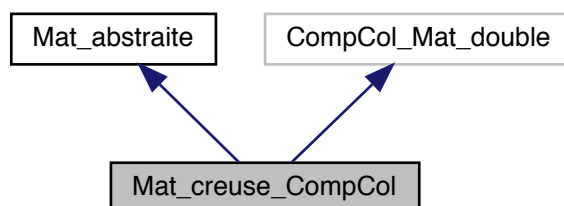
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/Mat\_abstraite.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/Mat\_abstraite.cc

## 6.489 Référence de la classe Mat\_creuse\_CompCol

Graphe d'héritage de Mat\_creuse\_CompCol:



Graphe de collaboration de Mat\_creuse\_CompCol:



### Fonctions membres publiques

- **Mat\_creuse\_CompCol** (int M, int N)
- **Mat\_creuse\_CompCol** (int M, int N, const [Tableau](#)< [Tableau](#)< int > > &petites\_matrices)
- **Mat\_creuse\_CompCol** (const [Mat\\_creuse\\_CompCol](#) &S)
- [Mat\\_abstraite](#) \* **NouvelElement** () const
- [Mat\\_abstraite](#) & **operator=** (const [Mat\\_abstraite](#) &)
- void **Transfert\_vers\_mat** ([Mat\\_abstraite](#) &A)
- [Mat\\_creuse\\_CompCol](#) & **operator=** (const [Mat\\_creuse\\_CompCol](#) &)
- void **operator+=** (const [Mat\\_abstraite](#) &mat\_pl)
- void **operator-=** (const [Mat\\_abstraite](#) &mat\_pl)
- void **operator\*=** (const double r)
- int **operator==** (const [Mat\\_abstraite](#) &mat\_pl) const
- double & **operator()** (int i, int j)
- bool **Existe** (int i, int j) const
- double **operator()** (int i, int j) const
- [Vecteur](#) & **Ligne\_set** (int i)
- [Vecteur](#) **operator()** (int i) const
- [Vecteur](#) **Ligne** (int i) const
- [Vecteur](#) **LigneSpe** (int i) const
- void **RemplaceLigneSpe** (int i, const [Vecteur](#) &v)
- void **MetValligne** (int i, double val)
- [Vecteur](#) **Colonne** (int j) const
- [Vecteur](#) **ColonneSpe** (int j) const
- void **RemplaceColonneSpe** (int j, const [Vecteur](#) &v)

- void `MetValColonne` (int j, double val)
- void `Affiche` () const
- void `Affiche1` (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j) const
- void `Affiche2` (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j, int nd) const
- void `Change_taille` (const int M, const int N, const `Tableau`< `Tableau`< int > > &petites\_matrices)
- void `Change_taille` (const int M, const int N, const `Tableau`< int > &pointeur\_colonne, const int nz=0)
- int `Nb_colonne` () const
- int `Nb_ligne` () const
- void `Initialise` (double a)
- void `Libere` ()
- int `Symetrie` () const
- `Vecteur Resol_syst` (const `Vecteur` &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- `Vecteur & Resol_systID` (`Vecteur` &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- `Tableau`< `Vecteur` > `Resol_syst` (const `Tableau`< `Vecteur` > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- `Tableau`< `Vecteur` > & `Resol_systID` (`Tableau`< `Vecteur` > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- `Vecteur & Resol_systID_2` (const `Vecteur` &b, `Vecteur` &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- void `Preparation_resol` ()
- `Vecteur Simple_Resol_syst` (const `Vecteur` &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- `Vecteur & Simple_Resol_systID` (`Vecteur` &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- `Tableau`< `Vecteur` > `Simple_Resol_syst` (const `Tableau`< `Vecteur` > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- `Tableau`< `Vecteur` > & `Simple_Resol_systID` (`Tableau`< `Vecteur` > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- `Vecteur & Simple_Resol_systID_2` (const `Vecteur` &b, `Vecteur` &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- `Vecteur Prod_vec_mat` (const `Vecteur` &vec) const
- `Vecteur & Prod_vec_mat` (const `Vecteur` &vec, `Vecteur` &resul) const
- `Vecteur Prod_mat_vec` (const `Vecteur` &vec) const
- `Vecteur & Prod_mat_vec` (const `Vecteur` &vec, `Vecteur` &resul) const
- double `Prod_Ligne_vec` (int iligne, const `Vecteur` &vec) const
- double `Prod_vec_col` (int icol, const `Vecteur` &vec) const
- double `vectT_mat_vec` (const `Vecteur` &vec1, const `Vecteur` &vec2) const
- int `Place` () const
- virtual `MV_Vector`< double > **operator\*** (const `MV_Vector`< double > &vec) const
- virtual `MV_Vector`< double > **trans\_mult** (const `MV_Vector`< double > &x) const

## Amis

- `istream & operator>>` (`istream` &, `Mat_creuse_CompCol` &)
- `ostream & operator<<` (`ostream` &, const `Mat_creuse_CompCol` &)

## Membres hérités additionnels

### 6.489.1 Documentation des fonctions membres

#### 6.489.1.1 `Affiche()`

`void Mat_creuse_CompCol::Affiche ( ) const` [virtual]  
 Implémente `Mat_abstraite`.

### 6.489.1.2 Affiche1()

```
void Mat_creuse_CompCol::Affiche1 (
    int min_i,
    int max_i,
    int pas_i,
    int min_j,
    int max_j,
    int pas_j ) const [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

### 6.489.1.3 Affiche2()

```
void Mat_creuse_CompCol::Affiche2 (
    int min_i,
    int max_i,
    int pas_i,
    int min_j,
    int max_j,
    int pas_j,
    int nd ) const [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

### 6.489.1.4 Colonne()

```
Vecteur Mat_creuse_CompCol::Colonne (
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.489.1.5 ColonneSpe()

```
Vecteur Mat_creuse_CompCol::ColonneSpe (
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.489.1.6 Existe()

```
bool Mat_creuse_CompCol::Existe (
    int i,
    int j ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.489.1.7 Initialise()

```
void Mat_creuse_CompCol::Initialise (
    double a ) [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.489.1.8 Libere()

```
void Mat_creuse_CompCol::Libere ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.9 Ligne()**

```
Vecteur Mat_creuse_CompCol::Ligne (
    int i ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.10 Ligne\_set()**

```
Vecteur & Mat_creuse_CompCol::Ligne_set (
    int i ) [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.11 LigneSpe()**

```
Vecteur Mat_creuse_CompCol::LigneSpe (
    int i ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.12 MetValColonne()**

```
void Mat_creuse_CompCol::MetValColonne (
    int j,
    double val ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.13 MetValLigne()**

```
void Mat_creuse_CompCol::MetValLigne (
    int i,
    double val ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.14 Nb\_colonne()**

```
int Mat_creuse_CompCol::Nb_colonne ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.15 Nb\_ligne()**

```
int Mat_creuse_CompCol::Nb_ligne ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.16 NouvelElement()**

```
Mat\_abstraite * Mat_creuse_CompCol::NouvelElement ( ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.17 operator>() [1/2]**

```
double & Mat_creuse_CompCol::operator() (
    int i,
    int j ) [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.18 operator>() [2/2]

```
double Mat_creuse_CompCol::operator() (
    int i,
    int j ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.19 operator\*=( )

```
void Mat_creuse_CompCol::operator*= (
    const double r ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.20 operator+=( )

```
void Mat_creuse_CompCol::operator+= (
    const Mat\_abstraite & mat_pl ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.21 operator-=( )

```
void Mat_creuse_CompCol::operator-= (
    const Mat\_abstraite & mat_pl ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.22 operator=( )

```
Mat\_abstraite & Mat_creuse_CompCol::operator= (
    const Mat\_abstraite & b ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.23 operator==( )

```
int Mat_creuse_CompCol::operator==(
    const Mat\_abstraite & mat_pl ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.24 Place()

```
int Mat_creuse_CompCol::Place ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.25 Preparation\_resol()

```
void Mat_creuse_CompCol::Preparation_resol ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).



**6.489.1.26 Prod\_Ligne\_vec()**

```
double Mat_creuse_CompCol::Prod_Ligne_vec (
    int iligne,
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.27 Prod\_mat\_vec() [1/2]**

```
Vecteur Mat_creuse_CompCol::Prod_mat_vec (
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.28 Prod\_mat\_vec() [2/2]**

```
Vecteur & Mat_creuse_CompCol::Prod_mat_vec (
    const Vecteur & vec,
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.29 Prod\_vec\_col()**

```
double Mat_creuse_CompCol::Prod_vec_col (
    int icol,
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.30 Prod\_vec\_mat() [1/2]**

```
Vecteur Mat_creuse_CompCol::Prod_vec_mat (
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.31 Prod\_vec\_mat() [2/2]**

```
Vecteur & Mat_creuse_CompCol::Prod_vec_mat (
    const Vecteur & vec,
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.32 RemplaceColonneSpe()**

```
void Mat_creuse_CompCol::RemplaceColonneSpe (
    int j,
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.33 RemplaceLigneSpe()**

```
void Mat_creuse_CompCol::RemplaceLigneSpe (
    int i,
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.34 Resol\_syst()** [1/2]

```
Tableau< Vecteur > Mat_creuse_CompCol::Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.35 Resol\_syst()** [2/2]

```
Vecteur Mat_creuse_CompCol::Resol_syst (
    const Vecteur & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.36 Resol\_systID()** [1/2]

```
Tableau< Vecteur > & Mat_creuse_CompCol::Resol_systID (
    Tableau< Vecteur > & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.37 Resol\_systID()** [2/2]

```
Vecteur & Mat_creuse_CompCol::Resol_systID (
    Vecteur & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.38 Resol\_systID\_2()**

```
Vecteur & Mat_creuse_CompCol::Resol_systID_2 (
    const Vecteur & b,
    Vecteur & vortie,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.489.1.39 Simple\_Resol\_syst()** [1/2]

```
Tableau< Vecteur > Mat_creuse_CompCol::Simple_Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_defaut,
```

```
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.40 `Simple_Resol_syst()` [2/2]

```
Vecteur Mat_creuse_CompCol::Simple_Resol_syst (  
    const Vecteur & b,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.41 `Simple_Resol_systID()` [1/2]

```
Tableau< Vecteur > & Mat_creuse_CompCol::Simple_Resol_systID (  
    Tableau< Vecteur > & b,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.42 `Simple_Resol_systID()` [2/2]

```
Vecteur & Mat_creuse_CompCol::Simple_Resol_systID (  
    Vecteur & b,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.43 `Simple_Resol_systID_2()`

```
Vecteur & Mat_creuse_CompCol::Simple_Resol_systID_2 (  
    const Vecteur & b,  
    Vecteur & vortie,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.44 `Symetrie()`

```
int Mat_creuse_CompCol::Symetrie ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.489.1.45 `trans_mult()`

```
virtual MV_Vector< double > Mat_creuse_CompCol::trans_mult (  
    const MV_Vector< double > & x ) const [inline], [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

### 6.489.1.46 Transfert\_vers\_mat()

```
void Mat_creuse_CompCol::Transfert_vers_mat (
    Mat_abstraite & A ) [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.489.1.47 vectT\_mat\_vec()

```
double Mat_creuse_CompCol::vectT_mat_vec (
    const Vecteur & vec1,
    const Vecteur & vec2 ) const [virtual]
```

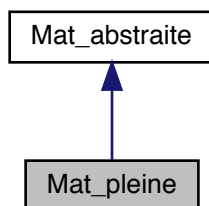
Implémente [Mat\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

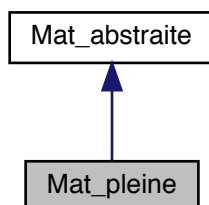
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/matrices\_creuses/Mat\_creuse\_↔  
CompCol.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/matrices\_creuses/Mat\_creuse\_↔  
CompCol.cc

## 6.490 Référence de la classe Mat\_pleine

Graphe d'héritage de Mat\_pleine:



Graphe de collaboration de Mat\_pleine:



## Fonctions membres publiques

- **Mat\_pleine** (int nb\_ligne, int nb\_colonne, double val\_init=0.0)

- **Mat\_pleine** (const [Mat\\_pleine](#) &mat\_pl)
- [Mat\\_abstraite](#) \* [NouvelElement](#) () const
- [Mat\\_abstraite](#) & [operator=](#) (const [Mat\\_abstraite](#) &mat\_pl)
- void [Transfert\\_vers\\_mat](#) ([Mat\\_abstraite](#) &A)
- void [operator+=](#) (const [Mat\\_abstraite](#) &mat\_pl)
- void [operator-=](#) (const [Mat\\_abstraite](#) &mat\_pl)
- int [operator==](#) (const [Mat\\_abstraite](#) &mat\_pl) const
- [Mat\\_pleine](#) & [operator=](#) (const [Mat\\_pleine](#) &mat\_pl)
- void [Affiche](#) () const
- void [Initialise](#) (double val\_init=0.0)
- void [Initialise](#) (int nb\_ligne, int nb\_colonne, double val\_init=0.0)
- void [Libere](#) ()
- int [Nb\\_ligne](#) () const
- int [Nb\\_colonne](#) () const
- [Vecteur](#) [Ligne](#) (int i) const
- [Vecteur](#) [LigneSpe](#) (int i) const
- void [RemplaceLigneSpe](#) (int i, const [Vecteur](#) &v)
- void [MetValLigne](#) (int i, double x)
- [Vecteur](#) [Colonne](#) (int j) const
- [Vecteur](#) [ColonneSpe](#) (int j) const
- void [RemplaceColonneSpe](#) (int j, const [Vecteur](#) &v)
- void [MetValColonne](#) (int j, double y)
- int [Symetrie](#) () const
- void [AfficheNonSymetries](#) () const
- [Vecteur](#) [Resol\\_syst](#) (const [Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- [Vecteur](#) & [Resol\\_systID](#) ([Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- [Tableau](#)< [Vecteur](#) > [Resol\\_syst](#) (const [Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- [Tableau](#)< [Vecteur](#) > & [Resol\\_systID](#) ([Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- [Vecteur](#) & [Resol\\_systID\\_2](#) (const [Vecteur](#) &b, [Vecteur](#) &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- void [Preparation\\_resol](#) ()
- [Vecteur](#) [Simple\\_Resol\\_syst](#) (const [Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur](#) & [Simple\\_Resol\\_systID](#) ([Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Tableau](#)< [Vecteur](#) > [Simple\\_Resol\\_syst](#) (const [Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Tableau](#)< [Vecteur](#) > & [Simple\\_Resol\\_systID](#) ([Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur](#) & [Simple\\_Resol\\_systID\\_2](#) (const [Vecteur](#) &b, [Vecteur](#) &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur](#) [Prod\\_vec\\_mat](#) (const [Vecteur](#) &vec) const
- [Vecteur](#) & [Prod\\_vec\\_mat](#) (const [Vecteur](#) &vec, [Vecteur](#) &resul) const
- [Vecteur](#) [Prod\\_mat\\_vec](#) (const [Vecteur](#) &vec) const
- [Vecteur](#) & [Prod\\_mat\\_vec](#) (const [Vecteur](#) &vec, [Vecteur](#) &resul) const
- [Mat\\_pleine](#) [Transpose](#) () const
- [Mat\\_pleine](#) [Inverse](#) () const
- [Mat\\_pleine](#) & [Inverse](#) ([Mat\\_pleine](#) &res) const
- double [Determinant](#) () const
- [Mat\\_pleine](#) [operator+](#) (const [Mat\\_pleine](#) &mat\_pl) const
- [Mat\\_pleine](#) [operator-](#) (const [Mat\\_pleine](#) &mat\_pl) const
- [Mat\\_pleine](#) [operator-](#) () const
- void [operator+=](#) (const [Mat\\_pleine](#) &mat\_pl)
- void [operator-=](#) (const [Mat\\_pleine](#) &mat\_pl)
- [Mat\\_pleine](#) [operator\\*](#) (const [Mat\\_pleine](#) &mat\_pl) const
- void [operator\\*=\[operator\\\*=" \\(const double r\\)\]\(#\)](#)
- [Vecteur](#) [operator\\*](#) (const [Vecteur](#) &vec) const
- [Mat\\_pleine](#) [operator\\*](#) (const double coeff) const
- int [operator==](#) (const [Mat\\_pleine](#) &mat\_pl) const
- int [operator!=](#) (const [Mat\\_pleine](#) &mat\_pl) const

- [Vecteur](#) & [Ligne\\_set](#) (int i)
- [Vecteur](#) & [operator\(\)](#) (int i)
- [Vecteur](#) [operator\(\)](#) (int i) const
- double & [operator\(\)](#) (int i, int j)
- bool [Existe](#) (int, int) const
- double [operator\(\)](#) (int i, int j) const
- double [Prod\\_Ligne\\_vec](#) (int iligne, const [Vecteur](#) &vec) const
- double [Prod\\_vec\\_col](#) (int icol, const [Vecteur](#) &vec) const
- double [vectT\\_mat\\_vec](#) (const [Vecteur](#) &vec1, const [Vecteur](#) &vec2) const
- void [Zero](#) ()
- int [Place](#) () const
- [Tableau](#)< [Coordonnee](#) > [Coordonnee\\_Base\\_associee](#) () const
- [Tableau](#)< [CoordonneeH](#) > [CoordonneeH\\_Base\\_associee](#) () const
- [Tableau](#)< [CoordonneeB](#) > [CoordonneeB\\_Base\\_associee](#) () const
- [Coordonnee](#) [Coordonnee\\_Base\\_associee](#) (int i) const
- [CoordonneeH](#) [CoordonneeH\\_Base\\_associee](#) (int i) const
- [CoordonneeB](#) [CoordonneeB\\_Base\\_associee](#) (int i) const
- double [MaxiValAbs](#) (int &i, int &j) const
- double [Maxi\\_ligne\\_ValAbs](#) (int &i) const

### Fonctions membres protégées

- void [Triangulation](#) (int N, [Mat\\_pleine](#) &B)
- void [Resolution](#) (int N, const [Mat\\_pleine](#) &B, const [Vecteur](#) &b, [Vecteur](#) &res) const
- void [Cramer](#) (const [Vecteur](#) &b, [Vecteur](#) &res) const
- void [Cramer](#) (const [Tableau](#)< [Vecteur](#) > &b, [Tableau](#)< [Vecteur](#) > &res) const
- void [Symetrisation](#) ()

### Attributs protégés

- [Tableau](#)< [Vecteur](#) > [val](#)

### Amis

- [istream](#) & [operator](#)>> ([istream](#) &, [Mat\\_pleine](#) &)
- [ostream](#) & [operator](#)<< ([ostream](#) &, const [Mat\\_pleine](#) &)

### Membres hérités additionnels

#### 6.490.1 Documentation des fonctions membres

##### 6.490.1.1 Affiche()

```
void Mat_pleine::Affiche ( ) const [virtual]
Implémente Mat\_abstraite.
```

##### 6.490.1.2 Colonne()

```
Vecteur Mat_pleine::Colonne (
    int j ) const [inline], [virtual]
Implémente Mat\_abstraite.
```

##### 6.490.1.3 ColonneSpe()

```
Vecteur Mat_pleine::ColonneSpe (
    int j ) const [inline], [virtual]
Implémente Mat\_abstraite.
```

#### 6.490.1.4 Existe()

```
bool Mat_pleine::Existe (
    int ,
    int ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.5 Initialise()

```
void Mat_pleine::Initialise (
    double val_init = 0.0 ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.6 Libere()

```
void Mat_pleine::Libere ( ) [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.7 Ligne()

```
Vecteur Mat_pleine::Ligne (
    int i ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.8 Ligne\_set()

```
Vecteur & Mat_pleine::Ligne_set (
    int i ) [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.9 LigneSpe()

```
Vecteur Mat_pleine::LigneSpe (
    int i ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.10 MetValColonne()

```
void Mat_pleine::MetValColonne (
    int j,
    double y ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.11 MetValLigne()

```
void Mat_pleine::MetValLigne (
    int i,
    double x ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.12 Nb\_colonne()**

```
int Mat_pleine::Nb_colonne ( ) const [inline], [virtual]  
Implémente Mat\_abstraite.
```

**6.490.1.13 Nb\_ligne()**

```
int Mat_pleine::Nb_ligne ( ) const [inline], [virtual]  
Implémente Mat\_abstraite.
```

**6.490.1.14 NouvelElement()**

```
Mat\_abstraite * Mat_pleine::NouvelElement ( ) const [virtual]  
Implémente Mat\_abstraite.
```

**6.490.1.15 operator>() [1/2]**

```
double & Mat_pleine::operator() (   
    int i,  
    int j ) [inline], [virtual]  
Implémente Mat\_abstraite.
```

**6.490.1.16 operator>() [2/2]**

```
double Mat_pleine::operator() (   
    int i,  
    int j ) const [inline], [virtual]  
Implémente Mat\_abstraite.
```

**6.490.1.17 operator\*=( )**

```
void Mat_pleine::operator*= (   
    const double r ) [virtual]  
Implémente Mat\_abstraite.
```

**6.490.1.18 operator+=( )**

```
void Mat_pleine::operator+= (   
    const Mat\_abstraite & mat_pl ) [virtual]  
Implémente Mat\_abstraite.
```

**6.490.1.19 operator-=( )**

```
void Mat_pleine::operator-= (   
    const Mat\_abstraite & mat_pl ) [virtual]  
Implémente Mat\_abstraite.
```

**6.490.1.20 operator=( )**

```
Mat\_abstraite & Mat_pleine::operator= (   
    const Mat\_abstraite & mat_pl ) [virtual]  
Implémente Mat\_abstraite.
```



**6.490.1.21 operator==( )**

```
int Mat_pleine::operator==(
    const Mat\_abstraite & mat_pl ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.22 Place( )**

```
int Mat_pleine::Place ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.23 Preparation\_resol( )**

```
void Mat_pleine::Preparation_resol ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.24 Prod\_Ligne\_vec( )**

```
double Mat_pleine::Prod_Ligne_vec (
    int iligne,
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.25 Prod\_mat\_vec( ) [1/2]**

```
Vecteur Mat_pleine::Prod_mat_vec (
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.26 Prod\_mat\_vec( ) [2/2]**

```
Vecteur & Mat_pleine::Prod_mat_vec (
    const Vecteur & vec,
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.27 Prod\_vec\_col( )**

```
double Mat_pleine::Prod_vec_col (
    int icol,
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.28 Prod\_vec\_mat( ) [1/2]**

```
Vecteur Mat_pleine::Prod_vec_mat (
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.29 Prod\_vec\_mat()** [2/2]

```
Vecteur & Mat_pleine::Prod_vec_mat (
    const Vecteur & vec,
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.30 RemplaceColonneSpe()**

```
void Mat_pleine::RemplaceColonneSpe (
    int j,
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.31 RemplaceLigneSpe()**

```
void Mat_pleine::RemplaceLigneSpe (
    int i,
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.32 Resol\_syst()** [1/2]

```
Tableau< Vecteur > Mat_pleine::Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.33 Resol\_syst()** [2/2]

```
Vecteur Mat_pleine::Resol_syst (
    const Vecteur & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.34 Resol\_systID()** [1/2]

```
Tableau< Vecteur > & Mat_pleine::Resol_systID (
    Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.490.1.35 Resol\_systID()** [2/2]

```
Vecteur & Mat_pleine::Resol_systID (
    Vecteur & b,
    const double & tol = tol_default,
```

```
    const int maxit = maxit_default,  
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.36 `Resol_systID_2()`

```
Vecteur & Mat_pleine::Resol_systID_2 (  
    const Vecteur & b,  
    Vecteur & vortie,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.37 `Simple_Resol_syst()` [1/2]

```
Tableau< Vecteur > Mat_pleine::Simple_Resol_syst (  
    const Tableau< Vecteur > & b,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.38 `Simple_Resol_syst()` [2/2]

```
Vecteur Mat_pleine::Simple_Resol_syst (  
    const Vecteur & b,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.39 `Simple_Resol_systID()` [1/2]

```
Tableau< Vecteur > & Mat_pleine::Simple_Resol_systID (  
    Tableau< Vecteur > & b,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.40 `Simple_Resol_systID()` [2/2]

```
Vecteur & Mat_pleine::Simple_Resol_systID (  
    Vecteur & b,  
    const double & tol = tol_default,  
    const int maxit = maxit_default,  
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.41 `Simple_Resol_systID_2()`

```
Vecteur & Mat_pleine::Simple_Resol_systID_2 (  
    const Vecteur & b,
```

```
Vecteur & vortie,  
const double & tol = tol_defaut,  
const int maxit = maxit_defaut,  
const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.42 Symetrie()

```
int Mat_pleine::Symetrie ( ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.43 Transfert\_vers\_mat()

```
void Mat_pleine::Transfert_vers_mat (  
    Mat_abstraite & A ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.490.1.44 vectT\_mat\_vec()

```
double Mat_pleine::vectT_mat_vec (  
    const Vecteur & vec1,  
    const Vecteur & vec2 ) const [virtual]
```

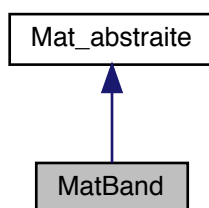
Implémente [Mat\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

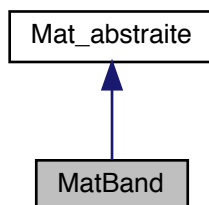
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/Mat\_pleine.h

## 6.491 Référence de la classe MatBand

Graphe d'héritage de MatBand:



Graphe de collaboration de MatBand:



## Fonctions membres publiques

- **MatBand** ([Enum\\_matrice](#) type\_mat, int lb, int dim)
- **MatBand** ([Enum\\_matrice](#) type\_mat, int lb, int dim, double a)
- **MatBand** (const [MatBand](#) &m)
- [Mat\\_abstraite](#) \* **NouvelElement** () const
- [Mat\\_abstraite](#) & **operator=** (const [Mat\\_abstraite](#) &)
- void **Transfert\_vers\_mat** ([Mat\\_abstraite](#) &A)
- [MatBand](#) & **operator=** (const [MatBand](#) &)
- void **operator+=** (const [Mat\\_abstraite](#) &mat\_pl)
- void **operator-=** (const [Mat\\_abstraite](#) &mat\_pl)
- void **operator\*=** (const double r)
- int **operator==** (const [Mat\\_abstraite](#) &mat\_pl) const
- double & **operator()** (int i, int j)
- bool **Existe** (int i, int j) const
- double **operator()** (int i, int j) const
- [Vecteur](#) & **Ligne\_set** (int i)
- [Vecteur](#) **operator()** (int i) const
- [Vecteur](#) **Ligne** (int i) const
- [Vecteur](#) **LigneSpe** (int i) const
- void **RemplaceLigneSpe** (int i, const [Vecteur](#) &v)
- void **MetValLigne** (int i, double val)
- [Vecteur](#) **Colonne** (int j) const
- [Vecteur](#) **ColonneSpe** (int j) const
- void **RemplaceColonneSpe** (int j, const [Vecteur](#) &v)
- void **MetValColonne** (int j, double val)
- void **Affiche** () const
- void **Affiche1** (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j) const
- void **Affiche2** (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j, int nd) const
- void **Change\_taille** (int lb, int dim)
- int **Nb\_colonne** () const
- int **Nb\_ligne** () const
- void **Initialise** (double a)
- void **Libere** ()
- int **Symetrie** () const
- [Vecteur](#) **Resol\_syst** (const [Vecteur](#) &b, const double &tol=tol\_defaut, const int maxit=maxit\_defaut, const int restart=restart\_defaut)
- [Vecteur](#) & **Resol\_systID** ([Vecteur](#) &b, const double &tol=tol\_defaut, const int maxit=maxit\_defaut, const int restart=restart\_defaut)
- [Tableau](#)< [Vecteur](#) > **Resol\_syst** (const [Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_defaut, const int maxit=maxit\_defaut, const int restart=restart\_defaut)
- [Tableau](#)< [Vecteur](#) > & **Resol\_systID** ([Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_defaut, const int maxit=maxit\_defaut, const int restart=restart\_defaut)
- [Vecteur](#) & **Resol\_systID\_2** (const [Vecteur](#) &b, [Vecteur](#) &vortie, const double &tol=tol\_defaut, const int maxit=maxit\_defaut, const int restart=restart\_defaut)

- void [Preparation\\_resol](#) ()
- [Vecteur Simple\\_Resol\\_syst](#) (const [Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur & Simple\\_Resol\\_systID](#) ([Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Tableau< Vecteur > Simple\\_Resol\\_syst](#) (const [Tableau< Vecteur >](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Tableau< Vecteur > & Simple\\_Resol\\_systID](#) ([Tableau< Vecteur >](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur & Simple\\_Resol\\_systID\\_2](#) (const [Vecteur](#) &b, [Vecteur](#) &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur Prod\\_vec\\_mat](#) (const [Vecteur](#) &vec) const
- [Vecteur & Prod\\_vec\\_mat](#) (const [Vecteur](#) &vec, [Vecteur](#) &resul) const
- [Vecteur Prod\\_mat\\_vec](#) (const [Vecteur](#) &vec) const
- [Vecteur & Prod\\_mat\\_vec](#) (const [Vecteur](#) &vec, [Vecteur](#) &resul) const
- double [Prod\\_Ligne\\_vec](#) (int iligne, const [Vecteur](#) &vec) const
- double [Prod\\_vec\\_col](#) (int icol, const [Vecteur](#) &vec) const
- double [vectT\\_mat\\_vec](#) (const [Vecteur](#) &vec1, const [Vecteur](#) &vec2) const
- int [Place](#) () const
- [Vecteur & Resol\\_systID\\_sans\\_triangu](#) ([Vecteur](#) &b)

## Fonctions membres protégées

- void **REDUCT** ([MatBand](#) &TRAID, int LB, int NDDL, [Vecteur](#) &CC)
- void **RESOLT** (const [MatBand](#) &TRAID, [Vecteur](#) &TSM, int NDDL, int LB) const
- void **ResoBand** ([MatBand](#) &TRAID, int LB, int NDDL, [Vecteur](#) &CC)
- void **ResoBand** ([MatBand](#) &TRAID, int LB, int NDDL, [Tableau< Vecteur >](#) &CC)
- double & **c** (int ii, int jj)
- const double & **cConst** (int ii, int jj) const

## Attributs protégés

- double \* **pt**
- int **lb**
- int **dim**

## Amis

- istream & **operator>>** (istream &, [MatBand](#) &)
- ostream & **operator<<** (ostream &, const [MatBand](#) &)

## Membres hérités additionnels

### 6.491.1 Documentation des fonctions membres

#### 6.491.1.1 Affiche()

void [MatBand::Affiche](#) ( ) const [virtual]  
 Implémente [Mat\\_abstraite](#).

#### 6.491.1.2 Affiche1()

```
void MatBand::Affiche1 (
    int min_i,
    int max_i,
    int pas_i,
    int min_j,
```

```
    int max_j,  
    int pas_j ) const [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

### 6.491.1.3 Affiche2()

```
void MatBand::Affiche2 (  
    int min_i,  
    int max_i,  
    int pas_i,  
    int min_j,  
    int max_j,  
    int pas_j,  
    int nd ) const [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

### 6.491.1.4 Colonne()

```
Vecteur MatBand::Colonne (  
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.491.1.5 ColonneSpe()

```
Vecteur MatBand::ColonneSpe (  
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.491.1.6 Existe()

```
bool MatBand::Existe (  
    int i,  
    int j ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.491.1.7 Initialise()

```
void MatBand::Initialise (  
    double a ) [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.491.1.8 Libere()

```
void MatBand::Libere ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.491.1.9 Ligne()

```
Vecteur MatBand::Ligne (  
    int i ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.10 Ligne\_set()**

```
Vecteur & MatBand::Ligne_set (
    int i ) [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.11 LigneSpe()**

```
Vecteur MatBand::LigneSpe (
    int i ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.12 MetValColonne()**

```
void MatBand::MetValColonne (
    int j,
    double val ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.13 MetValLigne()**

```
void MatBand::MetValLigne (
    int i,
    double val ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.14 Nb\_colonne()**

```
int MatBand::Nb_colonne ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.15 Nb\_ligne()**

```
int MatBand::Nb_ligne ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.16 NouvelElement()**

```
Mat_abstraite * MatBand::NouvelElement ( ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.17 operator>() [1/2]**

```
double & MatBand::operator() (
    int i,
    int j ) [virtual]
```

Implémente [Mat\\_abstraite](#).



**6.491.1.18 operator>() [2/2]**

```
double MatBand::operator() (
    int i,
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.19 operator\*=( )**

```
void MatBand::operator*= (
    const double r ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.20 operator+=( )**

```
void MatBand::operator+= (
    const Mat_abstraite & mat_pl ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.21 operator-=( )**

```
void MatBand::operator-= (
    const Mat_abstraite & mat_pl ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.22 operator=( )**

```
Mat_abstraite & MatBand::operator= (
    const Mat_abstraite & b ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.23 operator==( )**

```
int MatBand::operator==(
    const Mat_abstraite & mat_pl ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.24 Place()**

```
int MatBand::Place ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.25 Preparation\_resol()**

```
void MatBand::Preparation_resol ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.26 Prod\_Ligne\_vec()**

```
double MatBand::Prod_Ligne_vec (
    int iligne,
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.491.1.27 Prod\_mat\_vec() [1/2]

```
Vecteur MatBand::Prod_mat_vec (  
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.491.1.28 Prod\_mat\_vec() [2/2]

```
Vecteur & MatBand::Prod_mat_vec (  
    const Vecteur & vec,  
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.491.1.29 Prod\_vec\_col()

```
double MatBand::Prod_vec_col (  
    int icol,  
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.491.1.30 Prod\_vec\_mat() [1/2]

```
Vecteur MatBand::Prod_vec_mat (  
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.491.1.31 Prod\_vec\_mat() [2/2]

```
Vecteur & MatBand::Prod_vec_mat (  
    const Vecteur & vec,  
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.491.1.32 RemplaceColonneSpe()

```
void MatBand::RemplaceColonneSpe (  
    int j,  
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.491.1.33 RemplaceLigneSpe()

```
void MatBand::RemplaceLigneSpe (  
    int i,  
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.34 Resol\_syst()** [1/2]

```
Tableau< Vecteur > MatBand::Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.35 Resol\_syst()** [2/2]

```
Vecteur MatBand::Resol_syst (
    const Vecteur & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.36 Resol\_systID()** [1/2]

```
Tableau< Vecteur > & MatBand::Resol_systID (
    Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.37 Resol\_systID()** [2/2]

```
Vecteur & MatBand::Resol_systID (
    Vecteur & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.38 Resol\_systID\_2()**

```
Vecteur & MatBand::Resol_systID_2 (
    const Vecteur & b,
    Vecteur & vortie,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.39 Simple\_Resol\_syst()** [1/2]

```
Tableau< Vecteur > MatBand::Simple_Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.40 Simple\_Resol\_syst()** [2/2]

```
Vecteur MatBand::Simple_Resol_syst (
    const Vecteur & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.41 Simple\_Resol\_systID()** [1/2]

```
Tableau< Vecteur > & MatBand::Simple_Resol_systID (
    Tableau< Vecteur > & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.42 Simple\_Resol\_systID()** [2/2]

```
Vecteur & MatBand::Simple_Resol_systID (
    Vecteur & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.43 Simple\_Resol\_systID\_2()**

```
Vecteur & MatBand::Simple_Resol_systID_2 (
    const Vecteur & b,
    Vecteur & vortie,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.44 Symetrie()**

```
int MatBand::Symetrie ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.45 Transfert\_vers\_mat()**

```
void MatBand::Transfert_vers_mat (
    Mat_abstraite & A ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.491.1.46 vectT\_mat\_vec()**

```
double MatBand::vectT_mat_vec (
    const Vecteur & vec1,
    const Vecteur & vec2 ) const [virtual]
```

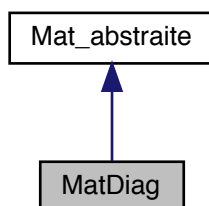
Implémente [Mat\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

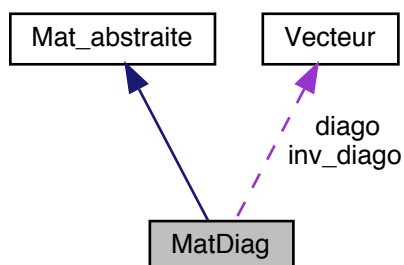
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/MatBand.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/MatBand.cc

## 6.492 Référence de la classe MatDiag

Graphe d'héritage de MatDiag:



Graphe de collaboration de MatDiag:



### Fonctions membres publiques

- **MatDiag** (int dim)
- **MatDiag** (int dim, double a)
- **MatDiag** (const [MatDiag](#) &m)
- [Mat\\_abstraite](#) \* **NouvelElement** () const
- [Mat\\_abstraite](#) & **operator=** (const [Mat\\_abstraite](#) &)
- void **Transfert\_vers\_mat** ([Mat\\_abstraite](#) &A)
- [MatDiag](#) & **operator=** (const [MatDiag](#) &)
- [MatDiag](#) & **operator=** (const [Vecteur](#) &)
- void **operator+=** (const [Mat\\_abstraite](#) &mat\_pl)
- void **operator-=** (const [Mat\\_abstraite](#) &mat\_pl)
- void **operator\*=** (const double r)
- int **operator==** (const [Mat\\_abstraite](#) &mat\_pl) const
- double & **operator()** (int i, int j)
- bool **Existe** (int i, int j) const
- double **operator()** (int i, int j) const

- void **set\_element** (int i, int j, double val)
- **Vecteur** & **Ligne\_set** (int i)
- **Vecteur operator()** (int i) const
- **Vecteur Ligne** (int i) const
- **Vecteur LigneSpe** (int i) const
- void **RemplaceLigneSpe** (int i, const **Vecteur** &v)
- void **MetValLigne** (int i, double val)
- **Vecteur Colonne** (int j) const
- **Vecteur ColonneSpe** (int j) const
- void **RemplaceColonneSpe** (int j, const **Vecteur** &v)
- void **MetValColonne** (int j, double val)
- void **Affiche** () const
- void **Affiche1** (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j) const
- void **Affiche2** (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j, int nd) const
- void **Change\_taille** (int dim)
- int **Nb\_colonne** () const
- int **Nb\_ligne** () const
- void **Initialise** (double a)
- void **Libere** ()
- int **Symetrie** () const
- **Vecteur Resol\_syst** (const **Vecteur** &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- **Vecteur** & **Resol\_systID** (**Vecteur** &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- **Tableau**< **Vecteur** > **Resol\_syst** (const **Tableau**< **Vecteur** > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- **Tableau**< **Vecteur** > & **Resol\_systID** (**Tableau**< **Vecteur** > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- **Vecteur** & **Resol\_systID\_2** (const **Vecteur** &b, **Vecteur** &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- void **Preparation\_resol** ()
- **Vecteur Simple\_Resol\_syst** (const **Vecteur** &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- **Vecteur** & **Simple\_Resol\_systID** (**Vecteur** &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- **Tableau**< **Vecteur** > **Simple\_Resol\_syst** (const **Tableau**< **Vecteur** > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- **Tableau**< **Vecteur** > & **Simple\_Resol\_systID** (**Tableau**< **Vecteur** > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- **Vecteur** & **Simple\_Resol\_systID\_2** (const **Vecteur** &b, **Vecteur** &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- **Vecteur Prod\_vec\_mat** (const **Vecteur** &vec) const
- **Vecteur** & **Prod\_vec\_mat** (const **Vecteur** &vec, **Vecteur** &resul) const
- **Vecteur Prod\_mat\_vec** (const **Vecteur** &vec) const
- **Vecteur** & **Prod\_mat\_vec** (const **Vecteur** &vec, **Vecteur** &resul) const
- double **Prod\_Ligne\_vec** (int iligne, const **Vecteur** &vec) const
- double **Prod\_vec\_col** (int icol, const **Vecteur** &vec) const
- double **vectT\_mat\_vec** (const **Vecteur** &vec1, const **Vecteur** &vec2) const
- int **Place** () const
- **MatDiag** & **operator+=** (const **Vecteur** &)
- **MatDiag** & **operator-=** (const **Vecteur** &)
- const **Vecteur** & **Vecteur\_MatDiag** () const

## Fonctions membres protégées

- void **ResoDiag** (const **Vecteur** &BB, **Vecteur** &CC)
- void **ResoDiag** (const **Tableau**< **Vecteur** > &BB, **Tableau**< **Vecteur** > &CC)
- void **Const\_ResoDiag** (const **Vecteur** &BB, **Vecteur** &CC) const
- void **Const\_ResoDiag** (const **Tableau**< **Vecteur** > &BB, **Tableau**< **Vecteur** > &CC) const
- double **Affiche\_elem** (int i, int j) const

## Attributs protégés

- [Vecteur diago](#)
- [Vecteur inv\\_diago](#)
- `bool inversion`

## Amis

- `istream & operator>>` (`istream &`, [MatDiag &](#))
- `ostream & operator<<` (`ostream &`, `const MatDiag &`)

## Membres hérités additionnels

### 6.492.1 Documentation des fonctions membres

#### 6.492.1.1 Affiche()

```
void MatDiag::Affiche ( ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.2 Affiche1()

```
void MatDiag::Affiche1 (
    int min_i,
    int max_i,
    int pas_i,
    int min_j,
    int max_j,
    int pas_j ) const [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

#### 6.492.1.3 Affiche2()

```
void MatDiag::Affiche2 (
    int min_i,
    int max_i,
    int pas_i,
    int min_j,
    int max_j,
    int pas_j,
    int nd ) const [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

#### 6.492.1.4 Colonne()

```
Vecteur MatDiag::Colonne (
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.5 ColonneSpe()

```
Vecteur MatDiag::ColonneSpe (
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.6 Existe()

```
bool MatDiag::Existe (
    int i,
    int j ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.7 Initialise()

```
void MatDiag::Initialise (
    double a ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.8 Libere()

```
void MatDiag::Libere ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.9 Ligne()

```
Vecteur MatDiag::Ligne (
    int i ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.10 Ligne\_set()

```
Vecteur & MatDiag::Ligne_set (
    int i ) [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.11 LigneSpe()

```
Vecteur MatDiag::LigneSpe (
    int i ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.12 MetValColonne()

```
void MatDiag::MetValColonne (
    int j,
    double val ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.13 MetValLigne()

```
void MatDiag::MetValLigne (
    int i,
    double val ) [virtual]
```

Implémente [Mat\\_abstraite](#).



**6.492.1.14 Nb\_colonne()**

```
int MatDiag::Nb_colonne ( ) const [inline], [virtual]
Implémente Mat\_abstraite.
```

**6.492.1.15 Nb\_ligne()**

```
int MatDiag::Nb_ligne ( ) const [inline], [virtual]
Implémente Mat\_abstraite.
```

**6.492.1.16 NouvelElement()**

```
Mat\_abstraite * MatDiag::NouvelElement ( ) const [virtual]
Implémente Mat\_abstraite.
```

**6.492.1.17 operator>() [1/2]**

```
double & MatDiag::operator() (
    int i,
    int j ) [virtual]
Implémente Mat\_abstraite.
```

**6.492.1.18 operator>() [2/2]**

```
double MatDiag::operator() (
    int i,
    int j ) const [virtual]
Implémente Mat\_abstraite.
```

**6.492.1.19 operator\*=( )**

```
void MatDiag::operator*= (
    const double r ) [virtual]
Implémente Mat\_abstraite.
```

**6.492.1.20 operator+=( )**

```
void MatDiag::operator+= (
    const Mat\_abstraite & mat_pl ) [virtual]
Implémente Mat\_abstraite.
```

**6.492.1.21 operator-=( )**

```
void MatDiag::operator-= (
    const Mat\_abstraite & mat_pl ) [virtual]
Implémente Mat\_abstraite.
```

**6.492.1.22 operator=( )**

```
Mat\_abstraite & MatDiag::operator= (
    const Mat\_abstraite & b ) [virtual]
Implémente Mat\_abstraite.
```

#### 6.492.1.23 operator==( )

```
int MatDiag::operator==(
    const Mat\_abstraite & mat_pl ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.24 Place( )

```
int MatDiag::Place ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.25 Preparation\_resol( )

```
void MatDiag::Preparation_resol ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.26 Prod\_Ligne\_vec( )

```
double MatDiag::Prod_Ligne_vec (
    int iligne,
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.27 Prod\_mat\_vec( ) [1/2]

```
Vecteur MatDiag::Prod_mat_vec (
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.28 Prod\_mat\_vec( ) [2/2]

```
Vecteur & MatDiag::Prod_mat_vec (
    const Vecteur & vec,
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.29 Prod\_vec\_col( )

```
double MatDiag::Prod_vec_col (
    int icol,
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.30 Prod\_vec\_mat( ) [1/2]

```
Vecteur MatDiag::Prod_vec_mat (
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.492.1.31 Prod\_vec\_mat()** [2/2]

```
Vecteur & MatDiag::Prod_vec_mat (
    const Vecteur & vec,
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.492.1.32 RemplaceColonneSpe()**

```
void MatDiag::RemplaceColonneSpe (
    int j,
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.492.1.33 RemplaceLigneSpe()**

```
void MatDiag::RemplaceLigneSpe (
    int i,
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.492.1.34 Resol\_syst()** [1/2]

```
Tableau< Vecteur > MatDiag::Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.492.1.35 Resol\_syst()** [2/2]

```
Vecteur MatDiag::Resol_syst (
    const Vecteur & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.492.1.36 Resol\_systID()** [1/2]

```
Tableau< Vecteur > & MatDiag::Resol_systID (
    Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.492.1.37 Resol\_systID()** [2/2]

```
Vecteur & MatDiag::Resol_systID (
    Vecteur & b,
    const double & tol = tol_default,
```

```

    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]

```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.38 Resol\_systID\_2()

```

Vecteur & MatDiag::Resol_systID_2 (
    const Vecteur & b,
    Vecteur & vortie,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]

```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.39 Simple\_Resol\_syst() [1/2]

```

Tableau< Vecteur > MatDiag::Simple_Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) const [virtual]

```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.40 Simple\_Resol\_syst() [2/2]

```

Vecteur MatDiag::Simple_Resol_syst (
    const Vecteur & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) const [virtual]

```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.41 Simple\_Resol\_systID() [1/2]

```

Tableau< Vecteur > & MatDiag::Simple_Resol_systID (
    Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) const [virtual]

```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.42 Simple\_Resol\_systID() [2/2]

```

Vecteur & MatDiag::Simple_Resol_systID (
    Vecteur & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) const [virtual]

```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.43 Simple\_Resol\_systID\_2()

```

Vecteur & MatDiag::Simple_Resol_systID_2 (
    const Vecteur & b,

```

```

    Vecteur & vortie,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]

```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.44 Symetrie()

```
int MatDiag::Symetrie ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.45 Transfert\_vers\_mat()

```
void MatDiag::Transfert_vers_mat (
    Mat_abstraite & A ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.492.1.46 vectT\_mat\_vec()

```
double MatDiag::vectT_mat_vec (
    const Vecteur & vec1,
    const Vecteur & vec2 ) const [virtual]
```

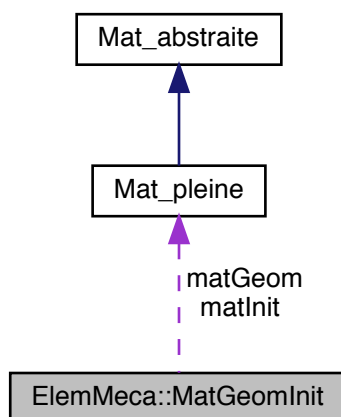
Implémente [Mat\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/MatDiag.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/MatDiag.cc

## 6.493 Référence de la classe ElemMeca::MatGeomInit

Graphe de collaboration de ElemMeca::MatGeomInit:



### Fonctions membres publiques

- **MatGeomInit** ([Mat\\_pleine](#) \*matG, [Mat\\_pleine](#) \*matI)

### Attributs publics

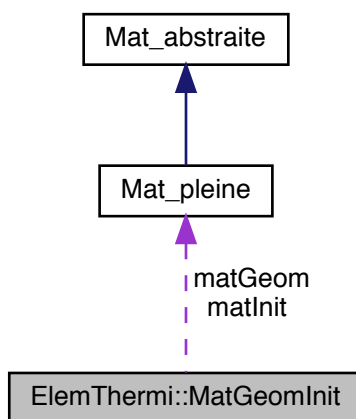
- [Mat\\_pleine](#) \* [matGeom](#)
- [Mat\\_pleine](#) \* [matInit](#)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca.h

## 6.494 Référence de la classe ElemThermi::MatGeomInit

Grphe de collaboration de ElemThermi::MatGeomInit:



### Fonctions membres publiques

- [MatGeomInit](#) ([Mat\\_pleine](#) \*matG, [Mat\\_pleine](#) \*matI)

### Attributs publics

- [Mat\\_pleine](#) \* [matGeom](#)
- [Mat\\_pleine](#) \* [matInit](#)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/ElemThermi.h

## 6.495 Référence de la classe MathUtil2

### Fonctions membres publiques statiques

- static [Vecteur VectValPropre](#) ([Mat\\_pleine](#) &A, int &cas)
 

*vecteurs propre et valeurs propres d'une matrice nxn c-a-d : en particulier 3 3 ou 2 2 ou 1 1 avec une limitation sur n (< 50) \*\*\* dans le cas où les matrices sont symétriques !! \*\*\* la méthode est itérative, --> méthode de Jacobi la matrice A est écrasée, à la sortie chaque colonne représente un vecteur propre le vecteur de retour contient les valeurs propres s'il y a une erreur dans le calcul : cas = -1*
- static [Tableau](#)< [Coordonnee](#) > [V\\_Propres3x3](#) (const [Mat\\_pleine](#) &A, const [Coordonnee](#) &VP, int &cas)
 

*calcul des vecteurs propres pour une matrice 3x3 réel uniquement, pas forcément symétrique dans le cas où on connaît déjà les valeurs propres la méthode est directe: --> pas d'itération, donc très rapide --> Utilisable que pour des matrices carrées en entrée : VP, de dimension 3, Vp contient les valeurs propres les 3 valeurs propres sont*

considérées classées:  $VP(1) \geq VP(2) \geq VP(3)$  cas : qui indique le type de valeurs propres , cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour) , cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour) , , cas = 2 si  $Vp(1)=Vp(2)$  (  $Vp(1)$ , et  $Vp(3)$  dans les 2 premières composantes du retour) , cas = 3 si  $Vp(2)=Vp(3)$  (  $Vp(1)$ , et  $Vp(2)$  dans les 2 premières composantes du retour) en sortie : le tableau des vecteurs propres, rangés selon l'ordre d'entrée cas = le cas de l'entrée, sauf s'il y a eu un pb, dans ce cas: cas = -1, et les vecteurs propres sont dans ce cas quelconques

- static int **CalculRangMatrice3x3Singuliere** (Mat\_pleine &A)
  - calcul du rang d'une matrice 3x3 singulière, qui peut être 0, 1 ou 2 modifie la matrice en conséquence en sortie: rang := le rang de la matrice rang = 0 ==> veut dire que c'est la matrice nulle (à ConstMath::pasmalpetit près) rang = 1 ==> la première ligne de la matrice contient la direction libre, génératrice rang = 2 ==> les deux premières ligne de la matrice contiennent les deux directions libres
- static void **Def\_vecteurs\_plan** (const Coordonnee &V1, Coordonnee &V2, Coordonnee &V3)
  - cas d'une dimension d'espace == 3 calcul deux vecteurs perpendiculaires à un troisième, générateurs d'un plan, et orthonormées Les deux vecteurs résultats sont quelconques a priori
- static void **Def\_vecteurs\_plan** (const CoordonneeB &V1, CoordonneeB &V2, CoordonneeB &V3)
- static void **Def\_vecteurs\_plan** (const CoordonneeH &V1, CoordonneeH &V2, CoordonneeH &V3)
- static void **Def\_vecteurs\_plan** (const Coordonnee &V1, Coordonnee &V2)
  - cas d'une dimension d'espace == 2 calcul d'un vecteur perpendiculaire à un premier, et orthonormées
- static void **Def\_vecteurs\_plan** (const CoordonneeB &V1, CoordonneeB &V2)
- static void **Def\_vecteurs\_plan** (const CoordonneeH &V1, CoordonneeH &V2)
- static void **ChBase** (const CoordonneeB &A\_B, const Mat\_pleine &beta, CoordonneeB &Ap\_B)
  - calcul deux vecteurs perpendiculaires à un troisième, générateurs d'un plan, et orthonormées Les deux vecteurs résultats sont quelconques a priori calcul également la variation des vecteurs résultats en entrée: V1 et d\_Vi(1)(ddl) : vecteur donné et sa variation en sortie: V2 et V3, d\_Vi(2)(ddl) et d\_Vi(3)(ddl) : vecteurs résultats et leurs variations
- static void **ChBase** (const CoordonneeH &A\_H, const Mat\_pleine &gamma, CoordonneeH &Ap\_H)
  - changement de base pour de une fois contravariant:  $[Ap^k] = [\gamma] * [A^i]$

## 6.495.1 Documentation des fonctions membres

### 6.495.1.1 ChBase()

```
static void MathUtil2::ChBase (
    const CoordonneeB & A_B,
    const Mat_pleine & beta,
    CoordonneeB & Ap_B ) [static]
```

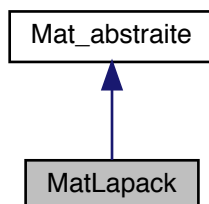
calcul deux vecteurs perpendiculaires à un troisième, générateurs d'un plan, et orthonormées Les deux vecteurs résultats sont quelconques a priori calcul également la variation des vecteurs résultats en entrée: V1 et d\_Vi(1)(ddl) : vecteur donné et sa variation en sortie: V2 et V3, d\_Vi(2)(ddl) et d\_Vi(3)(ddl) : vecteurs résultats et leurs variations calcul d'un changement de base: ceci n'est pas fait dans la classe [Coordonnee](#) car il faut y adjoindre la classe [Mat\\_pleine](#) qui intègre beaucoup de chose, du coup la classe [Coordonnee](#) deviendrait une usine changement de base (cf. théorie) : la matrice beta est telle que:  $gpB(i) = beta(i,j) * gB(j) \iff gp_i = beta_i^j * g_j$  et la matrice gamma telle que: gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH  $gpH(i) = gamma(i,j) * gH(j)$ , i indice de ligne, j indice de colonne c-a-d=  $gp^i = gamma^i_j * g^j$  rappel des différentes relations entre beta et gamma  $[beta]^{-1} = [gamma]^T$ ;  $[beta]^{-1T} = [gamma] [beta] = [gamma]^{-1T}$ ;  $[beta]^T = [gamma]^{-1}$  changement de base pour de une fois covariant:  $[Ap_k] = [beta] * [A_i]$

La documentation de cette classe a été générée à partir du fichier suivant :

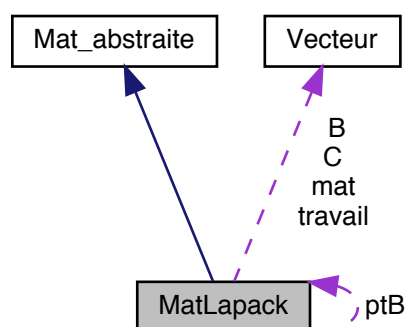
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/MathUtil2.h

## 6.496 Référence de la classe MatLapack

Graphe d'héritage de MatLapack:



Graphe de collaboration de MatLapack:



### Fonctions membres publiques

- **MatLapack** (int nb\_ligne, int nb\_colonne, bool symetrique, double val\_init, [Enum\\_type\\_resolution\\_matri](#) type\_resol, [Enum\\_preconditionnement](#) type\_precondi)
- **MatLapack** ([Enum\\_matrice](#) enu\_mat, int lb\_totale, int dim, bool symetrique, double val\_init, [Enum\\_type\\_resolution\\_matri](#) type\_resol, [Enum\\_preconditionnement](#) type\_precondi)
- **MatLapack** (const [MatLapack](#) &a)
- [MatLapack](#) & **operator=** (const [MatLapack](#) &)
- void **Transfert\_vers\_mat** ([Mat\\_abstraite](#) &A)
- [Mat\\_abstraite](#) & **operator=** (const [Mat\\_abstraite](#) &)
- void **operator+=** (const [MatLapack](#) &mat\_pl)
- void **operator+=** (const [Mat\\_abstraite](#) &mat\_pl)
- void **operator-=** (const [MatLapack](#) &mat\_pl)
- void **operator-=** (const [Mat\\_abstraite](#) &mat\_pl)
- void **operator\*=** (const double r)
- int **operator==** (const [Mat\\_abstraite](#) &mat\_pl) const
- int **operator==** (const [MatLapack](#) &mat\_pl) const
- [Mat\\_abstraite](#) \* **NouvelElement** () const
- void **Affiche** () const
- void **Affiche1** (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j) const



- void [Affiche2](#) (int min\_i, int max\_i, int pas\_i, int min\_j, int max\_j, int pas\_j, int nd) const
- void [Change\\_Choix\\_resolution](#) (Enum\_type\_resolution\_matri type\_resol, Enum\_preconditionnement type\_precondi)
- void [Initialise](#) (double val\_init)
- void [Libere](#) ()
- int [Nb\\_ligne](#) () const
- int [Nb\\_colonne](#) () const
- int [Symetrie](#) () const
- double & [operator\(\)](#) (int i, int j)
- double [operator\(\)](#) (int i, int j) const
- bool [Existe](#) (int i, int j) const
- [Vecteur](#) & [Ligne\\_set](#) (int i)
- [Vecteur](#) [Ligne](#) (int i) const
- [Vecteur](#) [LigneSpe](#) (int i) const
- void [RemplaceLigneSpe](#) (int i, const [Vecteur](#) &v)
- void [MetValLigne](#) (int i, double val)
- [Vecteur](#) [Colonne](#) (int j) const
- [Vecteur](#) [ColonneSpe](#) (int j) const
- void [RemplaceColonneSpe](#) (int j, const [Vecteur](#) &v)
- void [MetValColonne](#) (int j, double val)
- [Vecteur](#) [Resol\\_syst](#) (const [Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- [Vecteur](#) & [Resol\\_systID](#) ([Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- [Tableau](#)< [Vecteur](#) > [Resol\\_syst](#) (const [Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- [Tableau](#)< [Vecteur](#) > & [Resol\\_systID](#) ([Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- [Vecteur](#) & [Resol\\_systID\\_2](#) (const [Vecteur](#) &b, [Vecteur](#) &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default)
- void [Preparation\\_resol](#) ()
- [Vecteur](#) [Simple\\_Resol\\_syst](#) (const [Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur](#) & [Simple\\_Resol\\_systID](#) ([Vecteur](#) &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Tableau](#)< [Vecteur](#) > [Simple\\_Resol\\_syst](#) (const [Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Tableau](#)< [Vecteur](#) > & [Simple\\_Resol\\_systID](#) ([Tableau](#)< [Vecteur](#) > &b, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur](#) & [Simple\\_Resol\\_systID\\_2](#) (const [Vecteur](#) &b, [Vecteur](#) &vortie, const double &tol=tol\_default, const int maxit=maxit\_default, const int restart=restart\_default) const
- [Vecteur](#) [Prod\\_vec\\_mat](#) (const [Vecteur](#) &vec) const
- [Vecteur](#) & [Prod\\_vec\\_mat](#) (const [Vecteur](#) &vec, [Vecteur](#) &resul) const
- [Vecteur](#) [Prod\\_mat\\_vec](#) (const [Vecteur](#) &vec) const
- [Vecteur](#) & [Prod\\_mat\\_vec](#) (const [Vecteur](#) &vec, [Vecteur](#) &resul) const
- double [Prod\\_Ligne\\_vec](#) (int iligne, const [Vecteur](#) &vec) const
- double [Prod\\_vec\\_col](#) (int icol, const [Vecteur](#) &vec) const
- double [vectT\\_mat\\_vec](#) (const [Vecteur](#) &vec1, const [Vecteur](#) &vec2) const
- int [Place](#) () const
- [Tableau](#)< [VeurPropre](#) > \* [V\\_Propres](#) ([MatLapack](#) &KG, [Tableau](#)< [VeurPropre](#) > &VP)
- void [Change\\_taille](#) (int nb\_li, int nb\_col, const double &val\_init=0.)
- [MatLapack](#) [operator+](#) (const [MatLapack](#) &mat\_pl) const
- [MatLapack](#) [operator-](#) (const [MatLapack](#) &mat\_pl) const
- [MatLapack](#) [operator-](#) () const
- [MatLapack](#) [operator\\*](#) (const [MatLapack](#) &mat\_pl) const
- [MatLapack](#) [operator\\*](#) (const double &coeff) const
- [MatLapack](#) [Transpose](#) () const
- [MatLapack](#) [Inverse](#) ()
- [MatLapack](#) & [Inverse](#) ([MatLapack](#) &mat)

## Fonctions membres protégées

- double & **Coor\_GE** (int i, int j)
- double **Coor\_GE\_const** (int i, int j) const
- double & **Coor\_GB** (int i, int j)
- double **Coor\_GB\_const** (int i, int j) const
- double & **Coor\_PB** (int i, int j)
- double **Coor\_PB\_const** (int i, int j) const
- double & **Coor\_PP** (int i, int j)
- double **Coor\_PP\_const** (int i, int j) const
- void **Verif\_nb\_composante** (const [Vecteur](#) &b, string nom\_routine) const
- void **Verif\_nb\_composante** (const [Tableau](#)< [Vecteur](#) > &b, string nom\_routine) const
- void **GestionErreurDg\_sv** (const int &info) const
- void **GestionErreurDpbsv** (const int &info) const
- void **GestionErreurInverse** (const int &info, int cas, const string nom\_routine\_lapack) const

## Attributs protégés

- ENTIER\_POUR\_CLAPACK **nb\_lign**
- ENTIER\_POUR\_CLAPACK **nb\_col**
- ENTIER\_POUR\_CLAPACK **ldb**
- ENTIER\_POUR\_CLAPACK **kl**
- ENTIER\_POUR\_CLAPACK **ku**
- ENTIER\_POUR\_CLAPACK **lda**
- [Vecteur](#) **mat**
- [Vecteur](#) **B**
- [Vecteur](#) **C**
- ENTIER\_POUR\_CLAPACK \* **ipiv**
- [MatLapack](#) \* **ptB**
- [Vecteur](#) **travail**
- double &(MatLapack::\* **Coor**)(int i, int j)
- double(MatLapack::\* **Coor\_const**)(int i, int j) const

## Amis

- istream & **operator**>> (istream &, [MatLapack](#) &)
- ostream & **operator**<< (ostream &, const [MatLapack](#) &)

## Membres hérités additionnels

### 6.496.1 Documentation des fonctions membres

#### 6.496.1.1 Affiche()

void MatLapack::Affiche ( ) const [virtual]  
 Implémente [Mat\\_abstraite](#).

#### 6.496.1.2 Affiche1()

```
void MatLapack::Affiche1 (
    int min_i,
    int max_i,
    int pas_i,
    int min_j,
    int max_j,
    int pas_j ) const [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

### 6.496.1.3 Affiche2()

```
void MatLapack::Affiche2 (
    int min_i,
    int max_i,
    int pas_i,
    int min_j,
    int max_j,
    int pas_j,
    int nd ) const [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

### 6.496.1.4 Change\_Choix\_resolution()

```
void MatLapack::Change_Choix_resolution (
    Enum_type_resolution_matri type_resol,
    Enum_preconditionnement type_precondi ) [virtual]
```

Réimplémentée à partir de [Mat\\_abstraite](#).

### 6.496.1.5 Colonne()

```
Vecteur MatLapack::Colonne (
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.496.1.6 ColonneSpe()

```
Vecteur MatLapack::ColonneSpe (
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.496.1.7 Existe()

```
bool MatLapack::Existe (
    int i,
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.496.1.8 Initialise()

```
void MatLapack::Initialise (
    double val_init ) [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.496.1.9 Libere()

```
void MatLapack::Libere ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).

### 6.496.1.10 Ligne()

```
Vecteur MatLapack::Ligne (
    int i ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.11 Ligne\_set()

```
Vecteur & MatLapack::Ligne_set (
    int i ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.12 LigneSpe()

```
Vecteur MatLapack::LigneSpe (
    int i ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.13 MetValColonne()

```
void MatLapack::MetValColonne (
    int j,
    double val ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.14 MetValLigne()

```
void MatLapack::MetValLigne (
    int i,
    double val ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.15 Nb\_colonne()

```
int MatLapack::Nb_colonne ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.16 Nb\_ligne()

```
int MatLapack::Nb_ligne ( ) const [inline], [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.17 NouvelElement()

```
Mat_abstraite * MatLapack::NouvelElement ( ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.18 operator>() [1/2]

```
double & MatLapack::operator() (
    int i,
    int j ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.19 operator>() [2/2]**

```
double MatLapack::operator() (
    int i,
    int j ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.20 operator\*=( )**

```
void MatLapack::operator*= (
    const double r ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.21 operator+=( )**

```
void MatLapack::operator+= (
    const Mat_abstraite & mat_pl ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.22 operator-=( )**

```
void MatLapack::operator-= (
    const Mat_abstraite & mat_pl ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.23 operator=( )**

```
Mat_abstraite & MatLapack::operator= (
    const Mat_abstraite & ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.24 operator==( )**

```
int MatLapack::operator==(
    const Mat_abstraite & mat_pl ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.25 Place( )**

```
int MatLapack::Place ( ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.26 Preparation\_resol( )**

```
void MatLapack::Preparation_resol ( ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.27 Prod\_Ligne\_vec( )**

```
double MatLapack::Prod_Ligne_vec (
    int iligne,
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.28 Prod\_mat\_vec() [1/2]

```
Vecteur MatLapack::Prod_mat_vec (  
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.29 Prod\_mat\_vec() [2/2]

```
Vecteur & MatLapack::Prod_mat_vec (  
    const Vecteur & vec,  
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.30 Prod\_vec\_col()

```
double MatLapack::Prod_vec_col (  
    int icol,  
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.31 Prod\_vec\_mat() [1/2]

```
Vecteur MatLapack::Prod_vec_mat (  
    const Vecteur & vec ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.32 Prod\_vec\_mat() [2/2]

```
Vecteur & MatLapack::Prod_vec_mat (  
    const Vecteur & vec,  
    Vecteur & resul ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.33 RemplaceColonneSpe()

```
void MatLapack::RemplaceColonneSpe (  
    int j,  
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

#### 6.496.1.34 RemplaceLigneSpe()

```
void MatLapack::RemplaceLigneSpe (  
    int i,  
    const Vecteur & v ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.35 Resol\_syst()** [1/2]

```
Tableau< Vecteur > MatLapack::Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.36 Resol\_syst()** [2/2]

```
Vecteur MatLapack::Resol_syst (
    const Vecteur & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.37 Resol\_systID()** [1/2]

```
Tableau< Vecteur > & MatLapack::Resol_systID (
    Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.38 Resol\_systID()** [2/2]

```
Vecteur & MatLapack::Resol_systID (
    Vecteur & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.39 Resol\_systID\_2()**

```
Vecteur & MatLapack::Resol_systID_2 (
    const Vecteur & b,
    Vecteur & vortie,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.40 Simple\_Resol\_syst()** [1/2]

```
Tableau< Vecteur > MatLapack::Simple_Resol_syst (
    const Tableau< Vecteur > & b,
    const double & tol = tol_default,
    const int maxit = maxit_default,
    const int restart = restart_default ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.41 Simple\_Resol\_syst()** [2/2]

```
Vecteur MatLapack::Simple_Resol_syst (
    const Vecteur & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.42 Simple\_Resol\_systID()** [1/2]

```
Tableau< Vecteur > & MatLapack::Simple_Resol_systID (
    Tableau< Vecteur > & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.43 Simple\_Resol\_systID()** [2/2]

```
Vecteur & MatLapack::Simple_Resol_systID (
    Vecteur & b,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.44 Simple\_Resol\_systID\_2()**

```
Vecteur & MatLapack::Simple_Resol_systID_2 (
    const Vecteur & b,
    Vecteur & vortie,
    const double & tol = tol_defaut,
    const int maxit = maxit_defaut,
    const int restart = restart_defaut ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.45 Symetrie()**

```
int MatLapack::Symetrie ( ) const [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.46 Transfert\_vers\_mat()**

```
void MatLapack::Transfert_vers_mat (
    Mat_abstraite & A ) [virtual]
```

Implémente [Mat\\_abstraite](#).

**6.496.1.47 vectT\_mat\_vec()**

```
double MatLapack::vectT_mat_vec (
    const Vecteur & vec1,
    const Vecteur & vec2 ) const [virtual]
```

Implémente [Mat\\_abstraite](#).



La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices/matrices\_lapack/MatLapack.h

## 6.497 Référence de la structure Matrix\_

### Types publics

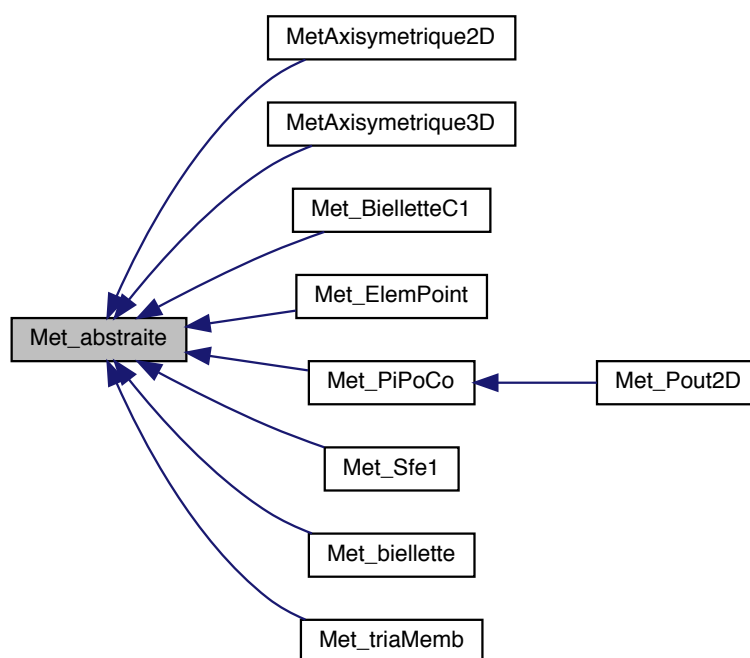
— enum **ref\_type** { **ref** = 1 }

La documentation de cette structure a été générée à partir du fichier suivant :

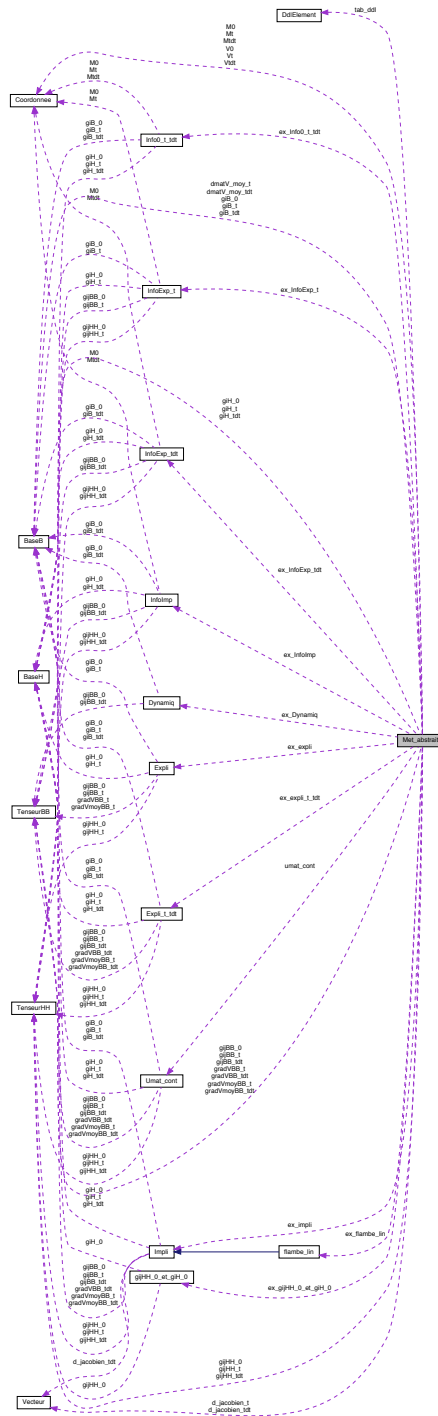
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices\_externes/MV++/mvmtpl\_GR.h

## 6.498 Référence de la classe Met\_abstraite

Graphe d'héritage de Met\_abstraite:



Graphe de collaboration de Met\_abstraite:



**Classes**

- class [Pour\\_def\\_Almansi](#)
- class [Pour\\_def\\_log](#)

**Fonctions membres publiques**

- **Met\_abstraite ()**  
*ceci dans la plupart des routines gourmandes !!!!!!!!!!!!!!!!!!!!!*

- **Met\_abstraite** (int dim\_base, int nvec\_base, const DdlElement &tabddl, const Tableau< Enum\_variable\_métrique > &tab, int nomb\_noeud, int nbddl\_sup\_xi=0)
- **Met\_abstraite** (const Met\_abstraite &)
- virtual Met\_abstraite & operator= (const Met\_abstraite &met)  
*surcharge de l'opérateur d'affectation*
- virtual void **Affiche** () const
- virtual const Expli & Cal\_explicit\_t (const Tableau< Noeud \* > &tab\_noeud, bool gradV\_instantane, const Mat\_pleine &tabDphi, int nombre\_noeud, const Vecteur &phi, bool premier\_calcul)  
*ceci dans la plupart des routines gourmandes !!!!!!!!!!!!!!!!!!!!!*
- virtual const Expli\_t\_tdt & Cal\_explicit\_tdt (const Tableau< Noeud \* > &tab\_noeud, bool gradV\_instantane, const Mat\_pleine &tabDphi, int nombre\_noeud, const Vecteur &phi, bool premier\_calcul)
- virtual const Expli\_t\_tdt & Conteneur\_Expli\_tdt () const
- virtual const Impli & Cal\_implicit (const Tableau< Noeud \* > &tab\_noeud, bool gradV\_instantane, const Mat\_pleine &tabDphi, int nombre\_noeud, const Vecteur &phi, bool premier\_calcul, bool avec\_var\_Xi=true)
- virtual const gijHH\_0\_et\_giH\_0 & Cal\_gijHH\_0\_et\_giH\_0\_apres\_im\_expli ()
- void **Recup\_grandeur\_0\_t** (BaseB &ggiB\_0, BaseH &ggiH\_0, BaseB &ggiB\_t, BaseH &ggiH\_t, TenseurBB &ggijBB\_0, TenseurHH &ggijHH\_0, TenseurBB &ggijBB\_t, TenseurHH &ggijHH\_t, TenseurBB &gggradVmoy, double &gjacobien\_t, double &gjacobien\_0)
- void **Recup\_grandeur\_0** (BaseB &ggiB\_0, BaseH &ggiH\_0, TenseurBB &ggijBB\_0, TenseurHH &ggijHH\_0, double &gjacobien\_0)
- virtual const Umat\_cont & Construction\_Umat (const Met\_abstraite::Impli &umat\_cont)
- virtual const Umat\_cont & Construction\_Umat (const Met\_abstraite::Umat\_cont &umat\_cont)
- virtual const Umat\_cont & Conteneur\_Umat () const
- virtual const Dynamiq & Cal\_pourMatMass (const Tableau< Noeud \* > &tab\_noeud, const Mat\_pleine &tabDphi, int nombre\_noeud, const Vecteur &phi)
- virtual const flambe\_lin & Cal\_flambe\_lin (const Tableau< Noeud \* > &tab\_noeud, bool gradV\_instantane, const Mat\_pleine &tabDphi, int nombre\_noeud, const Vecteur &phi, bool premier\_calcul)
- virtual double **DonneeInterpoleeScalaire** (const Tableau< Noeud \* > &tab\_noeud, const Vecteur &phi, int nombre\_noeud, Enum\_ddl enu, Enum\_dure temps) const
- virtual Tableau< double > & **DerDonneeInterpoleeScalaire** (Tableau< double > &d\_A, const Tableau< Noeud \* > &tab\_noeud, const Vecteur &phi, int nombre\_noeud, Enum\_ddl enu) const
- virtual CoordonneeB & **GradDonneeInterpoleeScalaire** (CoordonneeB &gradB, const Tableau< Noeud \* > &tab\_noeud, const Mat\_pleine &dphi, int nombre\_noeud, Enum\_ddl enu, Enum\_dure temps) const
- virtual Tableau< CoordonneeB > & **DerGradDonneeInterpoleeScalaire** (Tableau< CoordonneeB > &d\_gradB, const Tableau< Noeud \* > &tab\_noeud, const Vecteur &phi, const Mat\_pleine &dphi, int nombre\_noeud, Enum\_ddl enu) const
- virtual Tableau2< CoordonneeB > \* **Der2GradDonneeInterpoleeScalaire** (Tableau2< CoordonneeB > &d2\_gradB, const Tableau< Noeud \* > &tab\_noeud, const Vecteur &phi, const Mat\_pleine &dphi, int nombre\_noeud, Enum\_ddl enu) const
- const Tableau2< TenseurBB \* > & **CalVar2GijBBtdt** (bool total, const Mat\_pleine &dphi, int nombre\_noeud, const Vecteur &phi)
- virtual const Infolmp & Cal\_Infolmp (const Tableau< Noeud \* > &tab\_noeud, const Mat\_pleine &tabDphi, const Vecteur &phi, int nombre\_noeud)
- virtual const Infolmp & **Recup\_Infolmp** ()
- virtual const InfoExp\_t & Cal\_InfoExp\_t (const Tableau< Noeud \* > &tab\_noeud, const Mat\_pleine &tabDphi, const Vecteur &phi, int nombre\_noeud)
- virtual const InfoExp\_t & **Recup\_InfoExp\_t** ()
- virtual const InfoExp\_tdt & Cal\_InfoExp\_tdt (const Tableau< Noeud \* > &tab\_noeud, const Mat\_pleine &tabDphi, const Vecteur &phi, int nombre\_noeud)
- virtual const InfoExp\_tdt & **Recup\_InfoExp\_tdt** ()
- virtual const Info0\_t\_tdt & Cal\_Info0\_t\_tdt (const Tableau< Noeud \* > &tab\_noeud, const Mat\_pleine &tabDphi, const Vecteur &phi, int nombre\_noeud)
- virtual const Info0\_t\_tdt & **Recup\_Info0\_t\_tdt** ()
- virtual const Info\_et\_métrique\_0\_t\_tdt & **Recup\_Info\_et\_métrique\_0\_t\_tdt** ()
- virtual const Pour\_def\_Almansi Cal\_pour\_def\_Almansi\_au\_temps (Enum\_dure temps, const Tableau< Noeud \* > &tab\_noeud, const Mat\_pleine &dphi, int nombre\_noeud, const Vecteur &phi)
- virtual const Pour\_def\_log Cal\_pour\_def\_log\_au\_temps (Enum\_dure temps, const Tableau< Noeud \* > &tab\_noeud, const Mat\_pleine &dphi, int nombre\_noeud, const Vecteur &phi)
- virtual const Coordonnee & **PointM\_0** (const Tableau< Noeud \* > &tab\_noeud, const Vecteur &phi)
- virtual const Coordonnee & **PointM\_t** (const Tableau< Noeud \* > &tab\_noeud, const Vecteur &phi)

- virtual const `Tableau< Coordonnee > & D_PointM_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`)
- virtual const `Coordonnee & PointM_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`)
- virtual const `Tableau< Coordonnee > & D_PointM_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`)
- virtual const `Coordonnee & VitesseM_0` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`)
- virtual const `Coordonnee & VitesseM_0` (const `Noeud *noeud`)
- virtual const `Coordonnee & VitesseM_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`)
- virtual const `Coordonnee & VitesseM_t` (const `Noeud *noeud`)
- virtual const `Tableau< Coordonnee > & D_VitesseM_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, bool ddl\_vitesse)
- virtual const `Tableau< Coordonnee > & D_VitesseM_t` (const `Noeud *noeud`, bool ddl\_vitesse)
- virtual const `Coordonnee & VitesseM_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`)
- virtual const `Coordonnee & VitesseM_tdt` (const `Noeud *noeud`)
- virtual const `Tableau< Coordonnee > & D_VitesseM_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, bool ddl\_vitesse)
- virtual const `Tableau< Coordonnee > & D_VitesseM_tdt` (const `Noeud *noeud`, bool ddl\_vitesse)
- virtual const `BaseB & BaseNat_0` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, const `Vecteur &phi`)
- virtual const `BaseB & BaseNat_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, const `Vecteur &phi`)
- virtual const `BaseB & BaseNat_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, const `Vecteur &phi`)
- virtual const `Tableau< BaseB > & d_BaseNat_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, const `Vecteur &phi`)
- virtual void `BaseND_0` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, const `Vecteur &phi`, `BaseB` &bB, `BaseH` &bH)
- virtual void `BaseND_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, const `Vecteur &phi`, `BaseB` &bB, `BaseH` &bH)
- virtual void `BaseND_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, const `Vecteur &phi`, `BaseB` &bB, `BaseH` &bH)
- double `JacobienInitial` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre\_↵\_noeud, const `Vecteur &phi`)
- virtual void `PlusInitVariables` (`Tableau< Enum_variable_metrique >` &tab)
- virtual void `Dim_NbVec` (int dim\_base, int nbvec\_base, `DdlElement` &tabddl, int nb\_noeud\_interpol)
- const int & `Dim_vec_base` () const
- const int & `Nbvec_des_bases` () const
- const int & `Nombre_de_noeud` () const
- const `DdlElement` & `Tab_ddl_element` () const

## Fonctions membres protégées

- virtual void `Calcul_M0` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)  
*ceci dans la plupart des routines gourmandes !!!!!!!!!!!!!!!!!!!!!*
- virtual void `Calcul_Mt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_d_Mt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_Mtdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_d_Mtdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_V0` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_V0` (const `Noeud *noeud`)
- virtual void `Calcul_Vt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_Vt` (const `Noeud *noeud`)
- virtual void `Calcul_d_Vt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_d_Vt` (const `Noeud *noeud`)
- virtual void `Calcul_Vtdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_Vtdt` (const `Noeud *noeud`)
- virtual void `Calcul_d_Vtdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_noeud)
- virtual void `Calcul_d_Vtdt` (const `Noeud *noeud`)
- virtual void `Calcul_V_moy0` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_↵\_noeud)
- virtual void `Calcul_V_moy0` (const `Noeud *noeud`)
- virtual void `Calcul_V_moyt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur &phi`, int nombre\_↵\_noeud)

- virtual void **Calcul\_V\_moyt** (const [Noeud](#) \*noeud)
- virtual void **Calcul\_d\_V\_moyt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Vecteur](#) &phi, int nombre\_↔noeud)
- virtual void **Calcul\_d\_V\_moyt** (const [Noeud](#) \*noeud)
- virtual void **Calcul\_V\_moytdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Vecteur](#) &phi, int nombre\_↔noeud)
- virtual void **Calcul\_V\_moytdt** (const [Noeud](#) \*noeud)
- virtual void **Calcul\_d\_V\_moytdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Vecteur](#) &phi, int nombre\_↔noeud)
- virtual void **Calcul\_d\_V\_moytdt** (const [Noeud](#) \*noeud)
- virtual void **Calcul\_giB\_0** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &tabDphi, int nombre\_noeud, const [Vecteur](#) &phi)
- virtual void **Calcul\_giB\_t** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &tabDphi, int nombre\_↔noeud, const [Vecteur](#) &phi)
- virtual void **Calcul\_giB\_tdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &tabDphi, int nombre\_noeud, const [Vecteur](#) &phi)
- virtual void **Calcul\_gijBB\_0** ()
- virtual void **Calcul\_gijBB\_t** ()
- virtual void **Calcul\_gijBB\_tdt** ()
- virtual void **Calcul\_gijHH\_0** ()
- virtual void **Calcul\_gijHH\_t** ()
- virtual void **Calcul\_gijHH\_tdt** ()
- virtual void **Calcul\_gradVBB\_t** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &tabDphi, int nombre\_noeud)
- virtual void **Calcul\_gradVBB\_tdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &tabDphi, int nombre\_noeud)
- virtual void **Calcul\_gradVBB\_moyen\_t** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &tab↔Dphi, int nombre\_noeud)
- virtual void **Calcul\_gradVBB\_moyen\_tdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &tabDphi, int nombre\_noeud)
- virtual void **Calcul\_giH\_0** ()
- virtual void **Calcul\_giH\_t** ()
- virtual void **Calcul\_giH\_tdt** ()
- virtual void **Jacobien\_0** ()
- virtual void **Jacobien\_t** ()
- virtual void **Jacobien\_tdt** ()
- virtual void **D\_giB\_t** (const [Mat\\_pleine](#) &tabDphi, int nbnoeu, const [Vecteur](#) &phi)
- virtual void **D\_giB\_tdt** (const [Mat\\_pleine](#) &tabDphi, int nbnoeu, const [Vecteur](#) &phi)
- virtual void **D\_giH\_t** ()
- virtual void **D\_giH\_tdt** ()
- virtual void **DgradVBB\_t** (const [Mat\\_pleine](#) &dphi)
- virtual void **DgradVBB\_tdt** (const [Mat\\_pleine](#) &dphi)
- virtual void **DgradVmoyBB\_t** (const [Mat\\_pleine](#) &dphi, const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- virtual void **DgradVmoyBB\_tdt** (const [Mat\\_pleine](#) &dphi)
- virtual void **D\_gijBB\_t** ()
- virtual void **D\_gijBB\_tdt** ()
- virtual void **D\_gijHH\_t** ()
- virtual void **D\_gijHH\_tdt** ()
- virtual void **D2\_gijBB\_tdt** ()
- virtual void **Djacobien\_t** ()
- virtual void **Djacobien\_tdt** ()
- void **Calcul\_gijBB** ([BaseB](#) &giB, [TenseurBB](#) &gijBB)
- void **Calcul\_giH** ([BaseB](#) \*giB, [TenseurHH](#) \*gijHH, [BaseH](#) \*giH)
- bool **Existe** ([Tableau](#)< [Enum\\_variable\\_metrique](#) > &tab, const [Enum\\_variable\\_metrique](#) &x)
- int **Existe\_num** ([Tableau](#)< [Enum\\_variable\\_metrique](#) > &tab, const [Enum\\_variable\\_metrique](#) &x)
- void **Allocation** ()
- void **Deallocation** ()
- void **Deallocation** ([Tableau](#)< [Enum\\_variable\\_metrique](#) > &tib)
- void **Copie\_met** (const [Met\\_abstraite](#) &a)
- void **Init\_conteneur\_de\_retour\_methode** ()
- void **Change\_nbddl\_sup\_xi** (int nbddl\_sup\_xi\_new)

## Attributs protégés

- DdlElement **tab\_ddl**
- int **nbdll\_sup\_xi**
- int **dim\_base**
- int **nbvec\_base**
- int **nomb\_noeud**
- Tableau< Enum\_variable\_metrique > **tab**
- Coordonnee \* **M0**
- Coordonnee \* **Mt**
- Coordonnee \* **Mtdt**
- Tableau< Coordonnee > \* **d\_Mt**
- Tableau< Coordonnee > \* **d\_Mtdt**
- Coordonnee \* **V0**
- Coordonnee \* **Vt**
- Coordonnee \* **Vtdt**
- Tableau< Coordonnee > \* **d\_Vt**
- Tableau< Coordonnee > \* **d\_Vtdt**
- Tableau< Coordonnee > **V\_moy\_t**
- Tableau< Coordonnee > **V\_moy\_tdt**
- BaseB \* **dmatV\_moy\_t**
- BaseB \* **dmatV\_moy\_tdt**
- BaseB \* **giB\_0**
- BaseB \* **giB\_t**
- BaseB \* **giB\_tdt**
- BaseH \* **giH\_0**
- BaseH \* **giH\_t**
- BaseH \* **giH\_tdt**
- TenseurBB \* **gijBB\_0**
- TenseurBB \* **gijBB\_t**
- TenseurBB \* **gijBB\_tdt**
- TenseurHH \* **gijHH\_0**
- TenseurHH \* **gijHH\_t**
- TenseurHH \* **gijHH\_tdt**
- double **jacobien\_0**
- double **jacobien\_t**
- double **jacobien\_tdt**
- Tableau< BaseB > \* **d\_giB\_t**
- Tableau< BaseB > \* **d\_giB\_tdt**
- Tableau< BaseH > \* **d\_giH\_t**
- Tableau< BaseH > \* **d\_giH\_tdt**
- Tableau< TenseurBB \* > **d\_gijBB\_t**
- Tableau< TenseurBB \* > **d\_gijBB\_tdt**
- Tableau2< TenseurBB \* > **d2\_gijBB\_tdt**
- Tableau< TenseurHH \* > **d\_gijHH\_t**
- Tableau< TenseurHH \* > **d\_gijHH\_tdt**
- Vecteur **d\_jacobien\_t**
- Vecteur **d\_jacobien\_tdt**
- TenseurBB \* **gradVmoyBB\_t**
- TenseurBB \* **gradVmoyBB\_tdt**
- TenseurBB \* **gradVBB\_t**
- TenseurBB \* **gradVBB\_tdt**
- Tableau< TenseurBB \* > **d\_gradVmoyBB\_t**
- Tableau< TenseurBB \* > **d\_gradVmoyBB\_tdt**
- Tableau< TenseurBB \* > **d\_gradVBB\_t**
- Tableau< TenseurBB \* > **d\_gradVBB\_tdt**
- Impli **ex\_impli**
- Expli **ex\_expli**
- Expli\_t\_tdt **ex\_expli\_t\_tdt**
- **gijHH\_0\_et\_giH\_0** **ex\_gijHH\_0\_et\_giH\_0**
- Dynamiq **ex\_Dynamiq**
- flambe\_lin **ex\_flambe\_lin**
- Umat\_cont **umat\_cont**
- Infolmp **ex\_Infolmp**
- InfoExp\_t **ex\_InfoExp\_t**
- InfoExp\_tdt **ex\_InfoExp\_tdt**
- Info0\_t\_tdt **ex\_Info0\_t\_tdt**
- int **posi\_tab**

## 6.498.1 Documentation des fonctions membres

### 6.498.1.1 Cal\_explicit\_t()

```
const Met_abstraite::Expli & Met_abstraite::Cal_explicit_t (
    const Tableau< Noeud * > & tab_noeud,
    bool gradV_instantane,
    const Mat_pleine & tabDphi,
    int nombre_noeud,
    const Vecteur & phi,
    bool premier_calcul ) [virtual]
```

ceci dans la plupart des routines gourmandes!!!!!!!!!!!!!!!!!!!!!!

Calcul\_gradVBB\_moyen\_t(tab\_noeud,dphi,nombre\_noeud); // gradient de vitesse moyen

### 6.498.1.2 Cal\_explicit\_tdt()

```
const Met_abstraite::Expli_t_tdt & Met_abstraite::Cal_explicit_tdt (
    const Tableau< Noeud * > & tab_noeud,
    bool gradV_instantane,
    const Mat_pleine & tabDphi,
    int nombre_noeud,
    const Vecteur & phi,
    bool premier_calcul ) [virtual]
```

Calcul\_gradVBB\_moyen\_t(tab\_noeud,dphi,nombre\_noeud); // gradient de vitesse moyen

Calcul\_gradVBB\_moyen\_tdt(tab\_noeud,dphi,nombre\_noeud); // gradient de vitesse moyen

### 6.498.1.3 Cal\_implicit()

```
const Met_abstraite::Impli & Met_abstraite::Cal_implicit (
    const Tableau< Noeud * > & tab_noeud,
    bool gradV_instantane,
    const Mat_pleine & tabDphi,
    int nombre_noeud,
    const Vecteur & phi,
    bool premier_calcul,
    bool avec_var_Xi = true ) [virtual]
```

Calcul\_gradVBB\_moyen\_t(tab\_noeud,dphi,nombre\_noeud); // gradient de vitesse moyen gradVmoyBB\_t(dphi); // calcul variation du gradient moyen à t

Calcul\_gradVBB\_moyen\_tdt(tab\_noeud,dphi,nombre\_noeud); // gradient de vitesse moyen DgradVmoyBB\_tdt(dphi); // calcul variation du gradient moyen à tdt

### 6.498.1.4 Calcul\_M0()

```
void Met_abstraite::Calcul_M0 (
    const Tableau< Noeud * > & tab_noeud,
    const Vecteur & phi,
    int nombre_noeud ) [protected], [virtual]
```

ceci dans la plupart des routines gourmandes!!!!!!!!!!!!!!!!!!!!!!

Réimplémentée dans [Met\\_PiPoCo](#), et [Met\\_Sfe1](#).

### 6.498.1.5 operator=()

```
Met_abstraite & Met_abstraite::operator= (
    const Met_abstraite & met ) [virtual]
```

surcharge de l'opérateur d'affectation

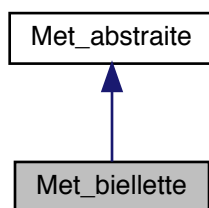
Réimplémentée dans [Met\\_biellette](#), [Met\\_BielletteC1](#), [Met\\_Pout2D](#), [Met\\_PiPoCo](#), [MetAxisymetrique2D](#), [MetAxisymetrique3D](#), [Met\\_ElemPoint](#), et [Met\\_Sfe1](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔ abstraite.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔ abstraite1s2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔ abstraite2s2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔ abstraite3s2.cc

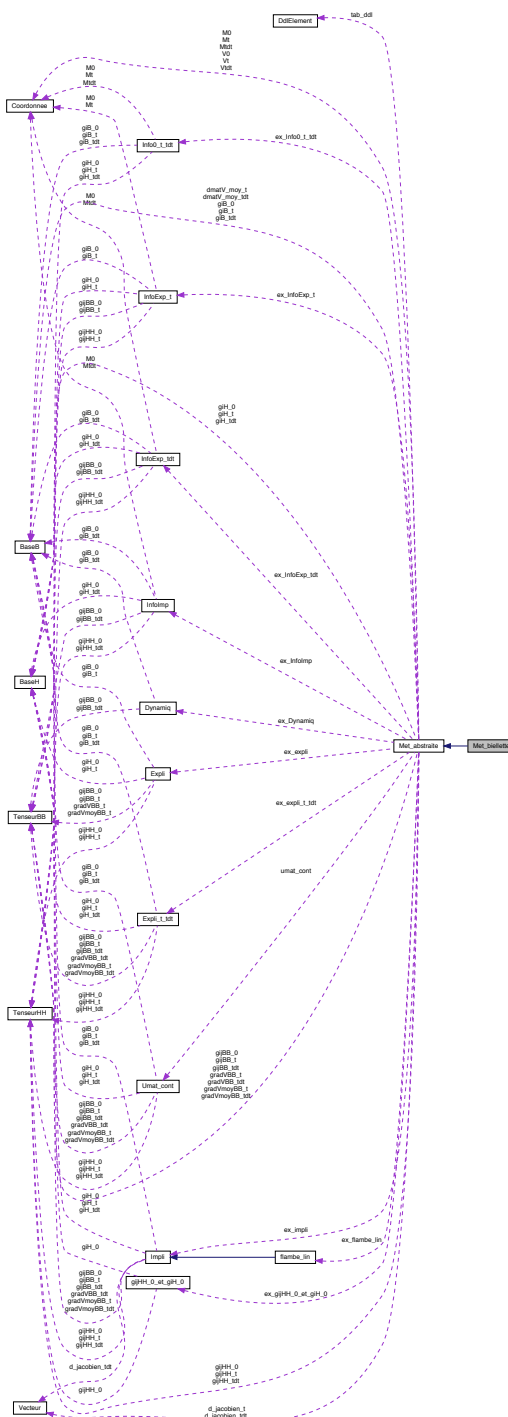
## 6.499 Référence de la classe Met\_biellette

Graphe d'héritage de Met\_biellette:





Graphe de collaboration de Met\_biellette:



**Fonctions membres publiques**

- **Met\_biellette** (int dim\_base, const DdlElement &tabddl, const Tableau< Enum\_variable\_metrrique > &tab, int nomb\_noeud)
- **Met\_biellette** (const Met\_biellette &)
- **Met\_abstraite** & operator= (const Met\_abstraite &met)  
*surcharge de l'opérateur d'affectation*

## Fonctions membres protégées

- void [Calcul\\_giB\\_0](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &dphi, int nombre\_noeud, const [Vecteur](#) &phi)
- void [Calcul\\_giB\\_t](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &dphi, int nombre\_noeud, const [Vecteur](#) &phi)
- void [Calcul\\_giB\\_tdt](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &dphi, int nombre\_noeud, const [Vecteur](#) &phi)
- void [Djacobien\\_t](#) ()
- void [Djacobien\\_tdt](#) ()

## Membres hérités additionnels

### 6.499.1 Documentation des fonctions membres

#### 6.499.1.1 Calcul\_giB\_0()

```
void Met_biellette::Calcul_giB_0 (
    const Tableau< Noeud * > & tab_noeud,
    const Mat\_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.499.1.2 Calcul\_giB\_t()

```
void Met_biellette::Calcul_giB_t (
    const Tableau< Noeud * > & tab_noeud,
    const Mat\_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.499.1.3 Calcul\_giB\_tdt()

```
void Met_biellette::Calcul_giB_tdt (
    const Tableau< Noeud * > & tab_noeud,
    const Mat\_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.499.1.4 Djacobien\_t()

```
void Met_biellette::Djacobien_t ( ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.499.1.5 Djacobien\_tdt()

```
void Met_biellette::Djacobien_tdt ( ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.499.1.6 operator=()

```
Met_abstraite & Met_biellette::operator= (  
    const Met_abstraite & met ) [inline], [virtual]  
surcharge de l'opérateur d'affectation
```

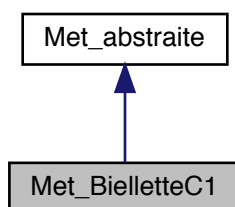
Réimplémentée à partir de [Met\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

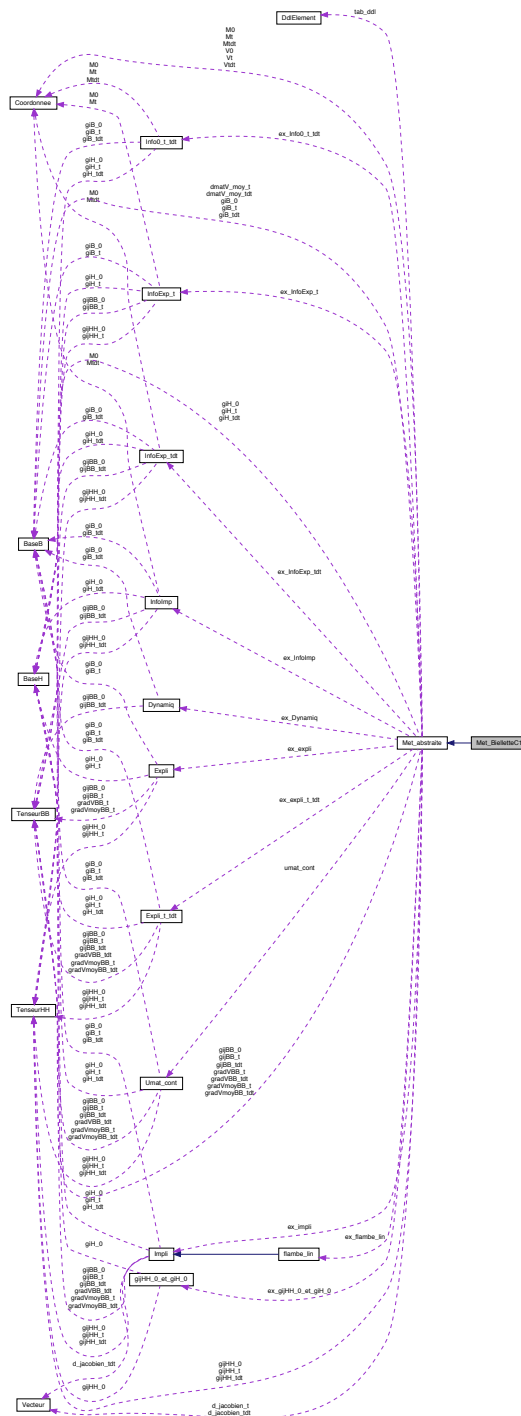
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met\_biellette.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met\_biellette.cc

## 6.500 Référence de la classe Met\_BielletteC1

Grphe d'héritage de Met\_BielletteC1:



Graphe de collaboration de Met\_BielletteC1:



## Fonctions membres publiques

- **Met\_BielletteC1** (int dim\_base, const DdElement &tabddl, const Tableau< Enum\_variable\_metrrique > &tab, int nomb\_noeud)
- **Met\_BielletteC1** (const Met\_BielletteC1 &)
- **Met\_abstraite** & **operator=** (const Met\_abstraite &met)  
*surcharge de l'opérateur d'affectation*

## Fonctions membres protégées

- void [Calcul\\_giB\\_0](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &dphi, int nombre\_noeud, const [Vecteur](#) &phi)
- void [Calcul\\_giB\\_t](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &dphi, int nombre\_noeud, const [Vecteur](#) &phi)
- void [Calcul\\_giB\\_tdt](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, const [Mat\\_pleine](#) &dphi, int nombre\_noeud, const [Vecteur](#) &phi)
- void [Djacobien\\_t](#) ()
- void [Djacobien\\_tdt](#) ()

## Membres hérités additionnels

### 6.500.1 Documentation des fonctions membres

#### 6.500.1.1 Calcul\_giB\_0()

```
void Met_BielletteC1::Calcul_giB_0 (
    const Tableau< Noeud * > & tab_noeud,
    const Mat\_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.500.1.2 Calcul\_giB\_t()

```
void Met_BielletteC1::Calcul_giB_t (
    const Tableau< Noeud * > & tab_noeud,
    const Mat\_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.500.1.3 Calcul\_giB\_tdt()

```
void Met_BielletteC1::Calcul_giB_tdt (
    const Tableau< Noeud * > & tab_noeud,
    const Mat\_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.500.1.4 Djacobien\_t()

```
void Met_BielletteC1::Djacobien_t ( ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.500.1.5 Djacobien\_tdt()

```
void Met_BielletteC1::Djacobien_tdt ( ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.500.1.6 operator=()

```
Met_abstraite & Met_BielletteC1::operator= (  
    const Met_abstraite & met ) [inline], [virtual]  
surcharge de l'opérateur d'affectation
```

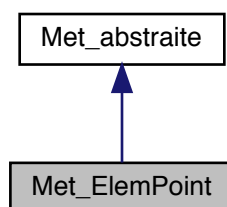
Réimplémentée à partir de [Met\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

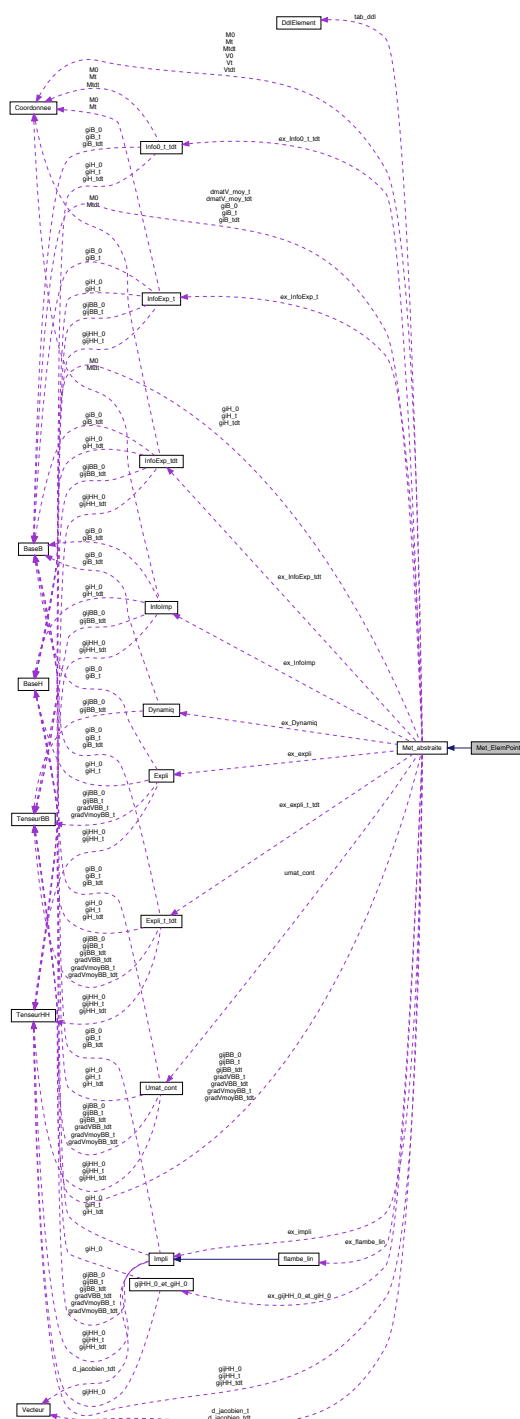
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met\_BielletteC1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met\_bielletteC1.cc

## 6.501 Référence de la classe Met\_ElemPoint

Graphe d'héritage de Met\_ElemPoint:



Graphe de collaboration de Met\_ElemPoint:



## Fonctions membres publiques

- **Met\_ElemPoint** (int dim\_base, int nbvec, const DdlElement &tabddl, const Tableau< Enum\_variable\_métrique > &tab, int nomb\_noeud)
- **Met\_ElemPoint** (int dim\_base, int nbvec, const DdlElement &tabddl, const Tableau< Enum\_variable\_métrique > &tab)
- **Met\_ElemPoint** (const Met\_ElemPoint &)
- **Met\_abstraite** & operator= (const Met\_abstraite &met)

*surcharge de l'opérateur d'affectation*

- const [Umat\\_cont](#) & [Calcul\\_grandeurs\\_Umat](#) (bool premier\_calcul)
- void [Mise\\_a\\_jour](#) (const [UmatAbaqus](#) &umat)

### Fonctions membres protégées

- void [Calcul\\_giB\\_0](#) (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &)
- void [Calcul\\_giB\\_t](#) (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &)
- void [Calcul\\_giB\\_tdt](#) (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &)

### Attributs protégés

- bool [metrique\\_imposee](#)

## 6.501.1 Documentation des fonctions membres

### 6.501.1.1 Calcul\_giB\_0()

```
void Met_ElemPoint::Calcul_giB_0 (
    const Tableau< Noeud * > & ,
    const Mat\_pleine & ,
    int ,
    const Vecteur & ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.501.1.2 Calcul\_giB\_t()

```
void Met_ElemPoint::Calcul_giB_t (
    const Tableau< Noeud * > & ,
    const Mat\_pleine & ,
    int ,
    const Vecteur & ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.501.1.3 Calcul\_giB\_tdt()

```
void Met_ElemPoint::Calcul_giB_tdt (
    const Tableau< Noeud * > & ,
    const Mat\_pleine & ,
    int ,
    const Vecteur & ) [inline], [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.501.1.4 Calcul\_grandeurs\_Umat()

```
const Met\_abstraite::Umat\_cont & Met\_ElemPoint::Calcul\_grandeurs\_Umat (
    bool premier_calcul )
Calcul\_gradVBB\_moyen\_t(tab\_noeud,dphi,nombre\_noeud); // gradient de vitesse moyen
Calcul\_gradVBB\_moyen\_tdt(tab\_noeud,dphi,nombre\_noeud); // gradient de vitesse moyen
```



### 6.501.1.5 operator=()

```
Met_abstraite & Met_ElemPoint::operator= (  
    const Met_abstraite & met ) [inline], [virtual]
```

surcharge de l'opérateur d'affectation

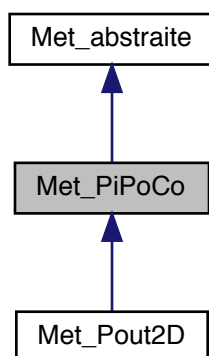
Réimplémentée à partir de [Met\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

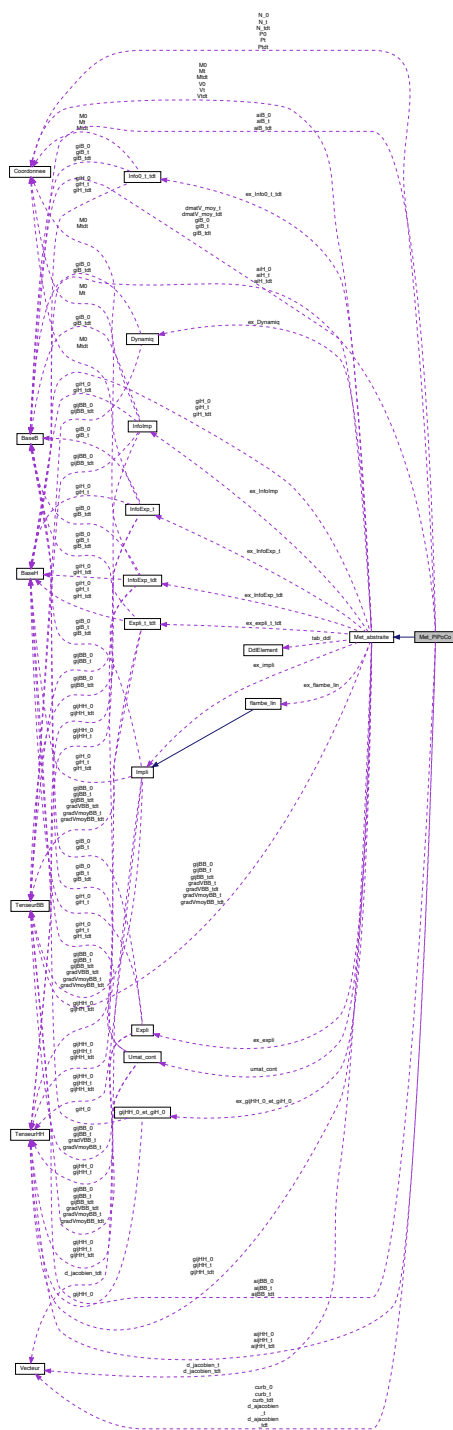
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/Met\_ElemPoint.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/Met\_ElemPoint.cc

## 6.502 Référence de la classe Met\_PiPoCo

Grphe d'héritage de Met\_PiPoCo:



Graphe de collaboration de Met\_PiPoCo:



## Fonctions membres publiques

- **Met\_PiPoCo** (int dim\_base, int nbvec, const DdiElement &tabddl, const Tableau< Enum\_variable\_metrrique > &tab, int nomb\_noeud)
- **Met\_PiPoCo** (const Met\_PiPoCo &)
- **Met\_abstraite** & operator= (const Met\_abstraite &met)  
*surcharge de l'opérateur d'affectation*
- virtual **Met\_PiPoCo** & operator= (const Met\_PiPoCo &met)

- const Met\_abstraite::Expli & **Cal\_explicit\_t** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, bool gradV\_instantane, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, int nombre\_noeud, [Tableau](#)< [Vecteur](#) const \* > &tabphi, bool premier\_calcul)
- const Met\_abstraite::Expli & **Cal\_explicit\_simple\_t** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, bool gradV\_instantane, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, int nombre\_noeud, [Tableau](#)< [Vecteur](#) const \* > &tabphi, bool premier\_calcul)
- const Met\_abstraite::Expli\_t\_tdt & **Cal\_explicit\_tdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, bool gradV\_instantane, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, int nombre\_noeud, [Tableau](#)< [Vecteur](#) const \* > &tabphi, bool premier\_calcul)
- const Met\_abstraite::Expli\_t\_tdt & **Cal\_explicit\_simple\_tdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, bool gradV\_instantane, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, int nombre\_noeud, [Tableau](#)< [Vecteur](#) const \* > &tabphi, bool premier\_calcul)
- const Met\_abstraite::Impli & **Cal\_implicit** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, bool gradV\_instantane, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, int nombre\_noeud, [Tableau](#)< [Vecteur](#) const \* > &tabphi, bool premier\_calcul)
- const Met\_abstraite::Impli & **Cal\_implicit\_simple** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, bool gradV\_instantane, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, int nombre\_noeud, [Tableau](#)< [Vecteur](#) const \* > &tabphi, bool premier\_calcul)
- const Met\_abstraite::InfoImp & **Cal\_InfoImp** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, [Tableau](#)< [Vecteur](#) const \* > &tabphi, int nombre\_noeud)
- const Met\_abstraite::InfoExp\_t & **Cal\_InfoExp\_t** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, [Tableau](#)< [Vecteur](#) const \* > &tabphi, int nombre\_noeud)
- const Met\_abstraite::InfoExp\_tdt & **Cal\_InfoExp\_tdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, [Tableau](#)< [Mat\\_pleine](#) const \* > &tabdphi, [Tableau](#)< [Vecteur](#) const \* > &tabphi, int nombre\_noeud)
- void **PlusInitVariables** ([Tableau](#)< [Enum\\_variable\\_metrrique](#) > &tab)

## Fonctions membres protégées

- virtual void **Calcul\_N\_0** ()
- virtual void **Calcul\_N\_t** ()
- virtual void **Calcul\_N\_tdt** ()
- void **Calcul\_M0** (const [Tableau](#)< [Noeud](#) \* > &, const [Vecteur](#) &phi, int)  
*ceci dans la plupart des routines gourmandes !!!!!!!!!!!!!!!!!!!!!*
- void **Calcul\_Mt** (const [Tableau](#)< [Noeud](#) \* > &, const [Vecteur](#) &phi, int)
- void **Calcul\_Mtdt** (const [Tableau](#)< [Noeud](#) \* > &, const [Vecteur](#) &phi, int)
- virtual void **Calcul\_giB\_0** (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- virtual void **Calcul\_giB\_t** (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- virtual void **Calcul\_giB\_tdt** (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- virtual void **D\_giB\_t** (const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- virtual void **D\_giB\_tdt** (const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- virtual [Vecteur](#) & **courbure\_0** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)=0
- virtual [Vecteur](#) & **courbure\_t** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)=0
- virtual [Vecteur](#) & **courbure\_tdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)=0
- virtual void **Dcourbure\_t** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, [Vecteur](#) &curb, [TabOper](#)< [Vecteur](#) > &dcurb)=0
- virtual void **Dcourbure\_tdt** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, [Vecteur](#) &curb, [TabOper](#)< [Vecteur](#) > &dcurb)=0
- void **AllocPiPoCo** ()
- void **DeAllocPiPoCo** ()
- void **DeAllocPiPoCo** ([Tableau](#)< [Enum\\_variable\\_metrrique](#) > &tib)
- void **Copie\_metPiPoCo** (const [Met\\_PiPoCo](#) &a)

## Attributs protégés

- [Tableau](#)< [Coordonnee](#) > **la**
- [Vecteur](#) **curb\_0**
- [Vecteur](#) **curb\_t**
- [Vecteur](#) **curb\_tdt**
- [TabOper](#)< [Vecteur](#) > **dcurb\_t**
- [TabOper](#)< [Vecteur](#) > **dcurb\_tdt**
- [Coordonnee](#) **N\_0**
- [Coordonnee](#) **N\_t**

- [Coordonnee N\\_tdt](#)
- [Coordonnee \\* P0](#)
- [Coordonnee \\* Pt](#)
- [Coordonnee \\* Ptdt](#)
- [Tableau< Coordonnee > \\* d\\_Pt](#)
- [Tableau< Coordonnee > \\* d\\_Ptdt](#)
- [BaseB \\* aiB\\_0](#)
- [BaseB \\* aiB\\_t](#)
- [BaseB \\* aiB\\_tdt](#)
- [BaseH \\* aiH\\_0](#)
- [BaseH \\* aiH\\_t](#)
- [BaseH \\* aiH\\_tdt](#)
- [TenseurBB \\* aijBB\\_0](#)
- [TenseurBB \\* aijBB\\_t](#)
- [TenseurBB \\* aijBB\\_tdt](#)
- [TenseurHH \\* aijHH\\_0](#)
- [TenseurHH \\* aijHH\\_t](#)
- [TenseurHH \\* aijHH\\_tdt](#)
- [double ajacobien\\_0](#)
- [double ajacobien\\_t](#)
- [double ajacobien\\_tdt](#)
- [Tableau< BaseB > \\* d\\_aiB\\_t](#)
- [Tableau< BaseB > \\* d\\_aiB\\_tdt](#)
- [Tableau< BaseH > \\* d\\_aiH\\_t](#)
- [Tableau< BaseH > \\* d\\_aiH\\_tdt](#)
- [Tableau< TenseurBB \\* > d\\_aijBB\\_t](#)
- [Tableau< TenseurBB \\* > d\\_aijBB\\_tdt](#)
- [Tableau< TenseurHH \\* > d\\_aijHH\\_t](#)
- [Tableau< TenseurHH \\* > d\\_aijHH\\_tdt](#)
- [Vecteur d\\_ajacobien\\_t](#)
- [Vecteur d\\_ajacobien\\_tdt](#)

## 6.502.1 Documentation des fonctions membres

### 6.502.1.1 Calcul\_giB\_0()

```
void Met_PiPoCo::Calcul_giB_0 (
    const Tableau< Noeud * > & ,
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.502.1.2 Calcul\_giB\_t()

```
void Met_PiPoCo::Calcul_giB_t (
    const Tableau< Noeud * > & ,
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.502.1.3 Calcul\_giB\_tdt()

```
void Met_PiPoCo::Calcul_giB_tdt (
    const Tableau< Noeud * > & ,
    const Mat_pleine & ,
```

```

    int ,
    const Vecteur & phi ) [protected], [virtual]

```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.502.1.4 Calcul\_M0()

```

void Met_PiPoCo::Calcul_M0 (
    const Tableau< Noeud * > & tab_noeud,
    const Vecteur & phi,
    int nombre_noeud ) [protected], [virtual]

```

ceci dans la plupart des routines gourmandes!!!!!!!!!!!!!!!!!!!!!!

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.502.1.5 Calcul\_Mt()

```

void Met_PiPoCo::Calcul_Mt (
    const Tableau< Noeud * > & ,
    const Vecteur & phi,
    int ) [protected], [virtual]

```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.502.1.6 Calcul\_Mtdt()

```

void Met_PiPoCo::Calcul_Mtdt (
    const Tableau< Noeud * > & ,
    const Vecteur & phi,
    int ) [protected], [virtual]

```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.502.1.7 D\_giB\_t()

```

void Met_PiPoCo::D_giB_t (
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]

```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.502.1.8 D\_giB\_tdt()

```

void Met_PiPoCo::D_giB_tdt (
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]

```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.502.1.9 operator=()

```

Met_abstraite & Met_PiPoCo::operator= (
    const Met_abstraite & met ) [virtual]

```

surcharge de l'opérateur d'affectation

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.502.1.10 PlusInitVariables()

```
void Met_PiPoCo::PlusInitVariables (
    Tableau< Enum_variable_métrique > & tab ) [virtual]
```

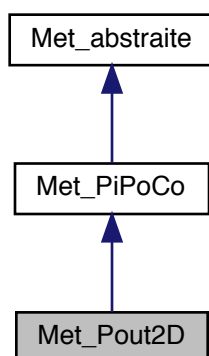
Réimplémentée à partir de [Met\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

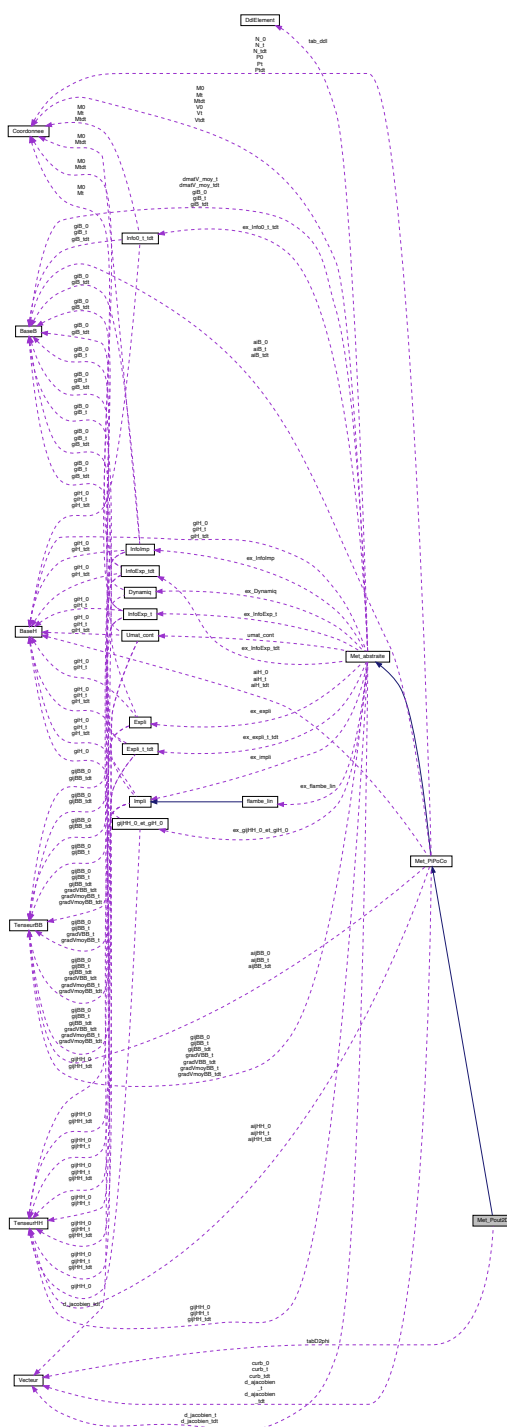
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_PiPoCo.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_PiPoCo1.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_PiPoCo2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_PiPoCo3.cc

## 6.503 Référence de la classe Met\_Pout2D

Graphe d'héritage de Met\_Pout2D:



Graphe de collaboration de Met\_Pout2D:



## Fonctions membres publiques

- **Met\_Pout2D** (int dim\_base, int nbvec, const [DdlElement](#) &tabddl, const [Tableau](#)< [Enum\\_variable\\_metrrique](#) > &tabb, int nomb\_noeud)
- **Met\_Pout2D** (const [Met\\_Pout2D](#) &)
- **Met\_abstraite** & **operator=** (const [Met\\_abstraite](#) &met)  
*surcharge de l'opérateur d'affectation*
- **Met\_PiPoCo** & **operator=** (const [Met\\_PiPoCo](#) &met)

- virtual [Met\\_Pout2D](#) & **operator=** (const [Met\\_Pout2D](#) &met)
- void **DeriveeSeconde** ([Vecteur](#) const &tabD2phi)

## Fonctions membres protégées

- void [Calcul\\_N\\_0](#) ()
- void [Calcul\\_N\\_t](#) ()
- void [Calcul\\_N\\_tdt](#) ()
- void [Calcul\\_giB\\_0](#) (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- void [Calcul\\_giB\\_t](#) (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- void [Calcul\\_giB\\_tdt](#) (const [Tableau](#)< [Noeud](#) \* > &, const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- void [D\\_giB\\_t](#) (const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- void [D\\_giB\\_tdt](#) (const [Mat\\_pleine](#) &, int, const [Vecteur](#) &phi)
- [Vecteur](#) & [courbure\\_0](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- [Vecteur](#) & [courbure\\_t](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- [Vecteur](#) & [courbure\\_tdt](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud)
- double **courbure** (const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [CoordonneeB](#) &aiB1, const [CoordonneeH](#) &)
- void [Dcourbure\\_t](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, [Vecteur](#) &curb, [TabOper](#)< [Vecteur](#) > &dcurb)
- void [Dcourbure\\_tdt](#) (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, [Vecteur](#) &curb, [TabOper](#)< [Vecteur](#) > &dcurb)
- void **Dcourbure** (const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [CoordonneeB](#) &aiB1, const [CoordonneeH](#) &, [Tableau](#)< [BaseB](#) > &DaiB, [Vecteur](#) &curb, [TabOper](#)< [Vecteur](#) > &dcurb)

## Attributs protégés

- [Vecteur](#) const \* **tabD2phi**

## 6.503.1 Documentation des fonctions membres

### 6.503.1.1 Calcul\_giB\_0()

```
void Met_Pout2D::Calcul_giB_0 (
    const Tableau< Noeud * > & ,
    const Mat\_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_PiPoCo](#).

### 6.503.1.2 Calcul\_giB\_t()

```
void Met_Pout2D::Calcul_giB_t (
    const Tableau< Noeud * > & ,
    const Mat\_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_PiPoCo](#).

### 6.503.1.3 Calcul\_giB\_tdt()

```
void Met_Pout2D::Calcul_giB_tdt (
    const Tableau< Noeud * > & ,
    const Mat\_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_PiPoCo](#).



#### 6.503.1.4 Calcul\_N\_0()

```
void Met_Pout2D::Calcul_N_0 ( ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_PiPoCo](#).

#### 6.503.1.5 Calcul\_N\_t()

```
void Met_Pout2D::Calcul_N_t ( ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_PiPoCo](#).

#### 6.503.1.6 Calcul\_N\_tdt()

```
void Met_Pout2D::Calcul_N_tdt ( ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_PiPoCo](#).

#### 6.503.1.7 courbure\_0()

```
Vecteur & Met_Pout2D::courbure_0 (
    const Tableau< Noeud * > & tab_noeud ) [protected], [virtual]
```

Implémente [Met\\_PiPoCo](#).

#### 6.503.1.8 courbure\_t()

```
Vecteur & Met_Pout2D::courbure_t (
    const Tableau< Noeud * > & tab_noeud ) [protected], [virtual]
```

Implémente [Met\\_PiPoCo](#).

#### 6.503.1.9 courbure\_tdt()

```
Vecteur & Met_Pout2D::courbure_tdt (
    const Tableau< Noeud * > & tab_noeud ) [protected], [virtual]
```

Implémente [Met\\_PiPoCo](#).

#### 6.503.1.10 D\_giB\_t()

```
void Met_Pout2D::D_giB_t (
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_PiPoCo](#).

#### 6.503.1.11 D\_giB\_tdt()

```
void Met_Pout2D::D_giB_tdt (
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_PiPoCo](#).

### 6.503.1.12 Dcourbure\_t()

```
void Met_Pout2D::Dcourbure_t (
    const Tableau< Noeud * > & tab_noeud,
    Vecteur & curb,
    TabOper< Vecteur > & dcurb ) [protected], [virtual]
```

Implémente [Met\\_PiPoCo](#).

### 6.503.1.13 Dcourbure\_tdt()

```
void Met_Pout2D::Dcourbure_tdt (
    const Tableau< Noeud * > & tab_noeud,
    Vecteur & curb,
    TabOper< Vecteur > & dcurb ) [protected], [virtual]
```

Implémente [Met\\_PiPoCo](#).

### 6.503.1.14 operator=() [1/2]

```
Met\_abstraite & Met_Pout2D::operator= (
    const Met\_abstraite & met ) [virtual]
```

surcharge de l'opérateur d'affectation

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.503.1.15 operator=() [2/2]

```
Met\_PiPoCo & Met_Pout2D::operator= (
    const Met\_PiPoCo & met ) [virtual]
```

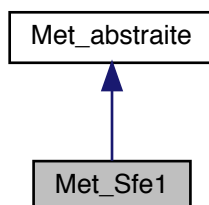
Réimplémentée à partir de [Met\\_PiPoCo](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met\_pout2D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Met\_pout2D.cc

## 6.504 Référence de la classe Met\_Sfe1

Grappe d'héritage de Met\_Sfe1:





- virtual void **Affiche** () const
- void **ChangeTypeCalculJacobien** (int nouveau\_type)
- const Met\_abstraite::Expli & **CalSfe1\_explicit\_t** (const **Tableau**< **Noeud** \* > &tab\_noeud, bool gradV\_↵ instantane, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, bool premier\_calcul, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabTypeCL, **Tableau**< **Coordonnee3** > const &vplan)
- const Met\_abstraite::Expli & **CalSfe1\_explicit\_simple\_t** (const **Tableau**< **Noeud** \* > &tab\_noeud, bool gradV\_instantane, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, bool premier\_↵\_calcul, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais)
- const Met\_abstraite::Expli\_t\_tdt & **CalSfe1\_explicit\_tdt** (const **Tableau**< **Noeud** \* > &tab\_noeud, bool gradV\_instantane, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, bool premier\_↵\_calcul, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabTypeCL, **Tableau**< **Coordonnee3** > const &vplan)
- const Met\_abstraite::Expli\_t\_tdt & **CalSfe1\_explicit\_simple\_tdt** (const **Tableau**< **Noeud** \* > &tab\_↵\_noeud, bool gradV\_instantane, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, bool premier\_calcul, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais)
- virtual const **Expli\_t\_tdt** & **Conteneur\_Expli\_tdtFac** () const
- const Met\_abstraite::Impli & **CalSfe1\_implicit** (const **Tableau**< **Noeud** \* > &tab\_noeud, bool gradV\_↵ instantane, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, bool premier\_calcul, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabTypeCL, **Tableau**< **Coordonnee3** > const &vplan)
- const Met\_abstraite::Impli & **CalSfe1\_implicit\_simple** (const **Tableau**< **Noeud** \* > &tab\_noeud, bool grad\_↵\_V\_instantane, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, bool premier\_calcul, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais)
- virtual const **Dynamiq** & **Cal\_pourMatMass** (const **Tableau**< **Noeud** \* > &tab\_noeud, const **Mat\_pleine** &tabDphi, int nombre\_noeud, const **Vecteur** &phi)
- int **Test\_courbure\_anormale** (**Enum\_dure** temps, double inf\_normale, const **Tableau**< **Noeud** \* > &tab\_↵\_noeud, const **Mat\_pleine** &dphiS, int nombre\_noeud, const **Vecteur** &phiS)
- const Met\_abstraite::InfoImp & **CalSfe1\_InfoImp** (const **Tableau**< **Noeud** \* > &tab\_noeud, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabTypeCL, **Tableau**< **Coordonnee3** > const &vplan)
- const Met\_abstraite::InfoExp\_t & **CalSfe1\_InfoExp\_t** (const **Tableau**< **Noeud** \* > &tab\_noeud, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabTypeCL, **Tableau**< **Coordonnee3** > const &vplan)
- const Met\_abstraite::InfoExp\_tdt & **CalSfe1\_InfoExp\_tdt** (const **Tableau**< **Noeud** \* > &tab\_noeud, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabTypeCL, **Tableau**< **Coordonnee3** > const &vplan)
- virtual const **Info0\_t\_tdt** & **CalSfe1\_Info0\_t\_tdt** (const **Tableau**< **Noeud** \* > &tab\_noeud, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabTypeCL, **Tableau**< **Coordonnee3** > const &vplan)
- virtual const **Pour\_def\_Almansi CalSfe1\_pour\_def\_Almansi\_au\_temps** (**Enum\_dure** temps, const **Tableau**< **Noeud** \* > &tab\_noeud, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabTypeCL, **Tableau**< **Coordonnee3** > const &vplan)
- virtual const **Pour\_def\_log CalSfe1\_pour\_def\_log\_au\_temps** (**Enum\_dure** temps, const **Tableau**< **Noeud** \* > &tab\_noeud, **Mat\_pleine** const &tabdphiS, int nombre\_noeud, **Vecteur** const &tabphiS, **Mat\_pleine** const &tabdphiH, **Vecteur** const &tabphiH, const **Epai** \*epais, **Tableau**< **EnumTypeCL** > const &tabType\_↵\_CL, **Tableau**< **Coordonnee3** > const &vplan)
- virtual const **Coordonnee** & **PointSfe1M\_0** (const **Tableau**< **Noeud** \* > &tab\_noeud, const **Vecteur** &phiS, **Mat\_pleine** const &dphiS, const **Epai** \*epais, const **Vecteur** &phiH)
- virtual const **Coordonnee** & **PointSfe1M\_t** (const **Tableau**< **Noeud** \* > &tab\_noeud, const **Vecteur** &phiS, **Mat\_pleine** const &dphiS, const **Epai** \*epais, const **Vecteur** &phiH)
- virtual const **Coordonnee** & **PointSfe1M\_tdt** (const **Tableau**< **Noeud** \* > &tab\_noeud, const **Vecteur** &phiS, **Mat\_pleine** const &dphiS, const **Epai** \*epais, const **Vecteur** &phiH)
- void **PlusInitVariables** (**Tableau**< **Enum\_variable\_metroique** > &tab)
- const Met\_Sfe1::Courbure\_t\_tdt & **RecupCourbure** () const

## Fonctions membres protégées

- void **Calcul\_N\_0** ()

```

— void Calcul_N_t ()
— void Calcul_N_tdt ()
— void Cal_d_N_t ()
— void Cal_d_N_tdt ()
— void Cal_N_alpha_0 ()
— void Cal_N_alpha_t ()
— void Cal_N_alpha_tdt ()
— void Cal_d_N_alpha_t ()
— void Cal_d_N_alpha_tdt ()
— void Calcul_MO (const Tableau< Noeud * > &, const Vecteur &phi, int)
    ceci dans la plupart des routines gourmandes !!!!!!!!!!!!!!!!!!!!!
— void Calcul_Mt (const Tableau< Noeud * > &, const Vecteur &phi, int)
— void Calcul_Mtdt (const Tableau< Noeud * > &, const Vecteur &phi, int)
— void Calcul_a_alpha_B_0 (const Tableau< Noeud * > &tab_noeud, const Mat_pleine &tabDphi, int
nombre_noeud, const Vecteur &phi)
— void Calcul_a_alpha_B_t (const Tableau< Noeud * > &tab_noeud, const Mat_pleine &tabDphi, int
nombre_noeud, const Vecteur &phi)
— void Calcul_a_alpha_B_tdt (const Tableau< Noeud * > &tab_noeud, const Mat_pleine &tabDphi, int
nombre_noeud, const Vecteur &phi)
— void Calcul_a_3_B_0 ()
— void Calcul_a_3_B_t ()
— void Calcul_a_3_B_tdt ()
— void D_a_alpha_B_t (const Mat_pleine &tabDphi, int nbnoeu, const Vecteur &phi)
— void D_a_alpha_B_tdt (const Mat_pleine &tabDphi, int nbnoeu, const Vecteur &phi)
— void D_a_3_B_t ()
— void D_a_3_B_tdt ()
— void Calcul_giB_0 (const Tableau< Noeud * > &, const Mat_pleine &, int, const Vecteur &phi)
— void Calcul_giB_t (const Tableau< Noeud * > &, const Mat_pleine &, int, const Vecteur &phi)
— void Calcul_giB_tdt (const Tableau< Noeud * > &, const Mat_pleine &, int, const Vecteur &phi)
— void D_giB_t (const Mat_pleine &, int, const Vecteur &phi)
— void D_giB_tdt (const Mat_pleine &, int, const Vecteur &phi)
— void Calcul_epaisseur_0 (const Tableau< Noeud * > &tab_noeud, const Vecteur &phiS)
— void Calcul_epaisseur_t (const Tableau< Noeud * > &tab_noeud, const Vecteur &phiS)
— void Calcul_epaisseur_tdt (const Tableau< Noeud * > &tab_noeud, const Vecteur &phiS)
— void D_epaisseur_t (const Vecteur &phiS)
— void D_epaisseur_tdt (const Vecteur &phiS)
— Tenseur_ns2BB & courbure_0 (const Tableau< Noeud * > &tab_noeud, Tableau< EnumTypeCL > const
&tabTypeCL, Tableau< Coordonnee3 > const &vplan)
— Tenseur_ns2BB & courbure_t (const Tableau< Noeud * > &tab_noeud, Tableau< EnumTypeCL > const
&tabTypeCL, Tableau< Coordonnee3 > const &vplan)
— Tenseur_ns2BB & courbure_tdt (const Tableau< Noeud * > &tab_noeud, Tableau< EnumTypeCL > const
&tabTypeCL, Tableau< Coordonnee3 > const &vplan)
— void Dcourbure_t (const Tableau< Noeud * > &tab_noeud, Tenseur_ns2BB &curb, TabOper<
Tenseur_ns2BB > &dcurb, Tableau< EnumTypeCL > const &tabTypeCL, Tableau< Coordonnee3 > const
&vplan)
— void Dcourbure_tdt (const Tableau< Noeud * > &tab_noeud, Tenseur_ns2BB &curb, TabOper<
Tenseur_ns2BB > &dcurb, Tableau< EnumTypeCL > const &tabTypeCL, Tableau< Coordonnee3 > const
&vplan)
— void AllocSfe1 ()
— void DeallocSfe1 ()
— void DeallocSfe1 (Tableau< Enum_variable_metrrique > &tib)
— void Copie_metSfe1 (const Met_Sfe1 &a)
— Tenseur_ns2BB Courbure1 (const Tableau< Coordonnee3 > &tab_coor, const BaseB &aiB, const BaseH
&aiH, Tableau< EnumTypeCL > const &tabTypeCL, Tableau< Coordonnee3 > const &vplan)
— void Dcourbure1 (const Tableau< Coordonnee3 > &tab_coor, const BaseB &aiB, const Tableau<
BaseB > &DaiB, const BaseH &aiH, const Tableau< BaseH > &DaiH, Tenseur_ns2BB &curb, TabOper<
Tenseur_ns2BB > &dcurb, Tableau< EnumTypeCL > const &tabTypeCL, Tableau< Coordonnee3 > const
&vplan)
— int Test_courbure_anormale1 (double inf_normale, const Tableau< Coordonnee3 > &tab_coor, const
BaseB &aiB, const BaseH &aiH)
— Tenseur_ns2BB Courbure2 (const Tableau< Coordonnee3 > &tab_coor, const BaseB &aiB, const BaseH
&aiH, Tableau< EnumTypeCL > const &tabTypeCL, Tableau< Coordonnee3 > const &vplan)

```

- void **Dcourbure2** (const [Tableau](#)< [Coordonnee3](#) > &tab\_coor, const [BaseB](#) &aiB, const [Tableau](#)< [BaseB](#) > &DaiB, const [BaseH](#) &aiH, const [Tableau](#)< [BaseH](#) > &DaiH, [Tenseur\\_ns2BB](#) &curb, [TabOper](#)< [Tenseur\\_ns2BB](#) > &dcurb, [Tableau](#)< [EnuTypeCL](#) > const &tabTypeCL, [Tableau](#)< [Coordonnee3](#) > const &vplan)
- int **Test\_courbure\_anormale2** (double inf\_normale, const [Tableau](#)< [Coordonnee3](#) > &tab\_coor, const [BaseB](#) &aiB, const [BaseH](#) &aiH)
- [Tenseur\\_ns2BB](#) **Courbure3** (const [Tableau](#)< [Coordonnee3](#) > &tab\_coor, const [BaseB](#) &aiB, const [BaseH](#) &aiH, [Tableau](#)< [EnuTypeCL](#) > const &tabTypeCL, [Tableau](#)< [Coordonnee3](#) > const &vplan)
- void **Dcourbure3** (const [Tableau](#)< [Coordonnee3](#) > &tab\_coor, const [BaseB](#) &aiB, const [Tableau](#)< [BaseB](#) > &DaiB, const [BaseH](#) &aiH, const [Tableau](#)< [BaseH](#) > &DaiH, [Tenseur\\_ns2BB](#) &curb, [TabOper](#)< [Tenseur\\_ns2BB](#) > &dcurb, [Tableau](#)< [EnuTypeCL](#) > const &tabTypeCL, [Tableau](#)< [Coordonnee3](#) > const &vplan)
- int **Test\_courbure\_anormale3** (double inf\_normale, const [Tableau](#)< [Coordonnee3](#) > &tab\_coor, const [BaseB](#) &aiB, const [BaseH](#) &aiH)
- [Tenseur\\_ns2BB](#) **Courbure4** (const [Tableau](#)< [Coordonnee3](#) > &tab\_coor, const [BaseB](#) &aiB, const [BaseH](#) &aiH, [Tableau](#)< [EnuTypeCL](#) > const &tabTypeCL, [Tableau](#)< [Coordonnee3](#) > const &vplan)
- void **Dcourbure4** (const [Tableau](#)< [Coordonnee3](#) > &tab\_coor, const [BaseB](#) &aiB, const [Tableau](#)< [BaseB](#) > &DaiB, const [BaseH](#) &aiH, const [Tableau](#)< [BaseH](#) > &DaiH, [Tenseur\\_ns2BB](#) &curb, [TabOper](#)< [Tenseur\\_ns2BB](#) > &dcurb, [Tableau](#)< [EnuTypeCL](#) > const &tabTypeCL, [Tableau](#)< [Coordonnee3](#) > const &vplan)
- int **Test\_courbure\_anormale4** (double inf\_normale, const [Tableau](#)< [Coordonnee3](#) > &tab\_coor, const [BaseB](#) &aiB, const [BaseH](#) &aiH)
- void **Verif\_Dcourbure** (const [Tableau](#)< [Noeud](#) \* > &tab\_noeud, bool gradV\_instantane, [Mat\\_pleine](#) const &tabdphiS, int nombre\_noeud, [Vecteur](#) const &tabphiS, [Mat\\_pleine](#) const &tabdphiH, [Vecteur](#) const &tabphiH, const [Epai](#) \*epais, [Tableau](#)< [EnuTypeCL](#) > const &tabTypeCL, [Tableau](#)< [Coordonnee3](#) > const &vplan)
- void **Init\_conteneur\_de\_retour\_pourFacette** ()

### Attributs protégés

- [Tableau](#)< [Coordonnee3](#) > **la**
- [Tenseur\\_ns2BB](#) **curb\_0**
- [Tenseur\\_ns2BB](#) **curb\_t**
- [Tenseur\\_ns2BB](#) **curb\_tdt**
- [TabOper](#)< [Tenseur\\_ns2BB](#) > **dcurb\_t**
- [TabOper](#)< [Tenseur\\_ns2BB](#) > **dcurb\_tdt**
- [Coordonnee3B](#) **N\_0**
- [Coordonnee3B](#) **N\_t**
- [Coordonnee3B](#) **N\_tdt**
- double **norme\_N\_t**
- double **norme\_N\_tdt**
- [Tableau](#)< [CoordonneeB](#) > \* **d\_N\_t**
- [Tableau](#)< [CoordonneeB](#) > \* **d\_N\_tdt**
- [CoordonneeB](#) **N\_alpha\_0\_1**
- [CoordonneeB](#) **N\_alpha\_0\_2**
- [CoordonneeB](#) **N\_alpha\_t\_1**
- [CoordonneeB](#) **N\_alpha\_t\_2**
- [CoordonneeB](#) **N\_alpha\_tdt\_1**
- [CoordonneeB](#) **N\_alpha\_tdt\_2**
- [Tableau](#)< [CoordonneeB](#) > \* **d\_N\_alpha\_t\_1**
- [Tableau](#)< [CoordonneeB](#) > \* **d\_N\_alpha\_t\_2**
- [Tableau](#)< [CoordonneeB](#) > \* **d\_N\_alpha\_tdt\_1**
- [Tableau](#)< [CoordonneeB](#) > \* **d\_N\_alpha\_tdt\_2**
- int **nbNoeudCentral**
- bool **tCalEpais**
- [Epai](#) **epais**
- [Tableau](#)< [Epai](#) > \* **d\_epais**
- int **type\_calcul\_jacobien**
- [Coordonnee3](#) \* **P0**
- [Coordonnee3](#) \* **Pt**
- [Coordonnee3](#) \* **Ptdt**
- [Tableau](#)< [Coordonnee3](#) > \* **d\_Pt**
- [Tableau](#)< [Coordonnee3](#) > \* **d\_Ptdt**
- [BaseB](#) \* **aiB\_0**

```

— BaseB * aiB_t
— BaseB * aiB_tdt
— BaseH * aiH_0
— BaseH * aiH_t
— BaseH * aiH_tdt
— TenseurBB * aijBB_0
— TenseurBB * aijBB_t
— TenseurBB * aijBB_tdt
— TenseurHH * aijHH_0
— TenseurHH * aijHH_t
— TenseurHH * aijHH_tdt
— double ajacobien_0
— double ajacobien_t
— double ajacobien_tdt
— Tableau< BaseB > * d_aiB_t
— Tableau< BaseB > * d_aiB_tdt
— Tableau< BaseH > * d_aiH_t
— Tableau< BaseH > * d_aiH_tdt
— Tableau< TenseurBB * > d_ajBB_t
— Tableau< TenseurBB * > d_ajBB_tdt
— Tableau< TenseurHH * > d_ajHH_t
— Tableau< TenseurHH * > d_ajHH_tdt
— Vecteur d_ajacobien_t
— Vecteur d_ajacobien_tdt
— Courbure_t_tdt courbure_t_tdt
— Impli ex_impliFac
— Expli ex_expliFac
— Expli_t_tdt ex_expli_t_tdtFac
— gijHH_0_et_giH_0 ex_ajHH_0_et_aiH_0Fac
— Dynamiq ex_DynamiqFac
— flambe_lin ex_flambe_linFac
— Umat_cont umat_contFac
— InfoImp ex_InfoImpFac
— InfoExp_t ex_InfoExp_tFac
— InfoExp_tdt ex_InfoExp_tdtFac
— Info0_t_tdt ex_Info0_t_tdtFac
— Tenseur_ns2BB(Met_Sfe1::* PtCourbure )(const Tableau< Coordonnee3 > &tab_coor, const BaseB &aiB,
const BaseH &aiH, Tableau< EnuTypeCL > const &tabTypeCL, Tableau< Coordonnee3 > const &vplan)
— void(Met_Sfe1::* PtDcourbure )(const Tableau< Coordonnee3 > &tab_coor, const BaseB &aiB, const
Tableau< BaseB > &DaiB, const BaseH &aiH, const Tableau< BaseH > &DaiH, Tenseur_ns2BB &curb,
TabOper< Tenseur_ns2BB > &dcurb, Tableau< EnuTypeCL > const &tabTypeCL, Tableau< Coordonnee3
> const &vplan)
— int(Met_Sfe1::* PtTest_courbure_anormale )(double inf_normale, const Tableau< Coordonnee3 > &tab_coor, const BaseB &aiB, const BaseH &aiH)

```

## Attributs protégés statiques

```

— static Tableau< int > indi
— static int indic_Verif_DcourbureSFE = 0

```

## 6.504.1 Documentation des fonctions membres

### 6.504.1.1 Affiche()

```
void Met_Sfe1::Affiche ( ) const [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.2 Cal\_pourMatMass()

```
const Met_abstraite::Dynamiq & Met_Sfel::Cal_pourMatMass (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud,
    const Vecteur & phi ) [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.3 Calcul\_giB\_0()

```
void Met_Sfel::Calcul_giB_0 (
    const Tableau< Noeud * > & ,
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.4 Calcul\_giB\_t()

```
void Met_Sfel::Calcul_giB_t (
    const Tableau< Noeud * > & ,
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.5 Calcul\_giB\_tdt()

```
void Met_Sfel::Calcul_giB_tdt (
    const Tableau< Noeud * > & ,
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.6 Calcul\_M0()

```
void Met_Sfel::Calcul_M0 (
    const Tableau< Noeud * > & tab_noeud,
    const Vecteur & phi,
    int nombre_noeud ) [protected], [virtual]
```

ceci dans la plupart des routines gourmandes!!!!!!!!!!!!!!!!!!!!!!

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.7 Calcul\_Mt()

```
void Met_Sfel::Calcul_Mt (
    const Tableau< Noeud * > & ,
    const Vecteur & phi,
    int ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).



### 6.504.1.8 Calcul\_Mtdt()

```
void Met_Sfel::Calcul_Mtdt (
    const Tableau< Noeud * > & ,
    const Vecteur & phi,
    int ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.9 D\_giB\_t()

```
void Met_Sfel::D_giB_t (
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.10 D\_giB\_tdt()

```
void Met_Sfel::D_giB_tdt (
    const Mat_pleine & ,
    int ,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.11 operator=()

```
Met\_abstraite & Met_Sfel::operator= (
    const Met\_abstraite & met ) [virtual]
```

surcharge de l'opérateur d'affectation

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.504.1.12 PlusInitVariables()

```
void Met_Sfel::PlusInitVariables (
    Tableau< Enum_variable_metrique > & tab ) [virtual]
```

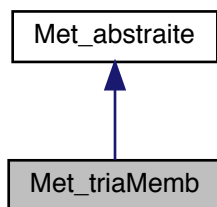
Réimplémentée à partir de [Met\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

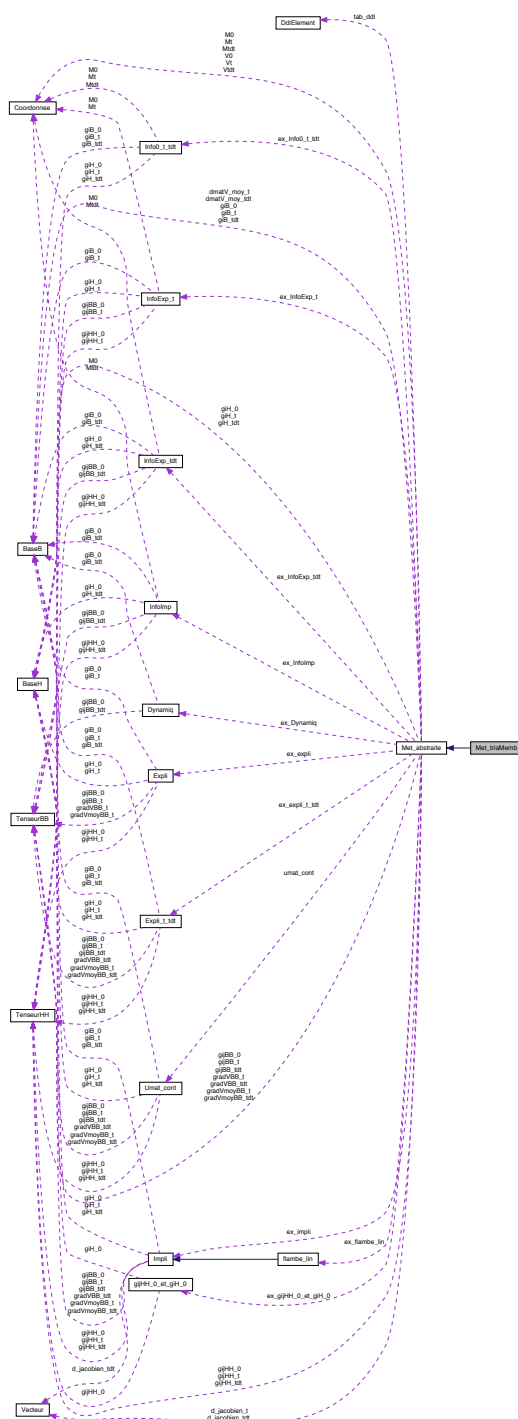
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met\_Sfe1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met\_Sfe1s1.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met\_Sfe1s2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met\_Sfe1s3.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met\_Sfe1s4.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Met\_Sfe2s2.cc

## 6.505 Référence de la classe Met\_triaMemb

Graphe d'héritage de Met\_triaMemb:



Graphe de collaboration de Met\_triaMemb:



## Fonctions membres publiques

- **Met\_triaMemb** (int dim\_base, DdlElement &tabddl, Tableau< Enum\_variable\_métrique > &tab)
- **Met\_triaMemb** (Met\_triaMemb &)

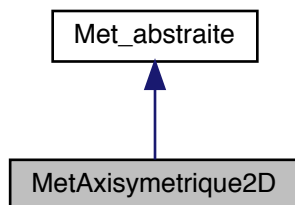
## Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

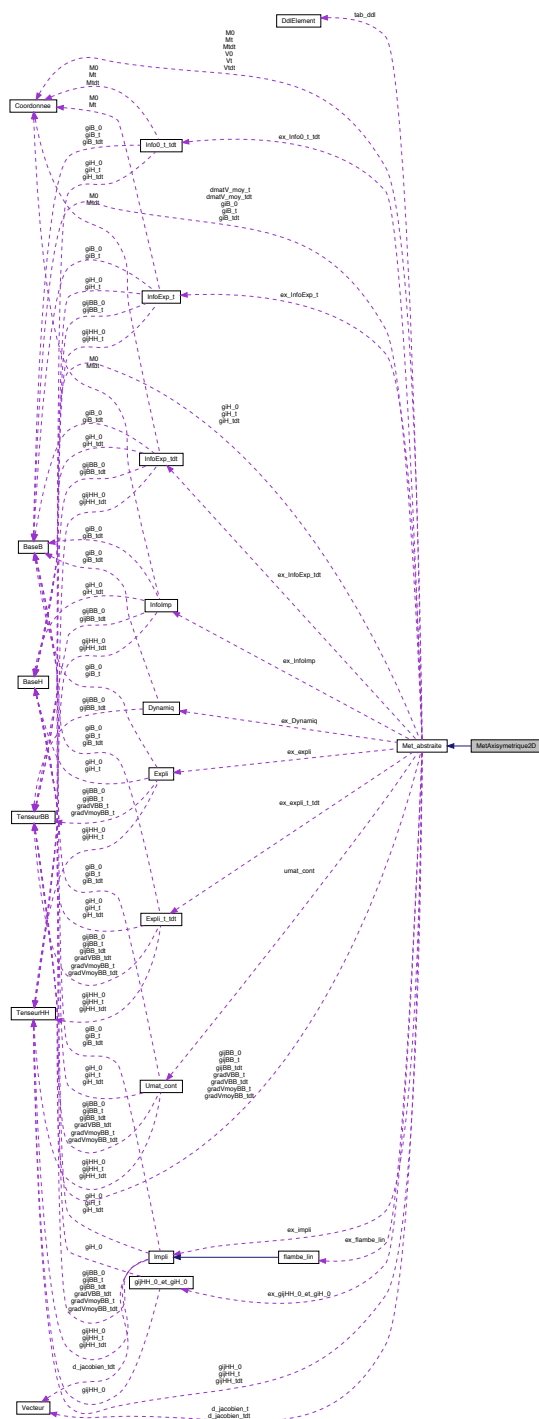
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/Met\_triaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/Met\_triaMemb.cc

## 6.506 Référence de la classe MetAxisymetrique2D

Graphes d'héritage de MetAxisymetrique2D:



Graphe de collaboration de MetAxisymetrique2D:



### Fonctions membres publiques

- **MetAxisymetrique2D** (int dim\_base, const DdlElement &tabddl, const Tableau< Enum\_variable\_metrrique > &tab, int nomb\_noeud)
- **MetAxisymetrique2D** (const MetAxisymetrique2D &)
- **Met\_abstraite & operator=** (const Met\_abstraite &met)  
*surcharge de l'opérateur d'affectation*

## Fonctions membres protégées

- virtual void `Calcul_d_Mtdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur` &phi, int nombre\_noeud)
- virtual void `Calcul_d_Vt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur` &phi, int nombre\_noeud)
- virtual void `Calcul_d_Vt` (const `Noeud *noeud`)
- virtual void `Calcul_d_Vtdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur` &phi, int nombre\_noeud)
- virtual void `Calcul_d_Vtdt` (const `Noeud *noeud`)
- virtual void `Calcul_d_V_moyt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur` &phi, int nombre\_↵noeud)
- virtual void `Calcul_d_V_moyt` (const `Noeud *noeud`)
- virtual void `Calcul_d_V_moytdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Vecteur` &phi, int nombre\_↵noeud)
- virtual void `Calcul_d_V_moytdt` (const `Noeud *noeud`)
- void `Calcul_giB_0` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &dphi, int nombre\_noeud, const `Vecteur` &phi)
- void `Calcul_giB_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &dphi, int nombre\_noeud, const `Vecteur` &phi)
- void `Calcul_giB_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &dphi, int nombre\_noeud, const `Vecteur` &phi)
- void `D_giB_t` (const `Mat_pleine` &tabDphi, int nbnoeu, const `Vecteur` &phi)
- void `D_giB_tdt` (const `Mat_pleine` &tabDphi, int nbnoeu, const `Vecteur` &phi)
- void `Calcul_gradVBB_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre\_↵noeud)
- void `Calcul_gradVBB_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre\_↵noeud)
- void `Calcul_gradVBB_moyen_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre\_noeud)
- void `Calcul_gradVBB_moyen_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre\_noeud)
- void `DgradVBB_t` (const `Mat_pleine` &dphi)
- void `DgradVBB_tdt` (const `Mat_pleine` &dphi)
- void `DgradVmoyBB_t` (const `Mat_pleine` &dphi, const `Tableau< Noeud * >` &tab\_noeud)
- void `DgradVmoyBB_tdt` (const `Mat_pleine` &dphi)

## Attributs protégés

- double `rho_0`
- double `rho_t`
- double `rho_tdt`

### 6.506.1 Documentation des fonctions membres

#### 6.506.1.1 `Calcul_d_Mtdt()`

```
void MetAxisymetrique2D::Calcul_d_Mtdt (
    const Tableau< Noeud * > & tab_noeud,
    const Vecteur & phi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.506.1.2 `Calcul_d_V_moyt()` [1/2]

```
void MetAxisymetrique2D::Calcul_d_V_moyt (
    const Noeud * noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.3 Calcul\_d\_V\_moyt()** [2/2]

```
void MetAxisymetrique2D::Calcul_d_V_moyt (
    const Tableau< Noeud * > & tab_noeud,
    const Vecteur & phi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.4 Calcul\_d\_V\_moytdt()** [1/2]

```
void MetAxisymetrique2D::Calcul_d_V_moytdt (
    const Noeud * noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.5 Calcul\_d\_V\_moytdt()** [2/2]

```
void MetAxisymetrique2D::Calcul_d_V_moytdt (
    const Tableau< Noeud * > & tab_noeud,
    const Vecteur & phi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.6 Calcul\_d\_Vt()** [1/2]

```
void MetAxisymetrique2D::Calcul_d_Vt (
    const Noeud * noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.7 Calcul\_d\_Vt()** [2/2]

```
void MetAxisymetrique2D::Calcul_d_Vt (
    const Tableau< Noeud * > & tab_noeud,
    const Vecteur & phi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.8 Calcul\_d\_Vtdt()** [1/2]

```
void MetAxisymetrique2D::Calcul_d_Vtdt (
    const Noeud * noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.9 Calcul\_d\_Vtdt()** [2/2]

```
void MetAxisymetrique2D::Calcul_d_Vtdt (
    const Tableau< Noeud * > & tab_noeud,
    const Vecteur & phi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.506.1.10 Calcul\_giB\_0()

```
void MetAxisymetrique2D::Calcul_giB_0 (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.506.1.11 Calcul\_giB\_t()

```
void MetAxisymetrique2D::Calcul_giB_t (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.506.1.12 Calcul\_giB\_tdt()

```
void MetAxisymetrique2D::Calcul_giB_tdt (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.506.1.13 Calcul\_gradVBB\_moyen\_t()

```
void MetAxisymetrique2D::Calcul_gradVBB_moyen_t (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.506.1.14 Calcul\_gradVBB\_moyen\_tdt()

```
void MetAxisymetrique2D::Calcul_gradVBB_moyen_tdt (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.506.1.15 Calcul\_gradVBB\_t()

```
void MetAxisymetrique2D::Calcul_gradVBB_t (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).



**6.506.1.16 Calcul\_gradVBB\_tdt()**

```
void MetAxisymetrique2D::Calcul_gradVBB_tdt (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.17 D\_giB\_t()**

```
void MetAxisymetrique2D::D_giB_t (
    const Mat_pleine & tabDphi,
    int nbnoeu,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.18 D\_giB\_tdt()**

```
void MetAxisymetrique2D::D_giB_tdt (
    const Mat_pleine & tabDphi,
    int nbnoeu,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.19 DgradVBB\_t()**

```
void MetAxisymetrique2D::DgradVBB_t (
    const Mat_pleine & dphi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.20 DgradVBB\_tdt()**

```
void MetAxisymetrique2D::DgradVBB_tdt (
    const Mat_pleine & dphi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.21 DgradVmoyBB\_t()**

```
void MetAxisymetrique2D::DgradVmoyBB_t (
    const Mat_pleine & dphi,
    const Tableau< Noeud * > & tab_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.22 DgradVmoyBB\_tdt()**

```
void MetAxisymetrique2D::DgradVmoyBB_tdt (
    const Mat_pleine & dphi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

**6.506.1.23 operator=()**

```
Met_abstraite & MetAxisymetrique2D::operator= (
    const Met_abstraite & met ) [inline], [virtual]
```

surcharge de l'opérateur d'affectation

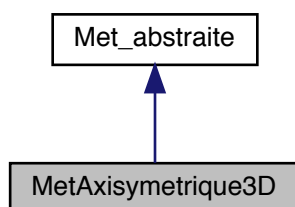
Réimplémentée à partir de [Met\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met↔Axisymetrique2D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met↔Axisymetrique2D.cc

## 6.507 Référence de la classe MetAxisymetrique3D

Graphe d'héritage de MetAxisymetrique3D:





## Fonctions membres protégées

- void `Calcul_giB_0` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &dphi, int nombre\_noeud, const `Vecteur` &phi)
- void `Calcul_giB_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &dphi, int nombre\_noeud, const `Vecteur` &phi)
- void `Calcul_giB_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &dphi, int nombre\_noeud, const `Vecteur` &phi)
- void `D_giB_t` (const `Mat_pleine` &tabDphi, int nbnoeu, const `Vecteur` &phi)
- void `D_giB_tdt` (const `Mat_pleine` &tabDphi, int nbnoeu, const `Vecteur` &phi)
- void `Calcul_gradVBB_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre↵\_noeud)
- void `Calcul_gradVBB_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre↵\_noeud)
- void `Calcul_gradVBB_moyen_t` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre\_noeud)
- void `Calcul_gradVBB_moyen_tdt` (const `Tableau< Noeud * >` &tab\_noeud, const `Mat_pleine` &tabDphi, int nombre\_noeud)
- void `DgradVBB_t` (const `Mat_pleine` &dphi)
- void `DgradVBB_tdt` (const `Mat_pleine` &dphi)
- void `DgradVmoyBB_t` (const `Mat_pleine` &dphi, const `Tableau< Noeud * >` &tab\_noeud)
- void `DgradVmoyBB_tdt` (const `Mat_pleine` &dphi)

## Membres hérités additionnels

### 6.507.1 Documentation des fonctions membres

#### 6.507.1.1 `Calcul_giB_0()`

```
void MetAxisymetrique3D::Calcul_giB_0 (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de `Met_abstraite`.

#### 6.507.1.2 `Calcul_giB_t()`

```
void MetAxisymetrique3D::Calcul_giB_t (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de `Met_abstraite`.

#### 6.507.1.3 `Calcul_giB_tdt()`

```
void MetAxisymetrique3D::Calcul_giB_tdt (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & dphi,
    int nombre_noeud,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de `Met_abstraite`.

#### 6.507.1.4 Calcul\_gradVBB\_moyen\_t()

```
void MetAxisymetrique3D::Calcul_gradVBB_moyen_t (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.507.1.5 Calcul\_gradVBB\_moyen\_tdt()

```
void MetAxisymetrique3D::Calcul_gradVBB_moyen_tdt (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.507.1.6 Calcul\_gradVBB\_t()

```
void MetAxisymetrique3D::Calcul_gradVBB_t (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.507.1.7 Calcul\_gradVBB\_tdt()

```
void MetAxisymetrique3D::Calcul_gradVBB_tdt (
    const Tableau< Noeud * > & tab_noeud,
    const Mat_pleine & tabDphi,
    int nombre_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.507.1.8 D\_giB\_t()

```
void MetAxisymetrique3D::D_giB_t (
    const Mat_pleine & tabDphi,
    int nbnoeu,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.507.1.9 D\_giB\_tdt()

```
void MetAxisymetrique3D::D_giB_tdt (
    const Mat_pleine & tabDphi,
    int nbnoeu,
    const Vecteur & phi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

#### 6.507.1.10 DgradVBB\_t()

```
void MetAxisymetrique3D::DgradVBB_t (
    const Mat_pleine & dphi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.507.1.11 DgradVBB\_tdt()

```
void MetAxisymetrique3D::DgradVBB_tdt (
    const Mat_pleine & dphi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.507.1.12 DgradVmoyBB\_t()

```
void MetAxisymetrique3D::DgradVmoyBB_t (
    const Mat_pleine & dphi,
    const Tableau< Noeud * > & tab_noeud ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.507.1.13 DgradVmoyBB\_tdt()

```
void MetAxisymetrique3D::DgradVmoyBB_tdt (
    const Mat_pleine & dphi ) [protected], [virtual]
```

Réimplémentée à partir de [Met\\_abstraite](#).

### 6.507.1.14 operator=()

```
Met_abstraite & MetAxisymetrique3D::operator= (
    const Met_abstraite & met ) [inline], [virtual]
```

surcharge de l'opérateur d'affectation

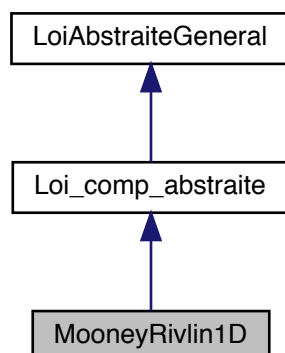
Réimplémentée à partir de [Met\\_abstraite](#).

La documentation de cette classe a été générée à partir du fichier suivant :

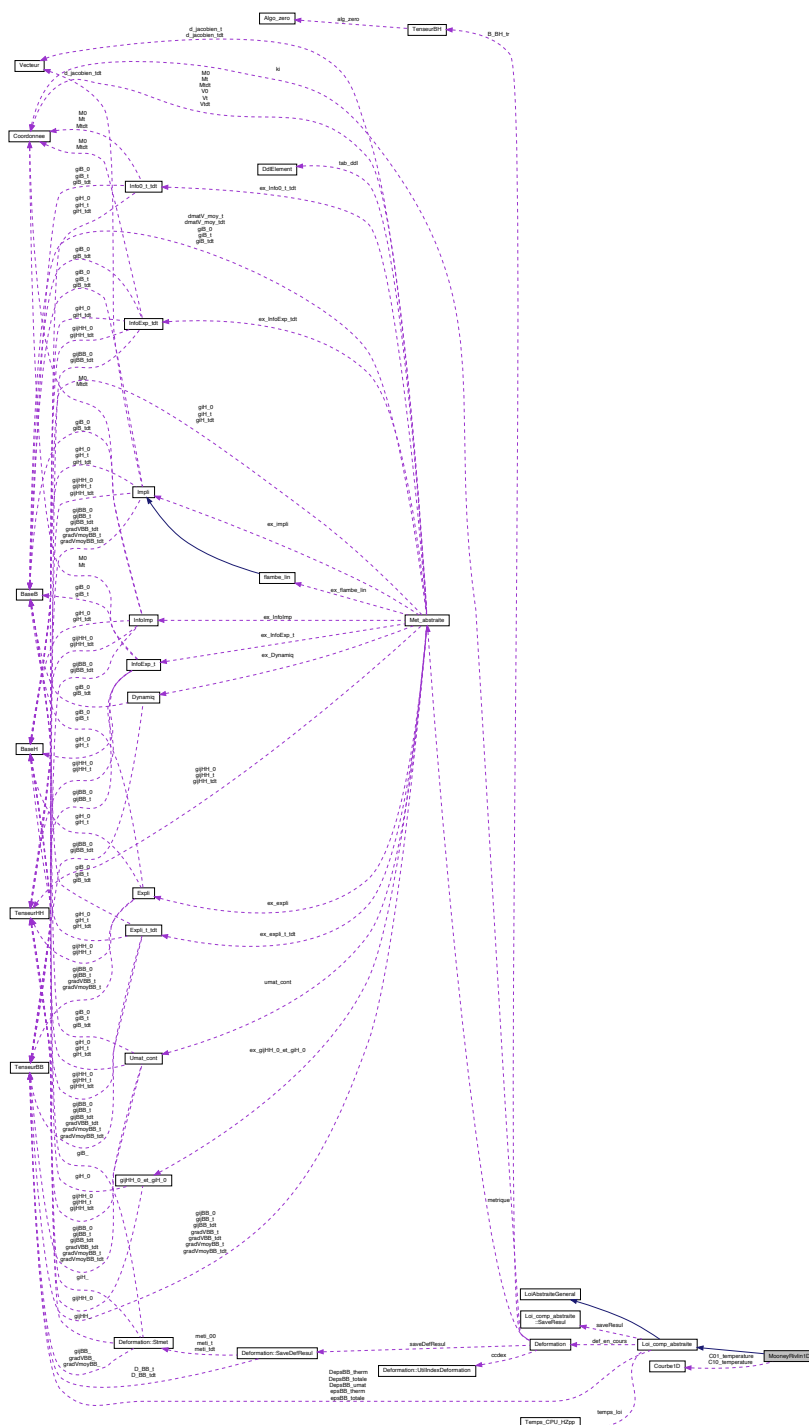
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met↔Axisymetrique3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met↔Axisymetrique3D.cc

## 6.508 Référence de la classe MooneyRivlin1D

Graphe d'héritage de MooneyRivlin1D:



Graphe de collaboration de MooneyRivlin1D:



## Fonctions membres publiques

- **MooneyRivlin1D** (const [MooneyRivlin1D](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)

- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure, const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gij↵  
BB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_,  
[TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double  
&jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_↵  
compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#)  
&giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#)  
&giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_eps↵  
BB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* >  
&d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#)  
&d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const  
[EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_↵  
abstraite::Impli](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const  
[ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const  
[Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const  
[TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- double [C10](#)
- double [C01](#)
- [Courbe1D](#) \* [C10\\_temperature](#)
- [Courbe1D](#) \* [C01\\_temperature](#)

## Membres hérités additionnels

### 6.508.1 Documentation des fonctions membres

#### 6.508.1.1 Affiche()

void [MooneyRivlin1D::Affiche](#) ( ) const [virtual]  
Implémente [LoiAbstraiteGeneral](#).

#### 6.508.1.2 Calcul\_DsigmaHH\_tdt()

```
void MooneyRivlin1D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
```



```

TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.508.1.3 Calcul\_SigmaHH()

```

void MooneyRivlin1D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.508.1.4 CalculGrandeurTravail()

```

virtual void MooneyRivlin1D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],

```

[virtual]

Implémente [Loi\\_comp\\_abstraite](#).

### 6.508.1.5 Ecriture\_base\_info\_loi()

```
void MooneyRivlin1D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.508.1.6 HsurH0()

```
virtual double MooneyRivlin1D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.508.1.7 Info\_commande\_LoisDeComp()

```
void MooneyRivlin1D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.508.1.8 Lecture\_base\_info\_loi()

```
void MooneyRivlin1D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.508.1.9 LectureDonneesParticulieres()

```
void MooneyRivlin1D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.508.1.10 Module\_young\_equivalent()

```
double MooneyRivlin1D::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.508.1.11 Nouvelle\_loi\_identique()

```
Loi\_comp\_abstraite * MooneyRivlin1D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.508.1.12 TestComplet()

```
int MooneyRivlin1D::TestComplet ( ) [virtual]
```

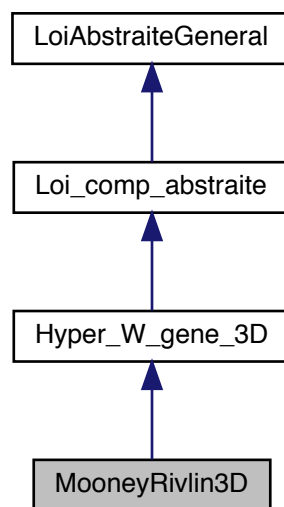
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

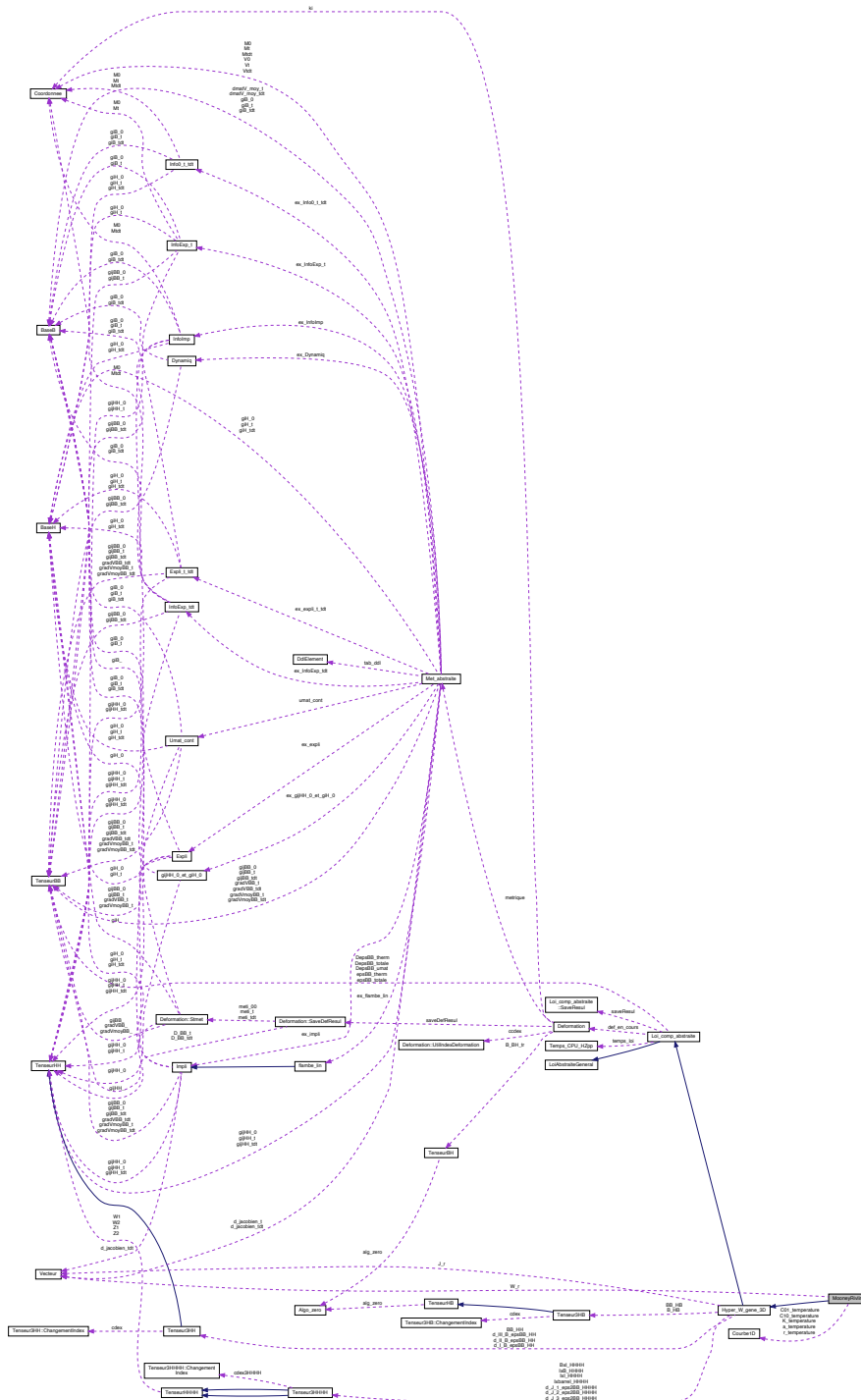
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/MooneyRivlin1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/MooneyRivlin1D.cc

## 6.509 Référence de la classe MooneyRivlin3D

Grphe d'héritage de MooneyRivlin3D:



Graphe de collaboration de MooneyRivlin3D:



## Fonctions membres publiques

- **MooneyRivlin3D** (const [MooneyRivlin3D](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)

- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- double `Module_young_equivalent` (Enum\_dure temps, const Deformation &, SaveResul \*)
- virtual double `HsurH0` (SaveResul \*saveResul) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- void `Info_commande_LoisDeComp` (UtilLecture &lec)

## Fonctions membres protégées

- void `Calcul_SigmaHH` (TenseurHH &sigHH\_t, TenseurBB &DepsBB, DdlElement &tab\_ddl, TenseurBB &gij←BB\_t, TenseurHH &gijHH\_t, BaseB &giB, BaseH &gi\_H, TenseurBB &epsBB\_, TenseurBB &delta\_epsBB\_, TenseurBB &gijBB\_, TenseurHH &gijHH\_, Tableau< TenseurBB \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, TenseurHH &sigHH, EnergieMeca &energ, const EnergieMeca &energ\_t, double &module\_←compressibilite, double &module\_cisaillement, const Met\_abstraite::Expli\_t\_tdt &ex)
- void `Calcul_DsigmaHH_tdt` (TenseurHH &sigHH\_t, TenseurBB &DepsBB, DdlElement &tab\_ddl, BaseB &giB\_t, TenseurBB &gijBB\_t, TenseurHH &gijHH\_t, BaseB &giB\_tdt, Tableau< BaseB > &d\_giB\_tdt, BaseH &giH\_tdt, Tableau< BaseH > &d\_giH\_tdt, TenseurBB &epsBB\_tdt, Tableau< TenseurBB \* > &d\_eps←BB, TenseurBB &delta\_epsBB, TenseurBB &gijBB\_tdt, TenseurHH &gijHH\_tdt, Tableau< TenseurBB \* > &d\_gijBB\_tdt, Tableau< TenseurHH \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, Vecteur &d\_jacobien\_tdt, TenseurHH &sigHH, Tableau< TenseurHH \* > &d\_sigHH, EnergieMeca &energ, const EnergieMeca &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const Met\_←abstraite::Impli &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, TenseurHH &sigHH\_t, TenseurBB &DepsBB, TenseurBB &epsBB\_tdt, TenseurBB &delta\_epsBB, double &jacobien\_0, double &jacobien, TenseurHH &sigHH, TenseurHHHH &d\_sigma\_deps, EnergieMeca &energ, const EnergieMeca &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const Met\_abstraite::Umat\_cont &ex)
- virtual void `CalculGrandeurTravail` (const PtIntegMecalInterne &, const Deformation &, Enum\_dure, const ThermoDonnee &, const Met\_abstraite::Impli \*ex\_impli, const Met\_abstraite::Expli\_t\_tdt \*ex\_expli\_tdt, const Met\_abstraite::Umat\_cont \*ex\_umat, const List\_io< Ddl\_etendu > \*exclure\_dd\_etend, const List\_io< const TypeQuelconque \* > \*exclure\_Q)

## Attributs protégés

- double `C10`
- double `C01`
- double `K`
- `Courbe1D` \* `C10_temperature`
- `Courbe1D` \* `C01_temperature`
- `Courbe1D` \* `K_temperature`
- int `type_pot_vol`
- bool `avec_courbure`
- double `a_courbure`
- double `r_courbure`
- `Courbe1D` \* `a_temperature`
- `Courbe1D` \* `r_temperature`
- double `W_d`
- double `W_v`
- `Vecteur` `W_r`
- double `W_d_J1`
- double `W_d_J2`
- double `W_v_J3`
- double `W_v_J3J3`
- `Tableau2`< double > `W_rs`
- double `W_c`
- double `W_c_J1`
- double `W_c_J3`
- double `W_c_J1_2`
- double `W_c_J3_2`
- double `W_c_J1_J3`

## Membres hérités additionnels

### 6.509.1 Documentation des fonctions membres

**6.509.1.1 Affiche()**

```
void MooneyRivlin3D::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.509.1.2 Calcul\_dsigma\_deps()**

```
void MooneyRivlin3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

[Tenseur3HHHH](#) d\_sigma\_depsHHHH; d\_sigma\_depsHHHH.TransfertDunTenseurGeneral(dSigdepsHHHH.↔  
Symetrise1et2\_3et4());

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.509.1.3 Calcul\_DsigmaHH\_tdt()**

```
void MooneyRivlin3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.509.1.4 Calcul\_SigmaHH()**

```
void MooneyRivlin3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.509.1.5 CalculGrandeurTravail()**

```
virtual void MooneyRivlin3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.509.1.6 Ecriture\_base\_info\_loi()**

```
void MooneyRivlin3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.509.1.7 HsurH0()**

```
virtual double MooneyRivlin3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.509.1.8 Info\_commande\_LoisDeComp()**

```
void MooneyRivlin3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.509.1.9 Lecture\_base\_info\_loi()**

```
void MooneyRivlin3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.509.1.10 LectureDonneesParticulieres()**

```
void MooneyRivlin3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.509.1.11 Module\_young\_equivalent()**

```
double MooneyRivlin3D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.509.1.12 Nouvelle\_loi\_identique()**

```
Loi\_comp\_abstraite * MooneyRivlin3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.509.1.13 TestComplet()**

```
int MooneyRivlin3D::TestComplet ( ) [virtual]
```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/MooneyRivlin3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/MooneyRivlin3D.cc

**6.510 Référence de la classe MotCle**

ici l'énuméré est remplacé par une classe

```
#include <MotCle.h>
```

**Fonctions membres publiques**

- **MotCle** (const [Tableau](#)< string > &TsousMot=tab\_Zero\_string)
- bool **SimotCle** (char \*tabcar)



*retourne true s'il y a un mot cle dans la chaine de character false sinon*

### 6.510.1 Description détaillée

ici l'énuméré est remplacé par une classe

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/MotCle.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/MotCle.cc

## 6.511 Référence du modèle de la classe MV\_ColMat< TYPE >

### Fonctions membres publiques

- **MV\_ColMat** (int, int)
- **MV\_ColMat** (int, int, const TYPE &)
- **MV\_ColMat** (TYPE \*, int m, int n)
- **MV\_ColMat** (TYPE \*, int m, int n, int lda)
- **MV\_ColMat** (TYPE \*, int m, int n, Matrix\_::ref\_type i)
- **MV\_ColMat** (TYPE \*, int m, int n, int lda, Matrix\_::ref\_type i)
- **MV\_ColMat** (const **MV\_ColMat**< TYPE > &)
- TYPE & **operator()** (int, int)
- const TYPE & **operator()** (int, int) const
- **MV\_ColMat**< TYPE > **operator()** (const **MV\_VecIndex** &I, const **MV\_VecIndex** &J)
- const **MV\_ColMat**< TYPE > **operator()** (const **MV\_VecIndex** &I, const **MV\_VecIndex** &J) const
- int **size** (int i) const
- **MV\_ColMat**< TYPE > & **newsize** (int, int)
- int **ref** () const
- **MV\_ColMat**< TYPE > & **operator=** (const **MV\_ColMat**< TYPE > &)
- **MV\_ColMat**< TYPE > & **operator=** (const TYPE &)

### Amis

- ostream & **operator**<< (ostream &s, const **MV\_ColMat**< TYPE > &V)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices\_externes/MV++/mvmtmp\_GR.h

## 6.512 Référence de la classe MV\_VecIndex

### Fonctions membres publiques

- **MV\_VecIndex** (int i1)
- **MV\_VecIndex** (int i1, int i2)
- **MV\_VecIndex** (const **MV\_VecIndex** &s)
- int **start** () const
- int **end** () const
- int **length** () const
- int **all** () const
- **MV\_VecIndex** & **operator=** (const **MV\_VecIndex** &I)
- **MV\_VecIndex** **operator+** (int i)
- **MV\_VecIndex** & **operator+=** (int i)
- **MV\_VecIndex** **operator-** (int i)
- **MV\_VecIndex** & **operator-=** (int i)

La documentation de cette classe a été générée à partir du fichier suivant :

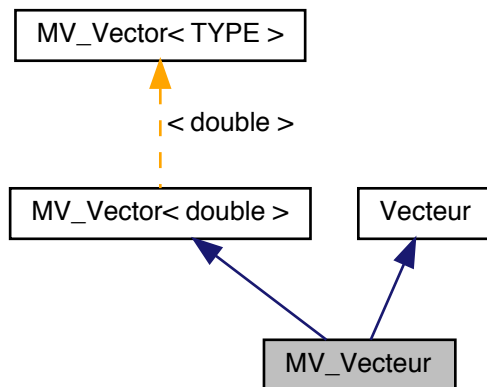
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices\_externes/MV++/mvvind\_GR.h

### 6.513 Référence de la classe MV\_Vecteur

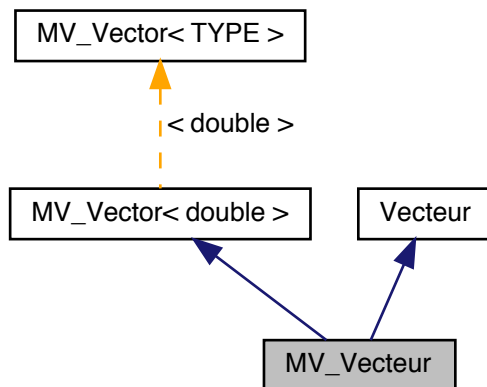
définition d'une classe de jonction entre les Vecteurs et les MV\_Vecteurs

```
#include <Vecteur.h>
```

Graphe d'héritage de MV\_Vecteur:



Graphe de collaboration de MV\_Vecteur:



#### Fonctions membres publiques

- **MV\_Vecteur** ()  
*par défaut*
- **MV\_Vecteur** (const [Vecteur](#) &vec)  
*utilise la même place mémoire que vec*
- **MV\_Vecteur** (const [MV\\_Vector](#)< double > &vec)  
*de copie*
- void **Change\_taille** (int)

- *Change la taille du vecteur (la nouvelle taille est n)*
- `void Libere ()`  
*Permet de desallouer l'ensemble des elements du vecteur.*

## Membres hérités additionnels

### 6.513.1 Description détaillée

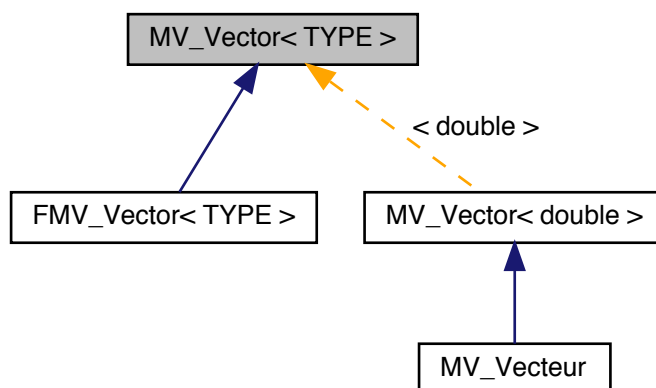
définition d'une classe de jonction entre les Vecteurs et les MV\_Vecteurs

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Vecteurs/Vecteur.h

## 6.514 Référence du modèle de la classe MV\_Vector< TYPE >

Graphe d'héritage de MV\_Vector< TYPE > :



## Fonctions membres publiques

- `MV_Vector (int)`
- `MV_Vector (int, const TYPE &)`
- `MV_Vector (TYPE *, int)`
- `MV_Vector (const TYPE *, int)`
- `MV_Vector (TYPE *, int, MV_Vector_::ref_type i)`
- `MV_Vector (const MV_Vector< TYPE > &)`
- `MV_Vector (MV_Vector< TYPE > &, MV_Vector_::ref_type i)`
- `TYPE & operator() (int i)`
- `const TYPE & operator() (int i) const`
- `TYPE & operator[] (int i)`
- `const TYPE & operator[] (int i) const`
- `MV_Vector< TYPE > operator() (const MV_VeclIndex &l)`
- `MV_Vector< TYPE > operator() (void)`
- `const MV_Vector< TYPE > operator() (void) const`
- `const MV_Vector< TYPE > operator() (const MV_VeclIndex &l) const`
- `int size () const`
- `int ref () const`
- `int null () const`
- `MV_Vector< TYPE > & newsize (int)`
- `MV_Vector< TYPE > & operator= (const MV_Vector< TYPE > &)`
- `MV_Vector< TYPE > & operator= (const TYPE &)`

## Attributs protégés

- TYPE \* **p\_**
- int **dim\_**
- int **ref\_**

## Amis

- istream & **operator**>> (istream &entree, [MV\\_Vector](#)< TYPE > &vec)
- ostream & **operator**<< (ostream &s, const [MV\\_Vector](#)< TYPE > &V)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices\_externes/MV++/mvvtp\_GR.h

## 6.515 Référence de la structure [MV\\_Vector\\_](#)

### Types publics

- enum **ref\_type** { **ref** = 1 }

La documentation de cette structure a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/Matrices\_externes/MV++/mvvrf\_GR.h

## 6.516 Référence de la classe [MvtSolide](#)

```
#include <MvtSolide.h>
```

### Fonctions membres publiques

- **MvtSolide** (const [MvtSolide](#) &nd)
- void **Affiche** () const
- [MvtSolide](#) & **operator=** (const [MvtSolide](#) &d)
- bool **operator==** (const [MvtSolide](#) &a) const
- bool **operator!=** (const [MvtSolide](#) &a) const
- void **Lecture\_mouvements\_solides** ([UtilLecture](#) \*entreePrinc)
- bool **ExisteMvtSolide** () const
- const [Coordonnee](#) & **Premiere\_rotation** () const
- bool **ExisteUneSeuleRotation** () const
- const list< [String\\_et\\_entier](#) > & **Liste\_ident\_centreNoeud** () const
- void **RenseigneCentreNoeud** (const list< [Coordonnee](#) > &lis\_coor)
- [Coordonnee](#) & **AppliqueMvtSolide** ([Coordonnee](#) &M) const
- [Coordonnee](#) & **AppliqueMvtSolide** ([Coordonnee](#) &M, const double &coef) const
- void **Info\_commande\_MvtSolide** (ostream &sort)
- void **EffaceMvtSolide** ()

### Fonctions membres publiques statiques

- static string **MotCleMvtSolide** ()

## Amis

- istream & **operator**>> (istream &, [MvtSolide](#) &)
- ostream & **operator**<< (ostream &, const [MvtSolide](#) &)

### 6.516.1 Description détaillée

Auteur

Gérard Rio

## Version

1.0

## Date

19/01/2007

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/MvtSolide/MvtSolide.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/MvtSolide/MvtSolide.cc

## 6.517 Référence de la classe Nb\_assemb

[Nb\\_assemb](#): description des numéros d'assemblage.

```
#include <Nb_assemb.h>
```

### Fonctions membres publiques

- [Nb\\_assemb](#) (int nn=0)
- [Nb\\_assemb](#) (const [Nb\\_assemb](#) &a)
- [Nb\\_assemb](#) & [operator=](#) (const [Nb\\_assemb](#) &c)
- bool [operator==](#) (const [Nb\\_assemb](#) &a) const
- bool [operator!=](#) (const [Nb\\_assemb](#) &a) const

### Attributs publics

- int n

### Amis

- istream & [operator>>](#) (istream &ent, [Nb\\_assemb](#) &a)
- ostream & [operator<<](#) (ostream &sort, const [Nb\\_assemb](#) &a)

#### 6.517.1 Description détaillée

[Nb\\_assemb](#): description des numéros d'assemblage.

## Auteur

Gérard Rio

## Version

1.0

## Date

23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Nb\_assemb.h

## 6.518 Référence de la classe Maillage::NBelemEtArete

### Fonctions membres publiques

- bool [operator<](#) (const [NBelemEtArete](#) &c) const
- bool [operator>](#) (const [NBelemEtArete](#) &c) const
- bool [operator==](#) (const [NBelemEtArete](#) &c) const
- bool [operator!=](#) (const [NBelemEtArete](#) &c) const

## Attributs publics

- int **nbElem**
- int **nbArete**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/maillage4.cc

## 6.519 Référence de la classe Maillage::NBelemEtFace

### Fonctions membres publiques

- bool **operator**< (const [NBelemEtFace](#) &c) const
- bool **operator**> (const [NBelemEtFace](#) &c) const
- bool **operator**== (const [NBelemEtFace](#) &c) const
- bool **operator**!= (const [NBelemEtFace](#) &c) const

### Attributs publics

- int **nbElem**
- int **nbFace**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/maillage4.cc

## 6.520 Référence de la classe Maillage::NBelemEtptInteg

### Fonctions membres publiques

- bool **operator**< (const [NBelemEtptInteg](#) &c) const
- bool **operator**> (const [NBelemEtptInteg](#) &c) const
- bool **operator**== (const [NBelemEtptInteg](#) &c) const
- bool **operator**!= (const [NBelemEtptInteg](#) &c) const

### Attributs publics

- int **nbElem**
- int **nbPtInteg**

### Amis

- istream & **operator**>> (istream &ent, [NBelemEtptInteg](#) &de)
- ostream & **operator**<< (ostream &sort, const [NBelemEtptInteg](#) &de)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/maillage4.cc

## 6.521 Référence de la classe Maillage::NBelemFAEtptInteg

### Fonctions membres publiques

- bool **operator**< (const [NBelemFAEtptInteg](#) &c) const
- bool **operator**> (const [NBelemFAEtptInteg](#) &c) const
- bool **operator**== (const [NBelemFAEtptInteg](#) &c) const
- bool **operator**!= (const [NBelemFAEtptInteg](#) &c) const

## Attributs publics

- int **nbElem**
- int **nbFA**
- int **nbPtInteg**

## Amis

- istream & **operator**>> (istream &ent, [NBelemFAEptInteg](#) &de)
- ostream & **operator**<< (ostream &sort, const [NBelemFAEptInteg](#) &de)

La documentation de cette classe a été générée à partir du fichier suivant :

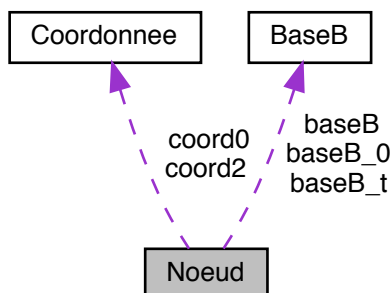
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/maillage4.cc

## 6.522 Référence de la classe Noeud

[Noeud](#): un noeud.

```
#include <Noeud.h>
```

Graphe de collaboration de Noeud:



## Fonctions membres publiques

- **Noeud** (int num\_id=-3, int num\_maill=0)
- **Noeud** (int num\_id, int dimension, int num\_maill)
- **Noeud** (int num\_id, const [Coordonnee](#) &c0, int num\_Mail)
- **Noeud** (int num\_id, const [Coordonnee](#) &c0, const [Tableau](#)< [Ddl](#) > &tab, int num\_Mail)
- **Noeud** (int num\_id, const [Coordonnee](#) &c0, const [Coordonnee](#) &c1, const [Tableau](#)< [Ddl](#) > &tab, int num↵\_Mail)
- **Noeud** (int num\_id, const [Coordonnee](#) &c0, const [Coordonnee](#) &c1, const [Coordonnee](#) &c2, const [Tableau](#)< [Ddl](#) > &tab, int num\_Mail)
- **Noeud** (const [Noeud](#) &nd)
- void **Affiche** (int niveau=0) const
- void **Affiche** (ofstream &sort) const
- void **Change\_num\_noeud** (int nouveau\_num)
- void **Change\_num\_Mail** (int nouveau\_num)
- [Enum\\_liaison\\_noeud](#) **Enu\_liaison** (int nb\_assemb) const
- const [Tableau](#)< [Posi\\_ddl\\_noeud](#) > \* **Ddl\_Noeud\_liier** (int nb\_assemb) const
- void **Change\_Enu\_liason** (int nb\_assemb, [Enum\\_liaison\\_noeud](#) enu\_liai, const [Tableau](#)< [Posi\\_ddl\\_noeud](#) > &tab\_liier)
- void **Suppression\_tous\_liaisons\_noeuds** ()
- void **Change\_coord0** (const [Coordonnee](#) &nouveau\_coord0)

---

```

— void Insert_coord1 (const Coordonnee &nouveau_coord1)
— void Insert_coord1 ()
— void Change_coord1 (const Coordonnee &nouveau_coord1)
— void Change_coord2 (const Coordonnee &nouveau_coord2)
— void Ajout_coord2 (const Coordonnee &delta_coord2)
— const Coordonnee & Coord0 () const
— bool ExisteCoord0 () const
— Coordonnee Coord1 () const
— bool ExisteCoord1 () const
— const Coordonnee & Coord2 () const
— double Max_var_coor_t_a_tdt () const
— bool ExisteCoord2 () const
— int Nombre_ddl () const
— int Nombre_var_ddl_actives () const
— int NB_ddl_actif_casAssemb (int nb_assemb) const
— int Nombre_var_ddl_actives (Enum_ddl en) const
— List\_io< Enum_ddl > Les_type_de_ddl (bool absolue)
— List\_io< Ddl\_enum\_etendu > Les_type_de_ddl_etendu (bool absolue)
— List\_io< TypeQuelconque > Les_TypeQuelconque (bool absolue)
— Tableau< double > Valeur_multi_et_Tensorielle (const List\_io< Ddl\_enum\_etendu > &li_enu_scal,
List\_io< TypeQuelconque > &li_quelc) const
— int Num_noeud () const
— int Num_Mail () const
— int Dimension () const
— Noeud & operator= (const Noeud &n)
— int PosiAssemb (int i) const
— void InitNouveauCasAssemb (int nb_cas)
— void Enreg_ordre_variable_ddl_actives (int nb_assemb)
— int Position_ddl (Enum_ddl enu, int nb_assemb) const
— void ChangePosiAssemb (int a, int i)
— int Pointeur_assemblage (Enum_ddl enu, int nb_assemb) const
— const BaseB * Const_BaseB_Noeud () const
— const BaseB * Const_BaseB_Noeud_t () const
— const BaseB * Const_BaseB_Noeud_0 () const
— BaseB * BaseB_Noeud () const
— BaseB * BaseB_Noeud_t () const
— BaseB * BaseB_Noeud_0 () const
— int TestComplet () const
— void PlusDdl (Ddl &a)
— void PlusTabDdl (Tableau< Ddl > &ta)
— void PlusTabDdl (const Noeud &noe)
— void Change_fixe (Enum_ddl en, bool val)
— void Change_fixe (int nb, bool val)
— bool Ddl_fixe (Enum_ddl en) const
— bool Ddl_sous_ou_sur_fixe (Enum_ddl en) const
— void Retour_a_la_normal_sur_ou_sous_fixe (Enum_ddl en)
— void Met_hors_service (Enum_ddl en)
— void Met_en_service (Enum_ddl en)
— void Met_hors_service (const Tableau< Enum_ddl > &taben)
— void Met_en_service (const Tableau< Enum_ddl > &taben)
— void Met_hors_service ()
— void Met_en_service ()
— void Met_hors_service_ddl ()
— void Met_en_service_ddl ()
— bool En_service (Enum_ddl en) const
— bool UneVariable (Enum_ddl en) const
— void ChangeVariable_a_Donnee (Enum_ddl en)
— void ChangeDonnee_a_Variable (Enum_ddl en)
— void ChangeVariable_a_Donnee (const Tableau< Enum_ddl > &taben)
— void ChangeDonnee_a_Variable (const Tableau< Enum_ddl > &taben)
— void ChangeToutesLesConditions (const Tableau< Ddl > &ta)
— void ChangeStatut (int cas, Enum_ddl enuta)
— void ChangeStatut (int cas, Enum\_boolddl enubold)
— void Change_val_0 (Enum_ddl en, double val)
— void Change_val_t (Enum_ddl en, double val)

```



- void **Change\_val\_tdt** (Enum\_ddl en, double val)
- void **Ajout\_val\_0** (Enum\_ddl en, double val)
- void **Ajout\_val\_t** (Enum\_ddl en, double val)
- void **Ajout\_val\_tdt** (Enum\_ddl en, double val)
- double **Valeur\_0** (Enum\_ddl en) const
- double **Valeur\_t** (Enum\_ddl en) const
- double **Valeur\_tdt** (Enum\_ddl en) const
- **Ddl Ddl\_noeud\_0** (Enum\_ddl enu) const
- **Ddl Ddl\_noeud\_t** (Enum\_ddl enu) const
- **Ddl Ddl\_noeud\_tdt** (Enum\_ddl enu) const
- void **Travail\_tdt** ()
- void **Travail\_t** ()
- bool **Existe\_ici** (const Enum\_ddl en) const
- void **ZeroVariablesDdl** (bool cas)
- void **TdtversT** ()
- void **TversTdt** ()
- bool **Tdt** ()
- bool **Contrainte** (**Vecteur** &Sig)
- **TenseurHB** & **Contrainte** (**TenseurHB** &Sig)
- void **Lecture** (**UtilLecture** \*entreePrinc)
- void **Info\_commande\_Noeud** (**UtilLecture** \*entreePrinc, **Coordonnee** &coor, int nb\_du\_noeud)
- void **LectureDeplacements** (**UtilLecture** \*entreePrinc)
- void **Lecture\_base\_info** (ifstream &ent, int cas)
- void **Ecriture\_base\_info** (ofstream &sort, int cas)
- void **AjoutTabTypeQuelconque** (const **Tableau**< **TypeQuelconque** > &tab\_t\_quel)
- void **AjoutUnTypeQuelconque** (const **TypeQuelconque** &t\_quel)
- void **SupprimeUnTypeQuelconque** (const **TypeQuelconque** &t\_quel)
- void **SupprimeTabTypeQuelconque** (const **Tableau**< **TypeQuelconque** > &tab\_t\_quel)
- bool **Existe\_ici** (**TypeQuelconque\_enum\_etendu** en) const
- **TypeQuelconque** & **ModifGrandeur\_quelconque** (**TypeQuelconque\_enum\_etendu** a)
- const **TypeQuelconque** & **Grandeur\_quelconque** (**TypeQuelconque\_enum\_etendu** enu) const
- int **Grandeur\_quelconque\_update** (**TypeQuelconque\_enum\_etendu** enu) const
- void **Mise\_non\_update\_grandeurs\_quelconques** ()
- void **Mise\_non\_update\_grandeurs\_quelconques** (**TypeQuelconque\_enum\_etendu** enu)
- void **AjoutTabDdl\_etendu** (const **Tableau**< **Ddl\_enum\_etendu** > &tab\_ddletendu)
- void **AjoutUnDdl\_etendu** (const **Ddl\_enum\_etendu** &ddletendu)
- void **SupprimeUnDdl\_etendu** (const **Ddl\_enum\_etendu** &ddletendu)
- void **SupprimeTabDdl\_etendu** (const **Tableau**< **Ddl\_enum\_etendu** > &tab\_ddletendu)
- bool **Existe\_ici\_ddlEtendu** (const **Ddl\_enum\_etendu** &ddletendu) const
- **Ddl\_etendu** & **ModifDdl\_etendu** (const **Ddl\_enum\_etendu** &ddletendu)
- const **Ddl\_etendu** & **DdlEtendue** (const **Ddl\_enum\_etendu** &ddlenumetendu) const
- int **DdlEtendue\_update** (const **Ddl\_enum\_etendu** &ddlenumetendu)
- void **Mise\_non\_update\_Ddl\_etendu** ()
- void **Mise\_non\_update\_Ddl\_etendu** (**Ddl\_enum\_etendu** ddlenumetendu)

## Fonctions membres publiques statiques

- static void **SchemaXML\_Noeud** (ofstream &sort, const **Enum\_IO\_XML** enu)

## Fonctions membres protégées

- void **Liaison\_t** (int nb)
- void **Liaison\_tdt** (int nb)
- void **LiaiSeulet** (int nb)
- void **MiseAJour** ()
- void **MiseAJourEnum** ()
- void **MiseAJourActif** ()
- int **Existe** (const Enum\_ddl en) const
- void **Lecture\_grandeurs\_quelconque** (istream &ent)
- void **Ecriture\_grandeurs\_quelconque** (ostream &sort) const
- int **Indice\_grandeur\_quelconque** (**TypeQuelconque\_enum\_etendu** a)
- void **MiseAJourTypeQuelconque** ()
- int **Existe** (**TypeQuelconque\_enum\_etendu** en) const
- void **Lecture\_Ddl\_etendu** (istream &ent)

- void **Ecriture\_Ddl\_etendu** (ostream &sort) const
- int **Indice\_Ddl\_etendu** (const [Ddl\\_enum\\_etendu](#) &a)
- void **MiseAJourDdl\_etendu** ()
- int **Existe\_ddlEtendu** (const [Ddl\\_enum\\_etendu](#) &en) const
- void **OrdonnerTableauDdl** ([Tableau](#)< [Ddl](#) > &ta)
- void **ListeDdlAjouter** (list< [Ddl](#) > &liDdl, [Tableau](#)< [Ddl](#) > &ta, bool &change\_donn\_var)

### Attributs protégés

- int **num\_noeud**
- int **num\_Mail**
- [Tableau](#)< [Ddl](#) > **tab\_ddl**
- [Coordonnee](#) \* **coord0**
- [Tableau](#)< [Enum\\_liaison\\_noeud](#) > \* **tab\_enu\_liaison**
- [Tableau](#)< [Tableau](#)< [Posi\\_ddl\\_noeud](#) > > \* **tab\_ddl\_lier**
- [Tableau](#)< double > **tab\_0**
- [Tableau](#)< double > **tab\_tdt**
- [Tableau](#)< double \* > **coord1**
- [Coordonnee](#) \* **coord2**
- [Tableau](#)< short int > **tab\_var\_actives**
- [Tableau](#)< short int > **tab\_actif**
- [Tableau](#)< int > **posiAssemb**
- [BaseB](#) \* **baseB\_0**
- [BaseB](#) \* **baseB\_t**
- [BaseB](#) \* **baseB**
- [Tableau](#)< int > **tab\_nb\_Elem**
- [Tableau](#)< [TypeQuelconque](#) > **tab\_type\_quel**
- [Tableau](#)< int > **update\_type\_quel**
- int **nbTypeQ\_update**
- [Tableau](#)< [Ddl\\_etendu](#) > **tab\_ddletendu**
- [Tableau](#)< int > **update\_ddletendu**
- int **nbddletendu\_update**
- list< [Tableau](#)< int > >::iterator **pos\_enum**
- [Tableau](#)< [List\\_io](#)< [Tableau](#)< int > >::iterator > **t\_enum\_s**
- list< [Tableau](#)< int > >::iterator **pos\_Quelconque**
- list< [Tableau](#)< int > >::iterator **pos\_ddletendu**

### Attributs protégés statiques

- static short int **posi\_type\_quel** = 1
- static short int **posi\_ddletendu** = 1
- static list< [Tableau](#)< int > > **list\_tab\_enum**
- static [List\\_io](#)< [Tableau](#)< int > > **list\_tab\_posi\_actif**
- static [List\\_io](#)< [Tableau](#)< int > > **list\_tab\_typeQuelconque**
- static [List\\_io](#)< [Tableau](#)< int > > **list\_tab\_ddletendu**

### Amis

- istream & **operator**>> (istream &, [Noeud](#) &)
- ostream & **operator**<< (ostream &, const [Noeud](#) &)

## 6.522.1 Description détaillée

[Noeud](#): un noeud.

Auteur

Gérard Rio

Version

1.0

Date

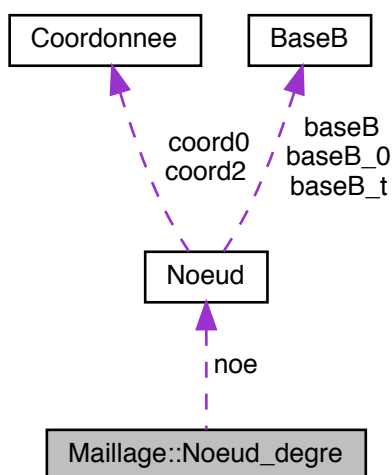
23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Noeud.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Noeud.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Noeud2.cc

## 6.523 Référence de la classe Maillage::Noeud\_degre

Grphe de collaboration de Maillage::Noeud\_degre:



### Fonctions membres publiques

- **Noeud\_degre** ([Noeud](#) \*no, int deg)
- **Noeud\_degre** (const [Noeud\\_degre](#) &no)
- bool **operator>** (const [Noeud\\_degre](#) &a) const
- bool **operator>=** (const [Noeud\\_degre](#) &a) const
- bool **operator<** (const [Noeud\\_degre](#) &a) const
- bool **operator<=** (const [Noeud\\_degre](#) &a) const
- [Noeud\\_degre](#) & **operator=** (const [Noeud\\_degre](#) &de)

### Attributs publics

- [Noeud](#) \* **noe**
- int **degre**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.h

## 6.524 Référence de la classe Biel\_axi::NombresConstruire

### Attributs publics

- int **nbne**

- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axi.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axi.cc

## 6.525 Référence de la classe `Biel_axiQ::NombresConstruire`

### Attributs publics

- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axiQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biel\_axiQ.cc

## 6.526 Référence de la classe `Biellette::NombresConstruire`

### Attributs publics

- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biellette.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/Biellette.cc

## 6.527 Référence de la classe `BielletteC1::NombresConstruire`

### Attributs publics

- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/BielletteC1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/BielletteC1.cc

## 6.528 Référence de la classe BielleQ::NombresConstruire

### Attributs publics

- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Bielle/BielleQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Bielle/BielleQ.cc

## 6.529 Référence de la classe BielleThermi::NombresConstruire

### Attributs publics

- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiA**
- int **nbiMas**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/Bielle/BielleThermi.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/Bielle/BielleThermi.cc

## 6.530 Référence de la classe ElemPoint::NombresConstruire

### Attributs publics

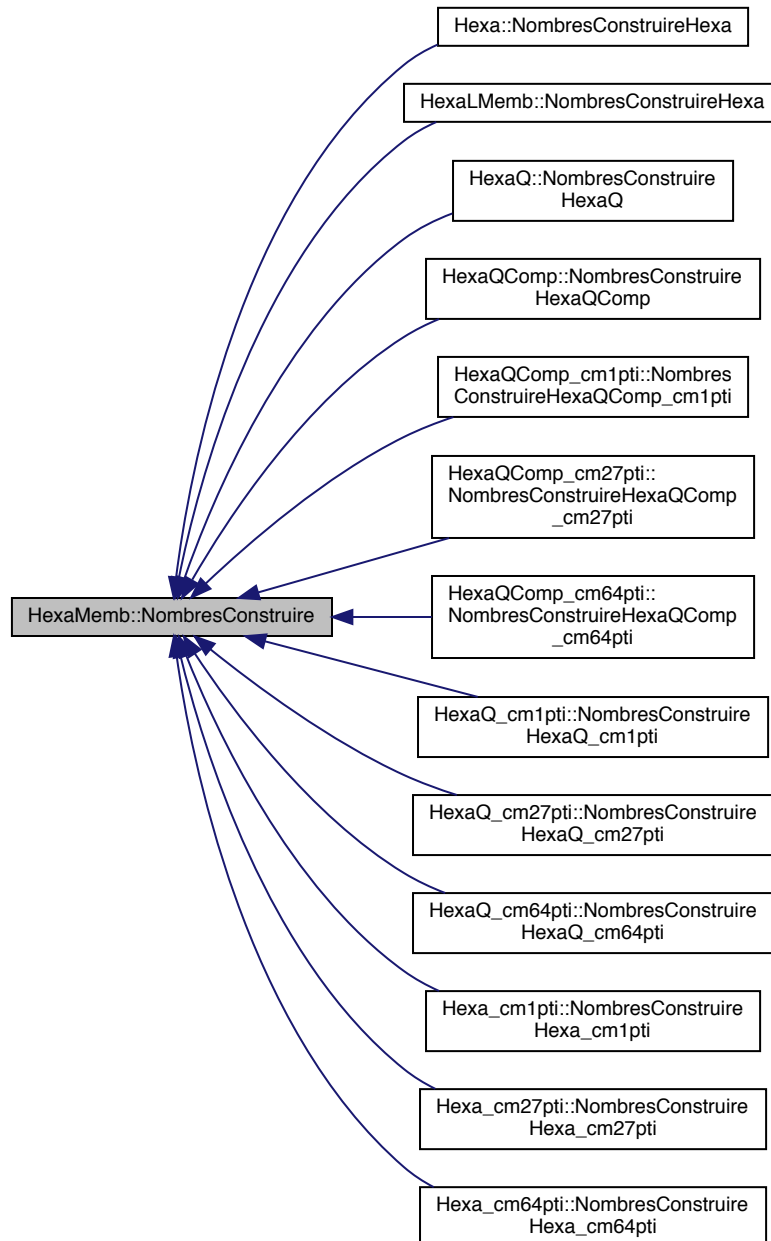
- int **nbne**
- int **nbiEr**
- int **nbiMas**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.cc

## 6.531 Référence de la classe HexaMemb::NombresConstruire

Graphe d'héritage de HexaMemb::NombresConstruire:



### Attributs publics

- int **nbne**
- int **nbneS**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiV**
- int **nbiS**

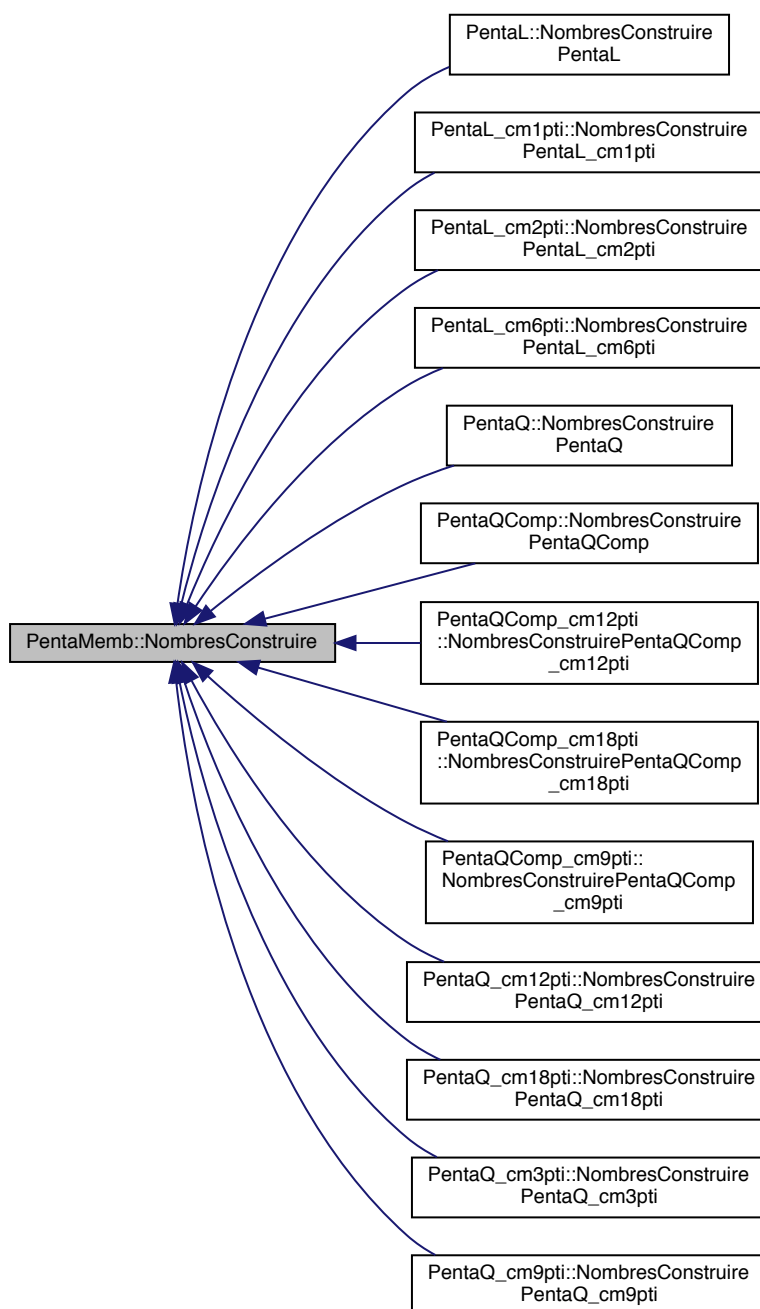
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaMemb.h

## 6.532 Référence de la classe PentaMemb::NombresConstruire

Graphe d'héritage de PentaMemb::NombresConstruire:



## Attributs publics

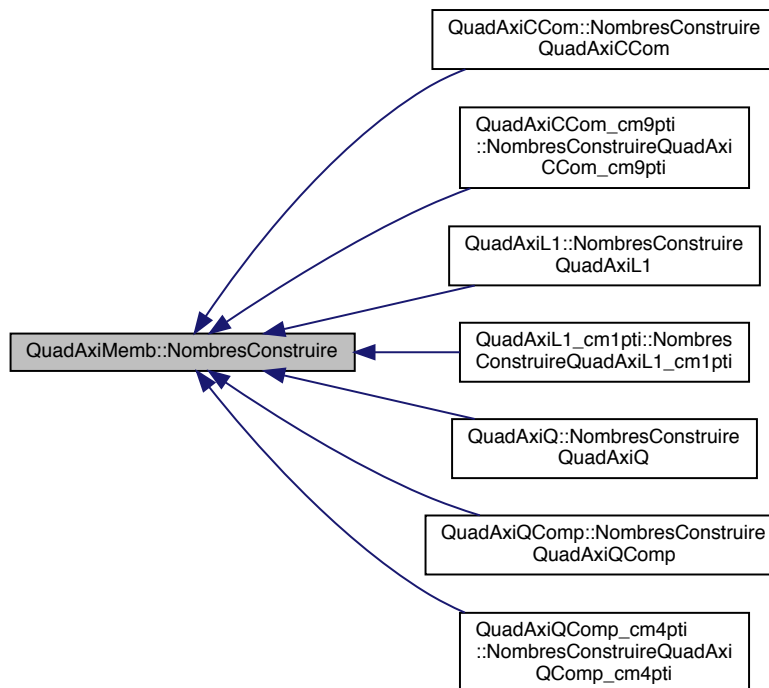
- int **nbne**
- int **nbneSQ**
- int **nbneST**
- int **nbneAQ**
- int **nbneAT**
- int **nbl**
- int **nbiQ**
- int **nbiT**
- int **nbiEr**
- int **nbiV**
- int **nbiSQ**
- int **nbiST**
- int **nbiAQ**
- int **nbiAT**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaMemb.h

## 6.533 Référence de la classe QuadAxiMemb::NombresConstruire

Graphe d'héritage de QuadAxiMemb::NombresConstruire:



## Attributs publics

- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**



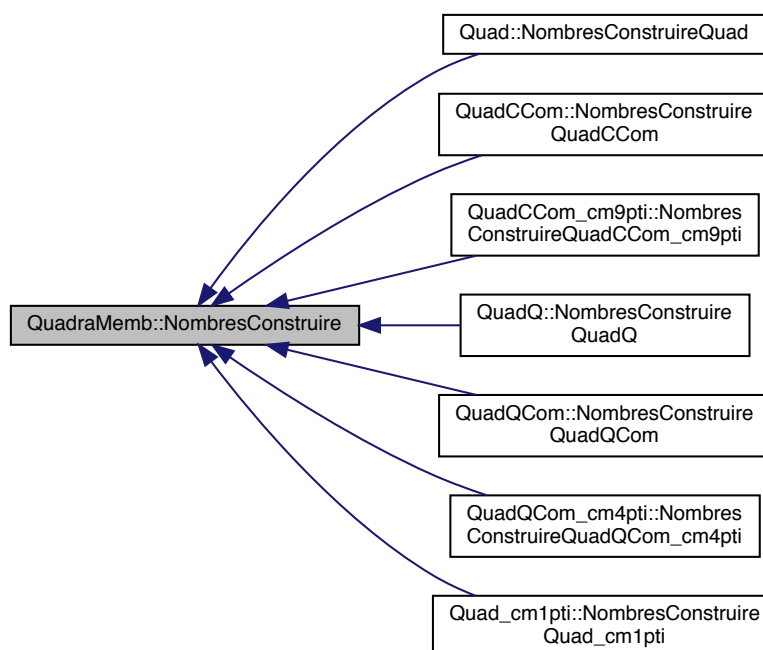
- int **nbiS**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxi←Memb.h

## 6.534 Référence de la classe QuadraMemb::NombresConstruire

Grappe d'héritage de QuadraMemb::NombresConstruire:



### Attributs publics

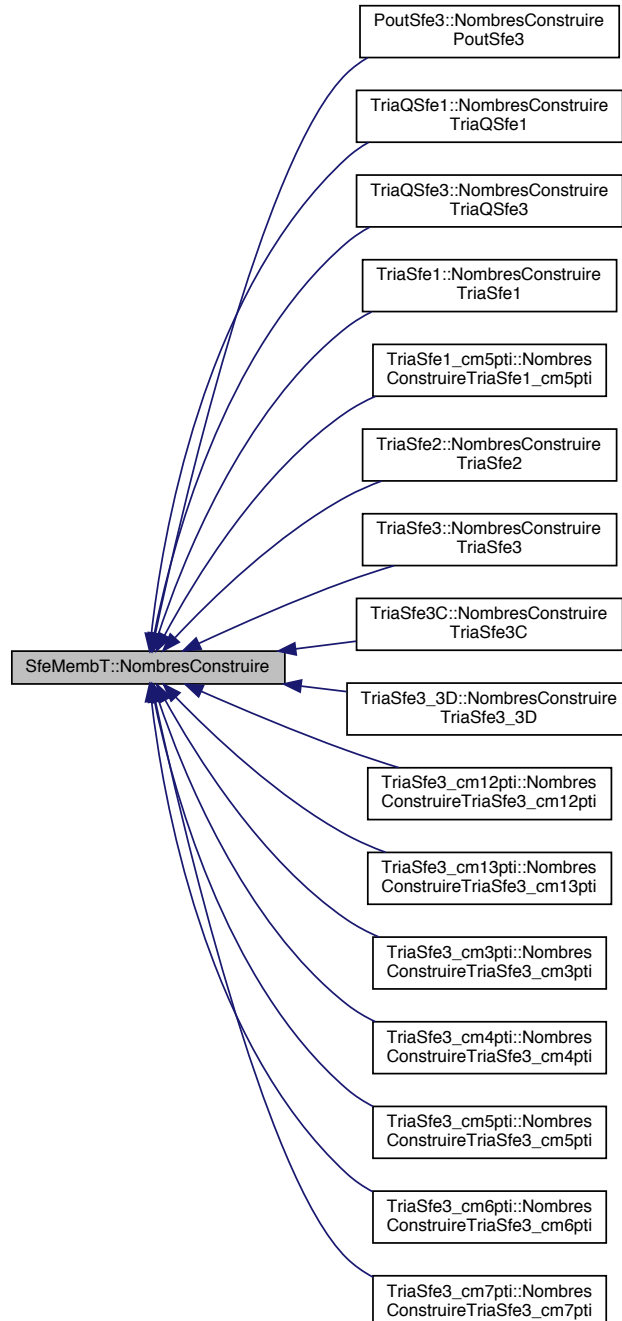
- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiS**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.h

## 6.535 Référence de la classe SfeMembT::NombresConstruire

Graphe d'héritage de SfeMembT::NombresConstruire:



### Attributs publics

- int **nbnce**
- int **nbnte**
- int **nbneA**
- int **nbis**
- int **nbie**

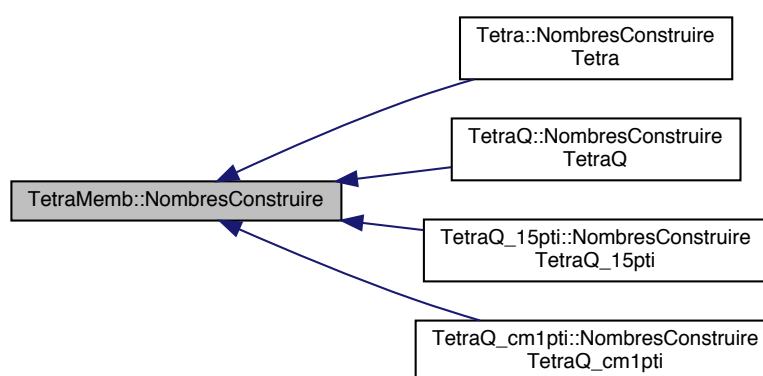
- int **nbisEr**
- int **nbieEr**
- int **nbisur**
- int **nbIA**
- int **nbisMas**
- int **nbieMas**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.h

## 6.536 Référence de la classe TetraMemb::NombresConstruire

Graphe d'héritage de TetraMemb::NombresConstruire:



### Attributs publics

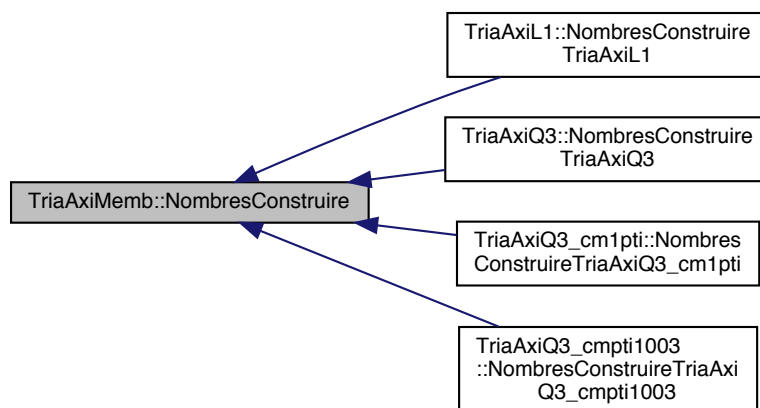
- int **nbne**
- int **nbneS**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiV**
- int **nbiS**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraMemb.h

## 6.537 Référence de la classe TriaAxiMemb::NombresConstruire

Graphe d'héritage de TriaAxiMemb::NombresConstruire:



### Attributs publics

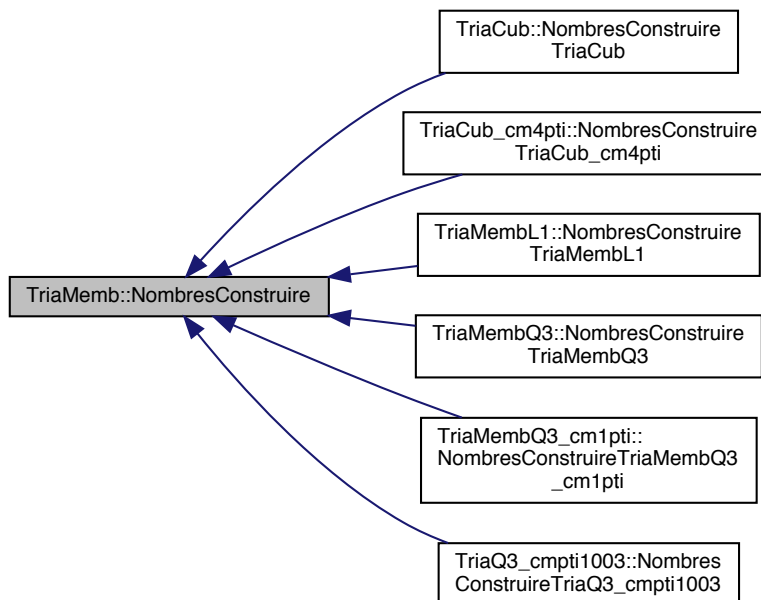
- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiS**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiMemb.h

## 6.538 Référence de la classe TriaMemb::NombresConstruire

Graphe d'héritage de TriaMemb::NombresConstruire:



### Attributs publics

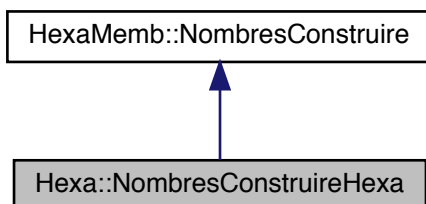
- int **nbne**
- int **nbneA**
- int **nbi**
- int **nbiEr**
- int **nbiS**
- int **nbiA**
- int **nbiMas**
- int **nbiHour**

La documentation de cette classe a été générée à partir du fichier suivant :

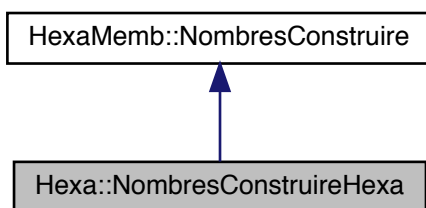
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.h

## 6.539 Référence de la classe Hexa::NombresConstruireHexa

Graphe d'héritage de Hexa::NombresConstruireHexa:



Graphe de collaboration de Hexa::NombresConstruireHexa:



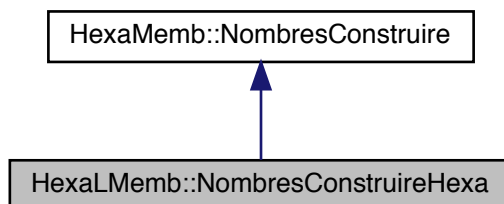
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

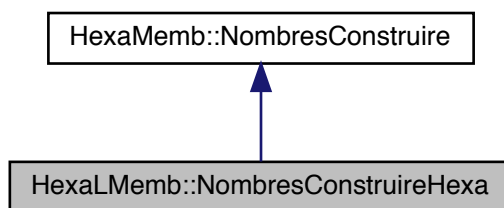
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaLMemb.cc

## 6.540 Référence de la classe HexaLMemb::NombresConstruireHexa

Graphe d'héritage de HexaLMemb::NombresConstruireHexa:



Graphe de collaboration de HexaLMemb::NombresConstruireHexa:



### Membres hérités additionnels

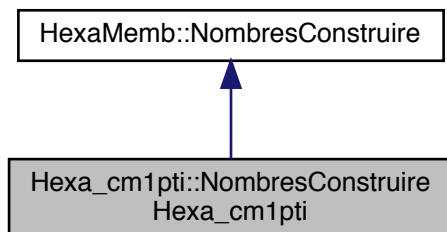
La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaLMemb.h

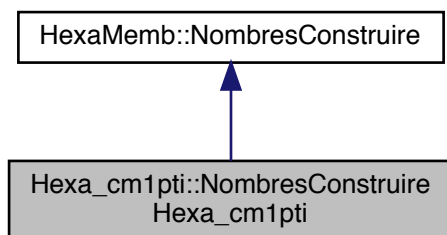
## 6.541 Référence de la classe

### Hexa\_cm1pti::NombresConstruireHexa\_cm1pti

Grphe d'héritage de Hexa\_cm1pti::NombresConstruireHexa\_cm1pti:



Grphe de collaboration de Hexa\_cm1pti::NombresConstruireHexa\_cm1pti:



### Membres hérités additionnels

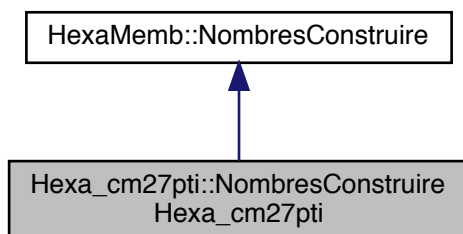
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm1pti.cc

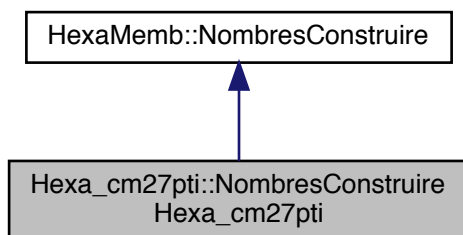


## 6.542 Référence de la classe Hexa\_cm27pti::NombresConstruireHexa\_cm27pti

Graphe d'héritage de Hexa\_cm27pti::NombresConstruireHexa\_cm27pti:



Graphe de collaboration de Hexa\_cm27pti::NombresConstruireHexa\_cm27pti:



### Membres hérités additionnels

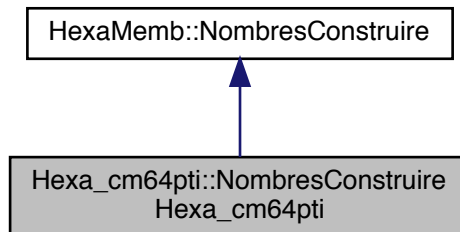
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm27pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm27pti.cc

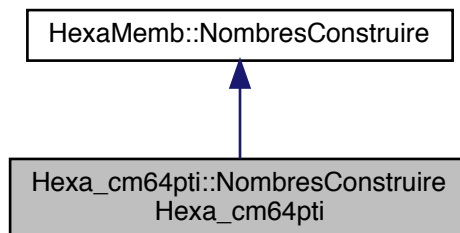
## 6.543 Référence de la classe

### Hexa\_cm64pti::NombresConstruireHexa\_cm64pti

Graphe d'héritage de Hexa\_cm64pti::NombresConstruireHexa\_cm64pti:



Graphe de collaboration de Hexa\_cm64pti::NombresConstruireHexa\_cm64pti:



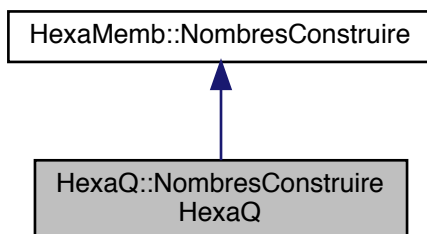
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

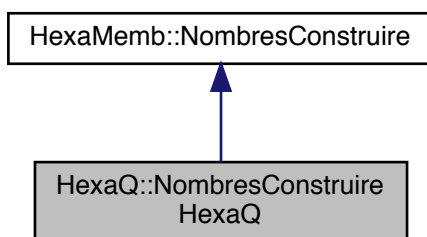
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm64pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/Hexa\_cm64pti.cc

## 6.544 Référence de la classe HexaQ::NombresConstruireHexaQ

Grappe d'héritage de HexaQ::NombresConstruireHexaQ:



Grappe de collaboration de HexaQ::NombresConstruireHexaQ:



### Membres hérités additionnels

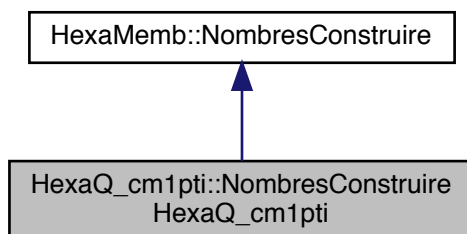
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ.cc

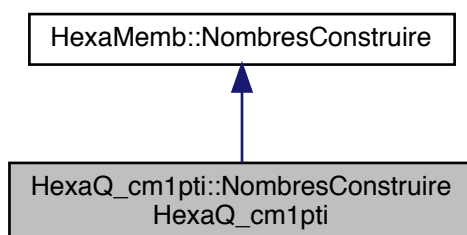
## 6.545 Référence de la classe

### HexaQ\_cm1pti::NombresConstruireHexaQ\_cm1pti

Grphe d'héritage de HexaQ\_cm1pti::NombresConstruireHexaQ\_cm1pti:



Grphe de collaboration de HexaQ\_cm1pti::NombresConstruireHexaQ\_cm1pti:



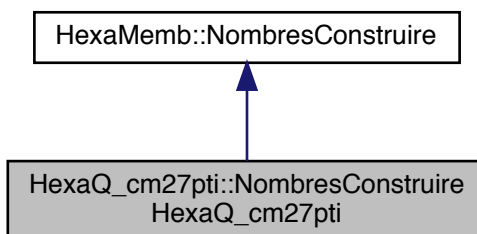
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

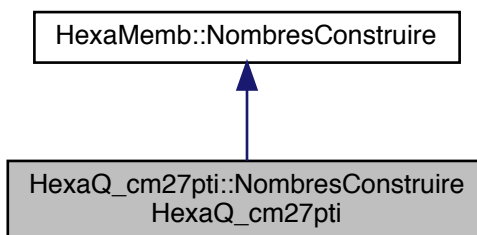
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm1pti.cc

## 6.546 Référence de la classe HexaQ\_cm27pti::NombresConstruireHexaQ\_cm27pti

Graphe d'héritage de HexaQ\_cm27pti::NombresConstruireHexaQ\_cm27pti:



Graphe de collaboration de HexaQ\_cm27pti::NombresConstruireHexaQ\_cm27pti:



### Membres hérités additionnels

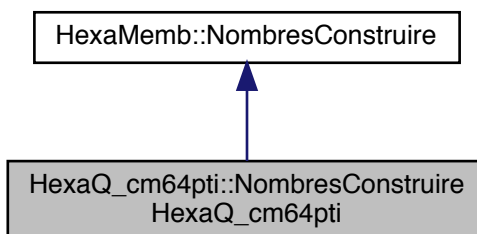
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm27pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm27pti.cc

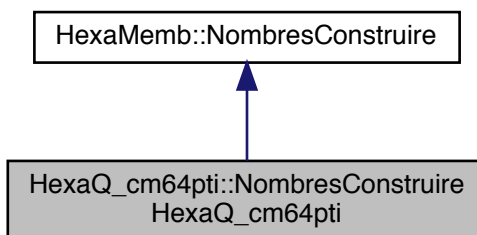
## 6.547 Référence de la classe

### HexaQ\_cm64pti::NombresConstruireHexaQ\_cm64pti

Grphe d'héritage de HexaQ\_cm64pti::NombresConstruireHexaQ\_cm64pti:



Grphe de collaboration de HexaQ\_cm64pti::NombresConstruireHexaQ\_cm64pti:



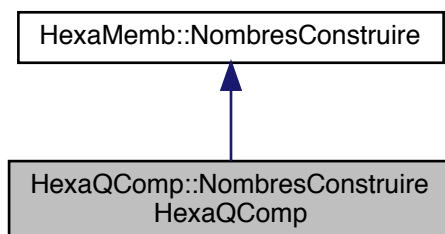
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

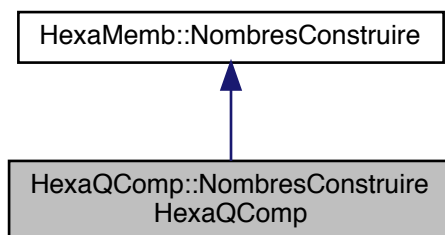
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm64pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQ\_cm64pti.cc

## 6.548 Référence de la classe HexaQComp::NombresConstruireHexaQComp

Graphe d'héritage de HexaQComp::NombresConstruireHexaQComp:



Graphe de collaboration de HexaQComp::NombresConstruireHexaQComp:



### Membres hérités additionnels

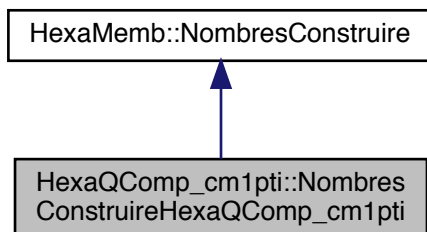
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp.cc

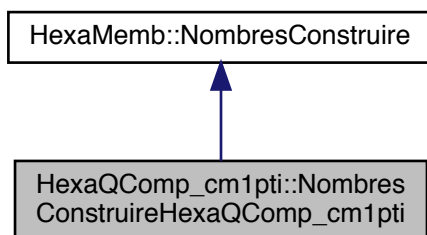
## 6.549 Référence de la classe

### HexaQComp\_cm1pti::NombresConstruireHexaQComp\_cm1pti

Grphe d'héritage de HexaQComp\_cm1pti::NombresConstruireHexaQComp\_cm1pti:



Grphe de collaboration de HexaQComp\_cm1pti::NombresConstruireHexaQComp\_cm1pti:



### Membres hérités additionnels

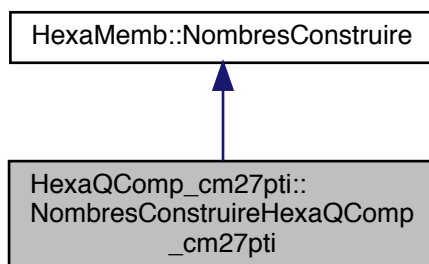
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_1pti.cc

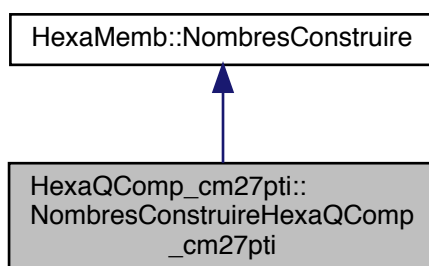


## 6.550 Référence de la classe HexaQComp\_cm27pti::NombresConstruireHexaQComp\_cm27pti

Graphe d'héritage de HexaQComp\_cm27pti::NombresConstruireHexaQComp\_cm27pti:



Graphe de collaboration de HexaQComp\_cm27pti::NombresConstruireHexaQComp\_cm27pti:



### Membres hérités additionnels

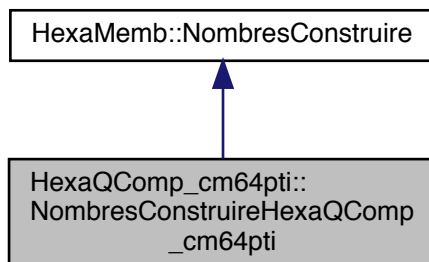
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm27pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm27pti.cc

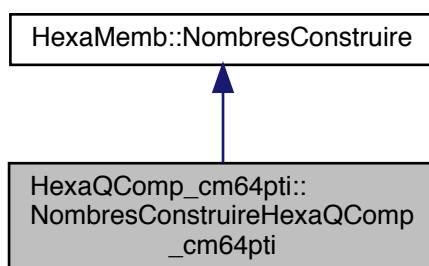
## 6.551 Référence de la classe

### HexaQComp\_cm64pti::NombresConstruireHexaQComp\_cm64pti

Graphe d'héritage de HexaQComp\_cm64pti::NombresConstruireHexaQComp\_cm64pti:



Graphe de collaboration de HexaQComp\_cm64pti::NombresConstruireHexaQComp\_cm64pti:



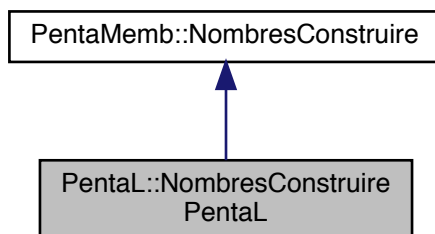
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

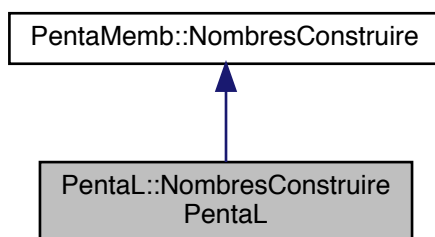
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm64pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaQComp\_cm64pti.cc

## 6.552 Référence de la classe PentaL::NombresConstruirePentaL

Grappe d'héritage de PentaL::NombresConstruirePentaL:



Grappe de collaboration de PentaL::NombresConstruirePentaL:



### Membres hérités additionnels

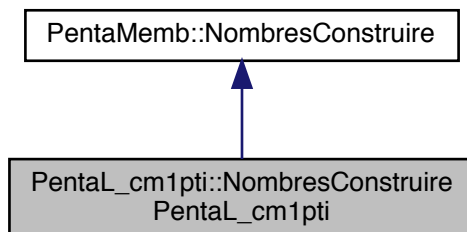
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL.cc

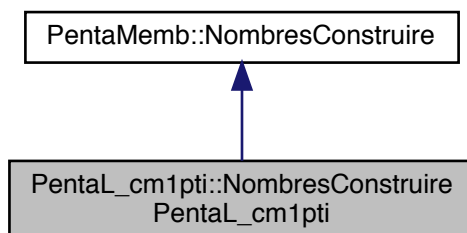
## 6.553 Référence de la classe

### PentaL\_cm1pti::NombresConstruirePentaL\_cm1pti

Grphe d'héritage de PentaL\_cm1pti::NombresConstruirePentaL\_cm1pti:



Grphe de collaboration de PentaL\_cm1pti::NombresConstruirePentaL\_cm1pti:



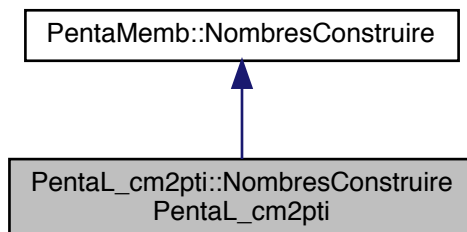
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

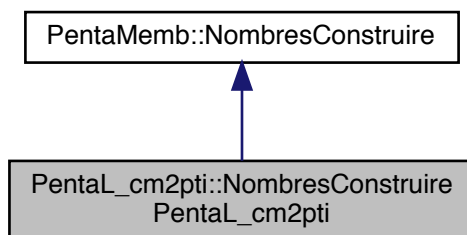
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm1pti.cc

## 6.554 Référence de la classe PentaL\_cm2pti::NombresConstruirePentaL\_cm2pti

Graphe d'héritage de PentaL\_cm2pti::NombresConstruirePentaL\_cm2pti:



Graphe de collaboration de PentaL\_cm2pti::NombresConstruirePentaL\_cm2pti:



### Membres hérités additionnels

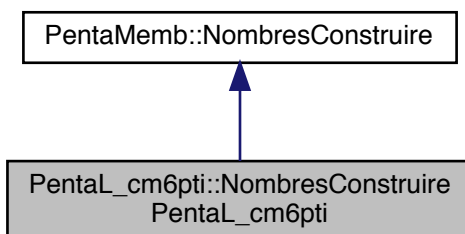
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm2pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm2pti.cc

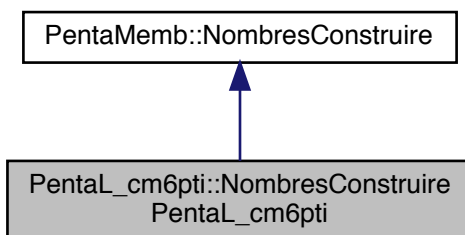
## 6.555 Référence de la classe

### PentaL\_cm6pti::NombresConstruirePentaL\_cm6pti

Grphe d'héritage de PentaL\_cm6pti::NombresConstruirePentaL\_cm6pti:



Grphe de collaboration de PentaL\_cm6pti::NombresConstruirePentaL\_cm6pti:



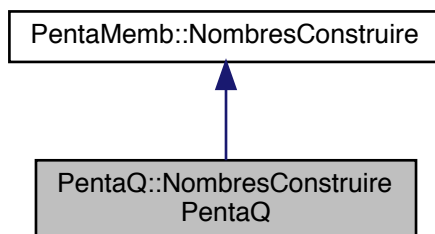
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

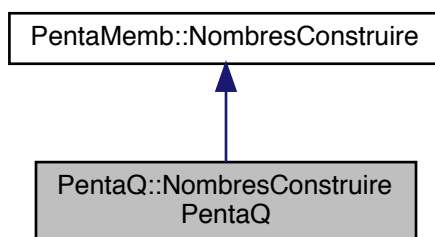
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm6pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm6pti.cc

## 6.556 Référence de la classe PentaQ::NombresConstruirePentaQ

Grphe d'héritage de PentaQ::NombresConstruirePentaQ:



Grphe de collaboration de PentaQ::NombresConstruirePentaQ:



### Membres hérités additionnels

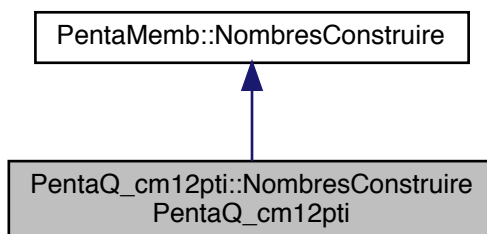
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ.cc

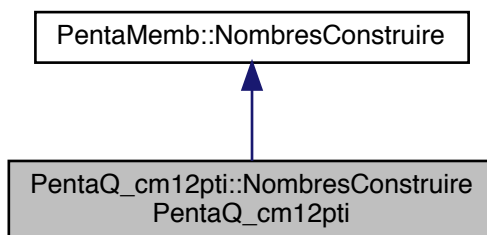
## 6.557 Référence de la classe

### PentaQ\_cm12pti::NombresConstruirePentaQ\_cm12pti

Grphe d'héritage de PentaQ\_cm12pti::NombresConstruirePentaQ\_cm12pti:



Grphe de collaboration de PentaQ\_cm12pti::NombresConstruirePentaQ\_cm12pti:



### Membres hérités additionnels

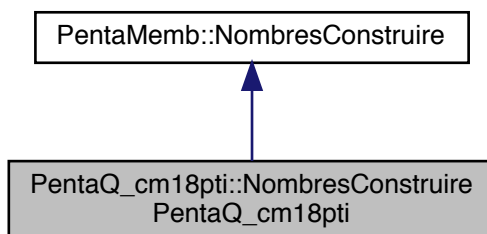
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm12pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm12pti.cc

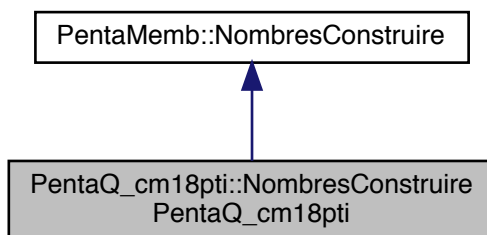


## 6.558 Référence de la classe PentaQ\_cm18pti::NombresConstruirePentaQ\_cm18pti

Grphe d'héritage de PentaQ\_cm18pti::NombresConstruirePentaQ\_cm18pti:



Grphe de collaboration de PentaQ\_cm18pti::NombresConstruirePentaQ\_cm18pti:



### Membres hérités additionnels

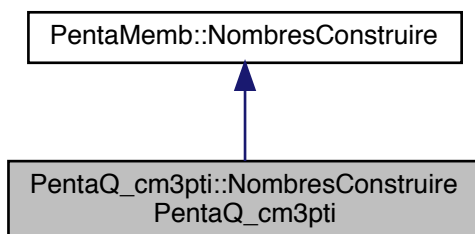
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm18pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm18pti.cc

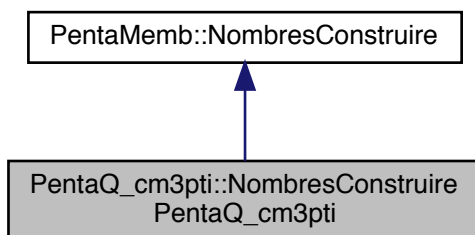
## 6.559 Référence de la classe

### PentaQ\_cm3pti::NombresConstruirePentaQ\_cm3pti

Grphe d'héritage de PentaQ\_cm3pti::NombresConstruirePentaQ\_cm3pti:



Grphe de collaboration de PentaQ\_cm3pti::NombresConstruirePentaQ\_cm3pti:



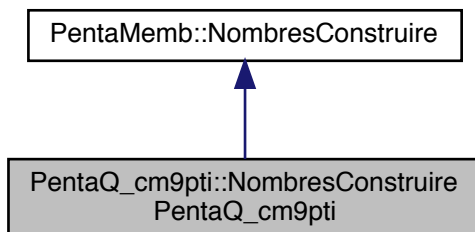
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

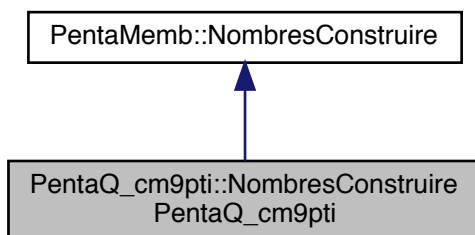
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm3pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm3pti.cc

## 6.560 Référence de la classe PentaQ\_cm9pti::NombresConstruirePentaQ\_cm9pti

Graphe d'héritage de PentaQ\_cm9pti::NombresConstruirePentaQ\_cm9pti:



Graphe de collaboration de PentaQ\_cm9pti::NombresConstruirePentaQ\_cm9pti:



### Membres hérités additionnels

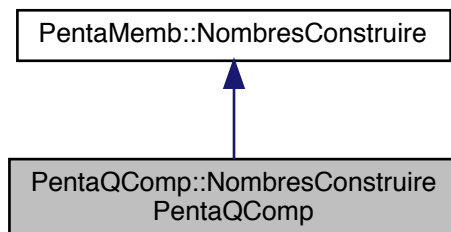
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm9pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm9pti.cc

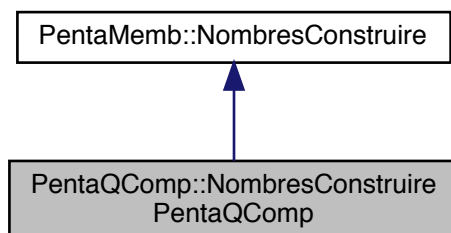
## 6.561 Référence de la classe

### PentaQComp::NombresConstruirePentaQComp

Grphe d'héritage de PentaQComp::NombresConstruirePentaQComp:



Grphe de collaboration de PentaQComp::NombresConstruirePentaQComp:



### Membres hérités additionnels

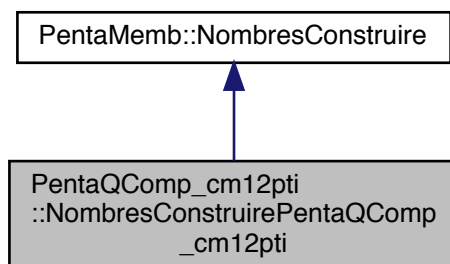
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp.cc

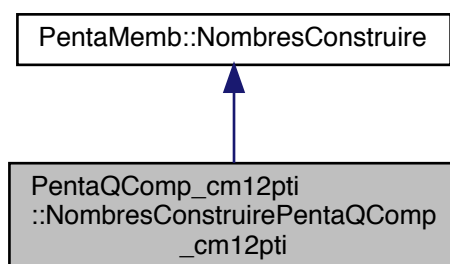
## 6.562 Référence de la classe

### PentaQComp\_cm12pti::NombresConstruirePentaQComp\_cm12pti

Graphe d'héritage de PentaQComp\_cm12pti::NombresConstruirePentaQComp\_cm12pti:



Graphe de collaboration de PentaQComp\_cm12pti::NombresConstruirePentaQComp\_cm12pti:



### Membres hérités additionnels

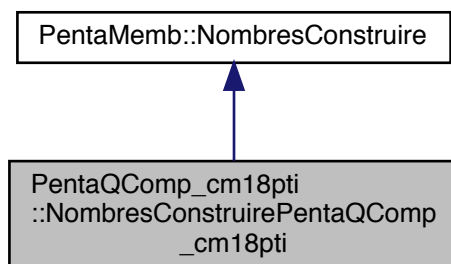
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm12pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm12pti.cc

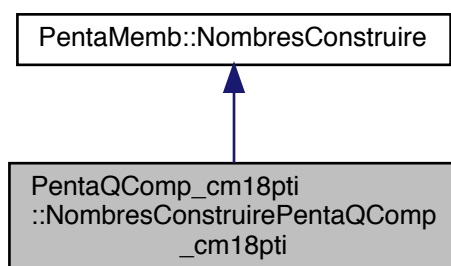
## 6.563 Référence de la classe

### PentaQComp\_cm18pti::NombresConstruirePentaQComp\_cm18pti

Graphe d'héritage de PentaQComp\_cm18pti::NombresConstruirePentaQComp\_cm18pti:



Graphe de collaboration de PentaQComp\_cm18pti::NombresConstruirePentaQComp\_cm18pti:



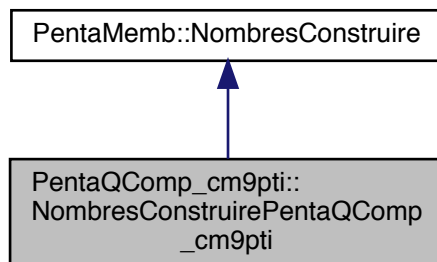
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

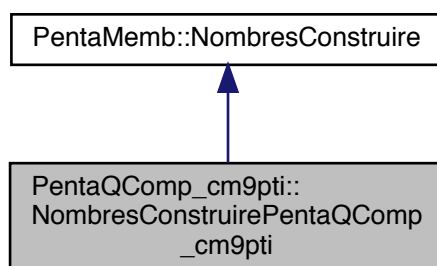
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm18pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm18pti.cc

## 6.564 Référence de la classe PentaQComp\_cm9pti::NombresConstruirePentaQComp\_cm9pti

Graphe d'héritage de PentaQComp\_cm9pti::NombresConstruirePentaQComp\_cm9pti:



Graphe de collaboration de PentaQComp\_cm9pti::NombresConstruirePentaQComp\_cm9pti:



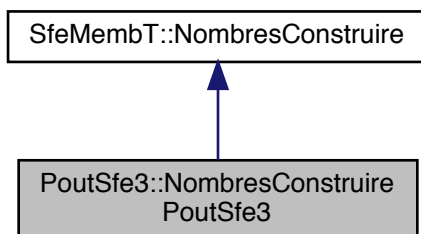
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

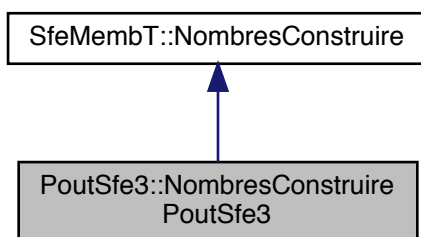
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm9pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm9pti.cc

## 6.565 Référence de la classe PoutSfe3::NombresConstruirePoutSfe3

Grphe d'héritage de PoutSfe3::NombresConstruirePoutSfe3:



Grphe de collaboration de PoutSfe3::NombresConstruirePoutSfe3:



### Membres hérités additionnels

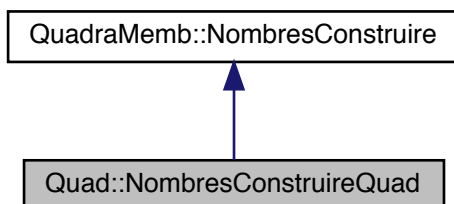
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/PoutSfe3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/PoutSfe3.cc

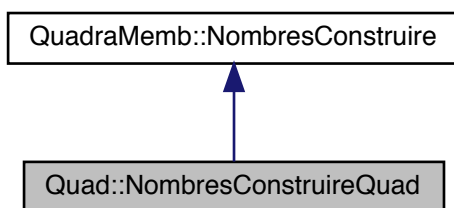


## 6.566 Référence de la classe Quad::NombresConstruireQuad

Grappe d'héritage de Quad::NombresConstruireQuad:



Grappe de collaboration de Quad::NombresConstruireQuad:



### Membres hérités additionnels

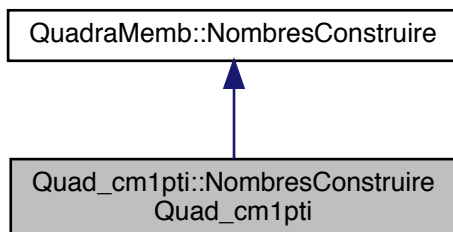
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad.cc

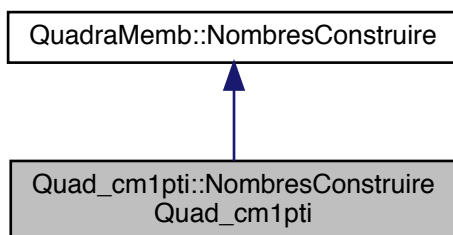
## 6.567 Référence de la classe

### Quad\_cm1pti::NombresConstruireQuad\_cm1pti

Graphe d'héritage de Quad\_cm1pti::NombresConstruireQuad\_cm1pti:



Graphe de collaboration de Quad\_cm1pti::NombresConstruireQuad\_cm1pti:



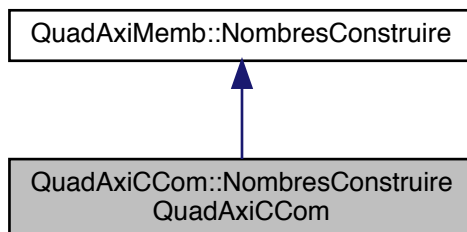
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

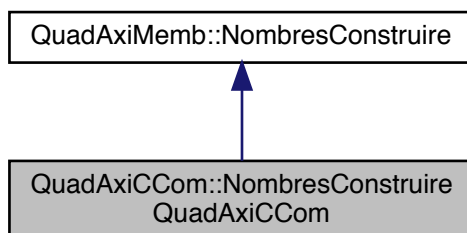
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad\_cm1pti.cc

## 6.568 Référence de la classe QuadAxiCCom::NombresConstruireQuadAxiCCom

Graphe d'héritage de QuadAxiCCom::NombresConstruireQuadAxiCCom:



Graphe de collaboration de QuadAxiCCom::NombresConstruireQuadAxiCCom:



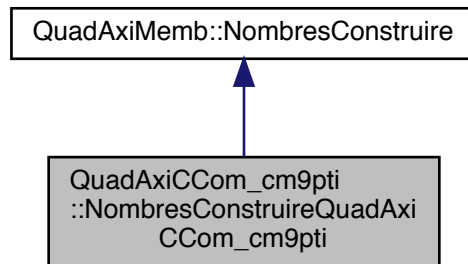
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

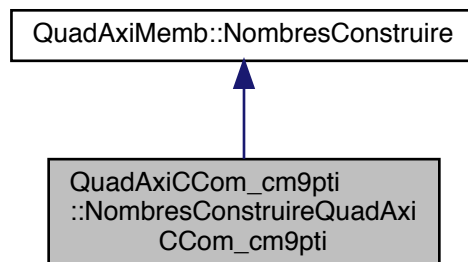
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom.cc

## 6.569 Référence de la classe QuadAxiCCom\_cm9pti::NombresConstruireQuadAxiCCom\_cm9pti

Graphe d'héritage de QuadAxiCCom\_cm9pti::NombresConstruireQuadAxiCCom\_cm9pti:



Graphe de collaboration de QuadAxiCCom\_cm9pti::NombresConstruireQuadAxiCCom\_cm9pti:



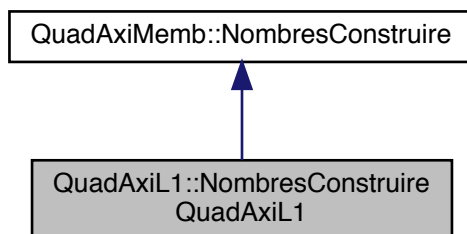
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

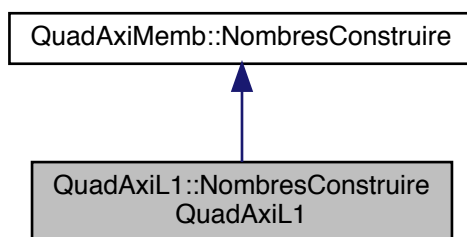
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom\_cm9pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom\_cm9pti.cc

## 6.570 Référence de la classe QuadAxiL1::NombresConstruireQuadAxiL1

Grphe d'héritage de QuadAxiL1::NombresConstruireQuadAxiL1:



Grphe de collaboration de QuadAxiL1::NombresConstruireQuadAxiL1:



### Membres hérités additionnels

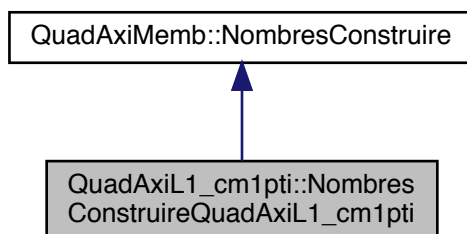
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1.cc

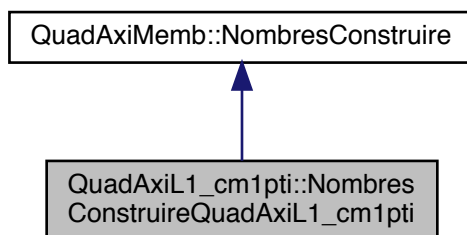
## 6.571 Référence de la classe

### QuadAxiL1\_cm1pti::NombresConstruireQuadAxiL1\_cm1pti

Grphe d'héritage de QuadAxiL1\_cm1pti::NombresConstruireQuadAxiL1\_cm1pti:



Grphe de collaboration de QuadAxiL1\_cm1pti::NombresConstruireQuadAxiL1\_cm1pti:



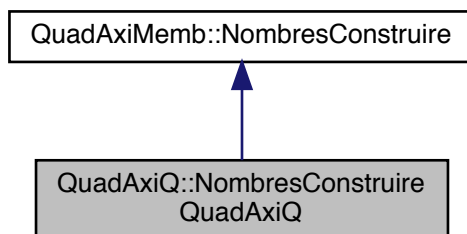
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

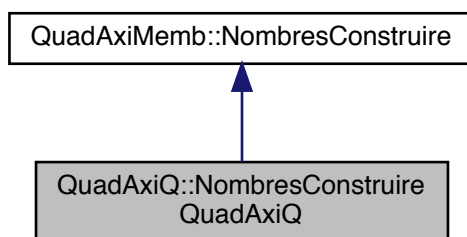
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1\_cm1pti.cc

## 6.572 Référence de la classe QuadAxiQ::NombresConstruireQuadAxiQ

Grphe d'héritage de QuadAxiQ::NombresConstruireQuadAxiQ:



Grphe de collaboration de QuadAxiQ::NombresConstruireQuadAxiQ:



### Membres hérités additionnels

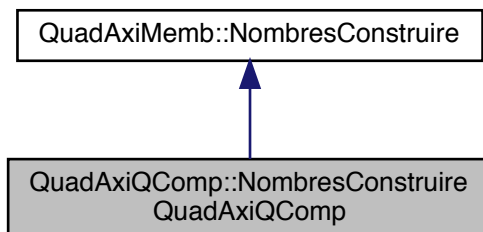
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQ.cc

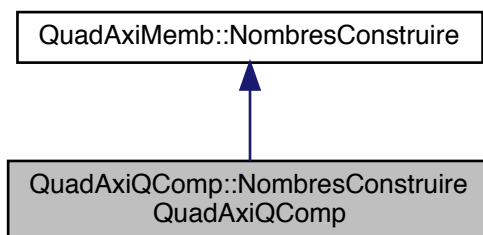
## 6.573 Référence de la classe

### QuadAxiQComp::NombresConstruireQuadAxiQComp

Graphe d'héritage de QuadAxiQComp::NombresConstruireQuadAxiQComp:



Graphe de collaboration de QuadAxiQComp::NombresConstruireQuadAxiQComp:



### Membres hérités additionnels

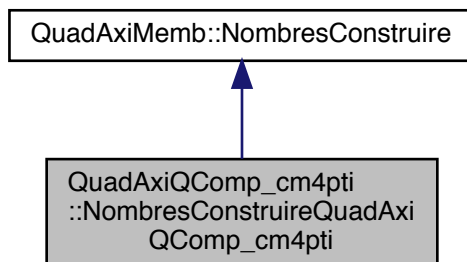
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQComp.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQComp.cc

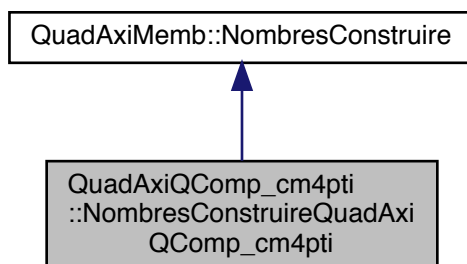


## 6.574 Référence de la classe QuadAxiQComp\_cm4pti::NombresConstruireQuadAxiQComp\_cm4pti

Graphe d'héritage de QuadAxiQComp\_cm4pti::NombresConstruireQuadAxiQComp\_cm4pti:



Graphe de collaboration de QuadAxiQComp\_cm4pti::NombresConstruireQuadAxiQComp\_cm4pti:



### Membres hérités additionnels

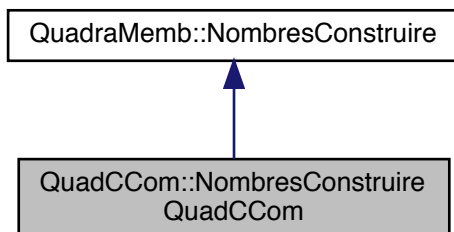
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQComp\_cm4pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQComp\_cm4pti.cc

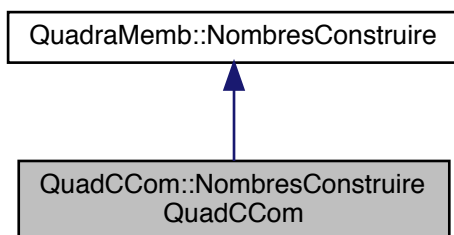
## 6.575 Référence de la classe

### QuadCCom::NombresConstruireQuadCCom

Graphe d'héritage de QuadCCom::NombresConstruireQuadCCom:



Graphe de collaboration de QuadCCom::NombresConstruireQuadCCom:



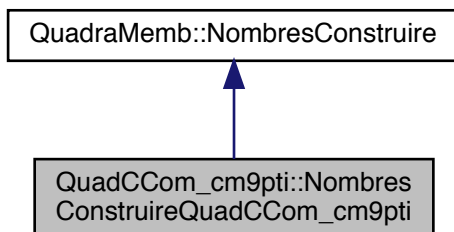
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

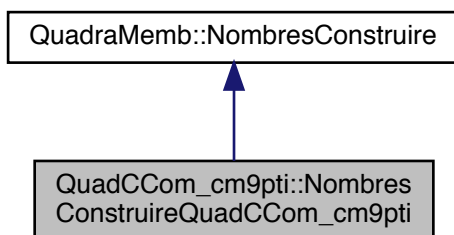
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom.cc

## 6.576 Référence de la classe QuadCCom\_cm9pti::NombresConstruireQuadCCom\_cm9pti

Graphe d'héritage de QuadCCom\_cm9pti::NombresConstruireQuadCCom\_cm9pti:



Graphe de collaboration de QuadCCom\_cm9pti::NombresConstruireQuadCCom\_cm9pti:



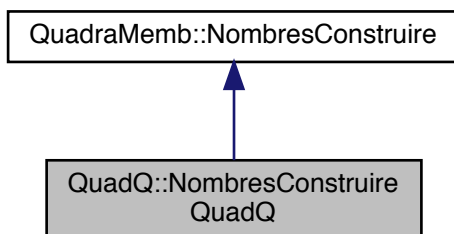
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

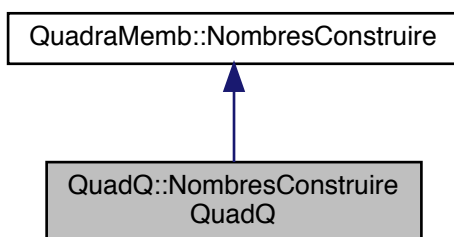
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom\_cm9pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom\_cm9pti.cc

## 6.577 Référence de la classe QuadQ::NombresConstruireQuadQ

Grphe d'héritage de QuadQ::NombresConstruireQuadQ:



Grphe de collaboration de QuadQ::NombresConstruireQuadQ:



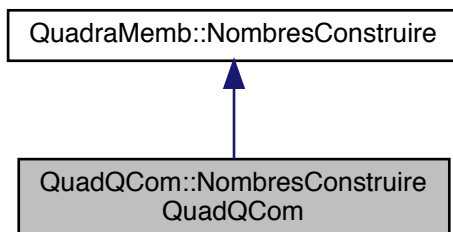
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

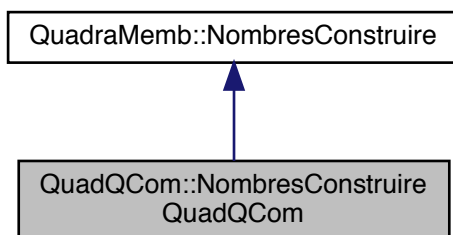
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQ.cc

## 6.578 Référence de la classe QuadQCom::NombresConstruireQuadQCom

Graphe d'héritage de QuadQCom::NombresConstruireQuadQCom:



Graphe de collaboration de QuadQCom::NombresConstruireQuadQCom:



### Membres hérités additionnels

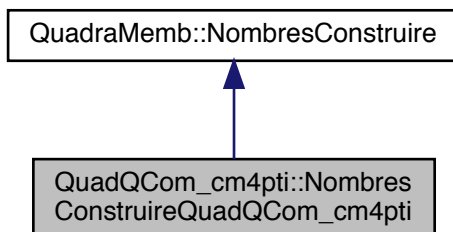
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom.cc

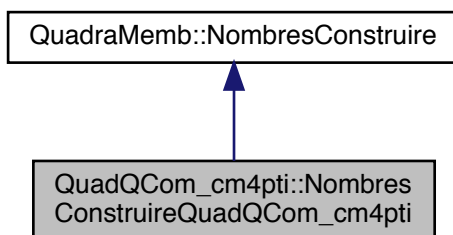
## 6.579 Référence de la classe

### QuadQCom\_cm4pti::NombresConstruireQuadQCom\_cm4pti

Grphe d'héritage de QuadQCom\_cm4pti::NombresConstruireQuadQCom\_cm4pti:



Grphe de collaboration de QuadQCom\_cm4pti::NombresConstruireQuadQCom\_cm4pti:



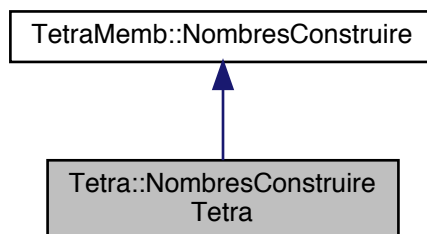
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

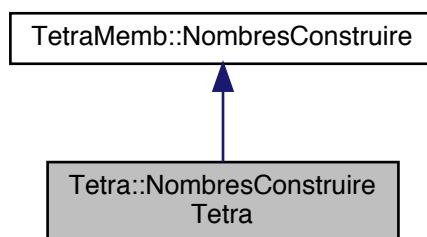
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom\_cm4pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom\_cm4pti.cc

## 6.580 Référence de la classe Tetra::NombresConstruireTetra

Grphe d'héritage de Tetra::NombresConstruireTetra:



Grphe de collaboration de Tetra::NombresConstruireTetra:



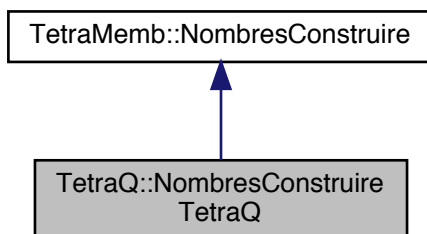
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

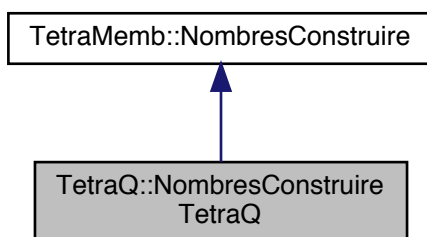
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/Tetra.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/Tetra.cc

## 6.581 Référence de la classe TetraQ::NombresConstruireTetraQ

Graphe d'héritage de TetraQ::NombresConstruireTetraQ:



Graphe de collaboration de TetraQ::NombresConstruireTetraQ:



### Membres hérités additionnels

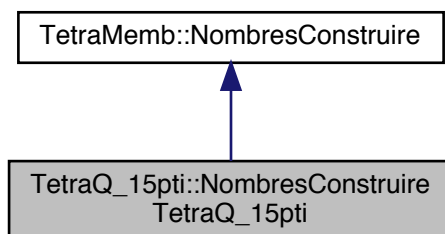
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ.cc

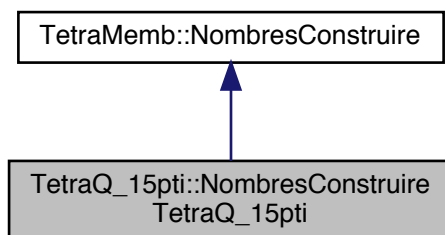


## 6.582 Référence de la classe TetraQ\_15pti::NombresConstruireTetraQ\_15pti

Graphe d'héritage de TetraQ\_15pti::NombresConstruireTetraQ\_15pti:



Graphe de collaboration de TetraQ\_15pti::NombresConstruireTetraQ\_15pti:



### Membres hérités supplémentaires

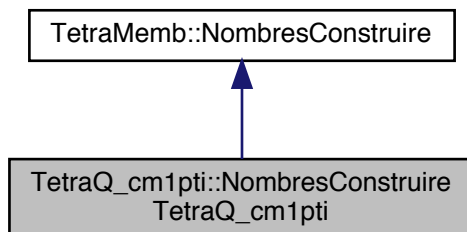
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm15pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm15pti.cc

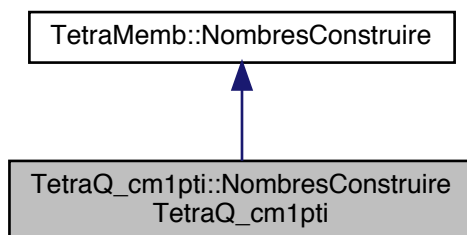
## 6.583 Référence de la classe

### TetraQ\_cm1pti::NombresConstruireTetraQ\_cm1pti

Grphe d'héritage de TetraQ\_cm1pti::NombresConstruireTetraQ\_cm1pti:



Grphe de collaboration de TetraQ\_cm1pti::NombresConstruireTetraQ\_cm1pti:



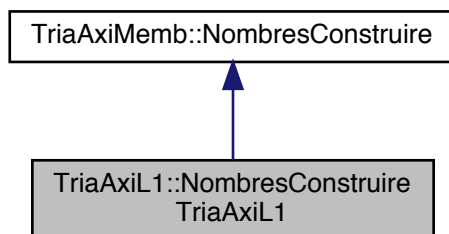
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

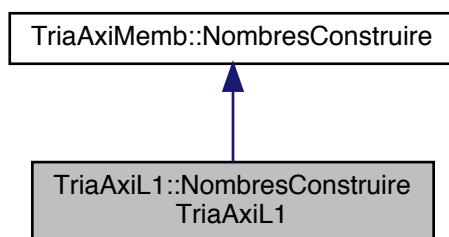
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm1pti.cc

## 6.584 Référence de la classe TriaAxiL1::NombresConstruireTriaAxiL1

Grphe d'héritage de TriaAxiL1::NombresConstruireTriaAxiL1:



Grphe de collaboration de TriaAxiL1::NombresConstruireTriaAxiL1:



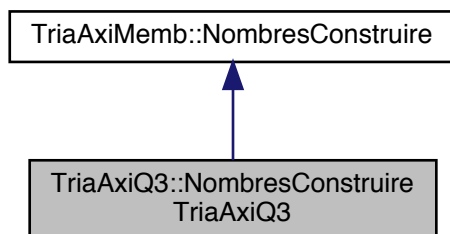
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

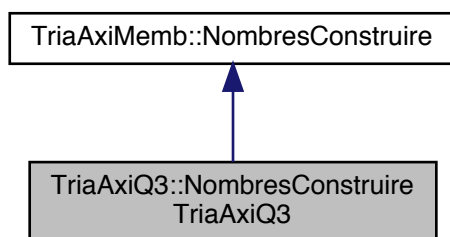
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiL1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiL1.cc

## 6.585 Référence de la classe TriaAxiQ3::NombresConstruireTriaAxiQ3

Grphe d'héritage de TriaAxiQ3::NombresConstruireTriaAxiQ3:



Grphe de collaboration de TriaAxiQ3::NombresConstruireTriaAxiQ3:



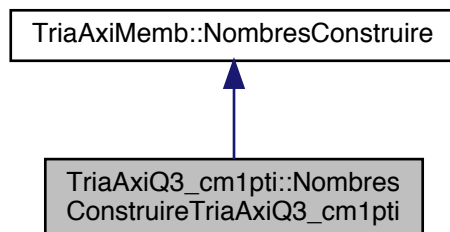
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

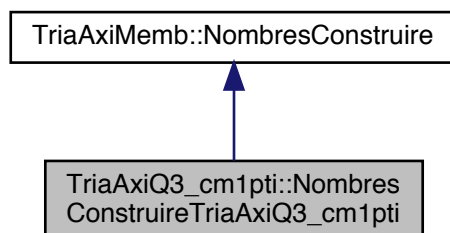
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3.cc

## 6.586 Référence de la classe TriaAxiQ3\_cm1pti::NombresConstruireTriaAxiQ3\_cm1pti

Graphe d'héritage de TriaAxiQ3\_cm1pti::NombresConstruireTriaAxiQ3\_cm1pti:



Graphe de collaboration de TriaAxiQ3\_cm1pti::NombresConstruireTriaAxiQ3\_cm1pti:



### Membres hérités additionnels

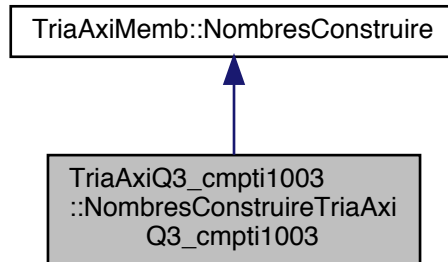
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_cm1pti.cc

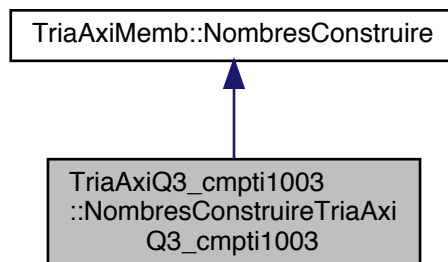
## 6.587 Référence de la classe

### TriaAxiQ3\_cmpti1003::NombresConstruireTriaAxiQ3\_cmpti1003

Graphe d'héritage de TriaAxiQ3\_cmpti1003::NombresConstruireTriaAxiQ3\_cmpti1003:



Graphe de collaboration de TriaAxiQ3\_cmpti1003::NombresConstruireTriaAxiQ3\_cmpti1003:



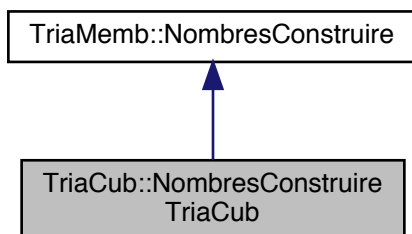
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

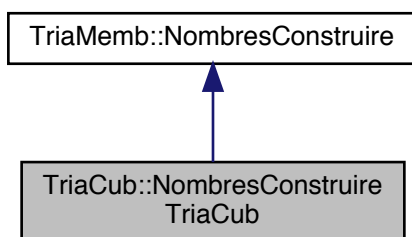
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_cmpti1003.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_cmpti1003.cc

## 6.588 Référence de la classe TriaCub::NombresConstruireTriaCub

Grphe d'héritage de TriaCub::NombresConstruireTriaCub:



Grphe de collaboration de TriaCub::NombresConstruireTriaCub:



### Membres hérités additionnels

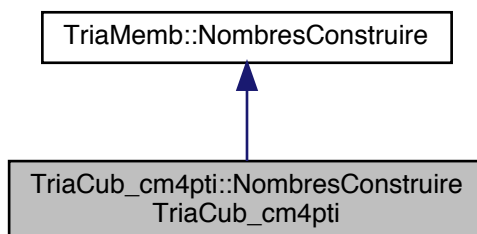
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub.cc

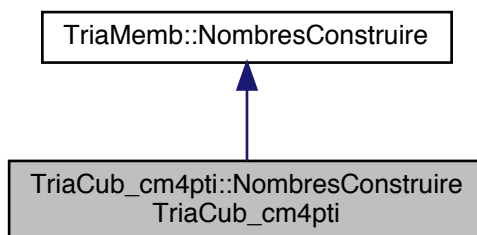
## 6.589 Référence de la classe

### TriaCub\_cm4pti::NombresConstruireTriaCub\_cm4pti

Grphe d'héritage de TriaCub\_cm4pti::NombresConstruireTriaCub\_cm4pti:



Grphe de collaboration de TriaCub\_cm4pti::NombresConstruireTriaCub\_cm4pti:



### Membres hérités additionnels

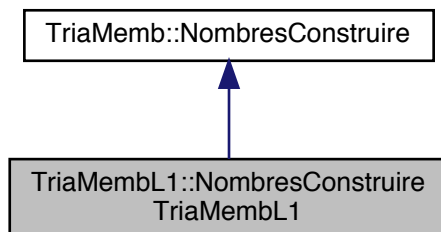
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub\_cm4pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub\_cm4pti.cc

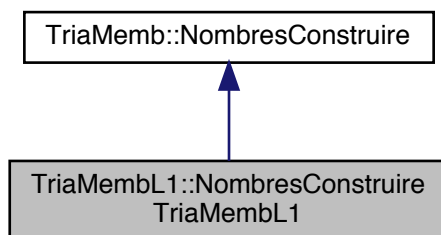


## 6.590 Référence de la classe TriaMembL1::NombresConstruireTriaMembL1

Grphe d'héritage de TriaMembL1::NombresConstruireTriaMembL1:



Grphe de collaboration de TriaMembL1::NombresConstruireTriaMembL1:



### Membres hérités additionnels

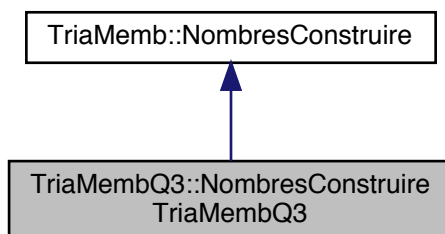
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembL1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembL1.cc

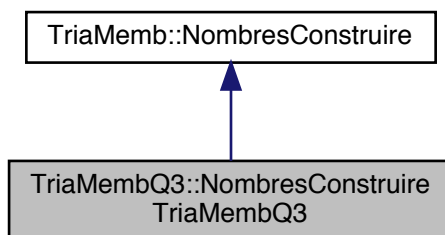
## 6.591 Référence de la classe

### TriaMembQ3::NombresConstruireTriaMembQ3

Grphe d'héritage de TriaMembQ3::NombresConstruireTriaMembQ3:



Grphe de collaboration de TriaMembQ3::NombresConstruireTriaMembQ3:



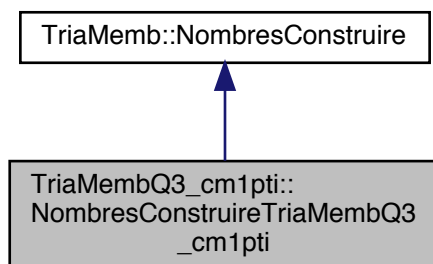
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

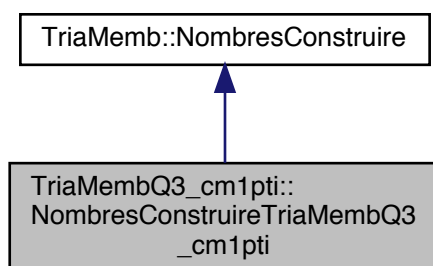
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3.cc

## 6.592 Référence de la classe TriaMembQ3\_cm1pti::NombresConstruireTriaMembQ3\_cm1pti

Graphe d'héritage de TriaMembQ3\_cm1pti::NombresConstruireTriaMembQ3\_cm1pti:



Graphe de collaboration de TriaMembQ3\_cm1pti::NombresConstruireTriaMembQ3\_cm1pti:



### Membres hérités supplémentaires

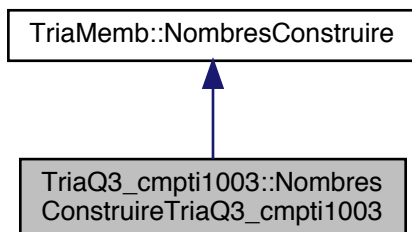
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3\_cm1pti.cc

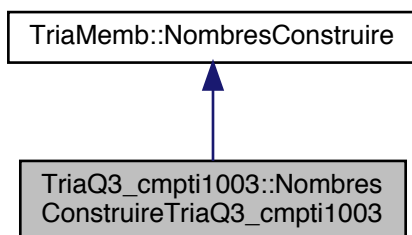
## 6.593 Référence de la classe

### TriaQ3\_cmpti1003::NombresConstruireTriaQ3\_cmpti1003

Grphe d'héritage de TriaQ3\_cmpti1003::NombresConstruireTriaQ3\_cmpti1003:



Grphe de collaboration de TriaQ3\_cmpti1003::NombresConstruireTriaQ3\_cmpti1003:



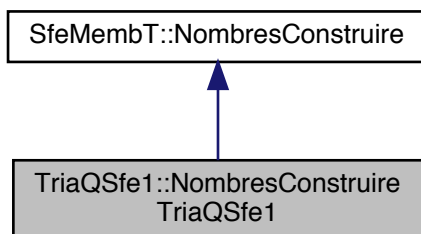
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

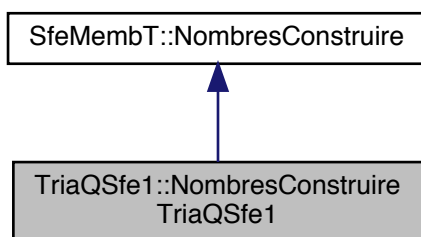
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaQ3\_cmpti1003.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaQ3\_cmpti1003.cc

## 6.594 Référence de la classe TriaQSfe1::NombresConstruireTriaQSfe1

Grphe d'héritage de TriaQSfe1::NombresConstruireTriaQSfe1:



Grphe de collaboration de TriaQSfe1::NombresConstruireTriaQSfe1:



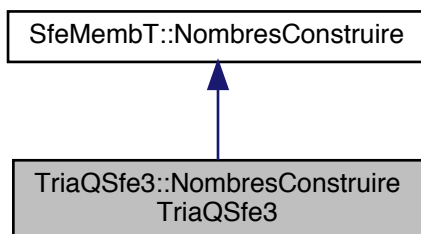
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

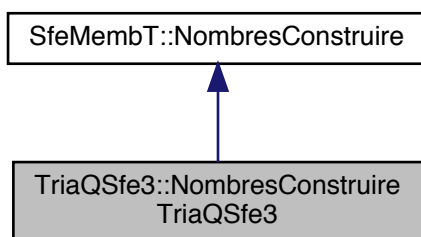
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe1.cc

## 6.595 Référence de la classe TriaQSfe3::NombresConstruireTriaQSfe3

Grphe d'héritage de TriaQSfe3::NombresConstruireTriaQSfe3:



Grphe de collaboration de TriaQSfe3::NombresConstruireTriaQSfe3:



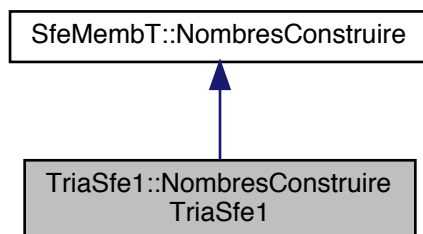
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

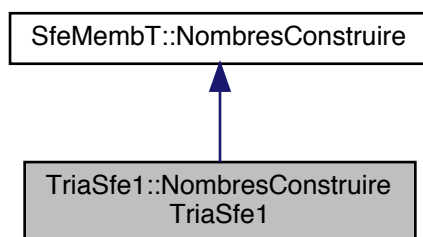
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe3.cc

## 6.596 Référence de la classe TriaSfe1::NombresConstruireTriaSfe1

Grphe d'héritage de TriaSfe1::NombresConstruireTriaSfe1 :



Grphe de collaboration de TriaSfe1::NombresConstruireTriaSfe1 :



### Membres hérités additionnels

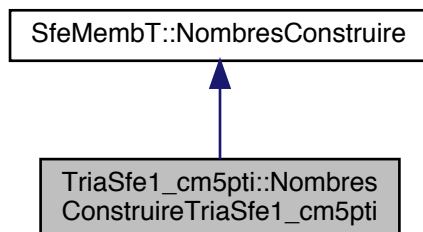
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1.cc

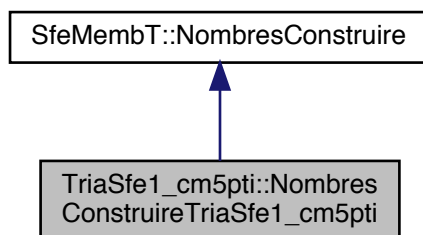
## 6.597 Référence de la classe

### TriaSfe1\_cm5pti::NombresConstruireTriaSfe1\_cm5pti

Grphe d'héritage de TriaSfe1\_cm5pti::NombresConstruireTriaSfe1\_cm5pti:



Grphe de collaboration de TriaSfe1\_cm5pti::NombresConstruireTriaSfe1\_cm5pti:



### Membres hérités additionnels

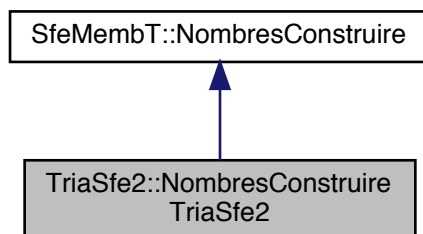
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1\_cm5pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1\_cm5pti.cc

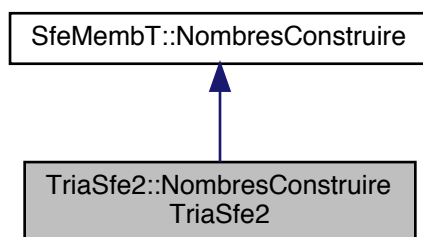


## 6.598 Référence de la classe TriaSfe2::NombresConstruireTriaSfe2

Grphe d'héritage de TriaSfe2::NombresConstruireTriaSfe2:



Grphe de collaboration de TriaSfe2::NombresConstruireTriaSfe2:



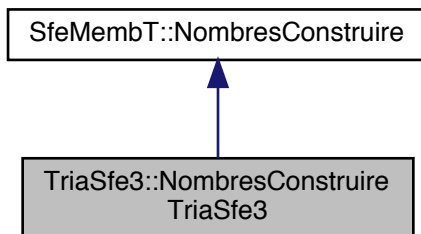
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

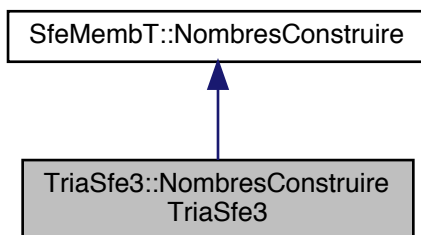
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe2.cc

## 6.599 Référence de la classe TriaSfe3::NombresConstruireTriaSfe3

Grphe d'héritage de TriaSfe3::NombresConstruireTriaSfe3:



Grphe de collaboration de TriaSfe3::NombresConstruireTriaSfe3:



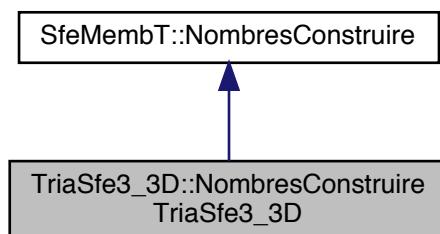
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

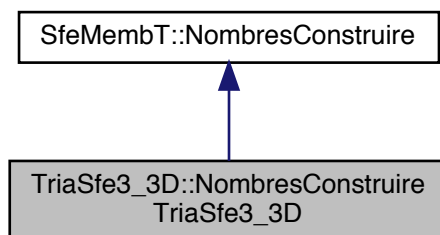
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3.cc

## 6.600 Référence de la classe TriaSfe3\_3D::NombresConstruireTriaSfe3\_3D

Graphe d'héritage de TriaSfe3\_3D::NombresConstruireTriaSfe3\_3D:



Graphe de collaboration de TriaSfe3\_3D::NombresConstruireTriaSfe3\_3D:



### Membres hérités additionnels

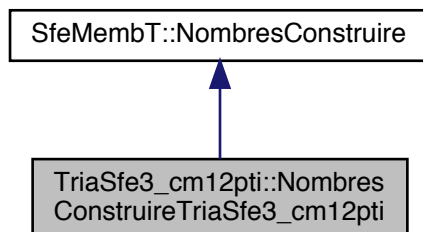
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_3D.cc

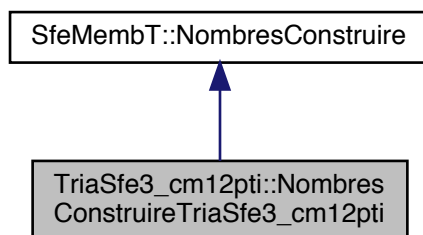
## 6.601 Référence de la classe

### TriaSfe3\_cm12pti::NombresConstruireTriaSfe3\_cm12pti

Graphe d'héritage de TriaSfe3\_cm12pti::NombresConstruireTriaSfe3\_cm12pti:



Graphe de collaboration de TriaSfe3\_cm12pti::NombresConstruireTriaSfe3\_cm12pti:



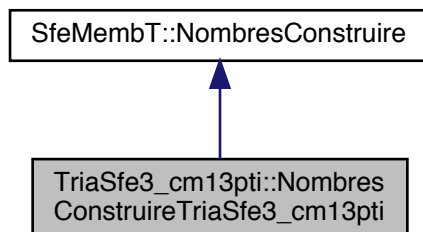
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

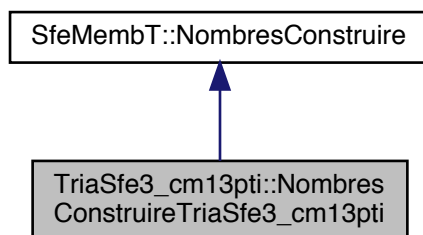
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm12pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm12pti.cc

## 6.602 Référence de la classe TriaSfe3\_cm13pti::NombresConstruireTriaSfe3\_cm13pti

Graphe d'héritage de TriaSfe3\_cm13pti::NombresConstruireTriaSfe3\_cm13pti:



Graphe de collaboration de TriaSfe3\_cm13pti::NombresConstruireTriaSfe3\_cm13pti:



### Membres hérités additionnels

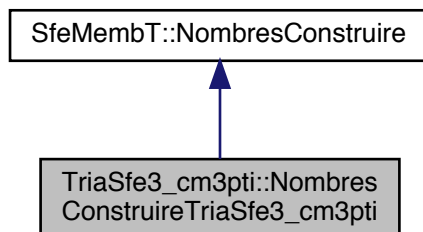
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm13pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm13pti.cc

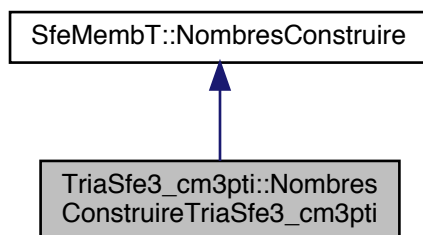
## 6.603 Référence de la classe

### TriaSfe3\_cm3pti::NombresConstruireTriaSfe3\_cm3pti

Grphe d'héritage de TriaSfe3\_cm3pti::NombresConstruireTriaSfe3\_cm3pti:



Grphe de collaboration de TriaSfe3\_cm3pti::NombresConstruireTriaSfe3\_cm3pti:



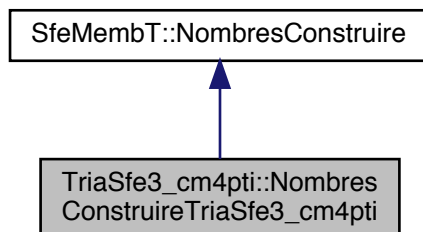
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

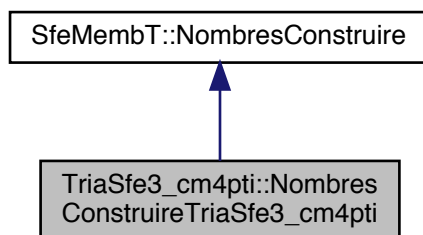
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm3pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm3pti.cc

## 6.604 Référence de la classe TriaSfe3\_cm4pti::NombresConstruireTriaSfe3\_cm4pti

Graphe d'héritage de TriaSfe3\_cm4pti::NombresConstruireTriaSfe3\_cm4pti:



Graphe de collaboration de TriaSfe3\_cm4pti::NombresConstruireTriaSfe3\_cm4pti:



### Membres hérités supplémentaires

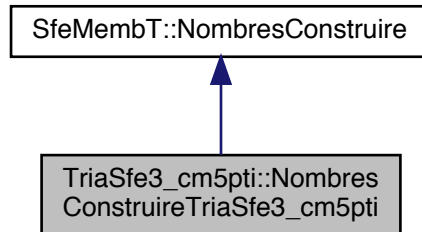
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm4pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm4pti.cc

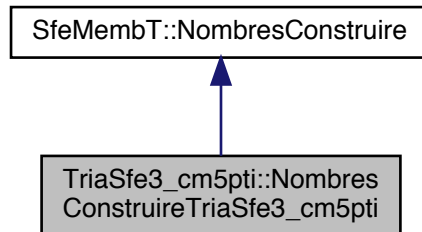
## 6.605 Référence de la classe

### TriaSfe3\_cm5pti::NombresConstruireTriaSfe3\_cm5pti

Grphe d'héritage de TriaSfe3\_cm5pti::NombresConstruireTriaSfe3\_cm5pti:



Grphe de collaboration de TriaSfe3\_cm5pti::NombresConstruireTriaSfe3\_cm5pti:



### Membres hérités additionnels

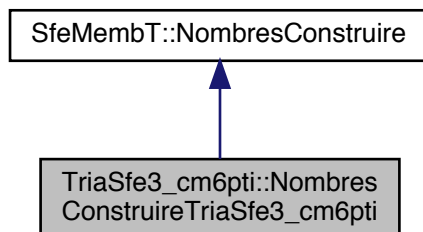
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm5pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm5pti.cc

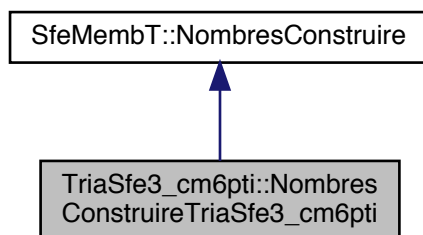


## 6.606 Référence de la classe TriaSfe3\_cm6pti::NombresConstruireTriaSfe3\_cm6pti

Graphe d'héritage de TriaSfe3\_cm6pti::NombresConstruireTriaSfe3\_cm6pti:



Graphe de collaboration de TriaSfe3\_cm6pti::NombresConstruireTriaSfe3\_cm6pti:



### Membres hérités supplémentaires

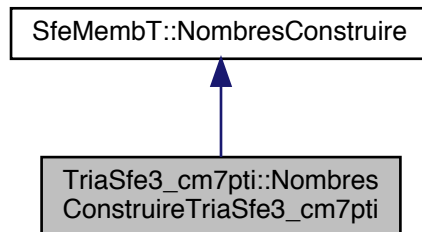
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm6pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm6pti.cc

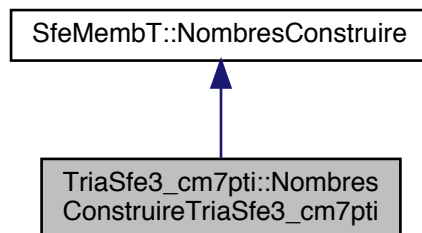
## 6.607 Référence de la classe

### TriaSfe3\_cm7pti::NombresConstruireTriaSfe3\_cm7pti

Graphe d'héritage de TriaSfe3\_cm7pti::NombresConstruireTriaSfe3\_cm7pti:



Graphe de collaboration de TriaSfe3\_cm7pti::NombresConstruireTriaSfe3\_cm7pti:



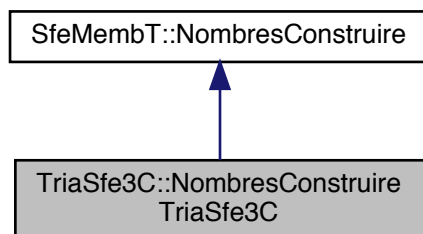
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

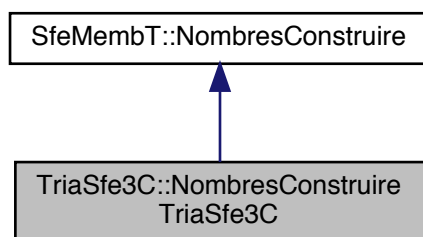
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm7pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm7pti.cc

## 6.608 Référence de la classe TriaSfe3C::NombresConstruireTriaSfe3C

Grphe d'héritage de TriaSfe3C::NombresConstruireTriaSfe3C:



Grphe de collaboration de TriaSfe3C::NombresConstruireTriaSfe3C:



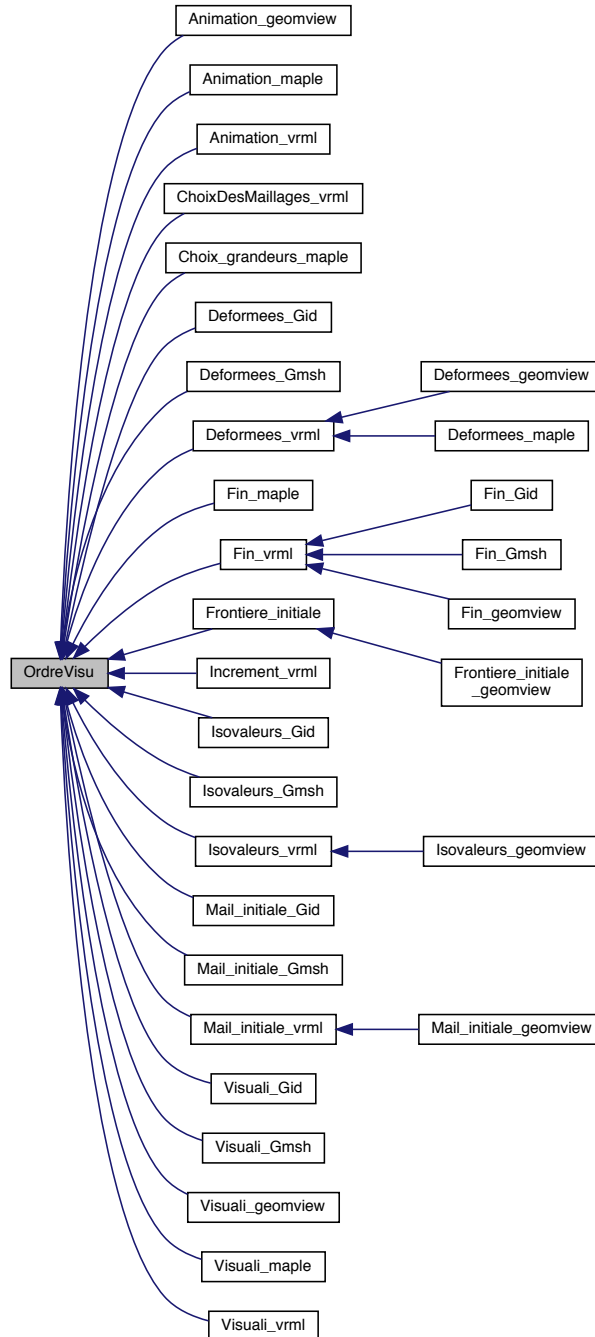
### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3C.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3C.cc

## 6.609 Référence de la classe OrdreVisu

Graphe d'héritage de OrdreVisu:



### Classes

— class [Deuxentiers](#)

### Types publics

— enum `EnumTypeIncre` { `INCRE_0 = 0` , `PREMIER_INCRE` , `INCRE_COURANT` , `DERNIER_INCRE` }

## Fonctions membres publiques

- string **Nom\_TypelIncre** (const `OrdreVisu::EnumTypeIncre` id\_TypelIncre)
- `EnumTypeIncre` **Id\_nom\_TypelIncre** (const string nom\_TypelIncre)
- **OrdreVisu** (const string &comment\_som, const string &explication, const string &ordre)
- **OrdreVisu** (const [OrdreVisu](#) &ord)
- void **Affiche\_ordre** ()
- void **Explication\_detailee** ()
- virtual void **Initialisation** ([ParaGlob](#) \*, [LesMaillages](#) \*, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, `EnumTypeIncre` type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob, bool fil\_calcul)
- virtual void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &tab\_mail, [LesMaillages](#) \*, bool unseul\_incre, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &entreePrinc, `OrdreVisu::EnumTypeIncre` type\_incre, int incre, bool animation, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVec←Glob)
- bool **OrdreVrai** (const string &ord)
- virtual void **ChoixOrdre** ()
- bool **Actif** () const
- void **Inactive\_ordre** ()
- virtual void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- virtual void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

## Fonctions membres protégées

- void **Propre\_liste** (list< int > &lis)
- void **Mise\_zero\_coordo** ()
- void **Lect\_para\_OrdreVisu\_general** ([UtilLecture](#) &entreePrinc)
- void **Ecrit\_para\_OrdreVisu\_general** ([UtilLecture](#) &entreePrinc)
- void **changeLibelle** (const string &comment\_som, const string &explication, const string &ord)

## Attributs protégés

- string **commentaire\_sommaire**
- string **explication\_detail**
- string **ordre**
- bool **actif**

## Attributs protégés statiques

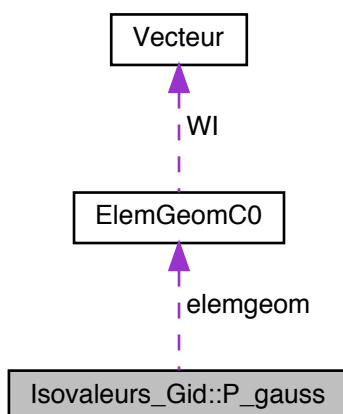
- static list< string > **list\_nom\_coordinate**
- static list< list< [Coordonnee](#) > > **list\_coordinate**
- static string **nom\_base\_couleur** = "base\_couleur"

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/OrdreVisu.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/OrdreVisu.cc

## 6.610 Référence de la classe Isovaleurs\_Gid::P\_gauss

Graphe de collaboration de Isovaleurs\_Gid::P\_gauss:



### Fonctions membres publiques

- **P\_gauss** (const ElemGeomC0 \*eg, string nom, Enum\_ddl en, const int &num\_m)
- **P\_gauss** (const P\_gauss &p)
- bool **operator==** (const P\_gauss &pg) const
- bool **operator!=** (const P\_gauss &pg) const

### Attributs publics

- const ElemGeomC0 \* **elemgeom**
- string **nom\_groupe\_pt\_integ**
- Enum\_ddl **enu**
- int **num\_maill**

### Amis

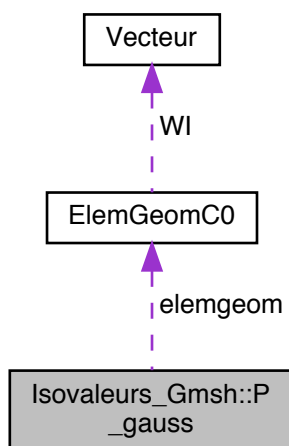
- ostream & **operator<<** (ostream &sort, const P\_gauss &a)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Isovaleurs\_Gid.h

## 6.611 Référence de la classe Isovaleurs\_Gmsh::P\_gauss

Grphe de collaboration de Isovaleurs\_Gmsh::P\_gauss:



### Fonctions membres publiques

- **P\_gauss** (const ElemGeomC0 \*eg, string nom, Enum\_ddl en, const int &num\_m)
- **P\_gauss** (const P\_gauss &p)
- bool **operator==** (const P\_gauss &pg) const
- bool **operator!=** (const P\_gauss &pg) const

### Attributs publics

- const ElemGeomC0 \* **elemgeom**
- string **nom\_groupe\_pt\_integ**
- Enum\_ddl **enu**
- int **num\_maill**

### Amis

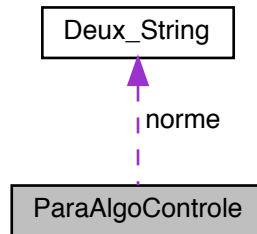
- ostream & **operator<<** (ostream &sort, const P\_gauss &a)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Isovaleurs\_Gmsh.h

## 6.612 Référence de la classe ParaAlgoControle

Grphe de collaboration de ParaAlgoControle:



### Fonctions membres publiques

- **ParaAlgoControle** (const [ParaAlgoControle](#) &p)
- [ParaAlgoControle](#) & **operator=** (const [ParaAlgoControle](#) &p)
- void **Lecture\_paraAlgoControle** ([UtilLecture](#) &entreePrinc)
- void **Affiche** () const
- void **Info\_commande\_ParaAlgoControle** ([UtilLecture](#) &lec)
- void **Ecriture\_base\_info\_Para** (ofstream &sort, const int cas) const
- void **Lecture\_base\_info\_Para** (ifstream &ent, const int cas)
- bool **Sauvegarde** () const
- bool **SauvegardeAutorisee** (int incre, const double &temps\_derniere\_sauvegarde, bool dernier\_calcul) const
- int **Iterations** () const
- double **Precision** () const
- double **Prectemps** () const
- [Deux\\_String](#) **Norme** () const
- bool **Cinematique** () const
- bool **Conv\_forcee** () const
- double **Multiplicateur** () const
- double **Deltat** () const
- double **Deltatmaxi** () const
- double **Deltatmini** () const
- double **Tempsfin** () const
- bool **DeltatOuDeltatmaxDependantTempsCritique** () const
- double **Coefficient\_pas\_critique\_deltat** () const
- double **Coefficient\_pas\_critique\_deltatmaxi** () const
- int **Maxincre** () const
- int **Max\_essai\_incre** () const
- int **Restart** () const
- double **Max\_puissance** () const
- bool **Line\_search** () const
- bool **Var\_charge\_externe** () const
- bool **Var\_jacobien** () const
- bool **Var\_D** () const
- const VariablesTemps & **Variables\_de\_temps\_specifiques\_algo** ()
- [Enum\\_matrice](#) **Type\_Matrice** () const
- bool **Symetrie\_matrice** () const
- [Enum\\_type\\_resolution\\_matri](#) **Type\_resolution** () const
- [Enum\\_preconditionnement](#) **Type\_preconditionnement** () const
- int **Nb\_iter\_nondirecte** () const
- double **Tolerance** () const
- int **Nb\_vect\_restart** () const



- const list< [Enum\\_matrice](#) > & **Type\_matrice\_secondaire** () const
- const list< [Enum\\_type\\_resolution\\_matri](#) > & **Type\_resolution\_secondaire** () const
- const list< [Enum\\_preconditionnement](#) > & **Type\_preconditionnement\_secondaire** () const
- const list< int > & **Nb\_iter\_nondirecte\_secondaire** () const
- const list< double > & **Tolerance\_secondaire** () const
- const list< int > & **NB\_vect\_restart\_secondaire** () const
- int **Optimisation\_pointeur\_assemblage** () const
- [EnumTypePilotage](#) **TypeDePilotage** () const
- double **Facteur\_diminution** () const
- double **Facteur\_augmentation** () const
- double **Fact\_dim\_en\_mauvaiseConv** () const
- int **Nb\_bonne\_convergence** () const
- int **Nb\_iter\_pour\_bonne\_convergence** () const
- int **Nb\_iter\_pour\_mauvaise\_convergence** () const
- int **Init\_comp\_tangent\_simple** () const
- double **SurSousRelaxation** () const
- double **NormeMax\_increment** () const
- double **NormeMax\_X\_increment** () const
- double **NormeMax\_V\_increment** () const
- double **VarMaxiDdl** () const
- double **VarMiniDdl** () const
- int **NbCycleControleResidu** () const
- int **PlageControleResidu** () const
- double **InilncreAvecDeltaDdlPrec** () const
- int **JacobienNegatif** () const
- double **MaxiVarJacobien** () const
- int **Cas\_fctnD\_charge** () const
- [Enum\\_calcul\\_masse](#) **Type\_calcul\_masse** () const
- bool **Limit\_temps\_stable** () const
- int **Amort\_visco\_artificielle** () const
- void **Change\_amort\_visco\_artificielle** (int val\_amort)
- double **Visco\_artificielle** () const
- double **CoefRayleighMasse** () const
- double **CoefRayleighRaideur** () const
- double **BorneMaxiSurCfonctionDeLaMasse** () const
- int **BulkViscosity** () const
- [DeuxDoubles](#) **CoefsBulk** () const
- int **Freq\_affich\_incre** () const
- int **freq\_affich\_iter** () const
- bool **Sortie\_fil\_calcul** () const
- bool **SauvegardeFilCalculAutorisee** (int incre, const double &temps\_derniere\_sauvegarde, bool dernier↔\_calcul) const
- bool **Vrai\_commande\_sortie** (int icharge, const double &temps\_derniere\_sauvegarde) const
- int **Nb\_diggit\_double\_calcul** () const
- int **Nb\_diggit\_double\_graphique** () const
- int **Nb\_diggit\_double\_ecran** () const
- double **Precision\_point\_interne\_debut** () const
- double **FacPourRayonAccostage** () const
- double **DistanceMaxiAuPtProjete** () const
- double **Extra\_boite\_prelocalisation** () const
- double **Rapport\_Extra\_boite\_mini\_prelocalisation** () const
- double **Ajout\_extra\_boite\_prelocalisation** () const
- int **ContactType** () const
- string **Fct\_nD\_bascul\_contact\_type\_4** () const
- double **PenalisationPenetrationContact** () const
- string **Fct\_nD\_penalisationPenetration** () const
- int **TypeCalculPenalisationPenetration** () const
- double **Penetration\_contact\_maxi** () const
- string **Fct\_nD\_penetration\_contact\_maxi** () const
- double **Penetration\_borne\_regularisation** () const
- string **Fct\_nD\_penetration\_borne\_regularisation** () const
- double **Force\_contact\_noeud\_maxi** () const
- string **Fct\_nD\_force\_contact\_noeud\_maxi** () const
- double **PenalisationTangentielleContact** () const
- string **Fct\_nD\_penalisationTangentielle** () const

- int **TypeCalculPenalisationTangentielle** () const
- double **Tangentielle\_contact\_maxi** () const
- string **Fct\_nD\_tangentielle\_contact\_maxi** () const
- double **Tangentielle\_borne\_regularisation** () const
- string **Fct\_nD\_tangentielle\_borne\_regularisation** () const
- double **Force\_tangentielle\_noeud\_maxi** () const
- string **Fct\_nD\_force\_tangentielle\_noeud\_maxi** () const
- double **Precision\_pt\_sur\_front** () const
- int **Nb\_boucle\_newton\_position\_sur\_frontiere** () const
- int **NbDecolAutorise** () const
- int **TypeDeDecolement** () const
- int **Nb\_moy\_glissant** () const
- int **Niveau\_commentaire\_contact** () const
- int **Optimisation\_numerotation** () const
- int **NblncrCalEnergie** () const
- bool **AfficheIncrEnergie** () const
- double **PointInterneDeltaThetaiMaxi** () const
- double **PointInternePrecThetaiInterne** () const
- int **PointInterneNbBoucleSurDeltaThetai** () const
- int **PointInterneNbExterne** () const
- bool **CalVolTotalEntreSurfaceEtPlansRef** () const
- double **Ratio\_maxi\_jacoMembrane\_jacoPti** () const
- [Enum\\_type\\_resolution\\_matri](#) **Type\_calnum\_inversion\_metrique** () const
- int **Modif\_Deltat** (double nouveau\_temps)
- bool **Modif\_Detat\_dans\_borne** (double &nouveau\_temps)
- void **Modif\_Temps** (double nouveau\_temps)
- bool **Modif\_Deltat\_DeltatMaxi** (double temps\_critique, double temps\_critiqueDFC)
- void **Modif\_temps** (const double &tem, const double &deltatem)
- void **Modif\_Var\_D** (bool va\_D)
- void **Modif\_Var\_jacobien** (bool va\_jacobien)
- void **ChangeSortieEquilibreGlobal** (bool val)
- bool **EtatSortieEquilibreGlobal** () const
- void **ChangeSortieEtatActuelDansBI** (bool val)
- bool **EtatSortieEtatActuelDansBI** () const
- void **ChangeSortieEtatActuelDansCVisu** (bool val)
- bool **EtatSortieEtatActuelDansCVisu** () const
- std::vector< double > & **Mise\_a\_jour\_Grandeurs\_globales** ()

## Fonctions membres publiques statiques

- static double **Coef\_defaut\_pa\_critique** ()
- static const VariablesTemps & **Variables\_de\_temps** ()

## Attributs protégés

- double **sauvegarde**
- int **cas\_de\_sauvegarde**
- int **iterations**
- double **precision**
- [Deux\\_String](#) **norme**
- bool **cinematique**
- bool **conv\_forcee**
- double **multiplicateur**
- VariablesTemps **tempo\_specifique\_algo**
- bool **force\_deltat**
- double **coef\_pas\_critique\_deltat**
- double **coef\_pas\_critique\_deltatmaxi**
- double **coef\_pas\_critique\_deltatmini**
- bool **typeDFC\_pas\_critique\_deltat**
- bool **typeDFC\_pas\_critique\_deltatmaxi**
- bool **typeDFC\_pas\_critique\_deltatmini**
- int **maxincre**
- int **max\_essai\_incre**
- int **restart**

- double **max\_puissance**
- bool **line\_search**
- bool **var\_charge\_extern**
- bool **var\_jacobien**
- bool **var\_D**
- bool **symetrie\_matrice**
- **Enum\_matrice** **type\_matrice**
- **Enum\_type\_resolution\_matri** **type\_resolution**
- **Enum\_preconditionnement** **type\_preconditionnement**
- int **nb\_iter\_nondirecte**
- double **tolerance**
- int **nb\_vect\_restart**
- list< **Enum\_matrice** > **type\_matrice\_secondaire**
- list< **Enum\_type\_resolution\_matri** > **type\_resolution\_secondaire**
- list< **Enum\_preconditionnement** > **type\_preconditionnement\_secondaire**
- list< int > **nb\_iter\_nondirecte\_secondaire**
- list< double > **tolerance\_secondaire**
- list< int > **nb\_vect\_restart\_secondaire**
- int **opti\_pointeur\_assemblage**
- **EnumTypePilotage** **type\_de\_pilotage**
- double **facteur\_diminution**
- double **facteur\_augmentation**
- double **fact\_dim\_en\_mauvaiseConv**
- int **nb\_bonne\_convergence**
- int **nb\_iter\_pour\_bonne\_convergence**
- int **nb\_iter\_pour\_mauvaise\_convergence**
- double **varMiniDdl**
- double **varMaxiDdl**
- int **nbCycleControleResidu**
- int **plageControleResidu**
- int **init\_comp\_tangent\_simple**
- double **sur\_sous\_relaxation**
- double **norme\_incre\_max**
- double **norme\_incre\_X\_max**
- double **norme\_incre\_V\_max**
- double **initIncreAvecDeltaDdlPrec**
- int **jabobien\_negatif**
- double **var\_maxi\_jacobien**
- int **cas\_fctnD\_charge**
- **Enum\_calcul\_masse** **type\_calcul\_masse**
- bool **limitation\_temps\_maxi\_stable**
- int **amort\_visco\_arti**
- double **visco\_arti**
- double **maxi\_C\_en\_fonction\_M**
- double **coef\_rayleigh\_masse**
- double **coef\_rayleigh\_raideur**
- int **bulk\_viscosity**
- double **c\_Tracebulk**
- double **c\_Trace2bulk**
- int **frequence\_affichage\_increment**
- int **frequence\_affichage\_iteration**
- double **sortie\_fil\_calcul**
- int **cas\_de\_sortie\_fil\_calcul**
- int **nb\_diggit\_double\_calcul**
- int **nb\_diggit\_double\_graphique**
- int **nb\_diggit\_double\_ecran**
- double **prec\_pt\_int\_deb**
- double **factPourRayonAccostage**
- double **distanceMaxiAuPtProjete**
- double **extra\_boite\_prelocalisation**
- double **mini\_extra\_boite\_prelocalisation**
- double **ajout\_extra\_boite\_prelocalisation**
- int **contact\_type**
- string **fct\_nD\_bascul\_contact\_type\_4**
- double **penalisationPenetration**

- string **fct\_nD\_penalisationPenetration**
- int **typePenalisationPenetration**
- double **penetration\_contact\_maxi**
- string **fct\_nD\_penetration\_contact\_maxi**
- double **penetration\_borne\_regularisation**
- string **fct\_nD\_penetration\_borne\_regularisation**
- double **force\_contact\_noeud\_maxi**
- string **fct\_nD\_force\_contact\_noeud\_maxi**
- double **penalisationTangentielle**
- string **fct\_nD\_penalisationTangentielle**
- int **typePenalisationTangentielle**
- double **tangentielle\_contact\_maxi**
- string **fct\_nD\_tangentielle\_contact\_maxi**
- double **tangentielle\_borne\_regularisation**
- string **fct\_nD\_tangentielle\_borne\_regularisation**
- double **force\_tangentielle\_noeud\_maxi**
- string **fct\_nD\_force\_tangentielle\_noeud\_maxi**
- double **prec\_pt\_sur\_frontiere**
- int **nb\_boucle\_newton\_position\_frontiere**
- int **nbDecolAutorise**
- int **typeDeDecolement**
- int **nb\_glissant**
- int **niveau\_commentaire\_contact**
- int **optimisation\_numerotation**
- int **nb\_incr\_cal\_ener**
- bool **affiche\_incr\_energie**
- double **point\_interne\_delta\_thetai\_maxi**
- double **point\_interne\_prec\_thetai\_interne**
- int **point\_interne\_nb\_boucle\_sur\_delta\_thetai**
- int **point\_interne\_nb\_externe**
- bool **cal\_vol\_total\_entre\_surface\_et\_plans\_ref**
- double **ratio\_maxi\_jacoMembrane\_jacoPti**
- [Enum\\_type\\_resolution\\_matri](#) **type\_calnum\_inversion\_metrique**
- bool **sortieEquilibreGlobal**
- bool **sortieEtatActuelDansBI**
- bool **sortieEtatActuelDansCVisu**

### Attributs protégés statiques

- static VariablesTemps **tempo**
- static double **coef\_defaut\_pa\_critique** = 0.853247689523

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ParaAlgoControle.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ParaAlgoControle.cc

## 6.613 Référence de la classe ParaGlob

Graphe de collaboration de ParaGlob:



## Fonctions membres publiques

- **ParaGlob** (int dim)
- **ParaGlob** (const [ParaGlob](#) &nd)
- const int **NombreMaillage** ()
- void **ChangeDimension** (int)
- const [ParaAlgoControle](#) & **ParaAlgoControleActifs** () const
- void **ChangeParaAlgoControleActifs** ([ParaAlgoControle](#) \*para)
- void **LecTypeCalcul** ([UtilLecture](#) &lec)
- void **Info\_commande\_TypeCalcul** ([UtilLecture](#) &lec)
- [EnumTypeCalcul](#) **TypeCalcul\_maitre** ()
- bool **TypeCalcul\_principal** ([EnumTypeCalcul](#) type, bool actif=false)
- bool **SousTypeCalcul** ([EnumSousTypeCalcul](#) soustype)
- bool **Avec\_typeDeCalcul** () const
- list< [EnumSousTypeCalcul](#) > const & **LesSousTypesDeCalcul** () const
- list< bool > const & **Avec\_soustypeDeCalcul** () const
- void **AfficheTypeEtSousTypes** ()
- void **Change\_niveau\_impression** (int n)
- void **Change\_Nb\_diggit\_double** (int cas, int nevez\_nb\_diggit)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- int **EtatDeLaLecturePointInfo** () const
- void **ChangeEtatDeLaLecturePointInfo** (int etat)
- void **ChangeDemandeLectureSecondaireInPointInfo** (int val)
- int **EtatDemandeLectureSecondaireInPointInfo** () const
- const void \* **GrandeurGlobal** ([Enum\\_GrandeurGlobale](#) enu) const
- const void \* **GrandeurGlobal** (const string &nom\_ref) const
- void **Ajout\_grandeur\_consultable** (void \*grandeur, [Enum\\_GrandeurGlobale](#) enu)
- void **Ajout\_grandeur\_consultable** (void \*grandeur, string nom\_ref)
- void \* **Mise\_a\_jour\_grandeur\_consultable** ([Enum\\_GrandeurGlobale](#) enu)
- void \* **Mise\_a\_jour\_grandeur\_consultable** (const string &nom\_ref)
- void **Mise\_a\_jour\_grandeur\_consultable\_Scalaire\_double** ([Enum\\_GrandeurGlobale](#) enu, const double &valeur)
- void **Mise\_a\_jour\_grandeur\_consultable\_Scalaire\_double** (const string &nom\_ref, const double &valeur)
- void **Modif\_interactive\_constant\_utilisateur** ()
- void **Suppression\_grandeur\_consultable** ([Enum\\_GrandeurGlobale](#) enu)
- void **Suppression\_grandeur\_consultable** (const string &nom\_ref)
- void **Affiche\_GrandeurGlobal** (ostream &sort) const
- void **Recup\_list\_GrandeurGlobal** (list< string > &list\_grandeurs\_globals) const
- void **Mise\_a\_jour\_TEMPS\_COURANT** ()

## Fonctions membres publiques statiques

- static const int **Dimension** ()
- static const int **NbCompTens** ()
- static int **NiveaulImpression** ()
- static string **NbVersion** ()
- static void **Sortie\_Version** (ofstream &sort)
- static void **Sortie\_Version** ()
- static string **Lecture\_Version** (ifstream &entr)
- static string **Lecture\_Version** (istream &entr)
- static string **NbVersion\_de\_fichier** ()
- static [EnumLangue](#) **Lecture\_Langue** (ifstream &entr)
- static [EnumLangue](#) **Lecture\_Langue** (istream &entr)
- static [EnumLangue](#) **Langage** ()
- static bool **Anglais** ()
- static bool **Francais** ()
- static void **Sortie\_Langue** (ofstream &sort)
- static int **NbdigdoCA** ()
- static int **NbdigdoGR** ()
- static int **NbdigdoEC** ()
- static bool **AxiSymetrie** ()
- static void **Change\_en\_AxiSymetrie** ()
- static std::vector< double > & **Grandeurs\_globales** ()
- static void **Init\_temps** (VariablesTemps &temps)
- static const VariablesTemps & **Variables\_de\_temps** ()

## Attributs publics statiques

- static [ParaGlob](#) \* **param** = NULL

## Fonctions membres protégées

- [ParaAlgoControle](#) & [ParaAlgoContActifs](#) ()
- void [Change\\_ParaAlgoControle](#) ([ParaAlgoControle](#) &pa)

## Fonctions membres protégées statiques

- static std::vector< double > & [Mise\\_a\\_jour\\_Grandeurs\\_globales](#) ()

## Attributs protégés

- std::map< [Enum\\_GrandeurGlobale](#), const void \*, std::less< [Enum\\_GrandeurGlobale](#) > > **listegrandeurs**
- std::map< string, const void \*, std::less< string > > **listegrandeurs\_par\_string**
- double \* **pt\_temps\_courant**

## Attributs protégés statiques

- static std::vector< double > **grandeurs\_globales**

## Amis

- class [ParaAlgoControle](#)
- class [AlgoriCombine](#)
- void [Handler\\_signal](#) (int theSignal)

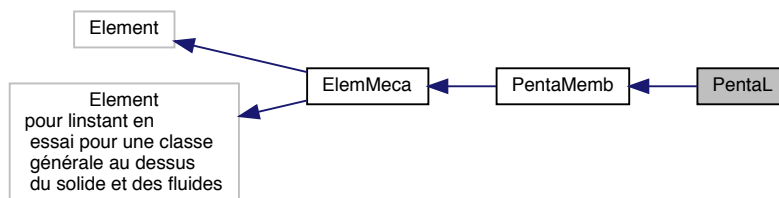
*méthode utilitaire, utilisable par les classes dérivées, permettant d'agir si un controle-c est émis intervient sur des paramètres stockés dans [ParaAlgoControle](#), qui sont actuellement actif (accessible via [ParaGlob](#)) c'est donc via ces paramètres que les autres programmes peuvent récupérer les infos et agir en conséquence éventuellement*

La documentation de cette classe a été générée à partir du fichier suivant :

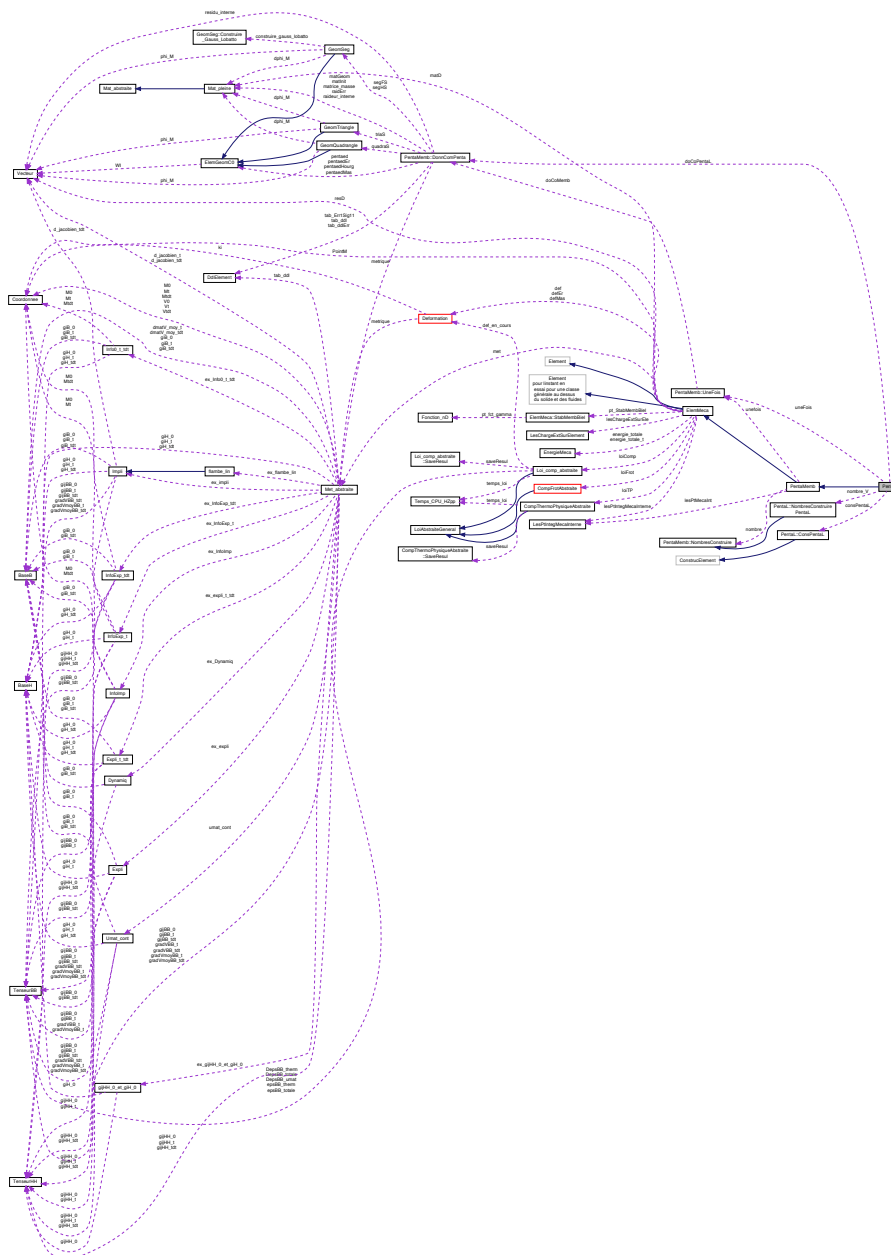
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ParaGlob.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/EnteteParaGlob.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ParaGlob.cc

## 6.614 Référence de la classe PentaL

Graphe d'héritage de PentaL:



Graphe de collaboration de PentaL:



## Classes

- class [ConsPentaL](#)
- class [NombresConstruirePentaL](#)

## Fonctions membres publiques

- **PentaL** (int num\_mail, int num\_id)
- **PentaL** (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **PentaL** (const [PentaL](#) &Penta)
- Element \* **Nevez\_copie** () const
- [PentaL](#) & **operator=** ([PentaL](#) &Penta)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_lin_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`

## Attributs protégés statiques

- static `PentaMemb::DonnComPenta * doCoPentaL = NULL`
- static `PentaMemb::UneFois uneFois`
- static `NombresConstruirePentaL nombre_V`
- static `ConsPentaL consPentaL`
- static int `bidon = 0`

## Membres hérités additionnels

### 6.614.1 Documentation des fonctions membres

#### 6.614.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaL::new_frontiere_lin_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.614.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaL::new_frontiere_lin_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.614.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaL::new_frontiere_surf_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.614.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaL::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

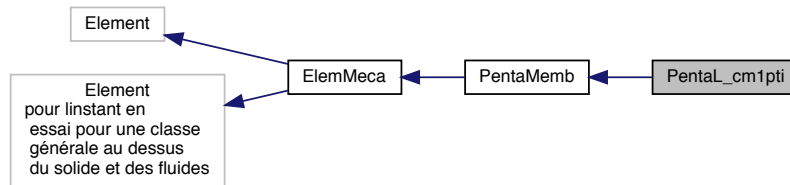
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL.cc`

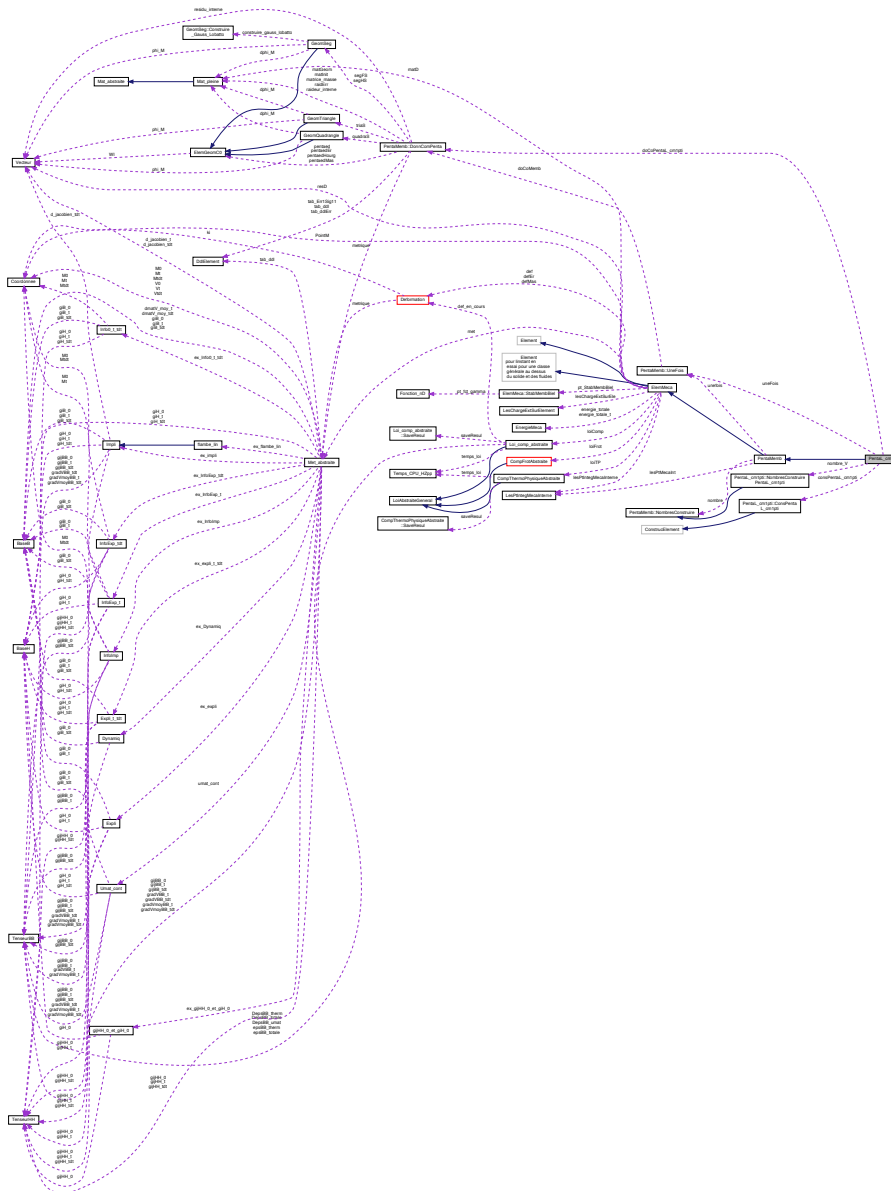


## 6.615 Référence de la classe PentaL\_cm1pti

Graphe d'héritage de PentaL\_cm1pti:



Graphe de collaboration de PentaL\_cm1pti:



## Classes

- class [ConsPentaL\\_cm1pti](#)
- class [NombresConstruirePentaL\\_cm1pti](#)

## Fonctions membres publiques

- [PentaL\\_cm1pti](#) (int num\_mail, int num\_id)
- [PentaL\\_cm1pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaL\\_cm1pti](#) (const [PentaL\\_cm1pti](#) &Penta)
- [Element](#) \* [Nevez\\_copie](#) () const
- [PentaL\\_cm1pti](#) & [operator=](#) ([PentaL\\_cm1pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [PentaMemb::DonnComPenta](#) \* [doCoPentaL\\_cm1pti](#) = NULL
- static [PentaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruirePentaL\\_cm1pti](#) [nombre\\_V](#)
- static [ConsPentaL\\_cm1pti](#) [consPentaL\\_cm1pti](#)
- static int [bidon](#) = 0

## Membres hérités additionnels

### 6.615.1 Documentation des fonctions membres

#### 6.615.1.1 [new\\_frontiere\\_lin\\_rec\(\)](#)

```
ElFrontiere * PentaL\_cm1pti::new\_frontiere\_lin\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.615.1.2 [new\\_frontiere\\_lin\\_tri\(\)](#)

```
ElFrontiere * PentaL\_cm1pti::new\_frontiere\_lin\_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.615.1.3 [new\\_frontiere\\_surf\\_rec\(\)](#)

```
ElFrontiere * PentaL\_cm1pti::new\_frontiere\_surf\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.615.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaL_cm1pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

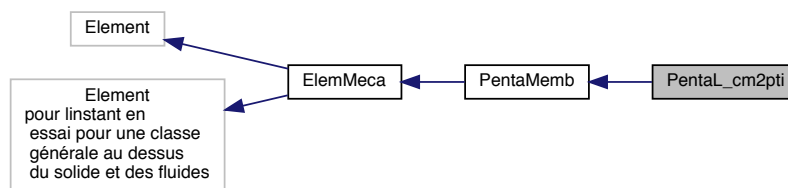
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

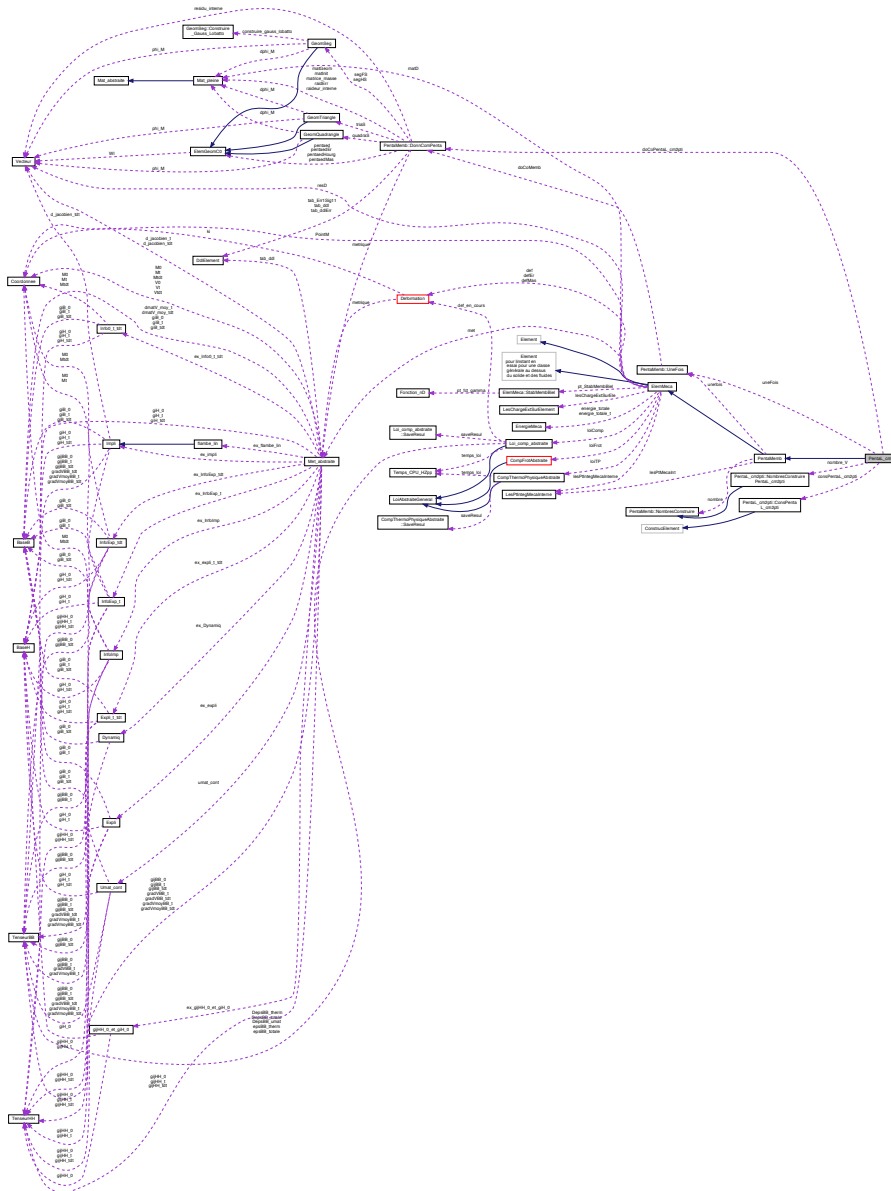
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm1pti.cc

## 6.616 Référence de la classe PentaL\_cm2pti

Grphe d'héritage de PentaL\_cm2pti:



Graphe de collaboration de PentaL\_cm2pti:



## Classes

- class [ConsPentaL\\_cm2pti](#)
- class [NombresConstruirePentaL\\_cm2pti](#)

## Fonctions membres publiques

- [PentaL\\_cm2pti](#) (int num\_mail, int num\_id)
- [PentaL\\_cm2pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaL\\_cm2pti](#) (const [PentaL\\_cm2pti](#) &Penta)
- [Element](#) \* [Nevez\\_copie](#) () const
- [PentaL\\_cm2pti](#) & [operator=](#) ([PentaL\\_cm2pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

- `ElFrontiere * new_frontiere_lin_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- static `PentaMemb::DonnComPenta * doCoPentaL_cm2pti = NULL`
- static `PentaMemb::UneFois uneFois`
- static `NombresConstruirePentaL_cm2pti nombre_V`
- static `ConsPentaL_cm2pti consPentaL_cm2pti`
- static int `bidon = 0`

### Membres hérités additionnels

#### 6.616.1 Documentation des fonctions membres

##### 6.616.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaL_cm2pti::new_frontiere_lin_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.616.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaL_cm2pti::new_frontiere_lin_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.616.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaL_cm2pti::new_frontiere_surf_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.616.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaL_cm2pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

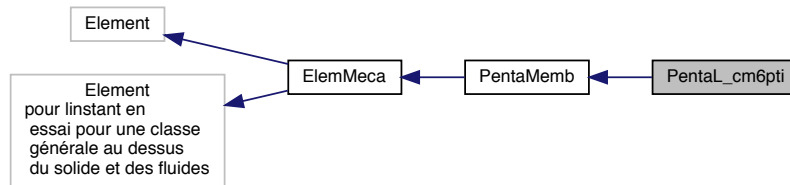
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

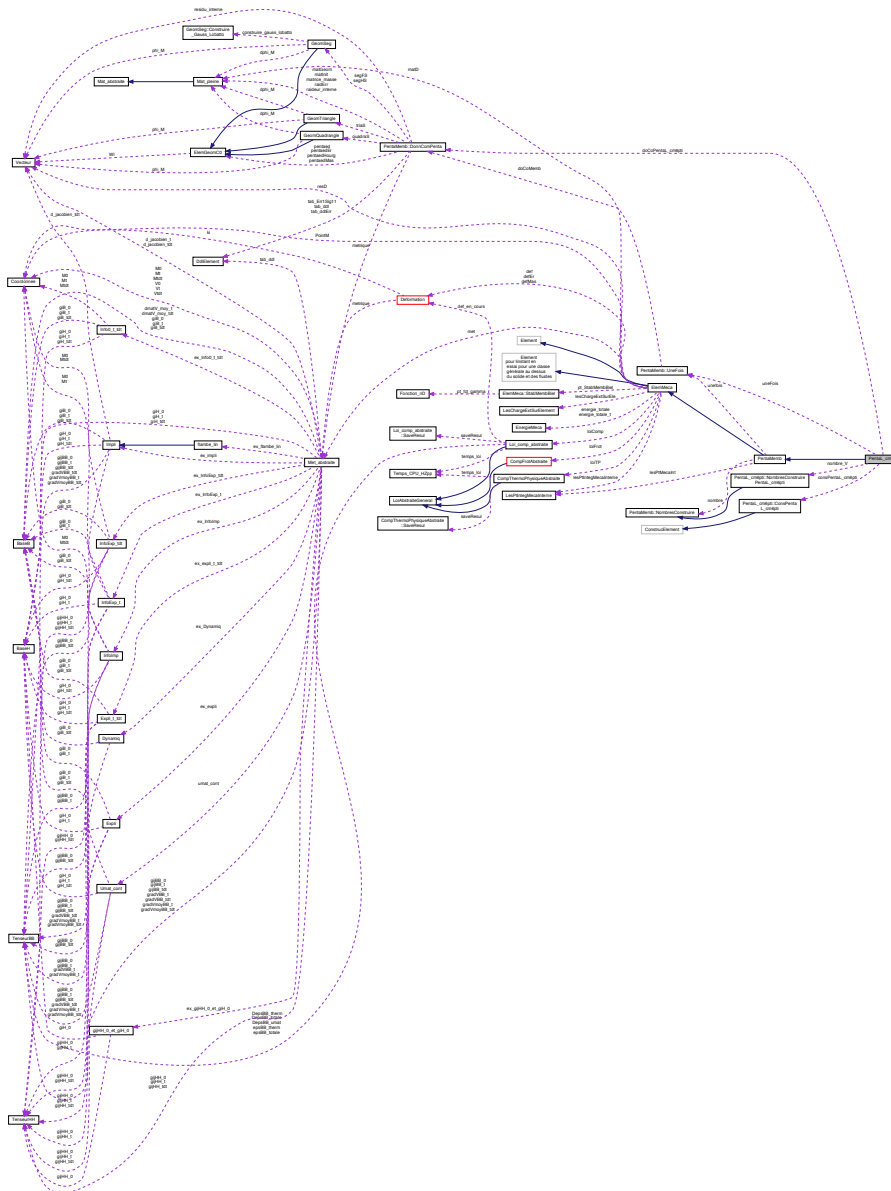
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL_cm2pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL_cm2pti.cc`

## 6.617 Référence de la classe PentaL\_cm6pti

Graphe d'héritage de PentaL\_cm6pti:



Graphe de collaboration de PentaL\_cm6pti:



## Classes

- class [ConsPentaL\\_cm6pti](#)
- class [NombresConstruirePentaL\\_cm6pti](#)

## Fonctions membres publiques

- [PentaL\\_cm6pti](#) (int num\_mail, int num\_id)
- [PentaL\\_cm6pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaL\\_cm6pti](#) (const [PentaL\\_cm6pti](#) &Penta)
- [Element](#) \* [Nevez\\_copie](#) () const
- [PentaL\\_cm6pti](#) & [operator=](#) ([PentaL\\_cm6pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [PentaMemb::DonnComPenta](#) \* [doCoPentaL\\_cm6pti](#) = NULL
- static [PentaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruirePentaL\\_cm6pti](#) [nombre\\_V](#)
- static [ConsPentaL\\_cm6pti](#) [consPentaL\\_cm6pti](#)
- static int [bidon](#) = 0

## Membres hérités additionnels

### 6.617.1 Documentation des fonctions membres

#### 6.617.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaL\_cm6pti::new\_frontiere\_lin\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.617.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaL\_cm6pti::new\_frontiere\_lin\_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.617.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaL\_cm6pti::new\_frontiere\_surf\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

### 6.617.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaL_cm6pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

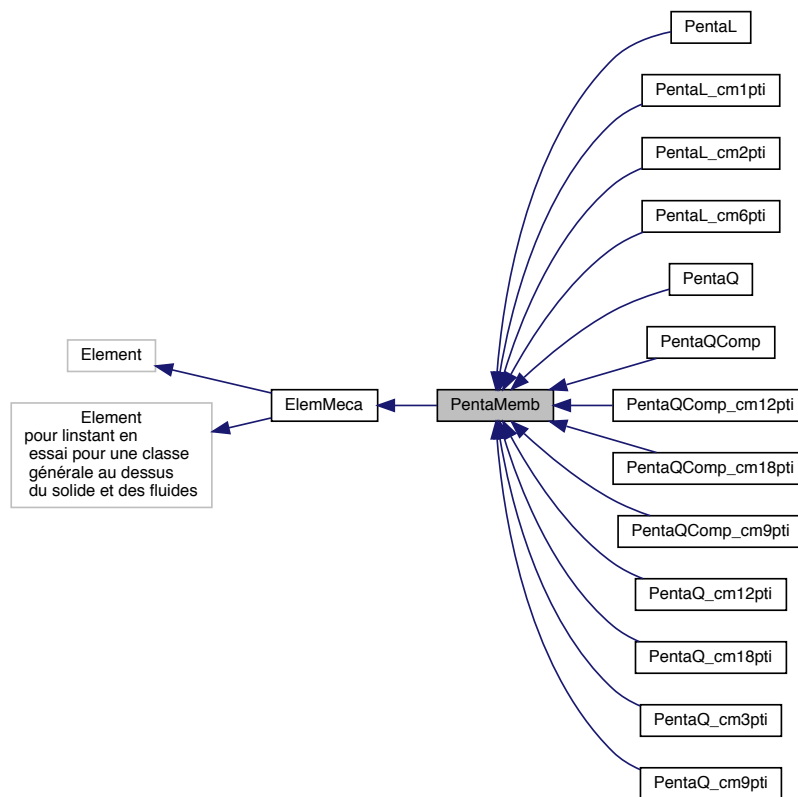
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm6pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaL\_cm6pti.cc

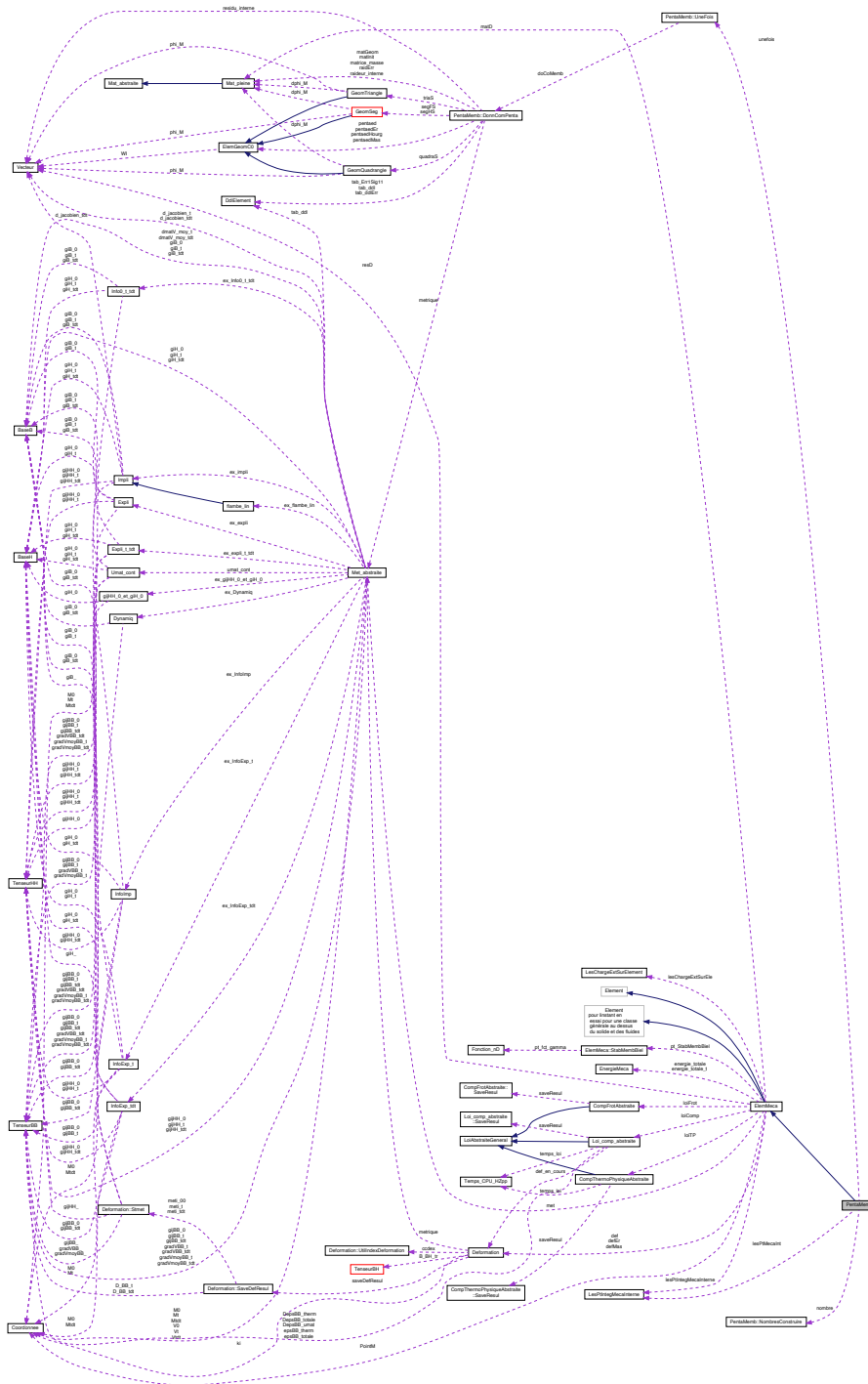
## 6.618 Référence de la classe PentaMemb

Grappe d'héritage de PentaMemb:





Graphe de collaboration de PentaMemb:



**Classes**

- class [DonnComPenta](#)
- class [NombresConstruire](#)
- class [UneFois](#)

**Fonctions membres publiques**

- **PentaMemb** (int num\_mail, int num\_id, [Enum\\_interp](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")

- **PentaMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, const [Tableau](#)< [Noeud](#) \* > &tab, string info="")
- **PentaMemb** (const [PentaMemb](#) &pentaMem)
- **PentaMemb** & **operator=** ([PentaMemb](#) &pentaMem)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- [ElemGeomC0](#) & [ElementGeometrique](#) () const
- const [ElemGeomC0](#) & [ElementGeometrique\\_const](#) () const
- [Coordonnee](#) & [Point\\_physique](#) (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComple** ()
- Element \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- Element \* **Comple** **Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- bool **SurExiste** (int ns) const
- bool **AreteExiste** (int na) const
- const [DdlElement](#) & [TableauDdl](#) () const
- Element::ResRaid **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- [DdlElement](#) & [Tableau\\_de\\_Sig1](#) () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- ResRaid **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_presUniDir\_E\_t** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_presUniDir\_E\_tdt** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)

- ResRaid **SMR\_charge\_presUniDir\_I** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_lineique\\_E\\_t](#) (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_lineique\\_E\\_tdt](#) (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_pression\\_E\\_t](#) (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_pression\\_E\\_tdt](#) (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_hydrostatique\\_E\\_t](#) (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- [Vecteur SM\\_charge\\_hydrostatique\\_E\\_tdt](#) (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- [Vecteur SM\\_charge\\_hydrodynamique\\_E\\_t](#) ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_hydrodynamique\\_E\\_tdt](#) ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Tableau](#)< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- [EIFrontiere](#) \*const [Frontiere\\_surfacique](#) (int num, bool force=false)
- virtual [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int num, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- virtual [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int num, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- void **ConstTabDdl** ()

## Fonctions membres protégées

- [PentaMemb::DonnComPenta](#) \* **Init** ([ElemGeomC0](#) \*penta, [ElemGeomC0](#) \*pentaEr, [ElemGeomC0](#) \*penta↔Mas, [ElemGeomC0](#) \*pentaHourg, bool sans\_init\_noeud=false)
- void **Destruction** ()
- virtual [EIFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)=0
- virtual [EIFrontiere](#) \* [new\\_frontiere\\_lin\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)=0
- virtual [EIFrontiere](#) \* [new\\_frontiere\\_surf\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)=0
- virtual [EIFrontiere](#) \* [new\\_frontiere\\_surf\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)=0
- int [Dim\\_sig\\_eps](#) () const

## Attributs protégés

- [UneFois](#) \* **unefois**
- [LesPtIntegMecaInterne](#) **lesPtMecaInt**
- [NombresConstruire](#) \* **nombre**

## Membres hérités additionnels

### 6.618.1 Documentation des fonctions membres

#### 6.618.1.1 Active\_ddl\_Sigma()

void [PentaMemb::Active\\_ddl\\_Sigma](#) ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

### 6.618.1.2 Active\_premier\_ddl\_Sigma()

```
void PentaMemb::Active_premier_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.618.1.3 ContraintesAbsolues()

```
bool PentaMemb::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.618.1.4 Dim\_sig\_eps()

```
int PentaMemb::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.618.1.5 ErreurElement()

```
void PentaMemb::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en Réimplémentée à partir de [ElemMeca](#).

### 6.618.1.6 Frontiere\_surfacique()

```
ElFrontiere *const PentaMemb::Frontiere_surfacique (
    int num,
    bool force = false ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.618.1.7 Inactive\_ddl\_Sigma()

```
void PentaMemb::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.618.1.8 LectureContraintes()

```
void PentaMemb::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.618.1.9 Long\_arrete\_mini\_sur\_c()

```
double PentaMemb::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

**6.618.1.10 new\_frontiere\_lin()**

```
ElFrontiere * PentaMemb::new_frontiere_lin (
    int num,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [virtual]
```

Implémente [ElemMeca](#).

**6.618.1.11 new\_frontiere\_surf()**

```
ElFrontiere * PentaMemb::new_frontiere_surf (
    int num,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [virtual]
```

Implémente [ElemMeca](#).

**6.618.1.12 Plus\_ddl\_Sigma()**

```
void PentaMemb::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

**6.618.1.13 Tableau\_de\_Sig1()**

```
DdlElement & PentaMemb::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

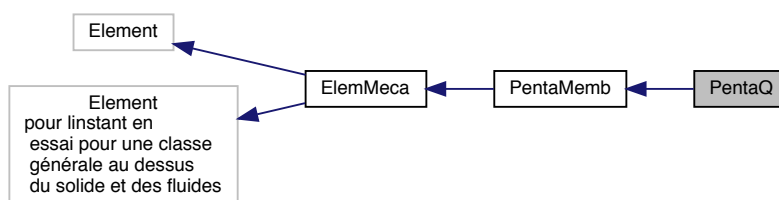
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

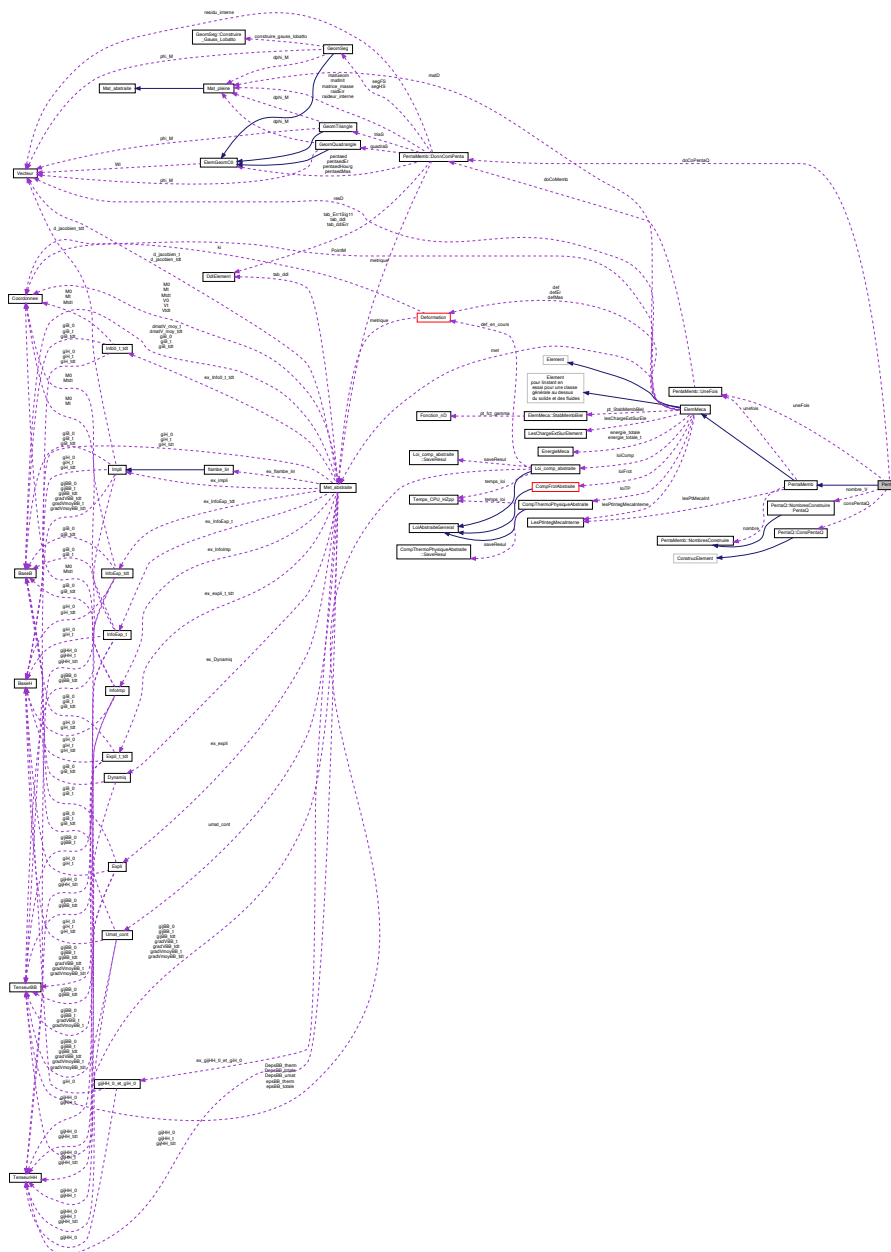
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaMemb.cc

**6.619 Référence de la classe PentaQ**

Graphe d'héritage de PentaQ:



Graphe de collaboration de PentaQ:



## Classes

- class [ConsPentaQ](#)
- class [NombresConstruirePentaQ](#)

## Fonctions membres publiques

- **PentaQ** (int num\_mail, int num\_id)
- **PentaQ** (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **PentaQ** (const [PentaQ](#) &Penta)
- Element \* **Nevez\_copie** () const
- [PentaQ](#) & **operator=** ([PentaQ](#) &Penta)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_lin_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`

## Attributs protégés statiques

- static `PentaMemb::DonnComPenta * doCoPentaQ = NULL`
- static `PentaMemb::UneFois uneFois`
- static `NombresConstruirePentaQ nombre_V`
- static `ConsPentaQ consPentaQ`
- static int `bidon`

## Membres hérités additionnels

### 6.619.1 Documentation des fonctions membres

#### 6.619.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaQ::new_frontiere_lin_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.619.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaQ::new_frontiere_lin_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.619.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaQ::new_frontiere_surf_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.619.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQ::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

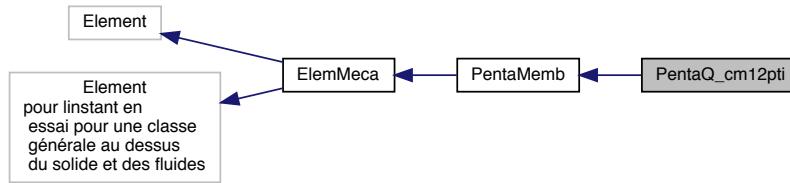
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

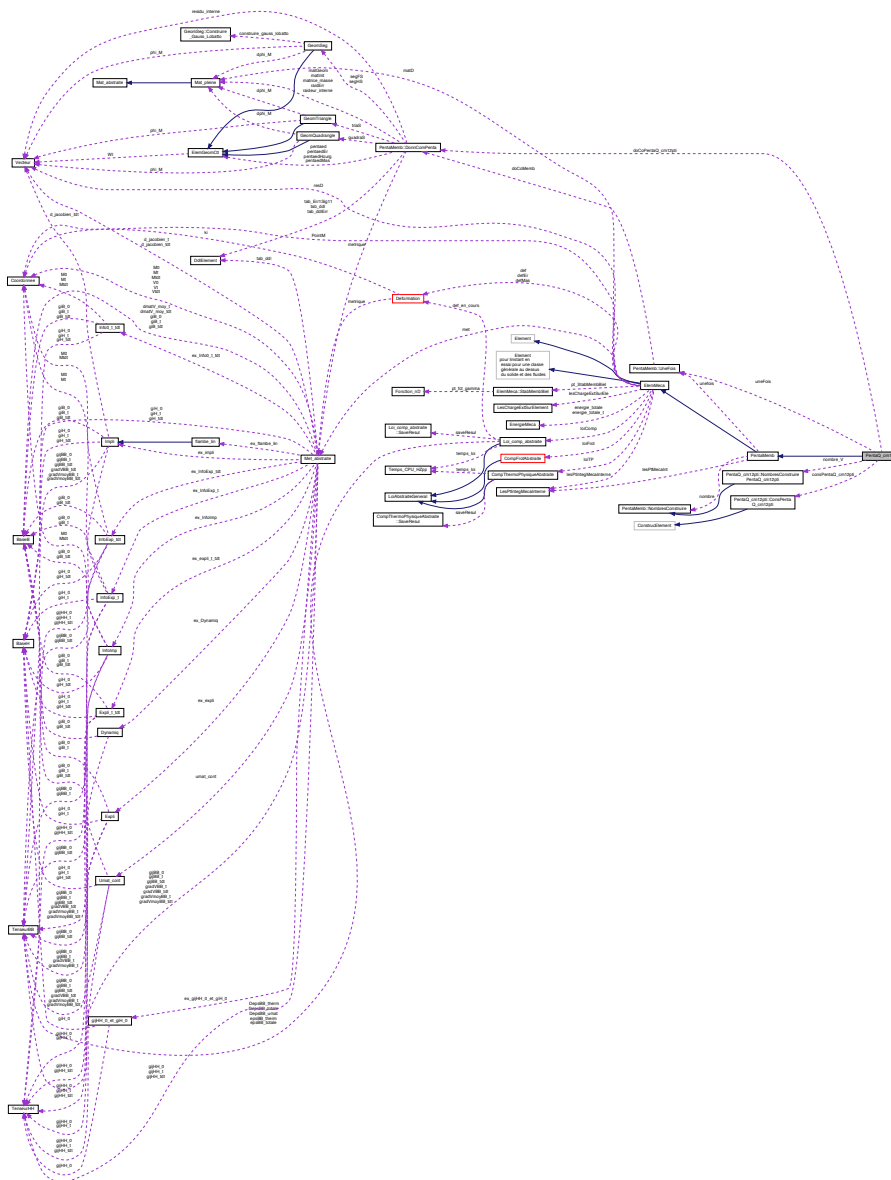
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ.cc`

## 6.620 Référence de la classe PentaQ\_cm12pti

Graphe d'héritage de PentaQ\_cm12pti:



Graphe de collaboration de PentaQ\_cm12pti:





## Classes

- class [ConsPentaQ\\_cm12pti](#)
- class [NombresConstruirePentaQ\\_cm12pti](#)

## Fonctions membres publiques

- [PentaQ\\_cm12pti](#) (int num\_mail, int num\_id)
- [PentaQ\\_cm12pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaQ\\_cm12pti](#) (const [PentaQ\\_cm12pti](#) &Penta)
- [Element](#) \* [Nevez\\_copie](#) () const
- [PentaQ\\_cm12pti](#) & [operator=](#) ([PentaQ\\_cm12pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [PentaMemb::DonnComPenta](#) \* [doCoPentaQ\\_cm12pti](#) = NULL
- static [PentaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruirePentaQ\\_cm12pti](#) [nombre\\_V](#)
- static [ConsPentaQ\\_cm12pti](#) [consPentaQ\\_cm12pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.620.1 Documentation des fonctions membres

#### 6.620.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaQ\_cm12pti::new\_frontiere\_lin\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.620.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaQ\_cm12pti::new\_frontiere\_lin\_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.620.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaQ\_cm12pti::new\_frontiere\_surf\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

### 6.620.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQ_cm12pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

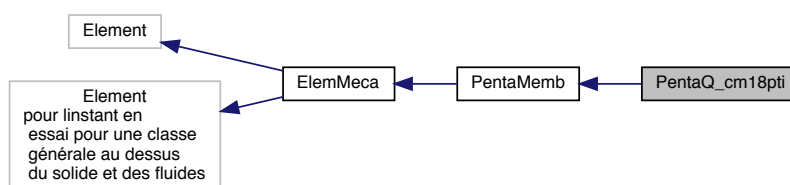
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

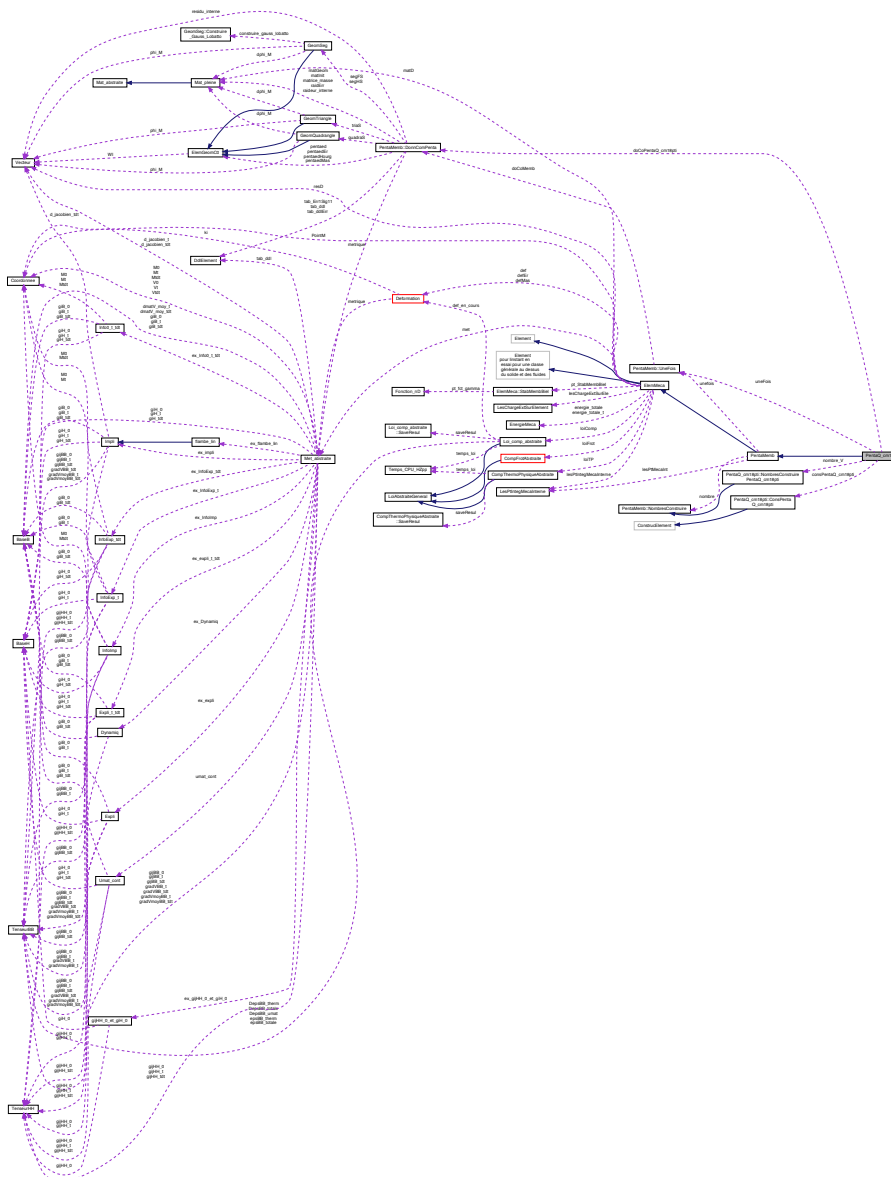
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm12pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm12pti.cc

## 6.621 Référence de la classe PentaQ\_cm18pti

Grphe d'héritage de PentaQ\_cm18pti:



Graphe de collaboration de PentaQ\_cm18pti:



## Classes

- class [ConsPentaQ\\_cm18pti](#)
- class [NombresConstruirePentaQ\\_cm18pti](#)

## Fonctions membres publiques

- [PentaQ\\_cm18pti](#) (int num\_mail, int num\_id)
- [PentaQ\\_cm18pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaQ\\_cm18pti](#) (const [PentaQ\\_cm18pti](#) &Penta)
- Element \* [Nevez\\_copie](#) () const
- [PentaQ\\_cm18pti](#) & [operator=](#) ([PentaQ\\_cm18pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdiElement](#) &ddelem)

- `ElFrontiere * new_frontiere_lin_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- static `PentaMemb::DonnComPenta * doCoPentaQ_cm18pti = NULL`
- static `PentaMemb::UneFois uneFois`
- static `NombresConstruirePentaQ_cm18pti nombre_V`
- static `ConsPentaQ_cm18pti consPentaQ_cm18pti`
- static int `bidon`

### Membres hérités additionnels

#### 6.621.1 Documentation des fonctions membres

##### 6.621.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaQ_cm18pti::new_frontiere_lin_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.621.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaQ_cm18pti::new_frontiere_lin_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.621.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaQ_cm18pti::new_frontiere_surf_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.621.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQ_cm18pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

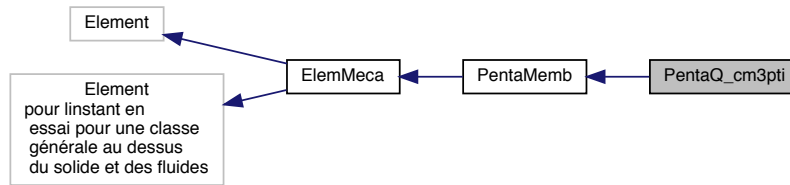
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

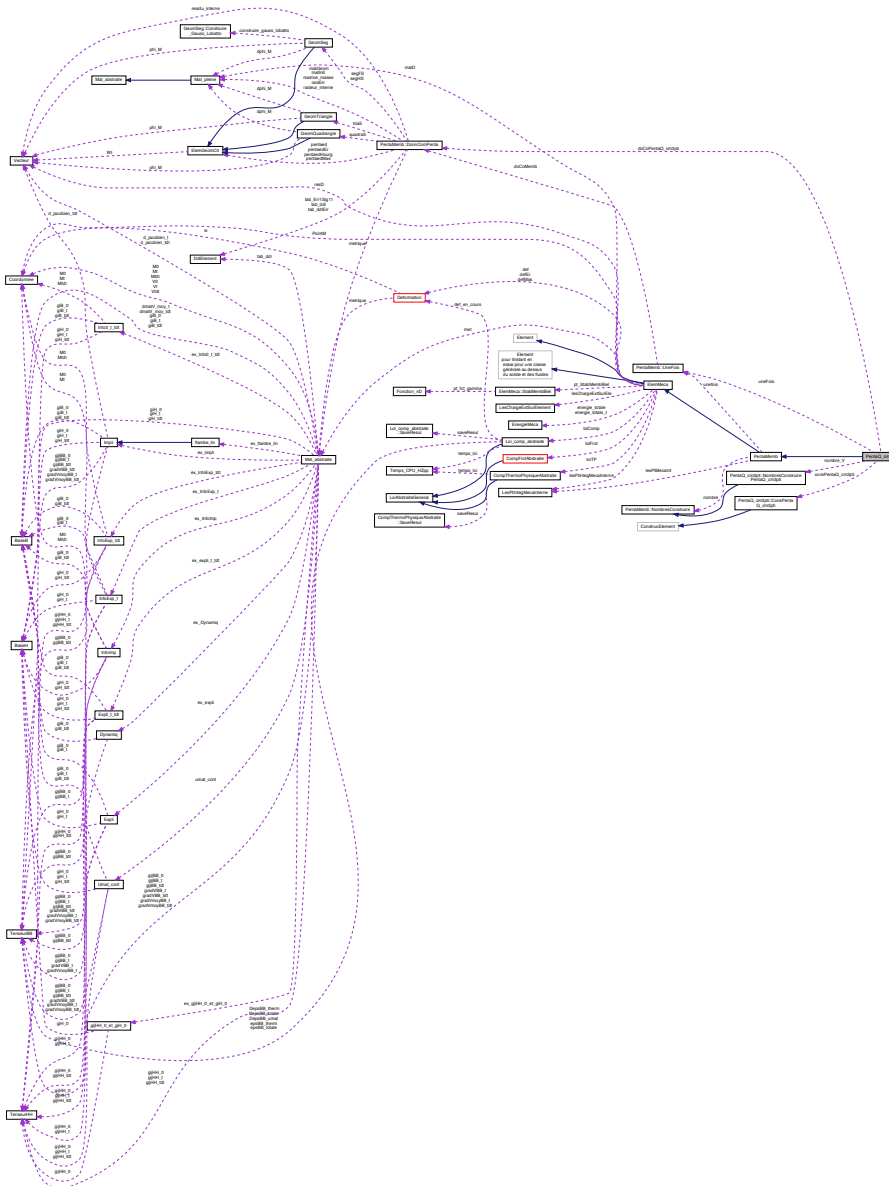
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm18pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm18pti.cc`

## 6.622 Référence de la classe PentaQ\_cm3pti

Graphe d'héritage de PentaQ\_cm3pti:



Graphe de collaboration de PentaQ\_cm3pti:



## Classes

- class [ConsPentaQ\\_cm3pti](#)
- class [NombresConstruirePentaQ\\_cm3pti](#)

## Fonctions membres publiques

- [PentaQ\\_cm3pti](#) (int num\_mail, int num\_id)
- [PentaQ\\_cm3pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaQ\\_cm3pti](#) (const [PentaQ\\_cm3pti](#) &Penta)
- [Element](#) \* [Nevez\\_copie](#) () const
- [PentaQ\\_cm3pti](#) & [operator=](#) ([PentaQ\\_cm3pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [PentaMemb::DonnComPenta](#) \* [doCoPentaQ\\_cm3pti](#) = NULL
- static [PentaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruirePentaQ\\_cm3pti](#) [nombre\\_V](#)
- static [ConsPentaQ\\_cm3pti](#) [consPentaQ\\_cm3pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.622.1 Documentation des fonctions membres

#### 6.622.1.1 [new\\_frontiere\\_lin\\_rec\(\)](#)

```
ElFrontiere * PentaQ\_cm3pti::new\_frontiere\_lin\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.622.1.2 [new\\_frontiere\\_lin\\_tri\(\)](#)

```
ElFrontiere * PentaQ\_cm3pti::new\_frontiere\_lin\_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.622.1.3 [new\\_frontiere\\_surf\\_rec\(\)](#)

```
ElFrontiere * PentaQ\_cm3pti::new\_frontiere\_surf\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.622.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQ_cm3pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

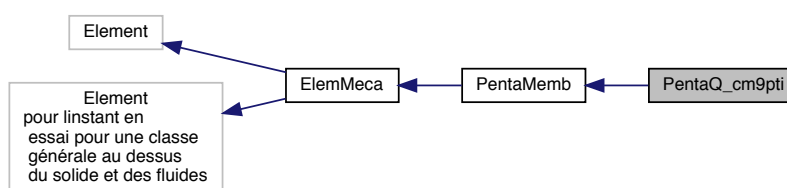
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

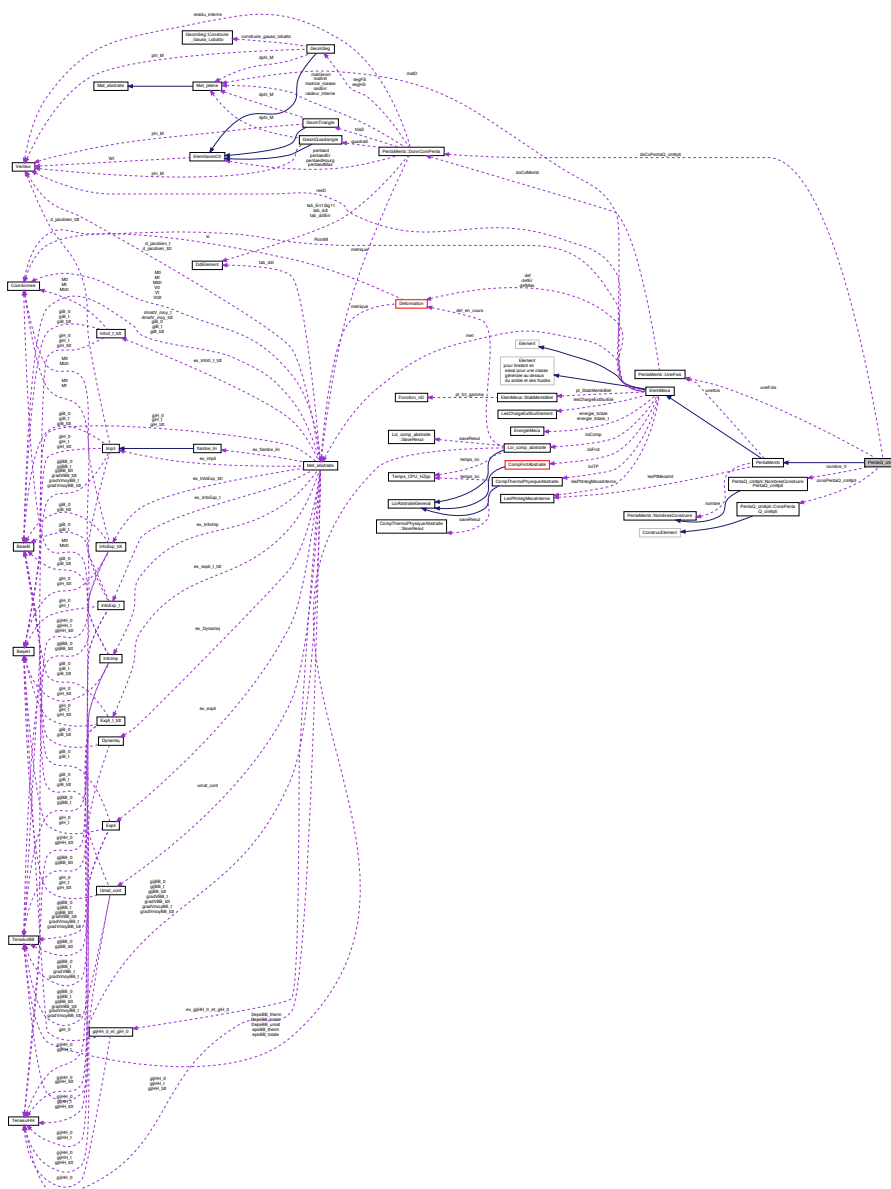
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm3pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ\_cm3pti.cc

## 6.623 Référence de la classe PentaQ\_cm9pti

Grphe d'héritage de PentaQ\_cm9pti:



Graphe de collaboration de PentaQ\_cm9pti:



## Classes

- class [ConsPentaQ\\_cm9pti](#)
- class [NombresConstruirePentaQ\\_cm9pti](#)

## Fonctions membres publiques

- [PentaQ\\_cm9pti](#) (int num\_mail, int num\_id)
- [PentaQ\\_cm9pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaQ\\_cm9pti](#) (const [PentaQ\\_cm9pti](#) &Penta)
- Element \* [Nevez\\_copie](#) () const
- [PentaQ\\_cm9pti](#) & [operator=](#) ([PentaQ\\_cm9pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)



- `ElFrontiere * new_frontiere_lin_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- static `PentaMemb::DonnComPenta * doCoPentaQ_cm9pti = NULL`
- static `PentaMemb::UneFois uneFois`
- static `NombresConstruirePentaQ_cm9pti nombre_V`
- static `ConsPentaQ_cm9pti consPentaQ_cm9pti`
- static int `bidon`

### Membres hérités additionnels

#### 6.623.1 Documentation des fonctions membres

##### 6.623.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaQ_cm9pti::new_frontiere_lin_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.623.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaQ_cm9pti::new_frontiere_lin_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.623.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaQ_cm9pti::new_frontiere_surf_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.623.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQ_cm9pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

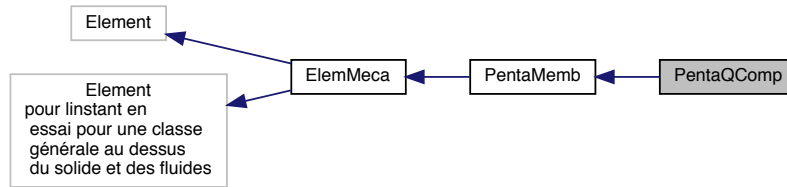
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

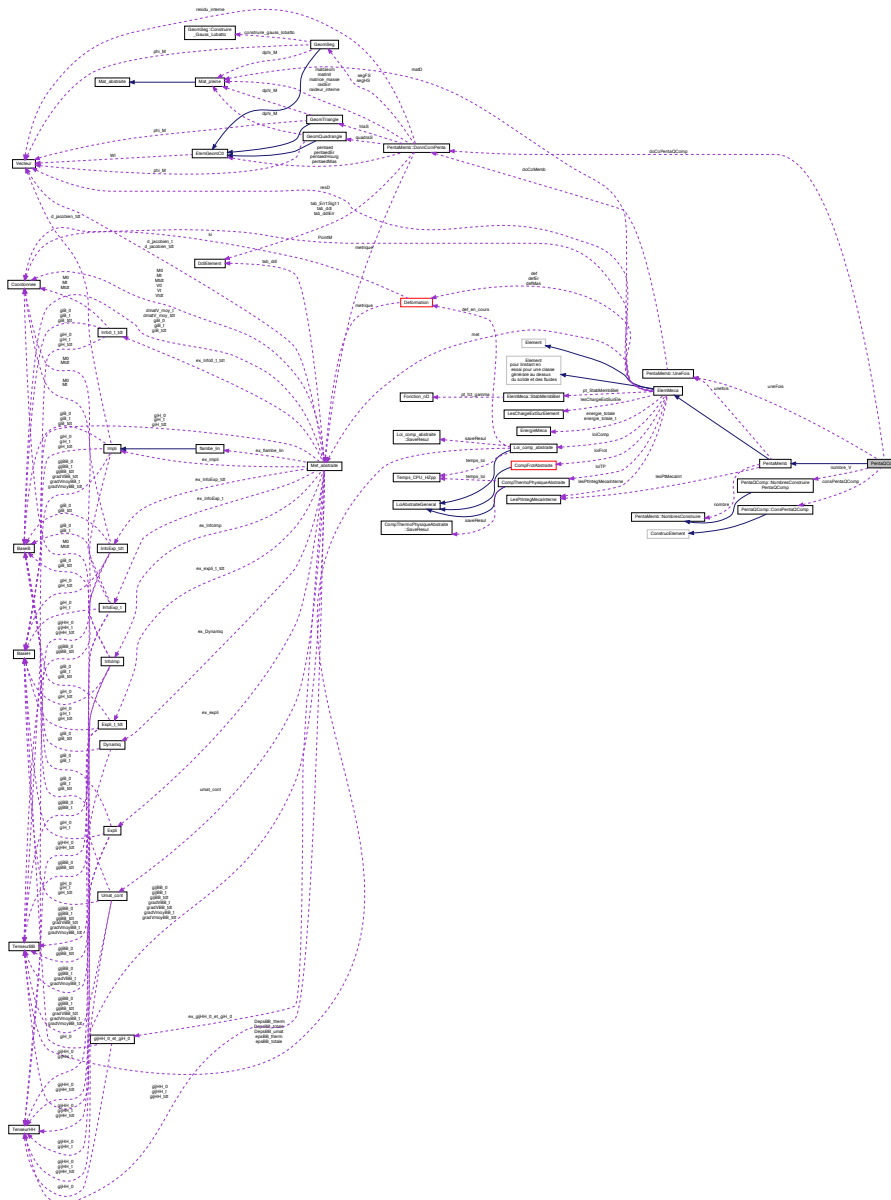
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm9pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQ_cm9pti.cc`

## 6.624 Référence de la classe PentaQComp

Graphe d'héritage de PentaQComp:



Graphe de collaboration de PentaQComp:



## Classes

- class [ConsPentaQComp](#)
- class [NombresConstruirePentaQComp](#)

## Fonctions membres publiques

- **PentaQComp** (int num\_mail, int num\_id)
- **PentaQComp** (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **PentaQComp** (const [PentaQComp](#) &Penta)
- [Element](#) \* **Nevez\_copie** () const
- [PentaQComp](#) & **operator=** ([PentaQComp](#) &Penta)
- list< [Noeud](#) \* > **Construct\_from\_imcomplet** (const [Element](#) &elem, list< [DeuxEntiers](#) > &li\_bornes, int nbnt)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin\_rec** ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_lin\_tri** ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf\_rec** ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf\_tri** ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [PentaMemb::DonnComPenta](#) \* **doCoPentaQComp** = NULL
- static [PentaMemb::UneFois](#) **uneFois**
- static [NombresConstruirePentaQComp](#) **nombre\_V**
- static [ConsPentaQComp](#) **consPentaQComp**
- static int **bidon**

## Membres hérités additionnels

### 6.624.1 Documentation des fonctions membres

#### 6.624.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaQComp::new\_frontiere\_lin\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.624.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaQComp::new\_frontiere\_lin\_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.624.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaQComp::new\_frontiere\_surf\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.624.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQComp::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

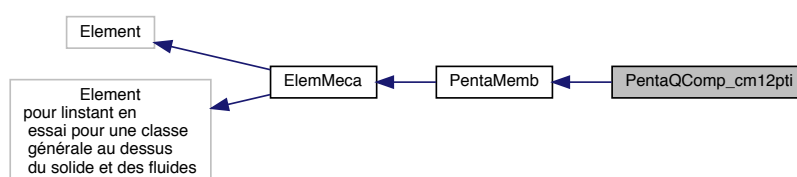
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

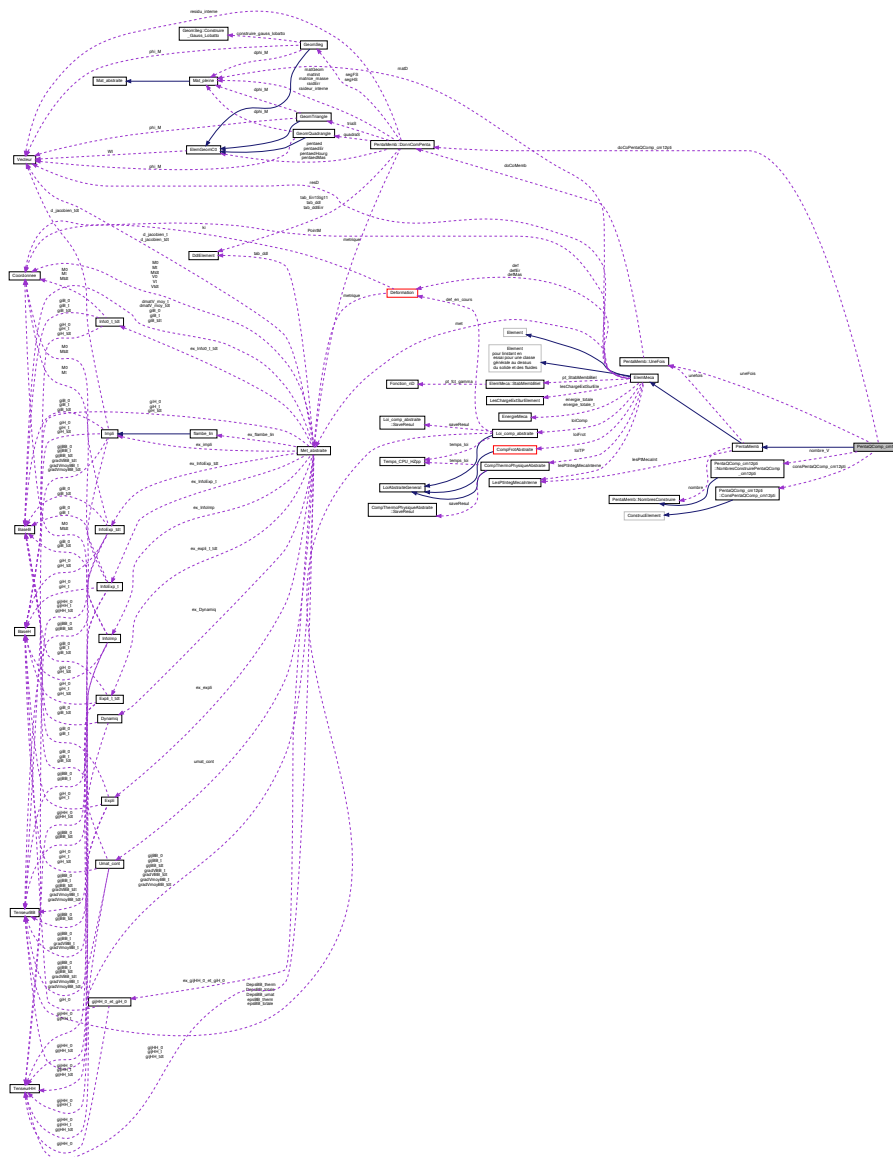
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp.cc

### 6.625 Référence de la classe PentaQComp\_cm12pti

Grphe d'héritage de PentaQComp\_cm12pti:



Graphe de collaboration de PentaQComp\_cm12pti:



## Classes

- class [ConsPentaQComp\\_cm12pti](#)
- class [NombresConstruirePentaQComp\\_cm12pti](#)

## Fonctions membres publiques

- [PentaQComp\\_cm12pti](#) (int num\_mail, int num\_id)
- [PentaQComp\\_cm12pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaQComp\\_cm12pti](#) (const [PentaQComp\\_cm12pti](#) &Penta)
- [Element](#) \* [Nevez\\_copie](#) () const
- [PentaQComp\\_cm12pti](#) & [operator=](#) ([PentaQComp\\_cm12pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EiFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdiElement](#) &ddelem)
- [EiFrontiere](#) \* [new\\_frontiere\\_lin\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdiElement](#) &ddelem)

- `ElFrontiere * new_frontiere_surf_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- static `PentaMemb::DonnComPenta * doCoPentaQComp_cm12pti = NULL`
- static `PentaMemb::UneFois uneFois`
- static `NombresConstruirePentaQComp_cm12pti nombre_V`
- static `ConsPentaQComp_cm12pti consPentaQComp_cm12pti`
- static int `bidon`

### Membres hérités additionnels

#### 6.625.1 Documentation des fonctions membres

##### 6.625.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaQComp_cm12pti::new_frontiere_lin_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.625.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaQComp_cm12pti::new_frontiere_lin_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.625.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaQComp_cm12pti::new_frontiere_surf_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.625.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQComp_cm12pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

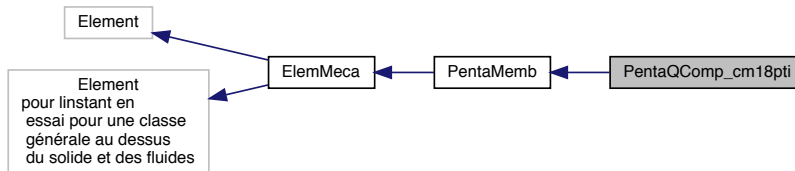
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

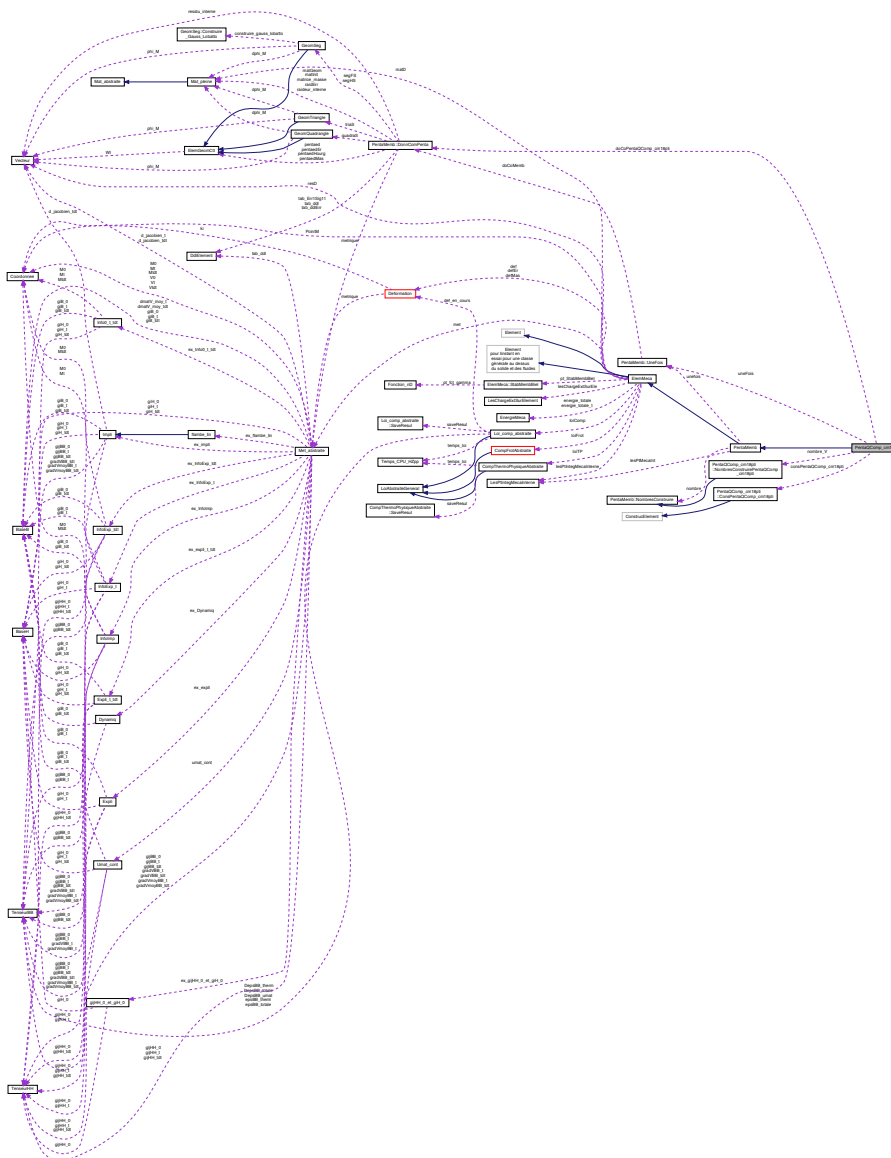
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp_↔cm12pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp_↔cm12pti.cc`

## 6.626 Référence de la classe PentaQComp\_cm18pti

Grappe d'héritage de PentaQComp\_cm18pti:



Grappe de collaboration de PentaQComp\_cm18pti:



## Classes

- class [ConsPentaQComp\\_cm18pti](#)
- class [NombresConstruirePentaQComp\\_cm18pti](#)

## Fonctions membres publiques

- [PentaQComp\\_cm18pti](#) (int num\_mail, int num\_id)
- [PentaQComp\\_cm18pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaQComp\\_cm18pti](#) (const [PentaQComp\\_cm18pti](#) &Penta)
- [Element](#) \* [Nevez\\_copie](#) () const
- [PentaQComp\\_cm18pti](#) & [operator=](#) ([PentaQComp\\_cm18pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_lin\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [PentaMemb::DonnComPenta](#) \* [doCoPentaQComp\\_cm18pti](#) = NULL
- static [PentaMemb::UneFois](#) [uneFois](#)
- static [NombresConstruirePentaQComp\\_cm18pti](#) [nombre\\_V](#)
- static [ConsPentaQComp\\_cm18pti](#) [consPentaQComp\\_cm18pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.626.1 Documentation des fonctions membres

#### 6.626.1.1 [new\\_frontiere\\_lin\\_rec\(\)](#)

```
ElFrontiere * PentaQComp\_cm18pti::new\_frontiere\_lin\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.626.1.2 [new\\_frontiere\\_lin\\_tri\(\)](#)

```
ElFrontiere * PentaQComp\_cm18pti::new\_frontiere\_lin\_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

#### 6.626.1.3 [new\\_frontiere\\_surf\\_rec\(\)](#)

```
ElFrontiere * PentaQComp\_cm18pti::new\_frontiere\_surf\_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).



#### 6.626.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQComp_cm18pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

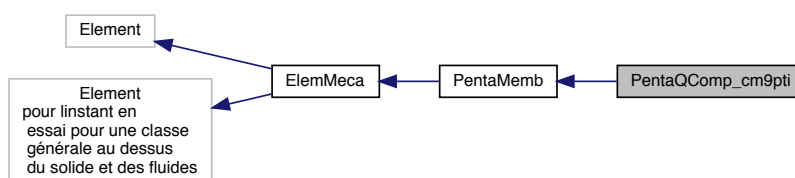
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

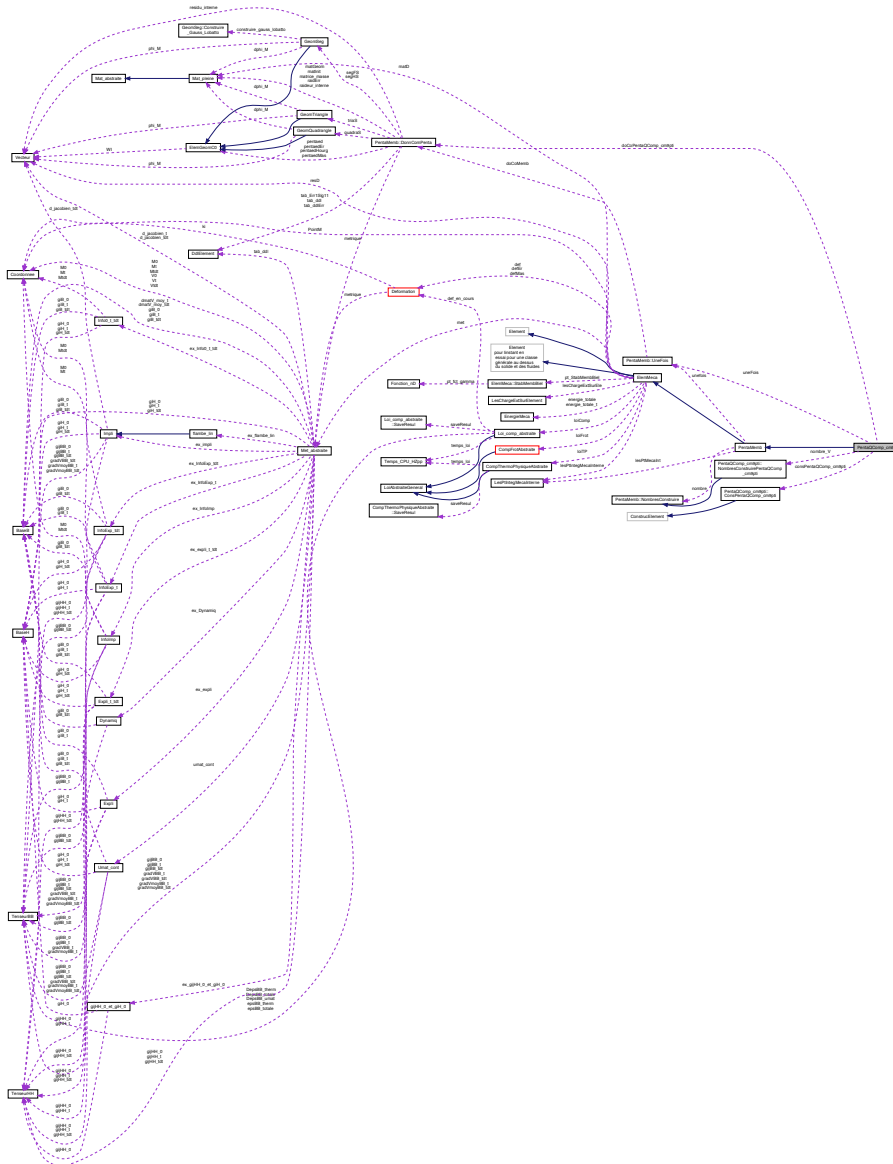
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm18pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp\_cm18pti.cc

## 6.627 Référence de la classe PentaQComp\_cm9pti

Graphe d'héritage de PentaQComp\_cm9pti:



Graphe de collaboration de PentaQComp\_cm9pti:



## Classes

- class [ConsPentaQComp\\_cm9pti](#)
- class [NombresConstruirePentaQComp\\_cm9pti](#)

## Fonctions membres publiques

- [PentaQComp\\_cm9pti](#) (int num\_mail, int num\_id)
- [PentaQComp\\_cm9pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [PentaQComp\\_cm9pti](#) (const [PentaQComp\\_cm9pti](#) &Penta)
- [Element](#) \* [Nevez\\_copie](#) () const
- [PentaQComp\\_cm9pti](#) & [operator=](#) ([PentaQComp\\_cm9pti](#) &Penta)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin\\_rec](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_lin\\_tri](#) ([Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

- `ElFrontiere * new_frontiere_surf_rec (Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf_tri (Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- static `PentaMemb::DonnComPenta * doCoPentaQComp_cm9pti = NULL`
- static `PentaMemb::UneFois uneFois`
- static `NombresConstruirePentaQComp_cm9pti nombre_V`
- static `ConsPentaQComp_cm9pti consPentaQComp_cm9pti`
- static int `bidon`

### Membres hérités additionnels

#### 6.627.1 Documentation des fonctions membres

##### 6.627.1.1 new\_frontiere\_lin\_rec()

```
ElFrontiere * PentaQComp_cm9pti::new_frontiere_lin_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.627.1.2 new\_frontiere\_lin\_tri()

```
ElFrontiere * PentaQComp_cm9pti::new_frontiere_lin_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.627.1.3 new\_frontiere\_surf\_rec()

```
ElFrontiere * PentaQComp_cm9pti::new_frontiere_surf_rec (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [PentaMemb](#).

##### 6.627.1.4 new\_frontiere\_surf\_tri()

```
ElFrontiere * PentaQComp_cm9pti::new_frontiere_surf_tri (
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

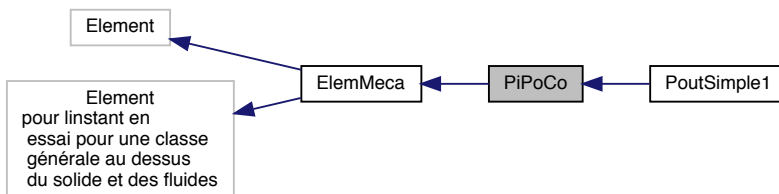
Implémente [PentaMemb](#).

La documentation de cette classe a été générée à partir du fichier suivant :

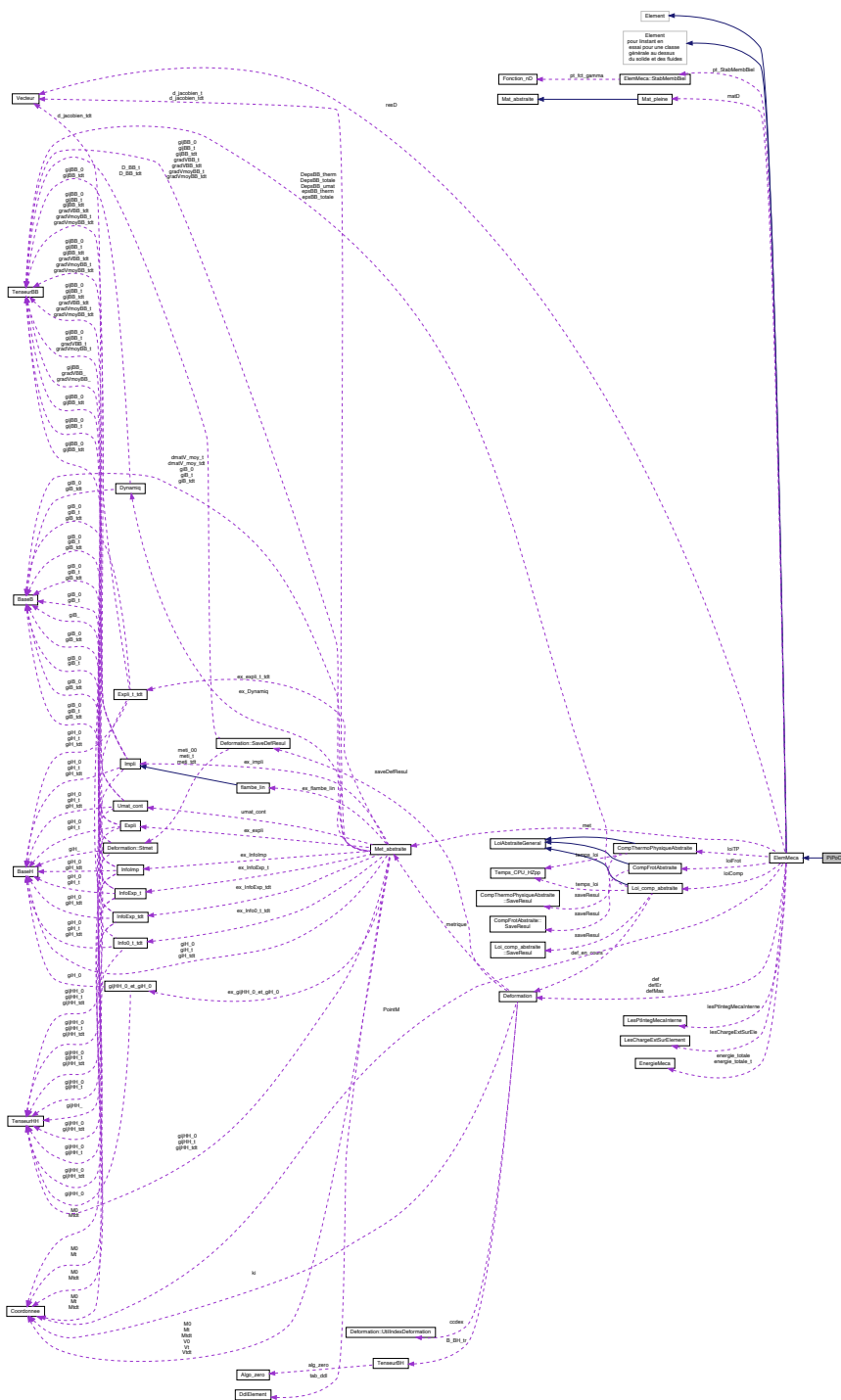
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp_↔cm9pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaQComp_↔cm9pti.cc`

## 6.628 Référence de la classe PiPoCo

Graphe d'héritage de PiPoCo:



Graphe de collaboration de PiPoCo:



### Fonctions membres publiques

- PiPoCo (int num\_mail=0, int num\_id=-3)
- PiPoCo (int num\_mail, int num\_id, const Tableau< Noeud \* > &tab)
- PiPoCo (int num\_mail, int num\_id, Enum\_interpol id\_interp\_elt, Enum\_geom id\_geom\_elt)
- PiPoCo (int num\_mail, int num\_id, char \*nom\_interpol, char \*nom\_geom)
- PiPoCo (int num\_mail, int num\_id, const Tableau< Noeud \* > &tab, Enum\_interpol id\_interp\_elt, Enum\_geom id\_geom\_elt)
- PiPoCo (int num\_mail, int num\_id, const Tableau< Noeud \* > &tab, char \*nom\_interpol, char \*nom\_geom)

- **PiPoCo** (const [PiPoCo](#) &ps)
- bool **SurfExiste** (int) const
- bool **AreteExiste** (int na) const
- int **Interne\_0** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- int **Interne\_t** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- int **Interne\_tdt** (const [Coordonnee](#) &M, [Coordonnee](#) \*coor\_locales=NULL)
- virtual int **Nb\_pt\_int\_epai** ()=0
- virtual int **Nb\_pt\_int\_surf** ()=0
- virtual double **H** ()=0
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#), [List\\_io](#)< [TypeQuelconque](#) > &, int)
- [ElemMeca::MatGeomInit](#) **MatricesGeometrique\_Et\_Initiale** (const [ParaAlgoControle](#) &pa)
- virtual double **KSlepais** (int i)=0

## Fonctions membres protégées

- void **Cal\_implicitPiPoCo** ([DdlElement](#) &tab\_ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > d2\_epsBB, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, int nbintS, [Vecteur](#) &poidsS, int nbintH, [Vecteur](#) &poidsH, const [ParaAlgoControle](#) &pa, bool cald\_Dvirtuelle)
- void **Cal\_explicitPiPoCo** ([DdlElement](#) &tab\_ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, int nbintS, [Vecteur](#) &poidsS, int nbintH, [Vecteur](#) &poidsH, const [ParaAlgoControle](#) &pa, bool atdt)
- void **Cal\_matGeom\_InitPiPoCo** ([Mat\\_pleine](#) &matGeom, [Mat\\_pleine](#) &matInit, [DdlElement](#) &tab\_ddl, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [Tableau](#)< [Tableau2](#)< [TenseurBB](#) \* > > d2\_epsBB, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, int nbintS, [Vecteur](#) &poidsS, int nbintH, [Vecteur](#) &poidsH, const [ParaAlgoControle](#) &pa, bool cald\_Dvirtuelle)

## Membres hérités additionnels

### 6.628.1 Documentation des fonctions membres

#### 6.628.1.1 [MatricesGeometrique\\_Et\\_Initiale\(\)](#)

```
ElemMeca::MatGeomInit PiPoCo::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

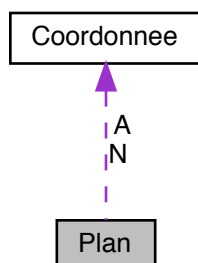
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/PiPoCo.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/PiPoCo.cc

## 6.629 Référence de la classe Plan

Graphe de collaboration de Plan:



### Fonctions membres publiques

- **Plan** (const [Coordonnee](#) &B, const [Coordonnee](#) &vec)
- **Plan** (int dim)
- **Plan** (const [Plan](#) &a)
- **Plan** & **operator=** (const [Plan](#) &P)
- const [Coordonnee](#) & **PointPlan** () const
- const [Coordonnee](#) & **Vecplan** () const
- void **Change\_dim** (int dima)
- void **Change\_ptref** (const [Coordonnee](#) &B)
- void **Change\_normal** (const [Coordonnee](#) &vec)
- void **change\_donnees** (const [Coordonnee](#) &B, const [Coordonnee](#) &vec)
- int **Intersection** (const [Droite](#) &D, [Coordonnee](#) &M) const
- double **Distance\_au\_plan** (const [Coordonnee](#) &M) const
- [Coordonnee](#) **Projete** (const [Coordonnee](#) &M) const
- bool **DuMemeCote** (const [Coordonnee](#) &M1, const [Coordonnee](#) &M2) const

### Attributs protégés

- [Coordonnee](#) **A**
- [Coordonnee](#) **N**

### Amis

- istream & **operator>>>** (istream &, [Plan](#) &)
- ostream & **operator<<<** (ostream &, const [Plan](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Plan.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Plan.cc

## 6.630 Référence de la classe Hyper3D::PoGrenoble\_V

### Attributs publics

- double **E**
- double **EV**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h

## 6.631 Référence de la classe Hyper3D::PoGrenoble\_VV

### Attributs publics

- double **E**
- double **EV**
- double **EVV**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h

## 6.632 Référence de la classe Hyper3D::PoGrenobleAvecPhaseAvecVar

### Attributs publics

- double **E**
- double **EV**
- double **EQ**
- double **Ecos3phi**
- double **EVV**
- double **EQV**
- double **EQQ**
- double **EVcos3phi**
- double **EQcos3phi**
- double **Ecos3phi2**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h

## 6.633 Référence de la classe Hyper3D::PoGrenobleAvecPhaseSansVar

### Attributs publics

- double **E**
- double **EV**
- double **EQ**
- double **Ecos3phi**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h

## 6.634 Référence de la classe Hyper3D::PoGrenobleSansPhaseAvecVar

### Attributs publics

- double **E**
- double **EV**
- double **EQ**
- double **EVV**
- double **EQV**
- double **EQQ**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h



## 6.635 Référence de la classe Hyper3D::PoGrenobleSansPhaseSansVar

### Attributs publics

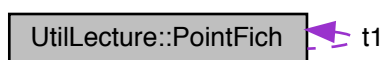
- double **E**
- double **EV**
- double **EQ**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper3D.h

## 6.636 Référence de la classe UtilLecture::PointFich

Grappe de collaboration de UtilLecture::PointFich:



### Fonctions membres publiques

- **PointFich** ([PointFich](#) \*x1, ifstream \*x2, string n)
- **PointFich** (const [PointFich](#) &a)
- [PointFich](#) & **operator=** (const [PointFich](#) &a)

### Attributs publics

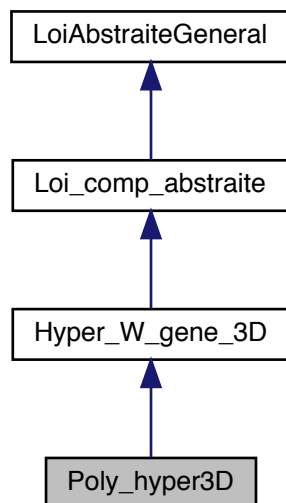
- [PointFich](#) \* **t1**
- string **nom**
- ifstream \* **sauvePtr**
- int **numLigne**

La documentation de cette classe a été générée à partir du fichier suivant :

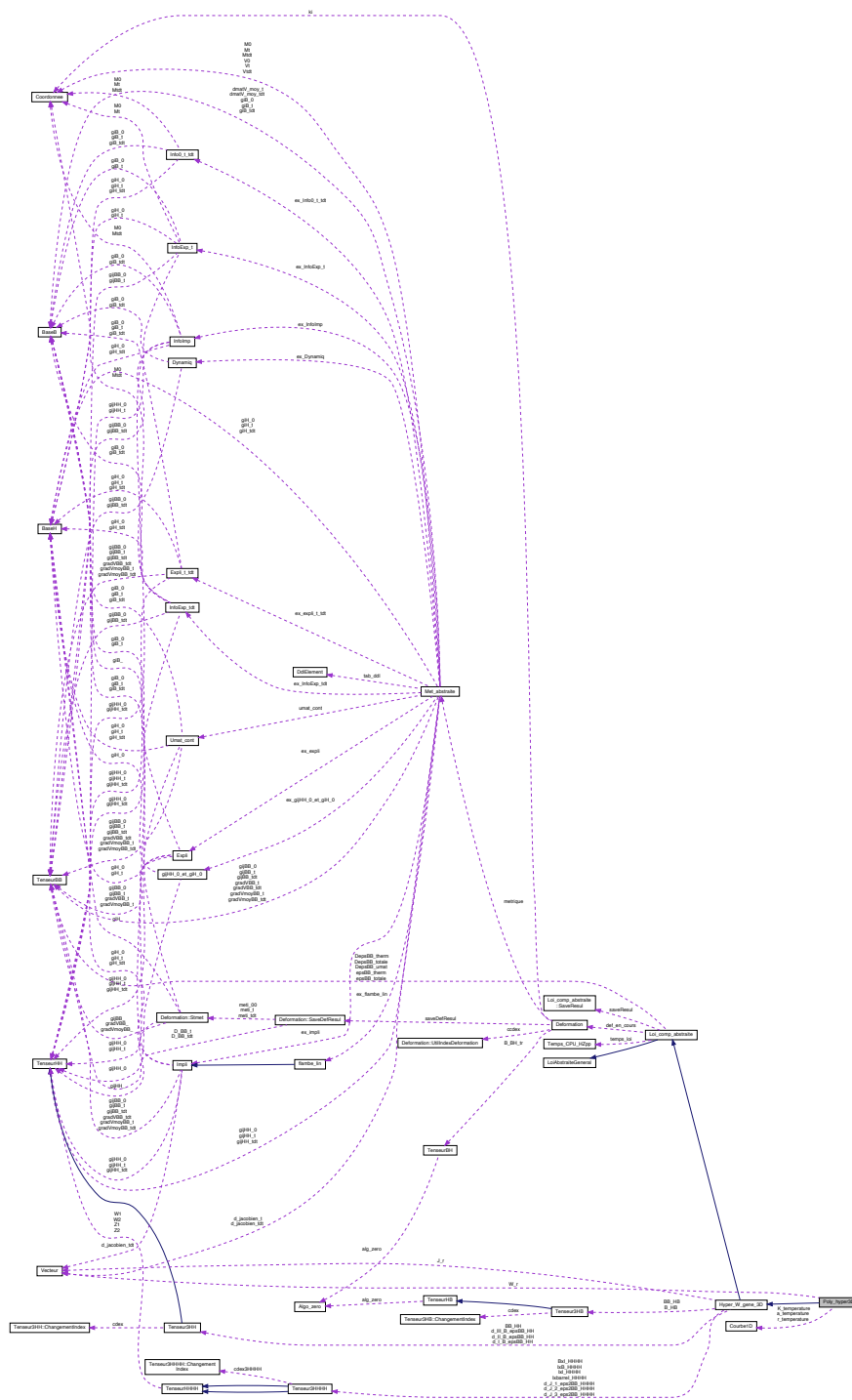
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.h

## 6.637 Référence de la classe Poly\_hyper3D

Graphe d'héritage de Poly\_hyper3D:



Graphe de collaboration de Poly\_hyper3D:



## Fonctions membres publiques

- `Poly_hyper3D` (const `Poly_hyper3D` &loi)
- void `LectureDonneesParticulieres` (`UtilLecture` \*, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComple` ()
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)

- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- double [Module\\_young\\_equivalent](#) (Enum\_dure, const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gij↵  
BB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_,  
[TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double  
&jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_↵  
compressibilite, double &module\_cisaillement, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#)  
&giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#)  
&giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_eps↵  
BB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* >  
&d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#)  
&d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const  
[EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_↵  
abstraite](#)::[Impli](#) &ex)
- void [Calcul\\_dsigma\\_deps](#) (bool en\_base\_orthonormee, [TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB,  
[TenseurBB](#) &epsBB\_tdt, [TenseurBB](#) &delta\_epsBB, double &jacobien\_0, double &jacobien, [TenseurHH](#)  
&sigHH, [TenseurHHHH](#) &d\_sigma\_deps, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double  
&module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite](#)::[Umat\\_cont](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const  
[ThermoDonnee](#) &, const [Met\\_abstraite](#)::[Impli](#) \*ex\_impli, const [Met\\_abstraite](#)::[Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const  
[Met\\_abstraite](#)::[Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const  
[TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- double **K**
- [Tableau](#)< [Tableau](#)< double > > **Cij**
- [Tableau](#)< [Tableau](#)< [Courbe1D](#) \* > > **Cij\_temperature**
- [Courbe1D](#) \* **K\_temperature**
- int **type\_pot\_vol**
- bool **avec\_courbure**
- double **a\_courbure**
- double **r\_courbure**
- [Courbe1D](#) \* **a\_temperature**
- [Courbe1D](#) \* **r\_temperature**
- double **W\_d**
- double **W\_v**
- [Vecteur](#) **W\_r**
- double **W\_d\_J1**
- double **W\_d\_J2**
- double **W\_d\_J1\_2**
- double **W\_d\_J1\_J2**
- double **W\_d\_J2\_2**
- double **W\_v\_J3**
- double **W\_v\_J3J3**
- [Tableau2](#)< double > **W\_rs**
- double **W\_c**
- double **W\_c\_J1**
- double **W\_c\_J3**
- double **W\_c\_J1\_2**
- double **W\_c\_J3\_2**
- double **W\_c\_J1\_J3**

## Membres hérités additionnels

### 6.637.1 Documentation des fonctions membres

#### 6.637.1.1 Affiche()

void Poly\_hyper3D::Affiche ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.637.1.2 Calcul\_dsigma\_deps()

```
void Poly_hyper3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
Tenseur3HHHH d_sigma_depsHHHH; d_sigma_depsHHHH.TransfertDunTenseurGeneral(dSigdepsHHHH.↔
Symetrise1et2_3et4());
Réimplémentée à partir de Loi\_comp\_abstraite.
```

#### 6.637.1.3 Calcul\_DsigmaHH\_tdt()

```
void Poly_hyper3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
```

```

    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.637.1.4 Calcul\_SigmaHH()

```

void Poly_hyper3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.637.1.5 CalculGrandeurTravail()

```

virtual void Poly_hyper3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * excludre_dd_etend,
    const List_io< const TypeQuelconque * > * excludre_Q ) [inline], [protected],
[virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.637.1.6 Ecriture\_base\_info\_loi()

```

void Poly_hyper3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

### 6.637.1.7 HsurH0()

```
virtual double Poly_hyper3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.637.1.8 Info\_commande\_LoisDeComp()

```
void Poly_hyper3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.637.1.9 Lecture\_base\_info\_loi()

```
void Poly_hyper3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.637.1.10 LectureDonneesParticulieres()

```
void Poly_hyper3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

### 6.637.1.11 Module\_young\_equivalent()

```
double Poly_hyper3D::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.637.1.12 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * Poly_hyper3D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.637.1.13 TestComplet()

```
int Poly_hyper3D::TestComplet ( ) [virtual]
```

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

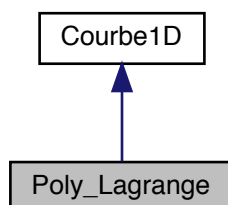
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Poly\_hyper3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Poly\_hyper3D.c

## 6.638 Référence de la classe Poly\_Lagrange

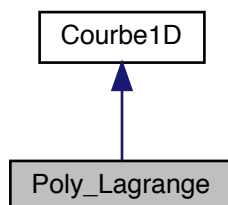
Classe permettant le calcul d'un polynôme 1D à l'aide d'une interpolation de Lagrange.

```
#include <Poly_Lagrange.h>
```

Graphe d'héritage de Poly\_Lagrange:



Graphe de collaboration de Poly\_Lagrange:



### Fonctions membres publiques

- **Poly\_Lagrange** (string nom="")
- **Poly\_Lagrange** (Tableau< Coordonnee2 > &pt, string nom="")
- **Poly\_Lagrange** (const Poly\_Lagrange &Co)
- **Poly\_Lagrange** (const Courbe1D &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Complet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, UtilLecture \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** (UtilLecture &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- **Courbe1D::ValDer Valeur\_Et\_derivee** (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*



- `Courbe1D::ValDer2 Valeur_Et_der12` (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double `Der_sec` (double x)  
*ramène la dérivée seconde*
- `Courbe1D::Valbool Valeur_stricte` (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- `Courbe1D::ValDerbool Valeur_Et_derivee_stricte` (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void `Change_tabPoints` (`Tableau< Coordonnee2 > &pt`)
- void `Lecture_base_info` (`ifstream &ent`, `const int cas`)  
*--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void `Ecriture_base_info` (`ofstream &sort`, `const int cas`)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- virtual void `SchemaXML_Courbes1D` (`ofstream &sort`, `const Enum_IO_XML enu`)  
*sortie du schemaXML: en fonction de enu*
- int `NombrePoint` ()
- const `Coordonnee2 & Pt_nbi` (int i)

### Attributs protégés

- `Tableau< Coordonnee2 > points`
- double `der_init`
- double `der_finale`
- int `indice_precedant`

### Membres hérités additionnels

#### 6.638.1 Description détaillée

Classe permettant le calcul d'un polynôme 1D à l'aide d'une interpolation de Lagrange.

BUT: Classe permettant le calcul d'un polynôme 1D à l'aide d'une interpolation de Lagrange ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

Auteur

Gérard Rio

Version

1.0

Date

19/01/2001

#### 6.638.2 Documentation des fonctions membres

##### 6.638.2.1 Affiche()

```
void Poly_Lagrange::Affiche ( ) const [virtual]
```

affichage de la courbe

Implémente `Courbe1D`.

### 6.638.2.2 Complet\_courbe()

```
bool Poly_Lagrange::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

### 6.638.2.3 Der\_sec()

```
double Poly_Lagrange::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

### 6.638.2.4 Derivee()

```
double Poly_Lagrange::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

### 6.638.2.5 Ecriture\_base\_info()

```
void Poly_Lagrange::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.638.2.6 Info\_commande\_Courbes1D()

```
void Poly_Lagrange::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

### 6.638.2.7 LectDonnParticulieres\_courbes()

```
void Poly_Lagrange::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.  
Implémente [Courbe1D](#).

### 6.638.2.8 Lecture\_base\_info()

```
void Poly_Lagrange::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

**6.638.2.9 SchemaXML\_Courbes1D()**

```
void Poly_Lagrange::SchemaXML_Courbes1D (
    ostream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

**6.638.2.10 Valeur()**

```
double Poly_Lagrange::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

**6.638.2.11 Valeur\_Et\_der12()**

```
Courbe1D::ValDer2 Poly_Lagrange::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

**6.638.2.12 Valeur\_Et\_derivee()**

```
Courbe1D::ValDer Poly_Lagrange::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

**6.638.2.13 Valeur\_Et\_derivee\_stricte()**

```
Courbe1D::ValDerbool Poly_Lagrange::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

**6.638.2.14 Valeur\_stricte()**

```
Courbe1D::Valbool Poly_Lagrange::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Poly\_Lagrange.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Poly\_Lagrange.cc

**6.639 Référence de la classe Ponderation**

une seconde classe de travail qui permet d'utiliser une pondération qui dépend de n [Courbe1D](#), chacune fonction d'un ddl étendu

```
#include <Ponderation.h>
```

## Fonctions membres publiques

- **Ponderation** (const [Ponderation](#) &a)
- void **LectureDonneesPonderation\_uneCourbe** ([Ddl\\_enum\\_etendu](#) ddl, [UtilLecture](#) \*entreePrinc, [LesCourbes1D](#) &lesCourbes1D)
- void **LectureDonneesPonderation** (const [List\\_io](#)< string > &list\_id\_ddl\_etendu, [UtilLecture](#) \*entreePrinc, [LesCourbes1D](#) &lesCourbes1D)
- void **Affectation\_fonctions** ([LesCourbes1D](#) &lesCourbes1D)
- [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & **Type\_grandeur** ()
- const [Tableau](#)< [Ddl\\_enum\\_etendu](#) > & **Const\_Type\_grandeur** () const
- [Tableau](#)< bool > & **Valeur\_aux\_noeuds** ()
- const [Tableau](#)< bool > & **Const\_Valeur\_aux\_noeuds** () const
- [Tableau](#)< [Courbe1D](#) \* > & **C\_proport** ()
- const [Tableau](#)< [Courbe1D](#) \* > & **Const\_C\_proport** () const
- [Tableau](#)< string > & **Tab\_nom\_fonction** ()
- void **Verif\_complet** () const
- void **Affiche** ()
- void **Activation\_donnees** ([Tableau](#)< [Noeud](#) \* > &tabnoeud, bool dilatation, [LesPtIntegMecalInterne](#) &les←PtMecalInt)
- double **CalculPonderMultiplicatif** (const [PtIntegMecalInterne](#) &ptintmeca, const [Deformation](#) &def, [Enum\\_dure](#) temps, const [ThermoDonnee](#) &dTP)
- void **Lecture\_base\_info** (ifstream &ent, const int cas, [LesCourbes1D](#) &lesCourbes1D)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **SchemaXML\_Fonctions\_nD** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)

## Attributs protégés

- [Tableau](#)< [Ddl\\_enum\\_etendu](#) > **type\_grandeur**
- [Tableau](#)< bool > **valeur\_aux\_noeuds**
- [Tableau](#)< [Courbe1D](#) \* > **c\_proport**
- [Tableau](#)< string > **tab\_nom\_fonction**

### 6.639.1 Description détaillée

une seconde classe de travail qui permet d'utiliser une pondération qui dépend de n [Courbe1D](#), chacune fonction d'un ddl étendu

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.cc

## 6.640 Référence de la classe [Ponderation\\_Consultable](#)

une classe de travail, qui permet d'utiliser une pondération qui dépend de plusieurs [courbe1D](#), chacune dépendant d'une grandeur consultable par un string

```
#include <Ponderation.h>
```

## Fonctions membres publiques

- **Ponderation\_Consultable** (const [Ponderation\\_Consultable](#) &a)
- void **Affectation\_fonctions** ([LesCourbes1D](#) &lesCourbes1D)
- [Tableau](#)< string > & **Type\_grandeur\_Consultable** ()
- [Tableau](#)< [Courbe1D](#) \* > & **C\_proport** ()
- [Tableau](#)< string > & **Tab\_nom\_fonction** ()
- void **Verif\_complet** () const
- void **Lecture\_base\_info** (ifstream &ent, const int cas, [LesCourbes1D](#) &lesCourbes1D)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **SchemaXML\_Fonctions\_nD** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)

## Attributs protégés

- [Tableau](#)< string > **type\_grandeur\_Consultable**
- [Tableau](#)< [Courbe1D](#) \* > **c\_proport**
- [Tableau](#)< string > **tab\_nom\_fonction**

### 6.640.1 Description détaillée

une classe de travail, qui permet d'utiliser une pondération qui dépend de plusieurs courbe1D, chacune dépendant d'une grandeur consultable par un string

La documentation de cette classe a été générée à partir du fichier suivant :

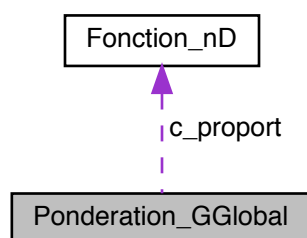
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.cc

## 6.641 Référence de la classe Ponderation\_GGlobal

une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble de grandeurs globales au travers d'une fonction nD

```
#include <Ponderation.h>
```

Graphes de collaboration de Ponderation\_GGlobal:



### Fonctions membres publiques

- **Ponderation\_GGlobal** (const [Ponderation\\_GGlobal](#) &a)
- void **LecturePonderation** (const [List\\_io](#)< string > &grandeurs, [UtilLecture](#) \*entreePrinc, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void **Affectation\_fonctions** ([LesFonctions\\_nD](#) &lesFonctionsnD)
- string & **Nom\_fonction** ()
- [Fonction\\_nD](#) \* **C\_proport** ()
- void **Assigne\_proport** ([Fonction\\_nD](#) \*pt)
- void **Verif\_complet** () const
- void **Affiche** ()
- void **Lecture\_base\_info** (ifstream &ent, const int cas, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas, bool sans\_courbe=false)
- void **SchemaXML\_Fonctions\_nD** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)

### Attributs protégés

- string **nom\_fonction**
- [Fonction\\_nD](#) \* **c\_proport**

### 6.641.1 Description détaillée

une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble de grandeurs globales au travers d'une fonction nD

La documentation de cette classe a été générée à partir du fichier suivant :

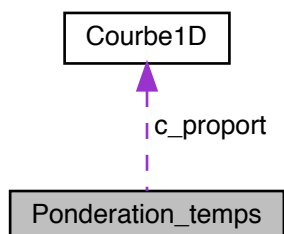
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.cc

## 6.642 Référence de la classe Ponderation\_temps

une classe de travail qui permet d'utiliser une pondération qui dépend du temps via une [Courbe1D](#)

```
#include <Ponderation.h>
```

Graphe de collaboration de Ponderation\_temps:



### Fonctions membres publiques

- **Ponderation\_temps** (const [Ponderation\\_temps](#) &a)
- void **LectureDonneesPonderation** ([UtilLecture](#) \*entreePrinc, [LesCourbes1D](#) &lesCourbes1D)
- void **Affectation\_fonctions** ([LesCourbes1D](#) &lesCourbes1D)
- string & **Nom\_fonction** ()
- [Courbe1D](#) \* **C\_proport** ()
- void **Verif\_complet** () const
- void **Affiche** ()
- double **CalculPonder** ()
- void **Lecture\_base\_info** (ifstream &ent, const int cas, [LesCourbes1D](#) &lesCourbes1D)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas, bool sans\_courbe=false)
- void **SchemaXML\_Fonctions\_nD** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)

### Attributs protégés

- string **nom\_fonction**
- [Courbe1D](#) \* **c\_proport**

#### 6.642.1 Description détaillée

une classe de travail qui permet d'utiliser une pondération qui dépend du temps via une [Courbe1D](#)

La documentation de cette classe a été générée à partir du fichier suivant :

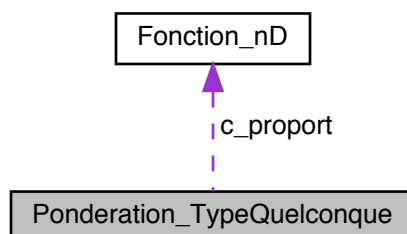
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.cc

## 6.643 Référence de la classe Ponderation\_TypeQuelconque

une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble de grandeurs quelconque au travers d'une fonction nD

```
#include <Ponderation.h>
```

Grappe de collaboration de Ponderation\_TypeQuelconque:



## Fonctions membres publiques

- **Ponderation\_TypeQuelconque** (const [Ponderation\\_TypeQuelconque](#) &a)
- void **LecturePonderation** (const [List\\_io](#)< string > &grandeurs, [UtilLecture](#) \*entreePrinc, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void **Affectation\_fonctions** ([LesFonctions\\_nD](#) &lesFonctionsnD)
- string & **Nom\_fonction** ()
- [Fonction\\_nD](#) \* **C\_proport** ()
- void **Assigne\_proport** ([Fonction\\_nD](#) \*pt)
- void **Verif\_complet** () const
- void **Lecture\_base\_info** (ifstream &ent, const int cas, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas, bool sans\_fct=false)
- void **SchemaXML\_Fonctions\_nD** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)

## Attributs protégés

- string **nom\_fonction**
- [Fonction\\_nD](#) \* **c\_proport**
- [List\\_io](#)< [EnumTypeQuelconque](#) > **type\_grandeur\_Quelconque**
- [Tableau](#)< double > **tab\_argument**

### 6.643.1 Description détaillée

une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble de grandeurs quelconque au travers d'une fonction nD

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Ponderation.cc

## 6.644 Référence de la classe Posi\_ddl\_noeud

[Posi\\_ddl\\_noeud](#): un conteneur qui associe numéro de noeud, de maillage, et enu ddl.

```
#include <Noeud.h>
```

## Fonctions membres publiques

- **Posi\_ddl\_noeud** (int aa, [Enum\\_ddl](#) enuu, int bb)
- **Posi\_ddl\_noeud** (const [Posi\\_ddl\\_noeud](#) &a)
- [Posi\\_ddl\\_noeud](#) & **operator=** (const [Posi\\_ddl\\_noeud](#) &a)
- bool **operator==** (const [Posi\\_ddl\\_noeud](#) &a)

- bool **operator!=** (const [Posi\\_ddl\\_noeud](#) &a)
- void **Affiche** (ofstream &sort) const
- int & **Nb\_maillage** ()
- int & **Nb\_noeud** ()
- Enum\_ddl & **Enu** ()
- const int & **Const\_Nb\_maillage** () const
- const int & **Const\_Nb\_noeud** () const
- const Enum\_ddl & **Const\_Enu** () const

### Attributs protégés

- int **nb\_maillage**
- int **nb\_noeud**
- Enum\_ddl **enu**

### Amis

- istream & **operator>>** (istream &, [Posi\\_ddl\\_noeud](#) &)
- ostream & **operator<<** (ostream &, const [Posi\\_ddl\\_noeud](#) &)

### 6.644.1 Description détaillée

[Posi\\_ddl\\_noeud](#): un conteneur qui associe numéro de noeud, de maillage, et enu ddl.

#### Auteur

Gérard Rio

#### Version

1.0



Date

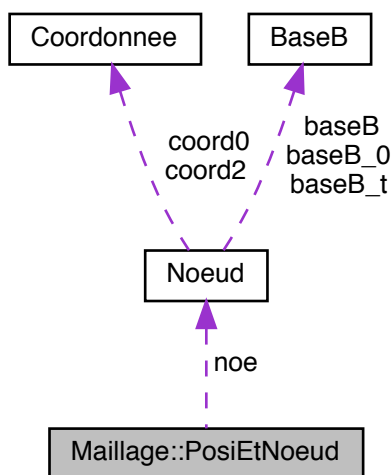
23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Noeud.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Noeud.cc

## 6.645 Référence de la classe Maillage::PosiEtNoeud

Graphes de collaboration de Maillage::PosiEtNoeud:



### Fonctions membres publiques

- **PosiEtNoeud** ([Noeud](#) \*no, [Element](#) \*e)
- **PosiEtNoeud** (const [PosiEtNoeud](#) &po)
- [PosiEtNoeud](#) & **operator=** (const [PosiEtNoeud](#) &po)
- bool **operator==** (const [PosiEtNoeud](#) &po) const
- bool **operator!=** (const [PosiEtNoeud](#) &po) const
- bool **operator<** (const [PosiEtNoeud](#) &po) const
- bool **operator<=** (const [PosiEtNoeud](#) &po) const
- bool **operator>** (const [PosiEtNoeud](#) &po) const
- bool **operator>=** (const [PosiEtNoeud](#) &po) const

### Attributs publics

- [Noeud](#) \* **noe**
- [Element](#) \* **el**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/Maillage2.cc

## 6.646 Référence de la classe UtilLecture::Position\_BI

### Fonctions membres publiques

- **Position\_BI** (streampos posi=(ios::beg), int num=0)

- **Position\_BI** (const [Position\\_BI](#) &a)
- bool **operator==** (const [Position\\_BI](#) &a) const
- bool **operator>** (const [Position\\_BI](#) &a) const
- bool **operator<** (const [Position\\_BI](#) &a) const
- [Position\\_BI](#) & **operator=** (const [Position\\_BI](#) &a)
- bool **operator!=** (const [Position\\_BI](#) &a)

### Attributs publics

- streampos **position**
- int **num\_incr**

### Amis

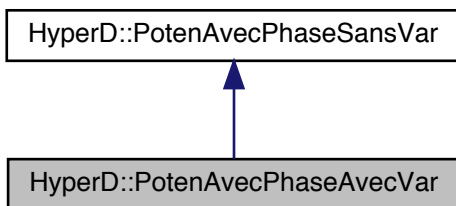
- istream & **operator>>** (istream &entree, [Position\\_BI](#) &a)
- ostream & **operator<<** (ostream &sort, const [Position\\_BI](#) &a)

La documentation de cette classe a été générée à partir du fichier suivant :

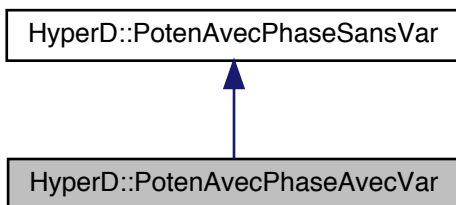
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.h

## 6.647 Référence de la classe `HyperD::PotenAvecPhaseAvecVar`

Grphe d'héritage de `HyperD::PotenAvecPhaseAvecVar`:



Grphe de collaboration de `HyperD::PotenAvecPhaseAvecVar`:



### Fonctions membres publiques

- **PotenAvecPhaseAvecVar** (const [PotenAvecPhaseAvecVar](#) &a)
- [PotenAvecPhaseAvecVar](#) & **operator=** (const [PotenAvecPhaseAvecVar](#) &a)

### Attributs publics

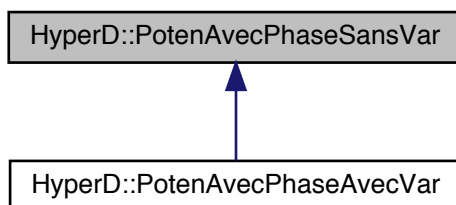
- double **EVV**
- double **EblIb2**
- double **Eleps2**
- double **EblIbV**
- double **ElepsV**
- double **EblIbleps**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.648 Référence de la classe HyperD::PotenAvecPhaseSansVar

Graphes d'héritage de HyperD::PotenAvecPhaseSansVar:



### Fonctions membres publiques

- **PotenAvecPhaseSansVar** (const [PotenAvecPhaseSansVar](#) &a)
- **PotenAvecPhaseSansVar & operator=** (const [PotenAvecPhaseSansVar](#) &a)

### Attributs publics

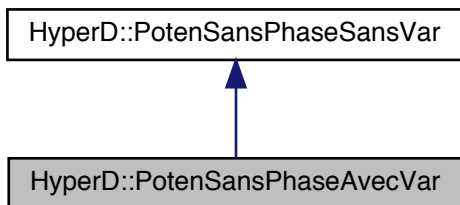
- double **E**
- double **EV**
- double **EblIb**
- double **Eleps**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

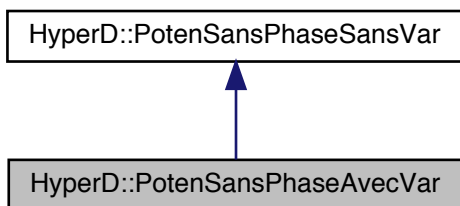
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.649 Référence de la classe HyperD::PotenSansPhaseAvecVar

Graphe d'héritage de HyperD::PotenSansPhaseAvecVar:



Graphe de collaboration de HyperD::PotenSansPhaseAvecVar:



### Fonctions membres publiques

- **PotenSansPhaseAvecVar** (const [PotenSansPhaseAvecVar](#) &a)
- [PotenSansPhaseAvecVar](#) & **operator=** (const [PotenSansPhaseAvecVar](#) &a)

### Attributs publics

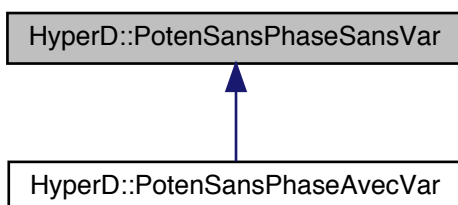
- double **EVV**
- double **Eb1lbV**
- double **Eb1lb2**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.650 Référence de la classe HyperD::PotenSansPhaseSansVar

Grphe d'héritage de HyperD::PotenSansPhaseSansVar:



### Fonctions membres publiques

- **PotenSansPhaseSansVar** (const [PotenSansPhaseSansVar](#) &a)
- [PotenSansPhaseSansVar](#) & **operator=** (const [PotenSansPhaseSansVar](#) &a)

### Attributs publics

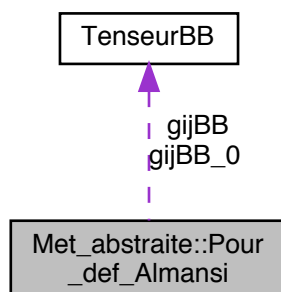
- double **E**
- double **EV**
- double **Ebllb**
- double **Ks**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h

## 6.651 Référence de la classe Met\_abstraite::Pour\_def\_Almansi

Grphe de collaboration de Met\_abstraite::Pour\_def\_Almansi:



### Fonctions membres publiques

- **Pour\_def\_Almansi** (const [Pour\\_def\\_Almansi](#) &a)

## Attributs publics

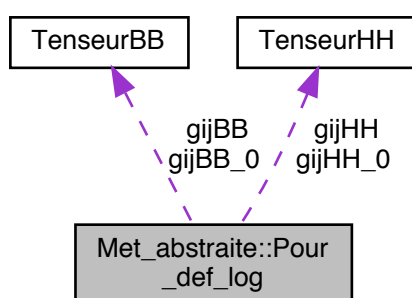
- [TenseurBB](#) \* [gijBB](#)
- [TenseurBB](#) \* [gijBB\\_0](#)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔ abstraite.h

## 6.652 Référence de la classe Met\_abstraite::Pour\_def\_log

Graphe de collaboration de Met\_abstraite::Pour\_def\_log:



## Fonctions membres publiques

- `Pour_def_log` (const [Pour\\_def\\_log](#) &a)

## Attributs publics

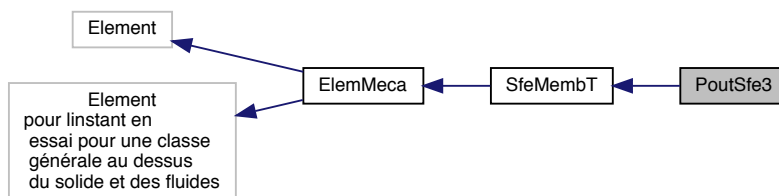
- [TenseurBB](#) \* [gijBB](#)
- [TenseurHH](#) \* [gijHH](#)
- [TenseurBB](#) \* [gijBB\\_0](#)
- [TenseurHH](#) \* [gijHH\\_0](#)

La documentation de cette classe a été générée à partir du fichier suivant :

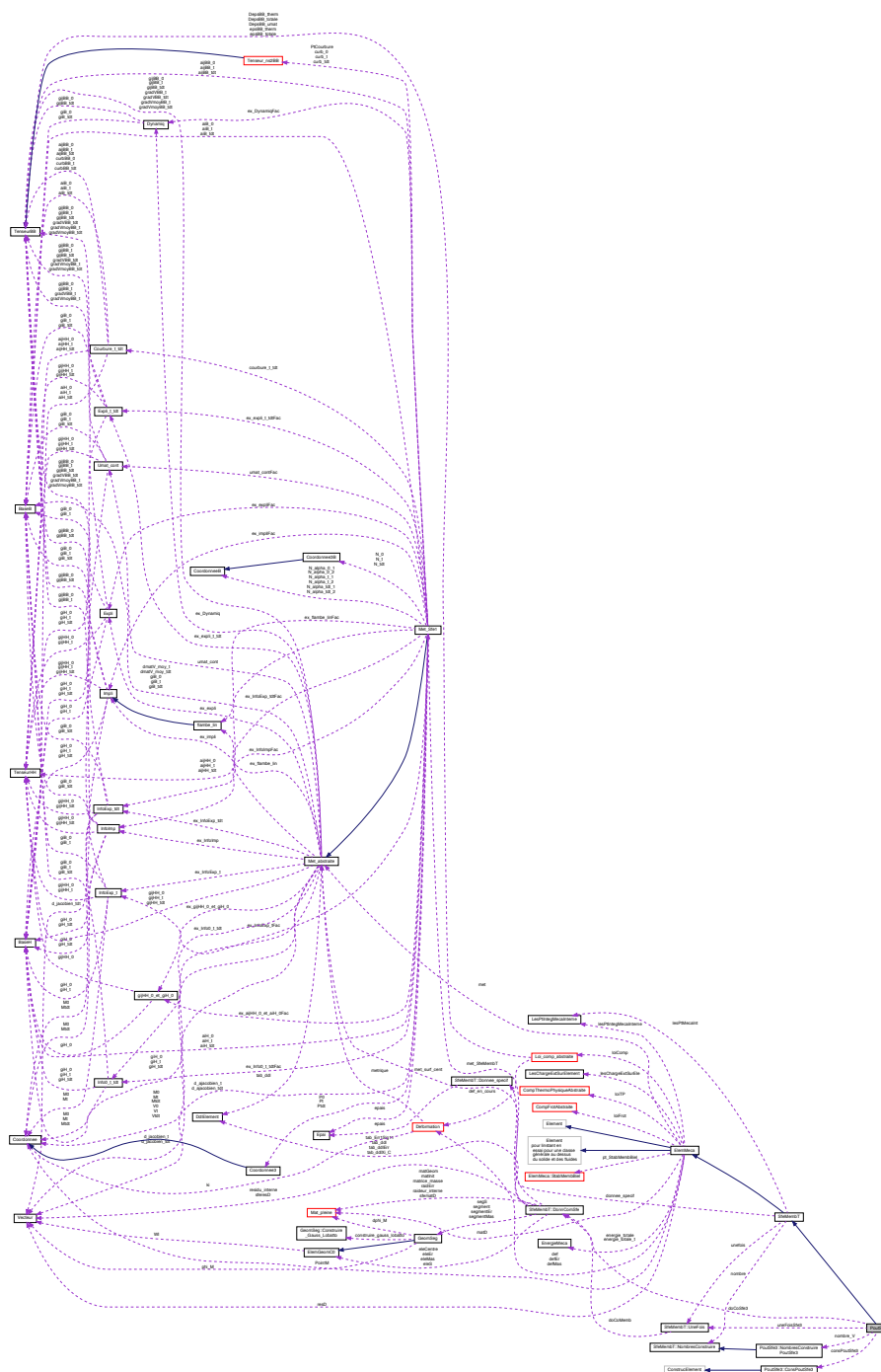
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔ abstraite.h

## 6.653 Référence de la classe PoutSfe3

Graphe d'héritage de PoutSfe3:



Graphe de collaboration de PoutSfe3:



## Classes

- class [ConsPoutSfe3](#)
- class [NombresConstruirePoutSfe3](#)

## Fonctions membres publiques

- **PoutSfe3** (double epaiss, int num\_mail=0, int num\_id=-3)
- **PoutSfe3** (int num\_mail, int num\_id)
- **PoutSfe3** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)



- **PoutSfe3** (const [PoutSfe3](#) &Pout)
- Element \* **Nevez\_copie** () const
- [PoutSfe3](#) & **operator=** ([PoutSfe3](#) &Pout)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- double **KSI** (int i)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [SfeMembT::DonnComSfe](#) \* **doCoSfe3** = NULL
- static [SfeMembT::UneFois](#) **uneFoisSfe3**
- static [NombresConstruirePoutSfe3](#) **nombre\_V**
- static [ConsPoutSfe3](#) **consPoutSfe3**

## Membres hérités additionnels

### 6.653.1 Documentation des fonctions membres

#### 6.653.1.1 KSI()

```
double PoutSfe3::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

#### 6.653.1.2 new\_frontiere\_lin()

```
ElFrontiere * PoutSfe3::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.653.1.3 new\_frontiere\_surf()

```
ElFrontiere * PoutSfe3::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

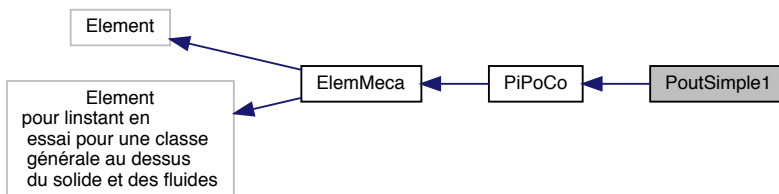
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

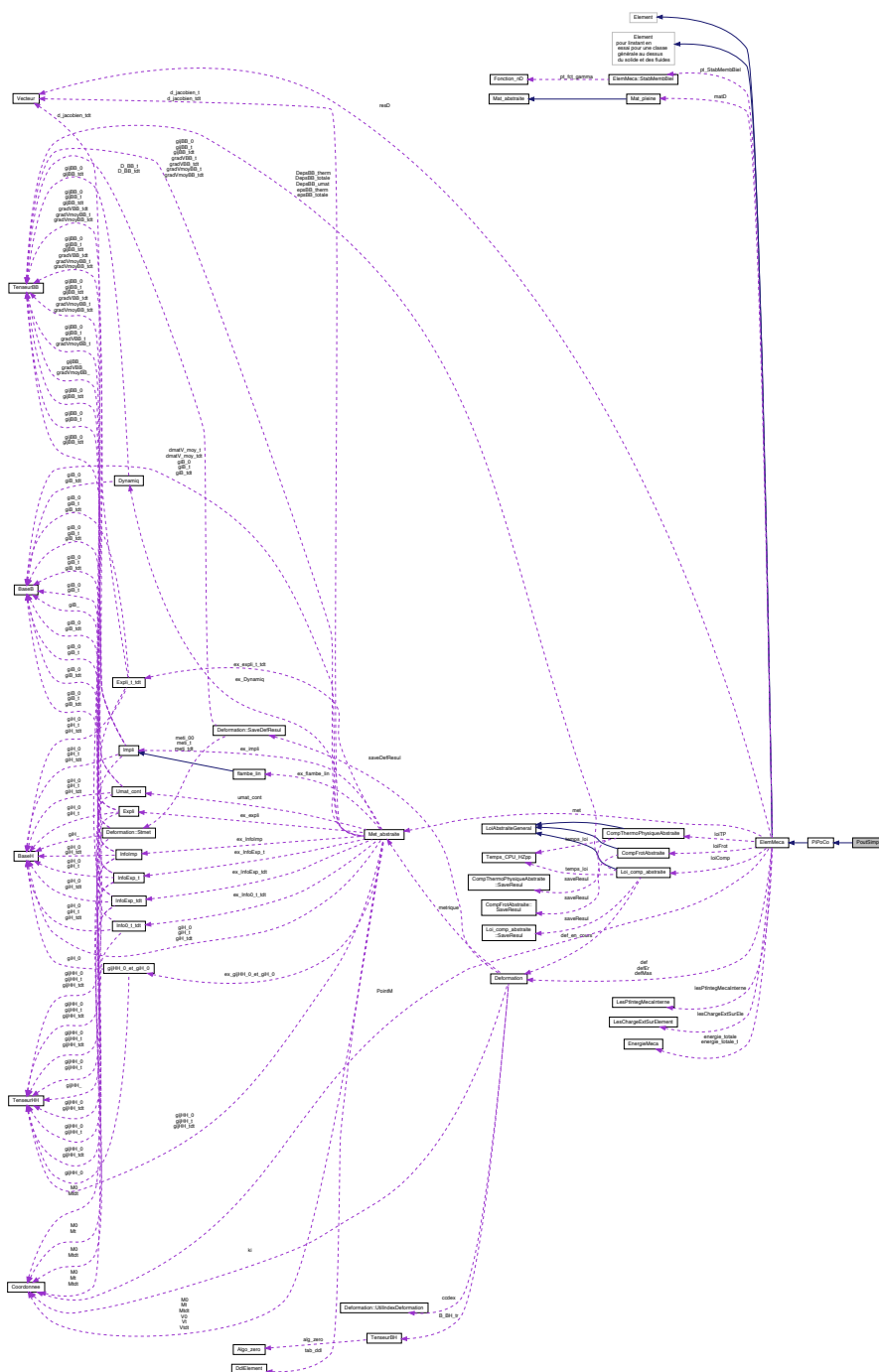
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/PoutSfe3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/PoutSfe3.cc

## 6.654 Référence de la classe PoutSimple1

Graphe d'héritage de PoutSimple1:



Graphe de collaboration de PoutSimple1:



### Fonctions membres publiques

- **PoutSimple1** (double epais, double larg, int num\_mail=0, int num\_id=-3)
- **PoutSimple1** (int num\_mail, int num\_id)
- **PoutSimple1** (double epais, double larg, int num\_mail, int num\_id, const **Tableau**< **Noeud** \* > &tab)
- **PoutSimple1** (const **PoutSimple1** &biel)
- **Element** \* **Nevez\_copie** () const
- **PoutSimple1** & **operator=** (**PoutSimple1** &pout)
- void **LectureDonneesParticulières** (**UtilLecture** \*, **Tableau**< **Noeud** \* > \*)
- void **Info\_com\_Element** (**UtilLecture** \*entreePrinc, string &ordre, **Tableau**< **Noeud** \* > \*tabMaillageNoeud)
- **Element::ResRaid Calcul\_implicit** (const **ParaAlgoControle** &pa)

- [Vecteur](#) \* [CalculResidu\\_t](#) (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* [CalculResidu\\_tdt](#) (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* [CalculMatriceMasse](#) ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double [Long\\_arrete\\_mini\\_sur\\_c](#) ([Enum\\_dure](#) temps)
- const [DdlElement](#) & [TableauDdl](#) () const
- void [TdtversT](#) ()
- void [TversTdt](#) ()
- void [Libere](#) ()
- void [DefLoi](#) ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int [TestComple](#) ()
- [Element](#) \* [Complete](#) ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- [Element](#) \* [Comple](#) [Hourglass](#) ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- [ElemGeomC0](#) & [ElementGeometrique](#) () const
- const [ElemGeomC0](#) & [ElementGeometrique\\_const](#) () const
- [Coordonnee](#) & [Point\\_physique](#) (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void [Point\\_physique](#) (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- int [Nb\\_pt\\_int\\_epai](#) ()
- int [Nb\\_pt\\_int\\_surf](#) ()
- double [H](#) ()
- [ElemMeca::MatGeomInit](#) [MatricesGeometrique\\_Et\\_Initiale](#) (const [ParaAlgoControle](#) &pa)
- void [Inactive\\_ddl\\_primaire](#) ()
- void [Active\\_ddl\\_primaire](#) ()
- void [Plus\\_ddl\\_Sigma](#) ()
- void [Inactive\\_ddl\\_Sigma](#) ()
- void [Active\\_ddl\\_Sigma](#) ()
- void [Active\\_premier\\_ddl\\_Sigma](#) ()
- void [LectureContraintes](#) ([UtilLecture](#) \*entreePrinc)
- bool [ContraintesAbsolues](#) ([Tableau](#)< [Vecteur](#) > &tabSig)
- [Tableau](#)< [ElFrontiere](#) \* > const & [Frontiere](#) (bool force=false)
- double [Section](#) ([Enum\\_dure](#), const [Coordonnee](#) &)
- double [SectionMoyenne](#) ([Enum\\_dure](#))
- double [Largeur](#) ()
- double [Hauteur](#) ()
- double [KSlepais](#) (int i)
- void [ConstTabDdl](#) ()

## Fonctions membres protégées

- int [Dim\\_sig\\_eps](#) () const
- virtual [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- virtual [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Membres hérités additionnels

### 6.654.1 Documentation des fonctions membres

#### 6.654.1.1 [Active\\_ddl\\_Sigma\(\)](#)

```
void PoutSimple1::Active_ddl_Sigma ( ) [inline], [virtual]
Implémente ElemMeca.
```

#### 6.654.1.2 [Active\\_premier\\_ddl\\_Sigma\(\)](#)

```
void PoutSimple1::Active_premier_ddl_Sigma ( ) [inline], [virtual]
Implémente ElemMeca.
```

### 6.654.1.3 ContraintesAbsolues()

```
bool PoutSimple1::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.654.1.4 Dim\_sig\_eps()

```
int PoutSimple1::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.654.1.5 H()

```
double PoutSimple1::H ( ) [inline], [virtual]
```

Implémente [PiPoCo](#).

### 6.654.1.6 Inactive\_ddl\_Sigma()

```
void PoutSimple1::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.654.1.7 KSlepais()

```
double PoutSimple1::KSlepais (
    int i ) [inline], [virtual]
```

Implémente [PiPoCo](#).

### 6.654.1.8 LectureContraintes()

```
void PoutSimple1::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.654.1.9 Long\_arrete\_mini\_sur\_c()

```
double PoutSimple1::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.654.1.10 MatricesGeometrique\_Et\_Initiale()

```
ElemMeca::MatGeomInit PoutSimple1::MatricesGeometrique_Et_Initiale (
    const ParaAlgoControle & pa ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.654.1.11 Nb\_pt\_int\_epai()

```
int PoutSimple1::Nb_pt_int_epai ( ) [inline], [virtual]
```

Implémente [PiPoCo](#).

#### 6.654.1.12 Nb\_pt\_int\_surf()

```
int PoutSimple1::Nb_pt_int_surf ( ) [inline], [virtual]  
Implémente PiPoCo.
```

#### 6.654.1.13 new\_frontiere\_lin()

```
virtual ElFrontiere * PoutSimple1::new_frontiere_lin (   
    int ,   
    Tableau< Noeud * > & tab,   
    DdlElement & ddelem ) [inline], [protected], [virtual]  
Implémente ElemMeca.
```

#### 6.654.1.14 new\_frontiere\_surf()

```
virtual ElFrontiere * PoutSimple1::new_frontiere_surf (   
    int ,   
    Tableau< Noeud * > & tab,   
    DdlElement & ddelem ) [inline], [protected], [virtual]  
Implémente ElemMeca.
```

#### 6.654.1.15 Plus\_ddl\_Sigma()

```
void PoutSimple1::Plus_ddl_Sigma ( ) [inline], [virtual]  
Implémente ElemMeca.
```

#### 6.654.1.16 Section()

```
double PoutSimple1::Section (   
    Enum\_dure ,   
    const Coordonnee & ) [inline], [virtual]  
Réimplémentée à partir de ElemMeca.
```

#### 6.654.1.17 SectionMoyenne()

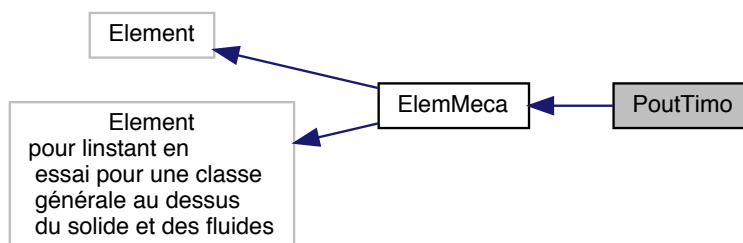
```
double PoutSimple1::SectionMoyenne (   
    Enum\_dure ) [inline], [virtual]  
Réimplémentée à partir de ElemMeca.
```

La documentation de cette classe a été générée à partir du fichier suivant :

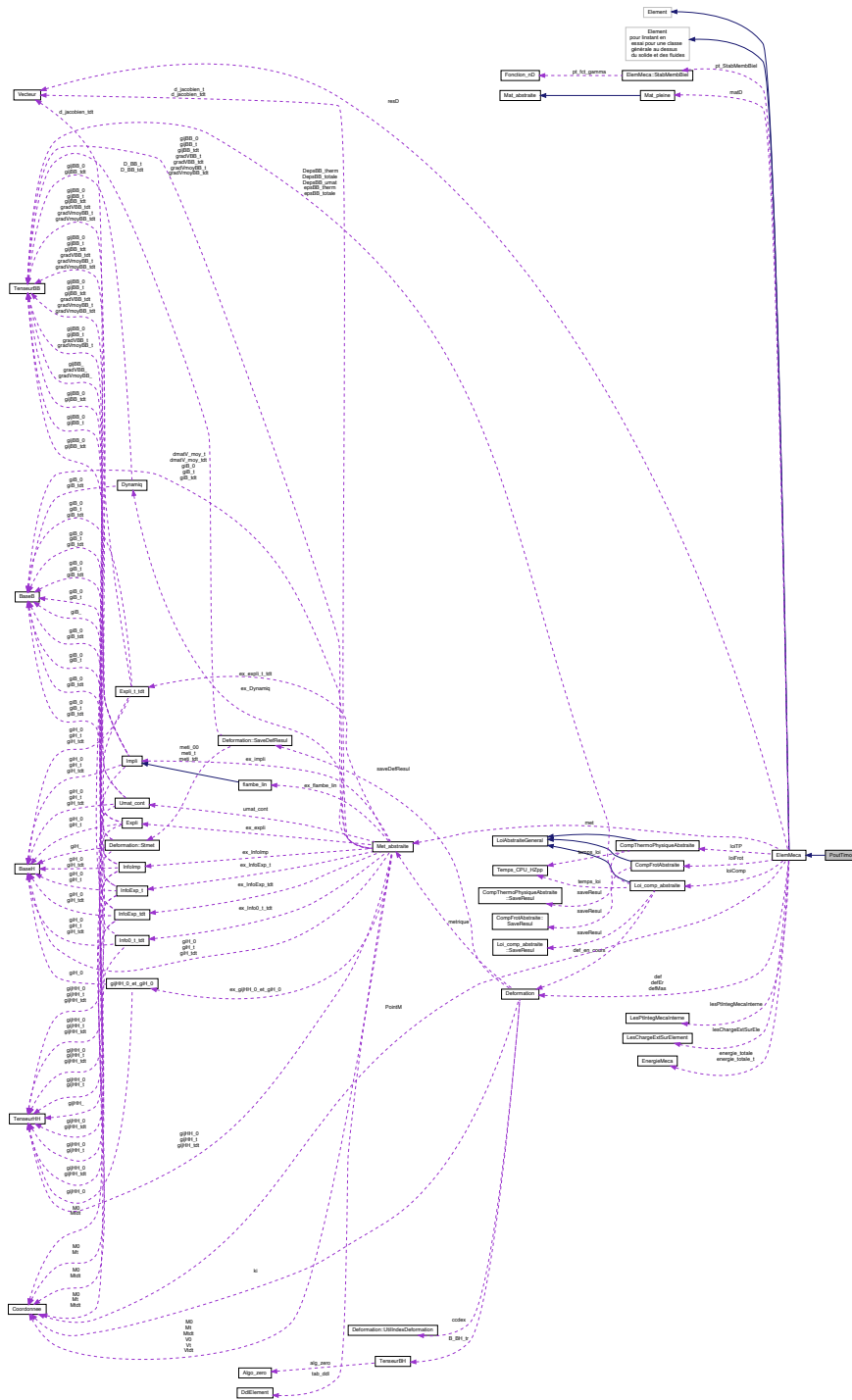
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/PoutSimple1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/PoutSimple1.cc

## 6.655 Référence de la classe PoutTimo

Graphe d'héritage de PoutTimo:



Graphe de collaboration de PoutTimo:



**Fonctions membres publiques**

- PoutTimo (double sect, int num\_id=-3)
- PoutTimo (int num\_id)
- PoutTimo (double sect, int num\_id, const Tableau< Noeud \* > &tab)
- PoutTimo (PoutTimo &pout)
- PoutTimo & operator= (PoutTimo &pout)
- void LectureDonneesParticulieres (UtilLecture \*, Tableau< Noeud \* > \*)
- Element::ResRaid Calcul\_implicit ()



- [Vecteur](#) \* [CalculResidu\\_t](#) ()
- [DdlElement](#) & [TableauDdl](#) ()
- void [Libere](#) ()
- void [DefLoi](#) ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int [TestComplet](#) ()
- [Element](#) \* [Complete](#) ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- [ElemGeom](#) & [ElementGeometrique](#) ()
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)
- [Tableau](#)< [ElFrontiere](#) \* > & [Frontiere](#) ()
- [ElemMeca::MatGeomInit](#) [MatricesGeometrique\\_Et\\_Initiale](#) ()
- double & [Section](#) ()
- [TenseurBB](#) \* [DeformationBB](#) ()
- [TenseurHH](#) \* [ContrainteHH](#) ()
- void [ConstTabDdl](#) ()

### Fonctions membres protégées

- virtual [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- virtual [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

### Membres hérités additionnels

#### 6.655.1 Documentation des fonctions membres

##### 6.655.1.1 [new\\_frontiere\\_lin\(\)](#)

```
virtual ElFrontiere * PoutTimo::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.655.1.2 [new\\_frontiere\\_surf\(\)](#)

```
virtual ElFrontiere * PoutTimo::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

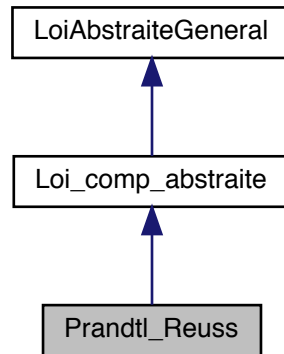
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

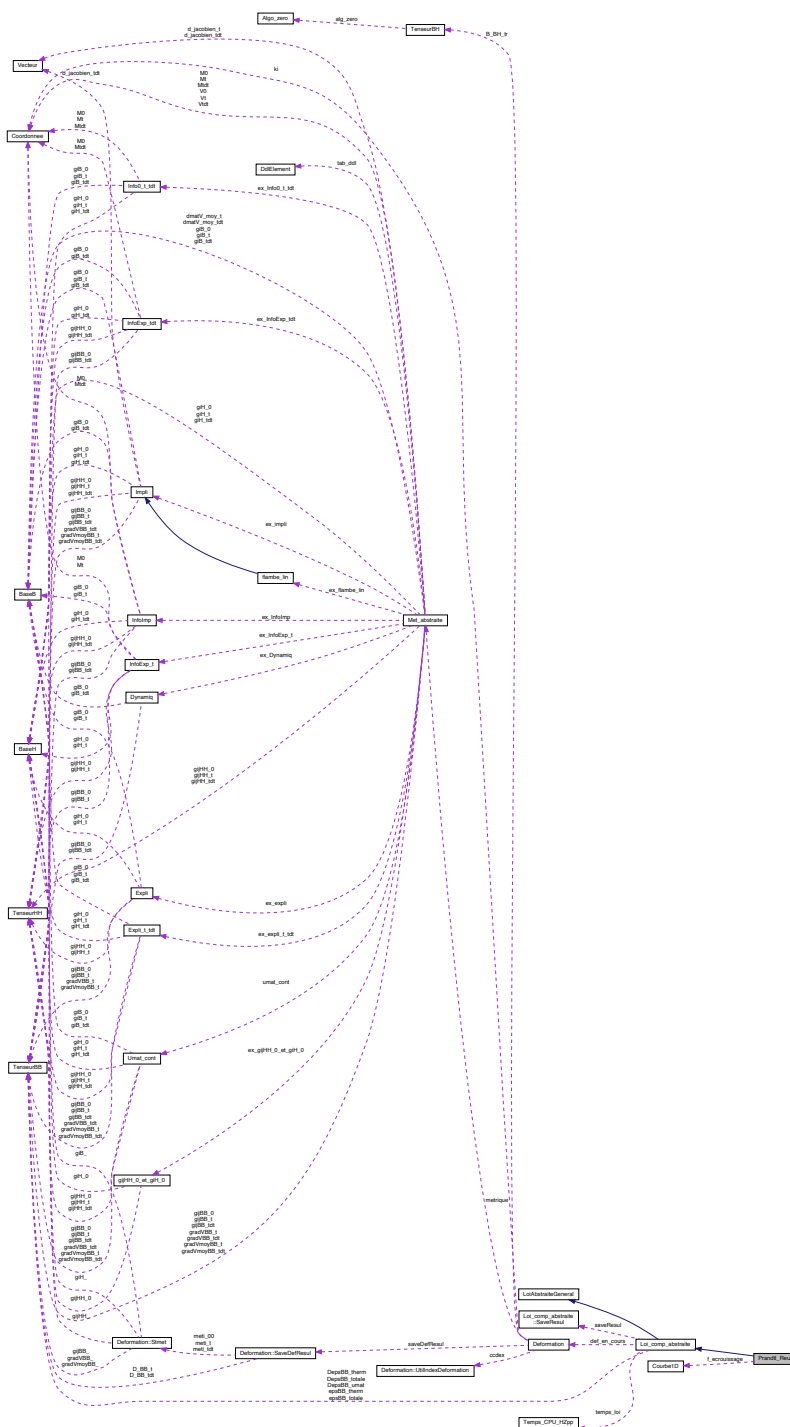
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/PoutTimo.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Biellette/PoutTimo.cc

## 6.656 Référence de la classe Prandtl\_Reuss

Graphe d'héritage de Prandtl\_Reuss:



Graphe de collaboration de Prandtl\_Reuss:



## Classes

- class [SaveResulPrandtl\\_Reuss](#)

## Fonctions membres publiques

- **Prandtl\_Reuss** (const [Prandtl\\_Reuss](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()

- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComplet](#) ()
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#), const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_, [TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#) &giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#) &giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB\_, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Impli](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- double [E](#)
- double [nu](#)
- [Courbe1D](#) \* [f\\_ecrouissage](#)
- double [tolerance\\_plas](#)
- int [nb\\_boucle\\_maxi](#)

## Membres hérités additionnels

### 6.656.1 Documentation des fonctions membres

#### 6.656.1.1 Affiche()

```
void Prandtl_Reuss::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.656.1.2 Calcul\_DsigmaHH\_tdt()

```
void Prandtl_Reuss::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
```

```

TenseurHH & gijHH_t,
BaseB & giB_tdt,
Tableau< BaseB > & d_giB_tdt,
BaseH & giH_tdt,
Tableau< BaseH > & d_giH_tdt,
TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]
Tenseur3BH dsigBH = dS_BH + (a3 * dleps ) * IdBH3;
dsigHH = gijHH * dsigBH + dgijHH * sigBH;
Implémente Loi\_comp\_abstraite.

```

### 6.656.1.3 Calcul\_SigmaHH()

```

void Prandtl_Reuss::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]
Implémente Loi\_comp\_abstraite.

```

### 6.656.1.4 CalculGrandeurTravail()

```

virtual void Prandtl_Reuss::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,

```

```

const ThermoDonnee & ,
const Met_abstraite::Impli * ex_impli,
const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
const Met_abstraite::Umat_cont * ex_umat,
const List_io< Ddl_etendu > * exclure_dd_etend,
const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],

```

[virtual]

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.656.1.5 Ecriture\_base\_info\_loi()

```

void Prandtl_Reuss::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.656.1.6 HsurH0()

```

virtual double Prandtl_Reuss::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.656.1.7 Info\_commande\_LoisDeComp()

```

void Prandtl_Reuss::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.656.1.8 Lecture\_base\_info\_loi()

```

void Prandtl_Reuss::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.656.1.9 LectureDonneesParticulieres()

```

void Prandtl_Reuss::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.656.1.10 Module\_young\_equivalent()

```

double Prandtl_Reuss::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.656.1.11 New\_et\_Initialise()

`SaveResul * Prandtl_Reuss::New_et_Initialise ( ) [inline], [virtual]`

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.656.1.12 Nouvelle\_loi\_identique()

`Loi_comp_abstraite * Prandtl_Reuss::Nouvelle_loi_identique ( ) const [inline], [virtual]`

Implémente [Loi\\_comp\\_abstraite](#).

### 6.656.1.13 TestComplet()

`int Prandtl_Reuss::TestComplet ( ) [virtual]`

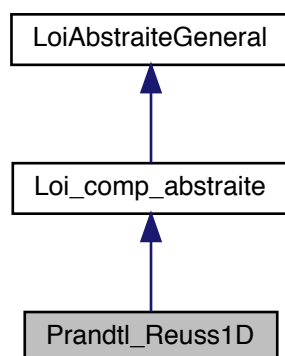
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

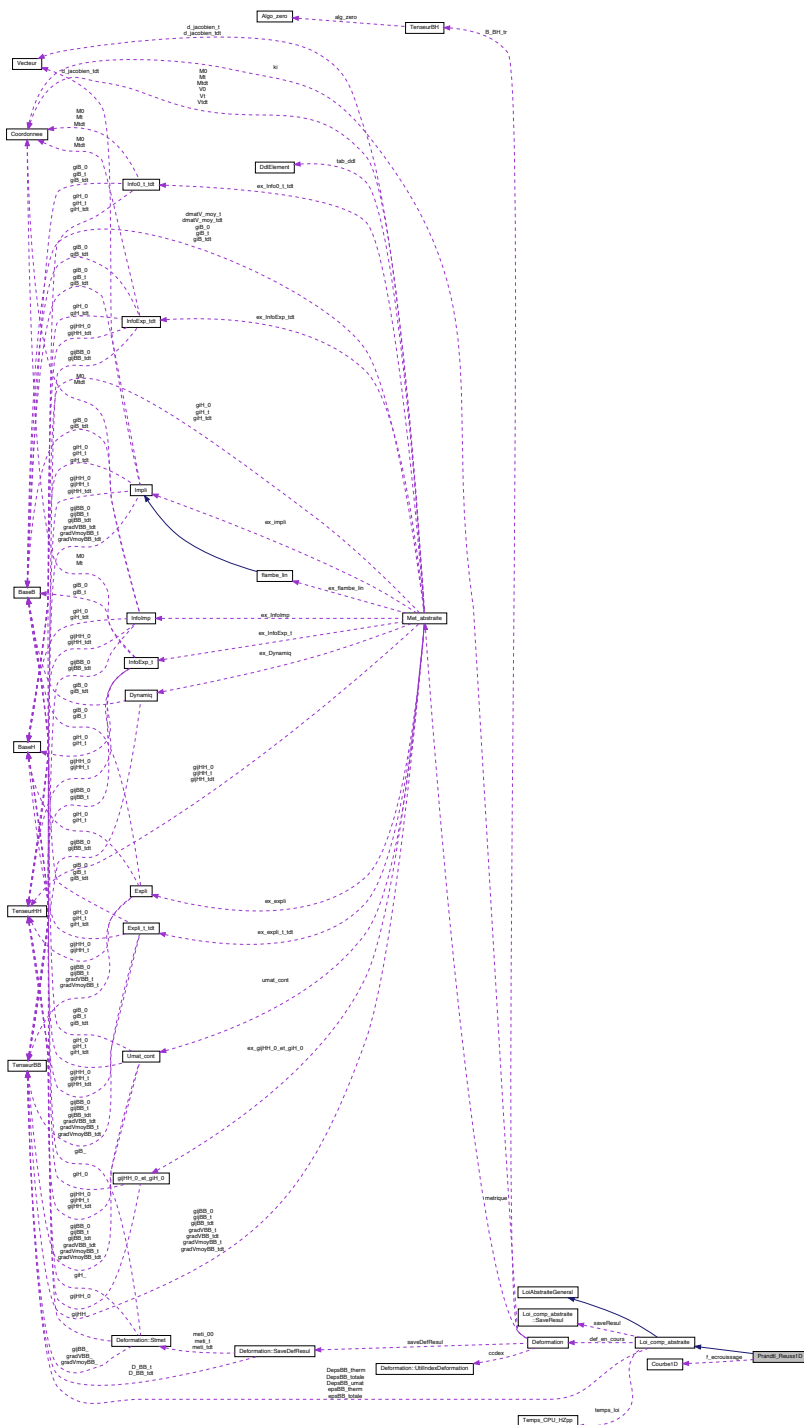
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss.cc

## 6.657 Référence de la classe Prandtl\_Reuss1D

Graphe d'héritage de Prandtl\_Reuss1D:



Graphe de collaboration de Prandtl\_Reuss1D:



### Classes

- class [SaveResulPrandtl\\_Reuss1D](#)

### Fonctions membres publiques

- [Prandtl\\_Reuss1D](#) (const [Prandtl\\_Reuss1D](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()



- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComplet](#) ()
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#), const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_, [TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#) &giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#) &giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB\_, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Impli](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- double [E](#)
- double [nu](#)
- [Courbe1D](#) \* [f\\_ecrouissage](#)
- double [tolerance\\_plas](#)
- int [nb\\_boucle\\_maxi](#)
- int [nb\\_sous\\_increment](#)

## Membres hérités additionnels

### 6.657.1 Documentation des fonctions membres

#### 6.657.1.1 Affiche()

```
void Prandtl_Reuss1D::Affiche ( ) const [virtual]
Implémente LoiAbstraiteGeneral.
```

#### 6.657.1.2 Calcul\_DsigmaHH\_tdt()

```
void Prandtl_Reuss1D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
```

```

TenseurBB & gijBB_t,
TenseurHH & gijHH_t,
BaseB & giB_tdt,
Tableau< BaseB > & d_giB_tdt,
BaseH & giH_tdt,
Tableau< BaseH > & d_giH_tdt,
TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.657.1.3 Calcul\_SigmaHH()

```

void Prandtl_ReussID::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.657.1.4 CalculGrandeurTravail()

```

virtual void Prandtl_ReussID::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,

```

```
const Met_abstraite::Impli * ex_impli,  
const Met_abstraite::Expli_t_tdt * ex_expli_tdt,  
const Met_abstraite::Umat_cont * ex_umat,  
const List_io< Ddl_etendu > * exclure_dd_etend,  
const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],
```

[virtual]

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.657.1.5 Ecriture\_base\_info\_loi()

```
void Prandtl_Reuss1D::Ecriture_base_info_loi (  
    ofstream & sort,  
    const int cas ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.657.1.6 HsurH0()

```
virtual double Prandtl_Reuss1D::HsurH0 (  
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.657.1.7 Info\_commande\_LoisDeComp()

```
void Prandtl_Reuss1D::Info_commande_LoisDeComp (  
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.657.1.8 Lecture\_base\_info\_loi()

```
void Prandtl_Reuss1D::Lecture_base_info_loi (  
    ifstream & ent,  
    const int cas,  
    LesReferences & lesRef,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.657.1.9 LectureDonneesParticulieres()

```
void Prandtl_Reuss1D::LectureDonneesParticulieres (  
    UtilLecture * entreePrinc,  
    LesCourbes1D & lesCourbes1D,  
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.657.1.10 Module\_young\_equivalent()

```
double Prandtl_Reuss1D::Module_young_equivalent (  
    Enum_dure ,  
    const Deformation & ,  
    SaveResul * ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.657.1.11 New\_et\_Initialise()

`SaveResul * Prandtl_Reuss1D::New_et_Initialise ( ) [inline], [virtual]`

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

### 6.657.1.12 Nouvelle\_loi\_identique()

`Loi_comp_abstraite * Prandtl_Reuss1D::Nouvelle_loi_identique ( ) const [inline], [virtual]`

Implémente [Loi\\_comp\\_abstraite](#).

### 6.657.1.13 TestComplet()

`int Prandtl_Reuss1D::TestComplet ( ) [virtual]`

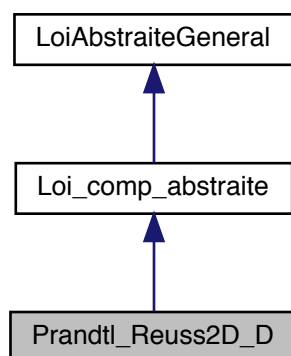
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

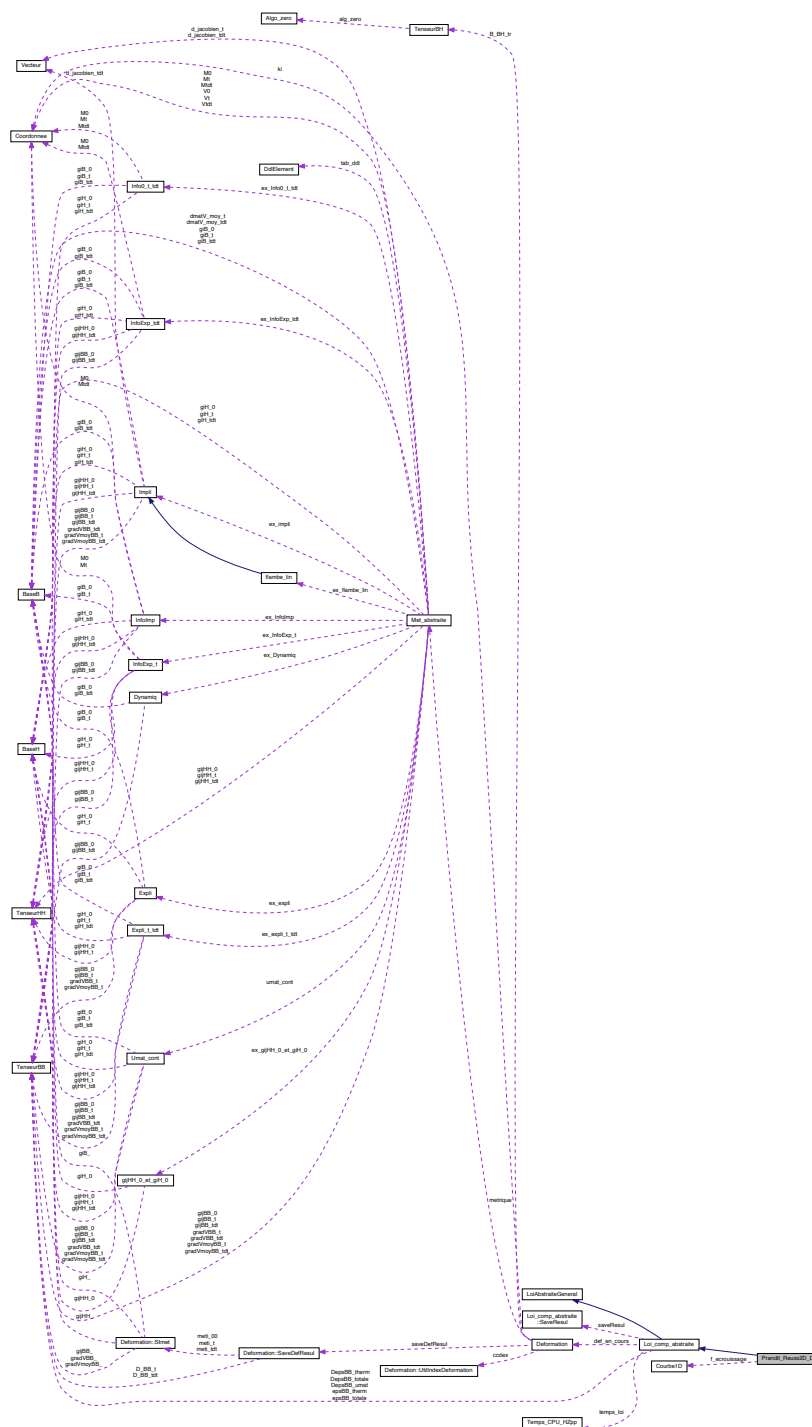
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss1D.cc

## 6.658 Référence de la classe Prandtl\_Reuss2D\_D

Graphe d'héritage de Prandtl\_Reuss2D\_D:



Graphe de collaboration de Prandtl\_Reuss2D\_D:



### Classes

- class [SaveResulPrandtl\\_Reuss2D\\_D](#)

### Fonctions membres publiques

- [Prandtl\\_Reuss2D\\_D](#) (const [Prandtl\\_Reuss2D\\_D](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()

- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComplet](#) ()
- double [Module\\_young\\_equivalent](#) ([Enum\\_dure](#), const [Deformation](#) &, [SaveResul](#) \*)
- virtual double [HsurH0](#) ([SaveResul](#) \*saveResul) const
- [Loi\\_comp\\_abstraite](#) \* [Nouvelle\\_loi\\_identique](#) () const
- void [Lecture\\_base\\_info\\_loi](#) (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Ecriture\\_base\\_info\\_loi](#) (ofstream &sort, const int cas)
- void [Info\\_commande\\_LoisDeComp](#) ([UtilLecture](#) &lec)

## Fonctions membres protégées

- void [Calcul\\_SigmaHH](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB, [BaseH](#) &gi\_H, [TenseurBB](#) &epsBB\_, [TenseurBB](#) &delta\_epsBB\_, [TenseurBB](#) &gijBB\_, [TenseurHH](#) &gijHH\_, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_, double &jacobien\_0, double &jacobien, [TenseurHH](#) &sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Expli\\_t\\_tdt](#) &ex)
- void [Calcul\\_DsigmaHH\\_tdt](#) ([TenseurHH](#) &sigHH\_t, [TenseurBB](#) &DepsBB, [DdlElement](#) &tab\_ddl, [BaseB](#) &giB\_t, [TenseurBB](#) &gijBB\_t, [TenseurHH](#) &gijHH\_t, [BaseB](#) &giB\_tdt, [Tableau](#)< [BaseB](#) > &d\_giB\_tdt, [BaseH](#) &giH\_tdt, [Tableau](#)< [BaseH](#) > &d\_giH\_tdt, [TenseurBB](#) &epsBB\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_epsBB, [TenseurBB](#) &delta\_epsBB, [TenseurBB](#) &gijBB\_tdt, [TenseurHH](#) &gijHH\_tdt, [Tableau](#)< [TenseurBB](#) \* > &d\_gijBB\_tdt, [Tableau](#)< [TenseurHH](#) \* > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, [Vecteur](#) &d\_jacobien\_tdt, [TenseurHH](#) &sigHH, [Tableau](#)< [TenseurHH](#) \* > &d\_sigHH, [EnergieMeca](#) &energ, const [EnergieMeca](#) &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const [Met\\_abstraite::Impli](#) &ex)
- virtual void [CalculGrandeurTravail](#) (const [PtIntegMecalInterne](#) &, const [Deformation](#) &, [Enum\\_dure](#), const [ThermoDonnee](#) &, const [Met\\_abstraite::Impli](#) \*ex\_impli, const [Met\\_abstraite::Expli\\_t\\_tdt](#) \*ex\_expli\_tdt, const [Met\\_abstraite::Umat\\_cont](#) \*ex\_umat, const [List\\_io](#)< [Ddl\\_etendu](#) > \*exclure\_dd\_etend, const [List\\_io](#)< const [TypeQuelconque](#) \* > \*exclure\_Q)

## Attributs protégés

- double [E](#)
- double [nu](#)
- [Courbe1D](#) \* [f\\_ecrouissage](#)
- double [tolerance\\_plas](#)
- int [nb\\_boucle\\_maxi](#)

## Membres hérités additionnels

### 6.658.1 Documentation des fonctions membres

#### 6.658.1.1 Affiche()

void [Prandtl\\_Reuss2D\\_D::Affiche](#) ( ) const [virtual]  
 Implémente [LoiAbstraiteGeneral](#).

#### 6.658.1.2 Calcul\_DsigmaHH\_tdt()

```
void Prandtl_Reuss2D_D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
```

```

TenseurHH & gijHH_t,
BaseB & giB_tdt,
Tableau< BaseB > & d_giB_tdt,
BaseH & giH_tdt,
Tableau< BaseH > & d_giH_tdt,
TenseurBB & epsBB_tdt,
Tableau< TenseurBB * > & d_epsBB,
TenseurBB & delta_epsBB,
TenseurBB & gijBB_tdt,
TenseurHH & gijHH_tdt,
Tableau< TenseurBB * > & d_gijBB_tdt,
Tableau< TenseurHH * > & d_gijHH_tdt,
double & jacobien_0,
double & jacobien,
Vecteur & d_jacobien_tdt,
TenseurHH & sigHH,
Tableau< TenseurHH * > & d_sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Impli & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.658.1.3 Calcul\_SigmaHH()

```

void Prandtl_Reuss2D_D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB,
    BaseH & gi_H,
    TenseurBB & epsBB_,
    TenseurBB & delta_epsBB_,
    TenseurBB & gijBB_,
    TenseurHH & gijHH_,
    Tableau< TenseurBB * > & d_gijBB_,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

### 6.658.1.4 CalculGrandeurTravail()

```

virtual void Prandtl_Reuss2D_D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ,
    const Deformation & ,
    Enum_dure ,
    const ThermoDonnee & ,
    const Met_abstraite::Impli * ex_impli,

```

```

const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
const Met_abstraite::Umat_cont * ex_umat,
const List_io< Ddl_etendu > * exclure_dd_etend,
const List_io< const TypeQuelconque * > * exclure_Q ) [inline], [protected],

```

[virtual]

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.658.1.5 Ecriture\_base\_info\_loi()

```

void Prandtl_Reuss2D_D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.658.1.6 HsurH0()

```

virtual double Prandtl_Reuss2D_D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.658.1.7 Info\_commande\_LoisDeComp()

```

void Prandtl_Reuss2D_D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.658.1.8 Lecture\_base\_info\_loi()

```

void Prandtl_Reuss2D_D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.658.1.9 LectureDonneesParticulieres()

```

void Prandtl_Reuss2D_D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.658.1.10 Module\_young\_equivalent()

```

double Prandtl_Reuss2D_D::Module_young_equivalent (
    Enum_dure ,
    const Deformation & ,
    SaveResul * ) [inline], [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).



**6.658.1.11 New\_et\_Initialise()**

```
SaveResul * Prandtl_Reuss2D_D::New_et_Initialise ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.658.1.12 Nouvelle\_loi\_identique()**

```
Loi_comp_abstraite * Prandtl_Reuss2D_D::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.658.1.13 TestComplet()**

```
int Prandtl_Reuss2D_D::TestComplet ( ) [virtual]
```

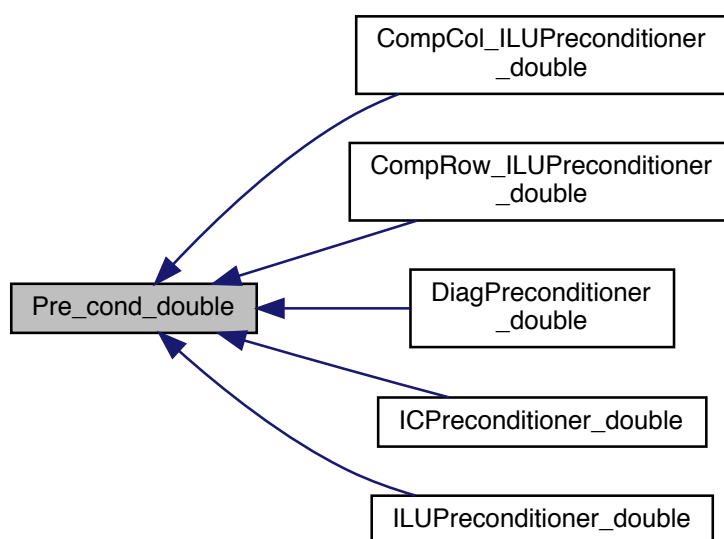
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss2D\_D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss2D\_D.cc

**6.659 Référence de la classe Pre\_cond\_double**

Grappe d'héritage de Pre\_cond\_double:

**Fonctions membres publiques**

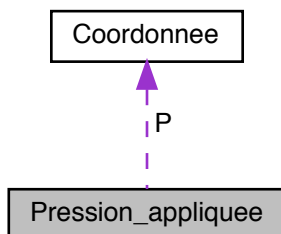
- virtual VECTOR\_double **solve** (const VECTOR\_double &x) const =0
- virtual VECTOR\_double **trans\_solve** (const VECTOR\_double &x) const =0

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resolin/preconditionnement/pre\_cond\_double.h

## 6.660 Référence de la classe `Pression_appliquee`

Graphe de collaboration de `Pression_appliquee`:



### Fonctions membres publiques

- `Pression_appliquee` (const `Pression_appliquee` &a)
- `Pression_appliquee` & `operator=` (const `Pression_appliquee` &a)
- void `Zero` ()

### Attributs publics

- double `press`
- `Coordonnee` `P`

### Amis

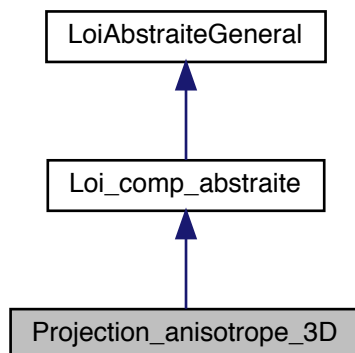
- `istream` & `operator>>` (`istream` &ent, `Pression_appliquee` &a)
- `ostream` & `operator<<` (`ostream` &sort, const `Pression_appliquee` &a)

La documentation de cette classe a été générée à partir du fichier suivant :

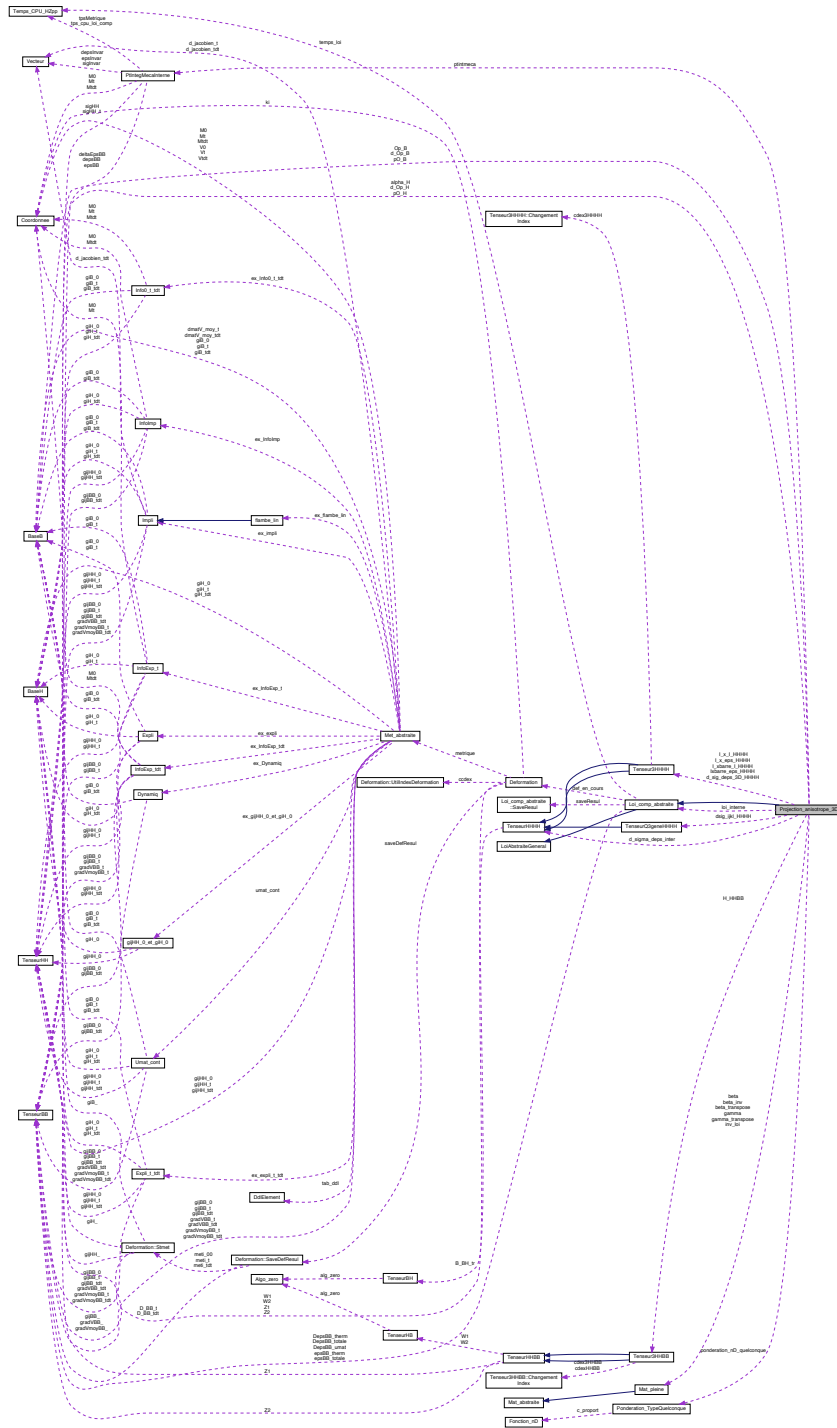
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/LesChargeExtSurElement.h`

## 6.661 Référence de la classe Projection\_anisotrope\_3D

Graphe d'héritage de Projection\_anisotrope\_3D:



Graphe de collaboration de Projection\_anisotrope\_3D:



**Classes**

- class [SaveResulProjection\\_anisotrope\\_3D](#)

**Fonctions membres publiques**

- **Projection\_anisotrope\_3D** (const [Projection\\_anisotrope\\_3D](#) &loi)
- [SaveResul](#) \* [New\\_et\\_Initialise](#) ()

- void `LectureDonneesParticulieres` (`UtilLecture *`, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Affiche` () const
- int `TestComplet` ()
- double `Module_young_equivalent` (`Enum_dure` temps, const `Deformation` &, `SaveResul` \*saveResul)
- double `Module_compressibilite_equivalent` (`Enum_dure` temps, const `Deformation` &def, `SaveResul` \*saveResul)
- virtual double `HsurH0` (`SaveResul` \*saveResul) const
- virtual void `Activation_donnees` (`Tableau`< `Noeud *` > &tabnoeud, bool dilatation, `LesPtIntegMecalInterne` &lesPtMecalInt)
- virtual void `Grandeur_particuliere` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ, `Loi_comp_abstraite::SaveResul` \*saveDon, list< int > &) const
- virtual void `ListeGrandeurs_particulieres` (bool absolue, `List_io`< `TypeQuelconque` > &liTQ) const
- `Loi_comp_abstraite` \* `Nouvelle_loi_identique` () const
- virtual `Enum_comp_3D_CP_DP_1D` `Comportement_3D_CP_DP_1D` ()
- void `Lecture_base_info_loi` (ifstream &ent, const int cas, `LesReferences` &lesRef, `LesCourbes1D` &lesCourbes1D, `LesFonctions_nD` &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` (`UtilLecture` &lec)
- const string & `NomRepere` () const
- const int & `Type_transport` () const

## Types protégés

- enum `Enumcompletudecalcul` { `CONTRAINTE_ET_TANGENT` =0 , `CONTRAINTE_UNIQUEMENT` , `TANGENT_UNIQUEMENT` }

## Fonctions membres protégées

- void `Calcul_SigmaHH` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB, `BaseH` &gi\_H, `TenseurBB` &epsBB\_, `TenseurBB` &delta\_epsBB\_, `TenseurBB` &gijBB\_, `TenseurHH` &gijHH\_, `Tableau`< `TenseurBB *` > &d\_gijBB\_, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Expli_t_tdt` &ex)
- void `Calcul_DsigmaHH_tdt` (`TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `DdlElement` &tab\_ddl, `BaseB` &giB\_t, `TenseurBB` &gijBB\_t, `TenseurHH` &gijHH\_t, `BaseB` &giB\_tdt, `Tableau`< `BaseB` > &d\_giB\_tdt, `BaseH` &giH\_tdt, `Tableau`< `BaseH` > &d\_giH\_tdt, `TenseurBB` &epsBB\_tdt, `Tableau`< `TenseurBB *` > &d\_epsBB, `TenseurBB` &delta\_epsBB, `TenseurBB` &gijBB\_tdt, `TenseurHH` &gijHH\_tdt, `Tableau`< `TenseurBB *` > &d\_gijBB\_tdt, `Tableau`< `TenseurHH *` > &d\_gijHH\_tdt, double &jacobien\_0, double &jacobien, `Vecteur` &d\_jacobien\_tdt, `TenseurHH` &sigHH, `Tableau`< `TenseurHH *` > &d\_sigHH, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Impli` &ex)
- void `Calcul_dsigma_deps` (bool en\_base\_orthonormee, `TenseurHH` &sigHH\_t, `TenseurBB` &DepsBB, `TenseurBB` &epsBB\_tdt, `TenseurBB` &delta\_epsBB, double &jacobien\_0, double &jacobien, `TenseurHH` &sigHH, `TenseurHHHH` &d\_sigma\_deps, `EnergieMeca` &energ, const `EnergieMeca` &energ\_t, double &module\_compressibilite, double &module\_cisaillement, const `Met_abstraite::Umat_cont` &ex)
- virtual void `CalculGrandeurTravail` (const `PtIntegMecalInterne` &, const `Deformation` &, `Enum_dure`, const `ThermoDonnee` &, const `Met_abstraite::Impli` \*ex\_impli, const `Met_abstraite::Expli_t_tdt` \*ex\_expli\_tdt, const `Met_abstraite::Umat_cont` \*ex\_umat, const `List_io`< `Ddl_etendu` > \*exclure\_dd\_etend, const `List_io`< const `TypeQuelconque *` > \*exclure\_Q)
- void `RepercuteChangeTemperature` (`Enum_dure` temps)
- void `Verif_et_preparation_acces_grandeurs_locale` ()

## Attributs protégés

- `Enum_proj_aniso` `type_projection`
- double `A1`
- double `A2`
- double `A3`
- double `B12`
- double `B13`

- double **B23**
- [Tableau2](#)< double \* > **hij**
- [Tableau](#)< [Fonction\\_nD](#) \* > **fct\_para**
- bool **BII\_fct\_para**
- string **nom\_repere**
- int **cas\_calcul**
- double **ratio\_inf\_module\_compressibilite**
- [Mat\\_pleine](#) **inv\_loi**
- int **type\_transport**
- [BaseB](#) **Op\_B**
- [BaseB](#) **d\_Op\_B**
- [BaseB](#) **pO\_B**
- [BaseH](#) **d\_Op\_H**
- [BaseH](#) **pO\_H**
- [BaseH](#) **alpha\_H**
- [Mat\\_pleine](#) **beta**
- [Mat\\_pleine](#) **gamma**
- [Mat\\_pleine](#) **beta\_transpose**
- [Mat\\_pleine](#) **gamma\_transpose**
- [Mat\\_pleine](#) **beta\_inv**
- int **sortie\_post**
- [Tenseur3HHHH](#) **I\_x\_I\_HHHH**
- [Tenseur3HHHH](#) **I\_xbarre\_I\_HHHH**
- [Tenseur3HHHH](#) **I\_x\_eps\_HHHH**
- [Tenseur3HHHH](#) **Ixbarre\_eps\_HHHH**
- [TenseurQ3geneHHHH](#) **dsig\_ijkl\_HHHH**
- [PtIntegMecaInterne](#) **ptintmeca**
- [Loi\\_comp\\_abstraite](#) \* **loi\_interne**
- Enumcompletudecalcul **completude\_calcul**
- [Ponderation\\_TypeQuelconque](#) \* **ponderation\_nD\_quelconque**
- [Tableau](#)< [TenseurHH](#) \* > **d\_sigRef\_HH**
- [TenseurHHHH](#) \* **d\_sigma\_deps\_inter**
- [Tenseur3HHHH](#) **d\_sig\_deps\_3D\_HHHH**
- [Tenseur3HHBB](#) **H\_HHBB**

## Amis

- class [SaveResulProjection\\_anisotrope\\_3D](#)

## Membres hérités additionnels

### 6.661.1 Documentation des fonctions membres

#### 6.661.1.1 Activation\_donnees()

```
void Projection_anisotrope_3D::Activation_donnees (
    Tableau< Noeud * > & tabnoeud,
    bool dilatation,
    LesPtIntegMecaInterne & lesPtMecaInt ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

#### 6.661.1.2 Affiche()

```
void Projection_anisotrope_3D::Affiche ( ) const [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.661.1.3 Calcul\_dsigma\_deps()**

```
void Projection_anisotrope_3D::Calcul_dsigma_deps (
    bool en_base_orthonormee,
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    TenseurBB & epsBB_tdt,
    TenseurBB & delta_epsBB,
    double & jacobien_0,
    double & jacobien,
    TenseurHH & sigHH,
    TenseurHHHH & d_sigma_deps,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Umat_cont & ex ) [protected], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.661.1.4 Calcul\_DsigmaHH\_tdt()**

```
void Projection_anisotrope_3D::Calcul_DsigmaHH_tdt (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
    BaseB & giB_t,
    TenseurBB & gijBB_t,
    TenseurHH & gijHH_t,
    BaseB & giB_tdt,
    Tableau< BaseB > & d_giB_tdt,
    BaseH & giH_tdt,
    Tableau< BaseH > & d_giH_tdt,
    TenseurBB & epsBB_tdt,
    Tableau< TenseurBB * > & d_epsBB,
    TenseurBB & delta_epsBB,
    TenseurBB & gijBB_tdt,
    TenseurHH & gijHH_tdt,
    Tableau< TenseurBB * > & d_gijBB_tdt,
    Tableau< TenseurHH * > & d_gijHH_tdt,
    double & jacobien_0,
    double & jacobien,
    Vecteur & d_jacobien_tdt,
    TenseurHH & sigHH,
    Tableau< TenseurHH * > & d_sigHH,
    EnergieMeca & energ,
    const EnergieMeca & energ_t,
    double & module_compressibilite,
    double & module_cisaillement,
    const Met_abstraite::Impli & ex ) [protected], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.661.1.5 Calcul\_SigmaHH()**

```
void Projection_anisotrope_3D::Calcul_SigmaHH (
    TenseurHH & sigHH_t,
    TenseurBB & DepsBB,
    DdlElement & tab_ddl,
```

```

TenseurBB & gijBB_t,
TenseurHH & gijHH_t,
BaseB & giB,
BaseH & gi_H,
TenseurBB & epsBB_,
TenseurBB & delta_epsBB_,
TenseurBB & gijBB_,
TenseurHH & gijHH_,
Tableau< TenseurBB * > & d_gijBB_,
double & jacobien_0,
double & jacobien,
TenseurHH & sigHH,
EnergieMeca & energ,
const EnergieMeca & energ_t,
double & module_compressibilite,
double & module_cisaillement,
const Met_abstraite::Expli_t_tdt & ex ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.661.1.6 CalculGrandeurTravail()

```

void Projection_anisotrope_3D::CalculGrandeurTravail (
    const PtIntegMecaInterne & ptintmeca,
    const Deformation & def,
    Enum_dure temps,
    const ThermoDonnee & dTP,
    const Met_abstraite::Impli * ex_impli,
    const Met_abstraite::Expli_t_tdt * ex_expli_tdt,
    const Met_abstraite::Umat_cont * ex_umat,
    const List_io< Ddl_etendu > * exclure_dd_etend,
    const List_io< const TypeQuelconque * > * exclure_Q ) [protected], [virtual]

```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.661.1.7 Comportement\_3D\_CP\_DP\_1D()

[Enum\\_comp\\_3D\\_CP\\_DP\\_1D](#) Projection\_anisotrope\_3D::Comportement\_3D\_CP\_DP\_1D ( ) [virtual]

Réimplémentée à partir de [LoiAbstraiteGeneral](#).

#### 6.661.1.8 Ecriture\_base\_info\_loi()

```

void Projection_anisotrope_3D::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]

```

Implémente [LoiAbstraiteGeneral](#).

#### 6.661.1.9 Grandeur\_particuliere()

```

void Projection_anisotrope_3D::Grandeur_particuliere (
    bool absolue,
    List_io< TypeQuelconque > & litQ,
    Loi_comp_abstraite::SaveResul * saveDon,
    list< int > & decal ) const [virtual]

```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).



**6.661.1.10 HsurH0()**

```
virtual double Projection_anisotrope_3D::HsurH0 (
    SaveResul * saveResul ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

**6.661.1.11 Info\_commande\_LoisDeComp()**

```
void Projection_anisotrope_3D::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.661.1.12 Lecture\_base\_info\_loi()**

```
void Projection_anisotrope_3D::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.661.1.13 LectureDonneesParticulieres()**

```
void Projection_anisotrope_3D::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

**6.661.1.14 ListeGrandeurs\_particulieres()**

```
void Projection_anisotrope_3D::ListeGrandeurs_particulieres (
    bool absolue,
    List_io< TypeQuelconque > & liTQ ) const [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.661.1.15 Module\_compressibilite\_equivalent()**

```
double Projection_anisotrope_3D::Module_compressibilite_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.661.1.16 Module\_young\_equivalent()**

```
double Projection_anisotrope_3D::Module_young_equivalent (
    Enum_dure temps,
    const Deformation & def,
    SaveResul * saveResul ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.661.1.17 New\_et\_Initialise()**

`Projection_anisotrope_3D::SaveResul * Projection_anisotrope_3D::New_et_Initialise ( ) [virtual]`  
 Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.661.1.18 Nouvelle\_loi\_identique()**

`Loi_comp_abstraite * Projection_anisotrope_3D::Nouvelle_loi_identique ( ) const [inline], [virtual]`  
 Implémente [Loi\\_comp\\_abstraite](#).

**6.661.1.19 RepercuteChangeTemperature()**

`void Projection_anisotrope_3D::RepercuteChangeTemperature ( Enum_dure temps ) [protected], [virtual]`  
 Réimplémentée à partir de [Loi\\_comp\\_abstraite](#).

**6.661.1.20 TestComplet()**

`int Projection_anisotrope_3D::TestComplet ( ) [virtual]`  
 Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Projection\_anisotrope\_↔3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Projection\_anisotrope\_↔3D.cc

**6.662 Référence de la classe Projet**

definition des operations de haut niveau : Lecture, Calcul, Sortie des [Resultats](#).

```
#include <Projet.h>
```

**Fonctions membres publiques**

- **Projet** (string &retour, int argc=0, const char \*argv[]=NULL)
- void **Lecture** ()
- bool **Lecture\_secondaire** ()
- void **Calcul** ()
- [EnumTypeCalcul](#) **TypeDeCalcul** () const
- void **SortieDesResultats** ()
- void **Visualisation\_interactive** ()
- void **Def\_fichier\_commande** (string &retour)
- void **Def\_fichier\_SchemaXML** (string &)
- void **FermetureFichiersVisualisation** ()
- [UtilLecture](#) \* **Sortie\_temps\_cpu** ()
- [UtilLecture](#) \* **Def\_sortie\_temps\_cpu** ()
- void **Arret\_du\_comptage\_CPU** ()

**6.662.1 Description détaillée**

definition des operations de haut niveau : Lecture, Calcul, Sortie des [Resultats](#).

Auteur

Gérard Rio

Version

1.0

Date

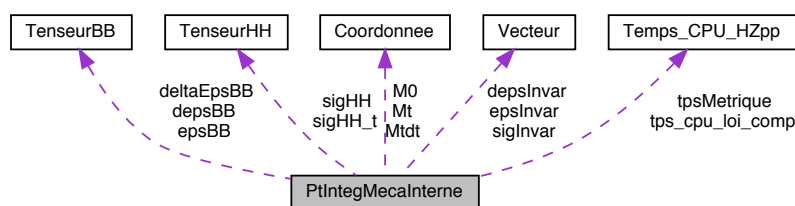
15/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/General/Projet.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/General/Projet.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/General/Projet2.cc

## 6.663 Référence de la classe PtIntegMecalInterne

Graphe de collaboration de PtIntegMecalInterne:



### Fonctions membres publiques

- **PtIntegMecalInterne** (int dimtens)
- **PtIntegMecalInterne** (const [PtIntegMecalInterne](#) &pti)
- [PtIntegMecalInterne](#) & **operator=** (const [PtIntegMecalInterne](#) &pti)
- [TenseurBB](#) \* **EpsBB** ()
- [TenseurBB](#) \* **DepsBB** ()
- [TenseurBB](#) \* **DeltaEpsBB** ()
- [TenseurHH](#) \* **SigHH** ()
- [TenseurHH](#) \* **SigHH\_t** ()
- double & **ModuleCompressibilite** ()
- double & **ModuleCisaillement** ()
- double & **Volume\_pti** ()
- [Tableau](#)< double > & **Deformation\_equi** ()
- [Tableau](#)< double > & **Deformation\_equi\_t** ()
- [Tableau](#)< double > & **Sig\_equi** ()
- [Tableau](#)< double > & **Sig\_equi\_t** ()
- [Coordonnee](#) & **M\_0** ()
- [Coordonnee](#) & **M\_t** ()
- [Coordonnee](#) & **M\_tdt** ()
- int **Nb\_mail** () const
- int **Nb\_ele** () const
- int **Nb\_pti** () const
- void **Change\_Nb\_mail** (int nb)
- void **Change\_Nb\_ele** (int nb)
- void **Change\_Nb\_pti** (int nb)
- void **Signature** () const
- [Temps\\_CPU\\_HZpp](#) & **TpsMetrique** ()
- [Temps\\_CPU\\_HZpp](#) & **Tps\_cpu\_loi\_comp** ()
- const [TenseurBB](#) & **EpsBB\_const** () const
- const [TenseurBB](#) & **DepsBB\_const** () const
- const [TenseurBB](#) & **DeltaEpsBB\_const** () const
- const [TenseurHH](#) & **SigHH\_const** () const
- const [TenseurHH](#) & **SigHH\_t\_const** () const
- const double & **ModuleCompressibilite\_const** () const
- const double & **ModuleCisaillement\_const** () const
- const double & **Volume\_pti\_const** ()

- const [Tableau](#)< double > & **Deformation\_equi\_const** () const
- const [Tableau](#)< double > & **Deformation\_equi\_t\_const** () const
- const [Tableau](#)< double > & **Sig\_equi\_const** () const
- const [Tableau](#)< double > & **Sig\_equi\_t\_const** () const
- const [Coordonnee](#) & **M0\_const** () const
- const [Coordonnee](#) & **Mt\_const** () const
- const [Coordonnee](#) & **Mtdt\_const** () const
- const [Temps\\_CPU\\_HZpp](#) & **TpsMetrique\_const** () const
- const [Temps\\_CPU\\_HZpp](#) & **Tps\_cpu\_loi\_comp\_const** () const
- bool **Statut\_Invariants\_deformation** () const
- bool **Statut\_Invariants\_vitesseDeformation** () const
- bool **Statut\_Invariants\_contrainte** () const
- void **Change\_statut\_Invariants\_deformation** (bool nevez\_statut)
- void **Change\_statut\_Invariants\_vitesseDeformation** (bool nevez\_statut)
- void **Change\_statut\_Invariants\_contrainte** (bool nevez\_statut)
- [Vecteur](#) & **EpsInvar** ()
- const [Vecteur](#) & **EpsInvar\_const** () const
- [Vecteur](#) & **Depsinvar** ()
- const [Vecteur](#) & **Depsinvar\_const** () const
- [Vecteur](#) & **SigInvar** ()
- const [Vecteur](#) & **SigInvar\_const** () const
- void **TdtversT** ()
- void **TversTdt** ()
- void **Affectation\_2D\_a\_3D** (const [PtIntegMecalInterne](#) &ptintmec, bool plusZero)
- void **Affectation\_3D\_a\_2D** (const [PtIntegMecalInterne](#) &ptintmec)
- void **Affectation\_1D\_a\_3D** (const [PtIntegMecalInterne](#) &ptintmec, bool plusZero)
- void **Affectation\_3D\_a\_1D** (const [PtIntegMecalInterne](#) &ptintmec)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

### Attributs protégés

- [TenseurBB](#) \* **epsBB**
- [TenseurBB](#) \* **depsBB**
- [TenseurBB](#) \* **deltaEpsBB**
- [TenseurHH](#) \* **sigHH**
- [TenseurHH](#) \* **sigHH\_t**
- double **module\_compressibilite**
- double **module\_cisaillement**
- double **volume\_pti**
- [Tableau](#)< double > **def\_equi\_t**
- [Tableau](#)< double > **def\_equi**
- [Tableau](#)< double > **sig\_equi\_t**
- [Tableau](#)< double > **sig\_equi**
- [Coordonnee](#) **M0**
- [Coordonnee](#) **Mt**
- [Coordonnee](#) **Mtdt**
- int **mail**
- int **nele**
- int **npti**
- [Vecteur](#) \* **epsInvar**
- [Vecteur](#) \* **depsInvar**
- [Vecteur](#) \* **sigInvar**
- [Temps\\_CPU\\_HZpp](#) **tpsMetrique**
- [Temps\\_CPU\\_HZpp](#) **tps\_cpu\_loi\_comp**

### Amis

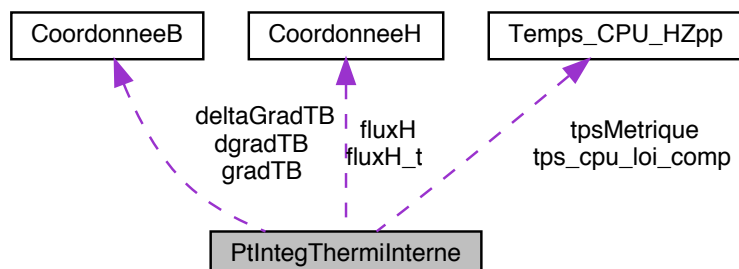
- istream & **operator**>> (istream &, [PtIntegMecalInterne](#) &)
- ostream & **operator**<< (ostream &, const [PtIntegMecalInterne](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/PtIntegMecalInterne.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/PtIntegMecalInterne.cc

## 6.664 Référence de la classe PtIntegThermilInterne

Grphe de collaboration de PtIntegThermilInterne:



### Fonctions membres publiques

- **PtIntegThermilInterne** (int dimtens)
- **PtIntegThermilInterne** (const [PtIntegThermilInterne](#) &pti)
- [PtIntegThermilInterne](#) & **operator=** (const [PtIntegThermilInterne](#) &pti)
- double & **Temperature** ()
- double & **Temperature\_t** ()
- [CoordonneeB](#) & **GradTB** ()
- [CoordonneeB](#) & **DgradTB** ()
- [CoordonneeB](#) & **DeltaGradTB** ()
- [CoordonneeH](#) & **FluxH** ()
- [CoordonneeH](#) & **FluxH\_t** ()
- [Temps\\_CPU\\_HZpp](#) & **TpsMetrique** ()
- [Temps\\_CPU\\_HZpp](#) & **Tps\_cpu\_loi\_comp** ()
- const double & **Temperature\_const** () const
- const double & **Temperature\_t\_const** () const
- const [CoordonneeB](#) & **GradTB\_const** () const
- const [CoordonneeB](#) & **DgradTB\_const** () const
- const [CoordonneeB](#) & **DeltaGradTB\_const** () const
- const [CoordonneeH](#) & **FluxH\_const** () const
- const [CoordonneeH](#) & **FluxH\_t\_const** () const
- const [Temps\\_CPU\\_HZpp](#) & **TpsMetrique\_const** () const
- const [Temps\\_CPU\\_HZpp](#) & **Tps\_cpu\_loi\_comp\_const** () const
- double & **Norme\_gradT** ()
- const double & **Norme\_gradT\_const** () const
- double & **Norme\_DGradT** ()
- const double & **Norme\_DGradT\_const** () const
- double & **Norme\_flux** ()
- const double & **Norme\_flux\_const** () const
- void **TdtversT** ()
- void **TversTdt** ()
- void **Affectation\_2D\_a\_3D** (const [PtIntegThermilInterne](#) &ptinther, bool plusZero)
- void **Affectation\_3D\_a\_2D** (const [PtIntegThermilInterne](#) &ptinther)
- void **Affectation\_1D\_a\_3D** (const [PtIntegThermilInterne](#) &ptinther, bool plusZero)
- void **Affectation\_3D\_a\_1D** (const [PtIntegThermilInterne](#) &ptinther)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

## Attributs protégés

- double **temperature**
- double **temperature\_t**
- [CoordonneeB](#) **gradTB**
- [CoordonneeB](#) **dgradTB**
- [CoordonneeB](#) **deltaGradTB**
- [CoordonneeH](#) **fluxH**
- [CoordonneeH](#) **fluxH\_t**
- double **norme\_gradT**
- double **norme\_dGradT**
- double **norme\_flux**
- [Temps\\_CPU\\_HZpp](#) **tpsMetrique**
- [Temps\\_CPU\\_HZpp](#) **tps\_cpu\_loi\_comp**

## Amis

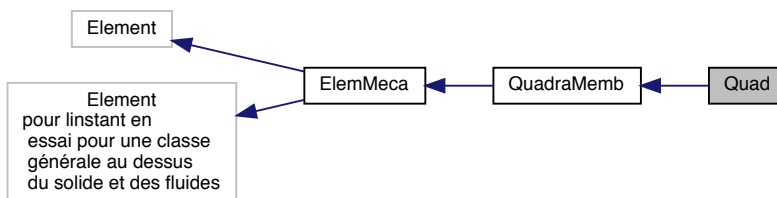
- `istream & operator>>` (`istream &`, [PtIntegThermiInterne &](#))
- `ostream & operator<<` (`ostream &`, `const` [PtIntegThermiInterne &](#))

La documentation de cette classe a été générée à partir du fichier suivant :

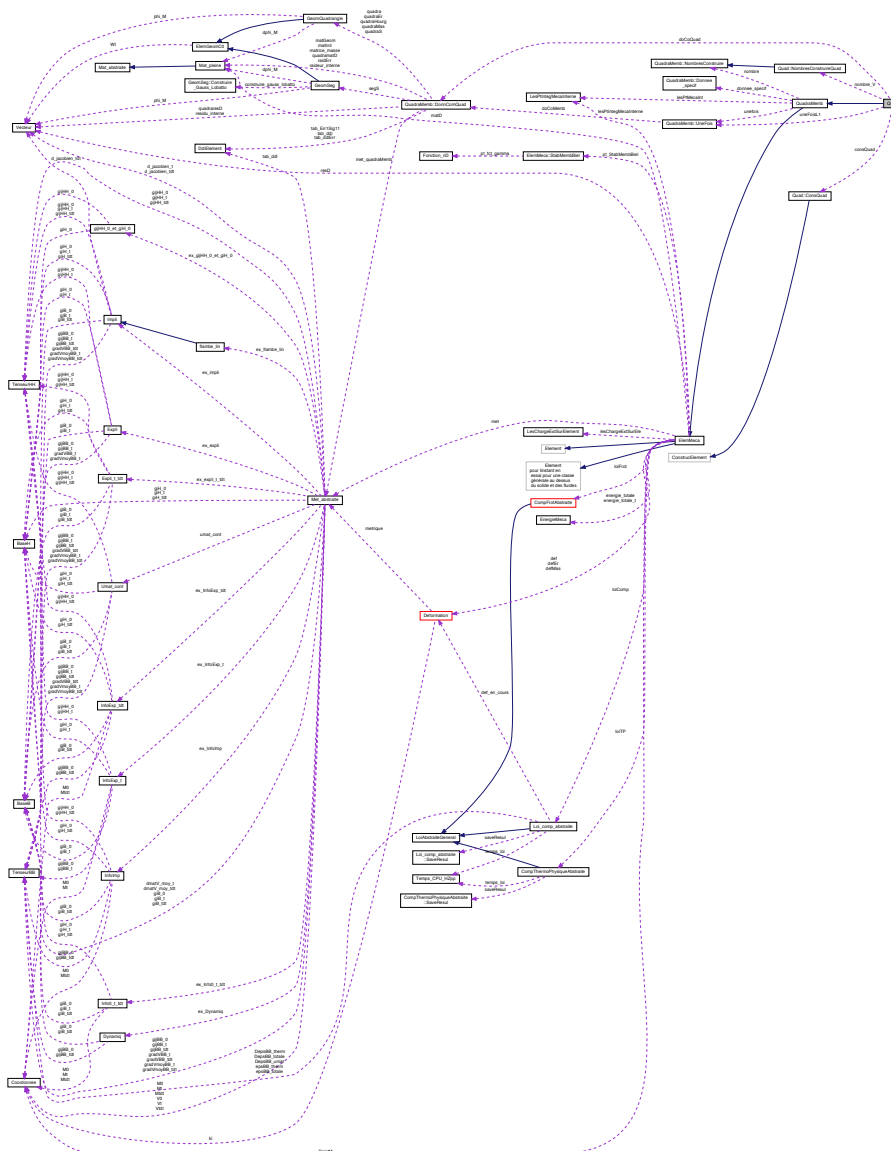
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/PtIntegThermiInterne.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Thermique/PtIntegThermiInterne.cc`

## 6.665 Référence de la classe Quad

Graphe d'héritage de Quad:



Graphe de collaboration de Quad:



## Classes

- class [ConsQuad](#)
- class [NombresConstruireQuad](#)

## Fonctions membres publiques

- **Quad** (double epaiss, int num\_mail=0, int num\_id=-3)
- **Quad** (int num\_mail, int num\_id)
- **Quad** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **Quad** (const [Quad](#) &quadra)
- [Element](#) \* **Nevez\_copie** () const
- [Quad](#) & **operator=** ([Quad](#) &quadra)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdElement](#) &ddelem)
- [EIFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadraMemb::DonnComQuad](#) \* **doCoQuad** = NULL
- static [QuadraMemb::UneFois](#) **uneFoisL1**
- static [NombresConstruireQuad](#) **nombre\_V**
- static [ConsQuad](#) **consQuad**

## Membres hérités additionnels

### 6.665.1 Documentation des fonctions membres

#### 6.665.1.1 new\_frontiere\_lin()

```
ElFrontiere * Quad::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.665.1.2 new\_frontiere\_surf()

```
ElFrontiere * Quad::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

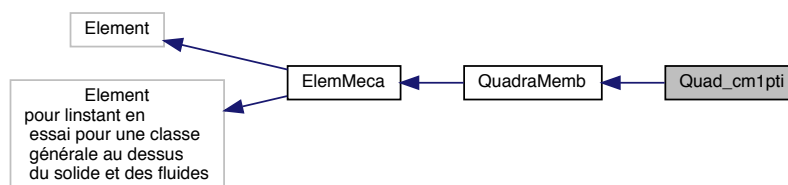
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad.cc

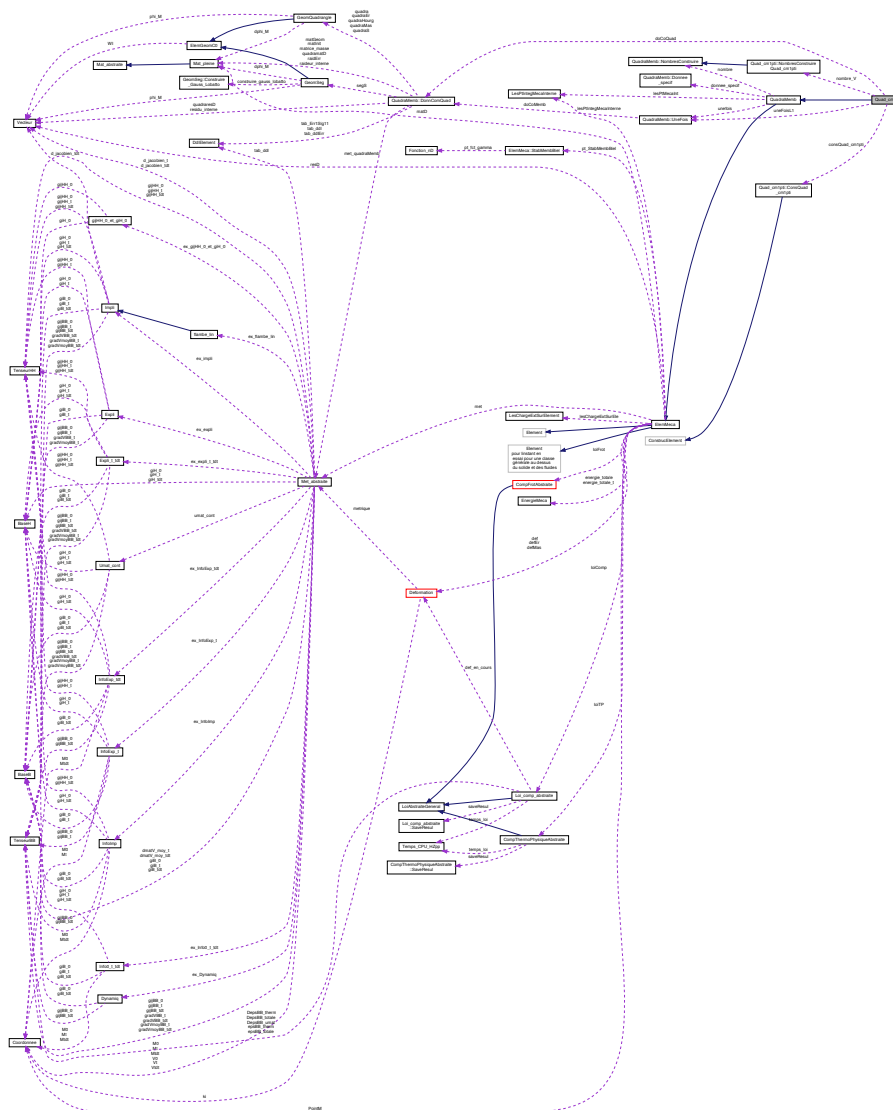
## 6.666 Référence de la classe Quad\_cm1pti

Graphe d'héritage de Quad\_cm1pti:





Graphe de collaboration de Quad\_cm1pti:



## Classes

- class [ConsQuad\\_cm1pti](#)
- class [NombresConstruireQuad\\_cm1pti](#)

## Fonctions membres publiques

- [Quad\\_cm1pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [Quad\\_cm1pti](#) (int num\_mail, int num\_id)
- [Quad\\_cm1pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [Quad\\_cm1pti](#) (const [Quad\\_cm1pti](#) &quadra)
- Element \* [Nevez\\_copie](#) () const
- [Quad\\_cm1pti](#) & [operator=](#) ([Quad\\_cm1pti](#) &quadra)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EifFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdiElement](#) &ddelem)
- [EifFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdiElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadraMemb::DonnComQuad](#) \* **doCoQuad** = NULL
- static [QuadraMemb::UneFois](#) **uneFoisL1**
- static [NombresConstruireQuad\\_cm1pti](#) **nombre\_V**
- static [ConsQuad\\_cm1pti](#) **consQuad\_cm1pti**

## Membres hérités additionnels

### 6.666.1 Documentation des fonctions membres

#### 6.666.1.1 new\_frontiere\_lin()

```
ElFrontiere * Quad_cm1pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.666.1.2 new\_frontiere\_surf()

```
ElFrontiere * Quad_cm1pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

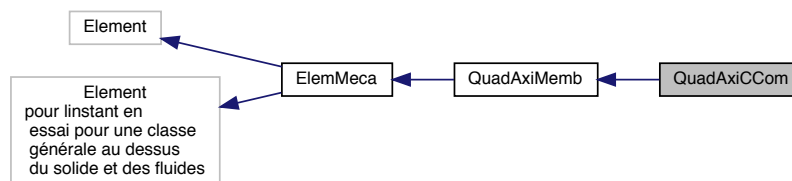
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

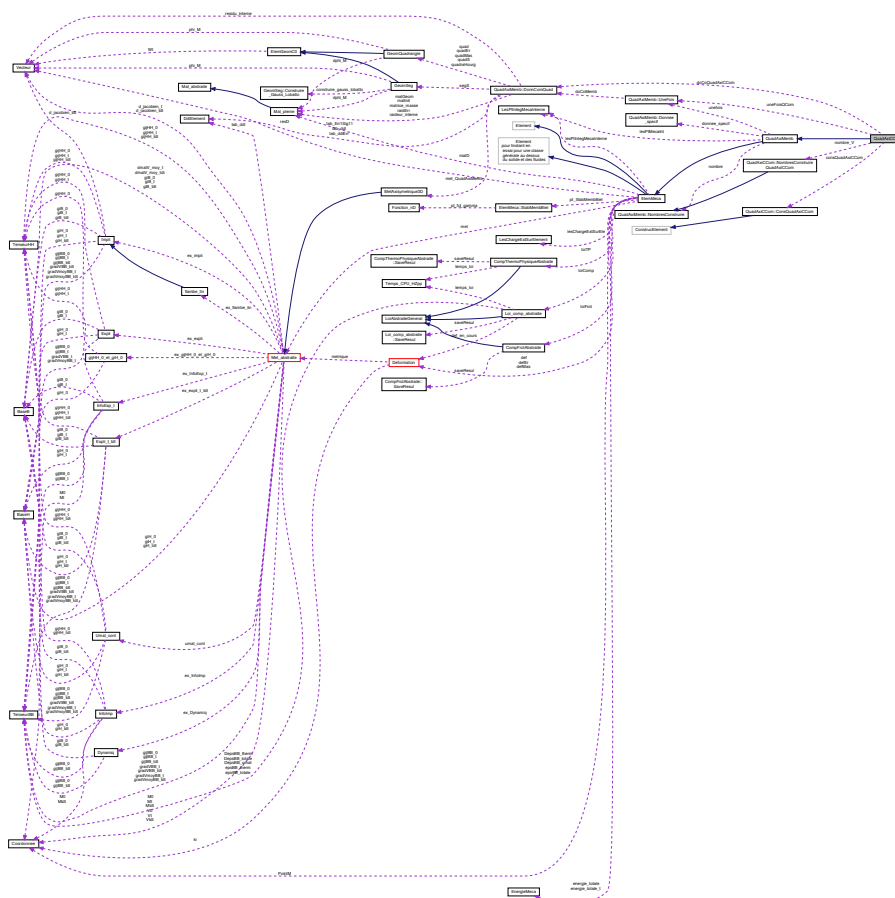
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/Quad\_cm1pti.cc

## 6.667 Référence de la classe QuadAxiCCom

Graphe d'héritage de QuadAxiCCom:



Graphe de collaboration de QuadAxiCCom:



## Classes

- class [ConsQuadAxiCCom](#)
- class [NombresConstruireQuadAxiCCom](#)

## Fonctions membres publiques

- [QuadAxiCCom](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [QuadAxiCCom](#) (int num\_mail, int num\_id)
- [QuadAxiCCom](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [QuadAxiCCom](#) (const [QuadAxiCCom](#) &quadra)
- [Element](#) \* [Nevez\\_copie](#) () const
- [QuadAxiCCom](#) & [operator=](#) ([QuadAxiCCom](#) &quadra)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadAxiMemb::DonnComQuad](#) \* [doCoQuadAxiCCom](#) = NULL
- static [QuadAxiMemb::UneFois](#) [uneFoisQCom](#)
- static [NombresConstruireQuadAxiCCom](#) [nombre\\_V](#)
- static [ConsQuadAxiCCom](#) [consQuadAxiCCom](#)

## Membres hérités additionnels

### 6.667.1 Documentation des fonctions membres

#### 6.667.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadAxiCCom::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.667.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadAxiCCom::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

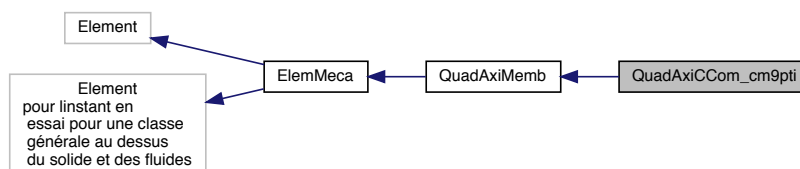
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

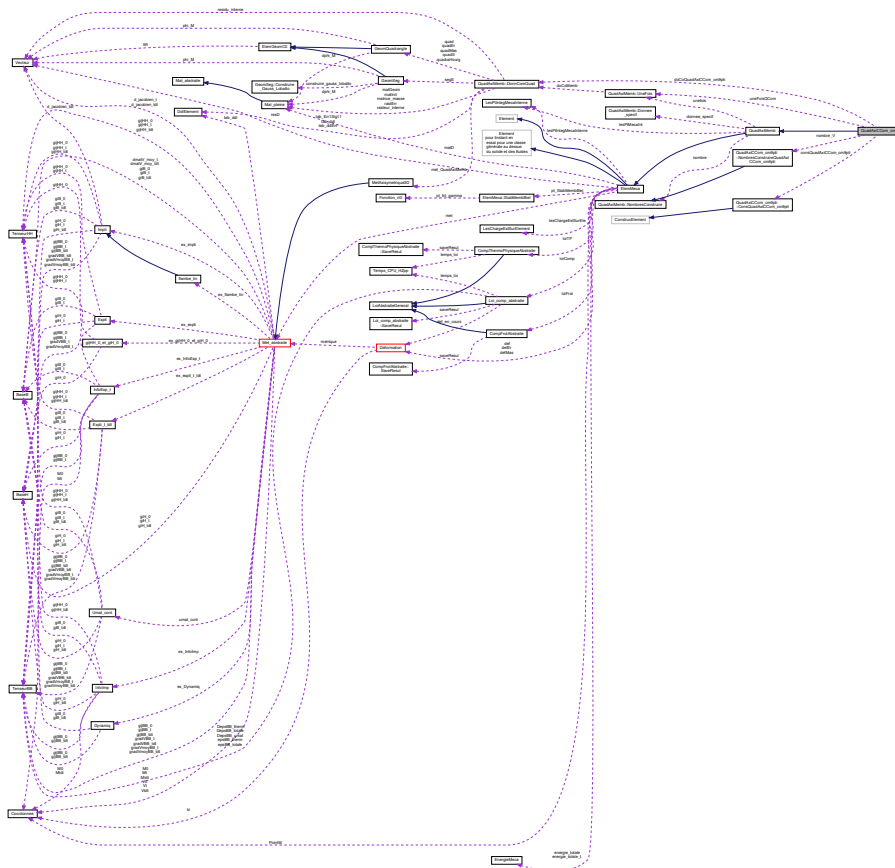
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom.cc

## 6.668 Référence de la classe QuadAxiCCom\_cm9pti

Grappe d'héritage de QuadAxiCCom\_cm9pti:



Graphe de collaboration de QuadAxiCCom\_cm9pti:



## Classes

- class [ConsQuadAxiCCom\\_cm9pti](#)
- class [NombresConstruireQuadAxiCCom\\_cm9pti](#)

## Fonctions membres publiques

- [QuadAxiCCom\\_cm9pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [QuadAxiCCom\\_cm9pti](#) (int num\_mail, int num\_id)
- [QuadAxiCCom\\_cm9pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [QuadAxiCCom\\_cm9pti](#) (const [QuadAxiCCom\\_cm9pti](#) &quadra)
- [Element](#) \* [Nevez\\_copie](#) () const
- [QuadAxiCCom\\_cm9pti](#) & [operator=](#) ([QuadAxiCCom\\_cm9pti](#) &quadra)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadAxiMemb::DonnComQuad](#) \* [doCoQuadAxiCCom\\_cm9pti](#) = NULL
- static [QuadAxiMemb::UneFois](#) [uneFoisQCom](#)
- static [NombresConstruireQuadAxiCCom\\_cm9pti](#) [nombre\\_V](#)
- static [ConsQuadAxiCCom\\_cm9pti](#) [consQuadAxiCCom\\_cm9pti](#)

## Membres hérités additionnels

### 6.668.1 Documentation des fonctions membres

#### 6.668.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadAxiCCom_cm9pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.668.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadAxiCCom_cm9pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

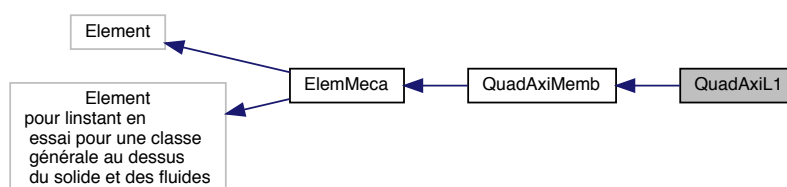
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

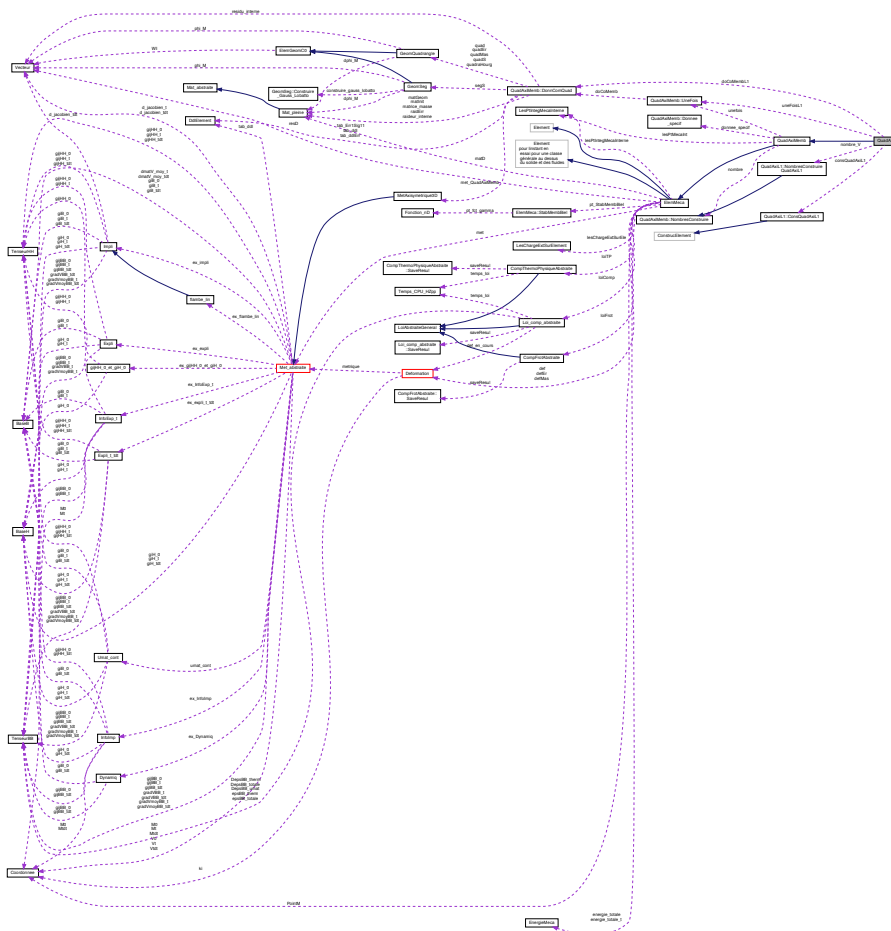
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom\_cm9pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiCCom\_cm9pti.cc

## 6.669 Référence de la classe QuadAxiL1

Graphe d'héritage de QuadAxiL1:



Graphe de collaboration de QuadAxil1:



## Classes

- class [ConsQuadAxil1](#)
- class [NombresConstruireQuadAxil1](#)

## Fonctions membres publiques

- [QuadAxil1](#) (int num\_mail, int num\_id)
- [QuadAxil1](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [QuadAxil1](#) (const [QuadAxil1](#) &quad)
- Element \* [Nevez\\_copie](#) () const
- [QuadAxil1](#) & [operator=](#) ([QuadAxil1](#) &quad)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadAxilMemb::DonnComQuad](#) \* [doCoMembL1](#) = NULL
- static [QuadAxilMemb::UneFois](#) [uneFoisL1](#)
- static [NombresConstruireQuadAxil1](#) [nombre\\_V](#)
- static [ConsQuadAxil1](#) [consQuadAxil1](#)

## Membres hérités additionnels

### 6.669.1 Documentation des fonctions membres

#### 6.669.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadAxiL1::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.669.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadAxiL1::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

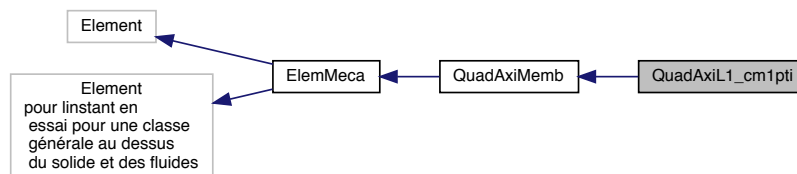
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1.cc

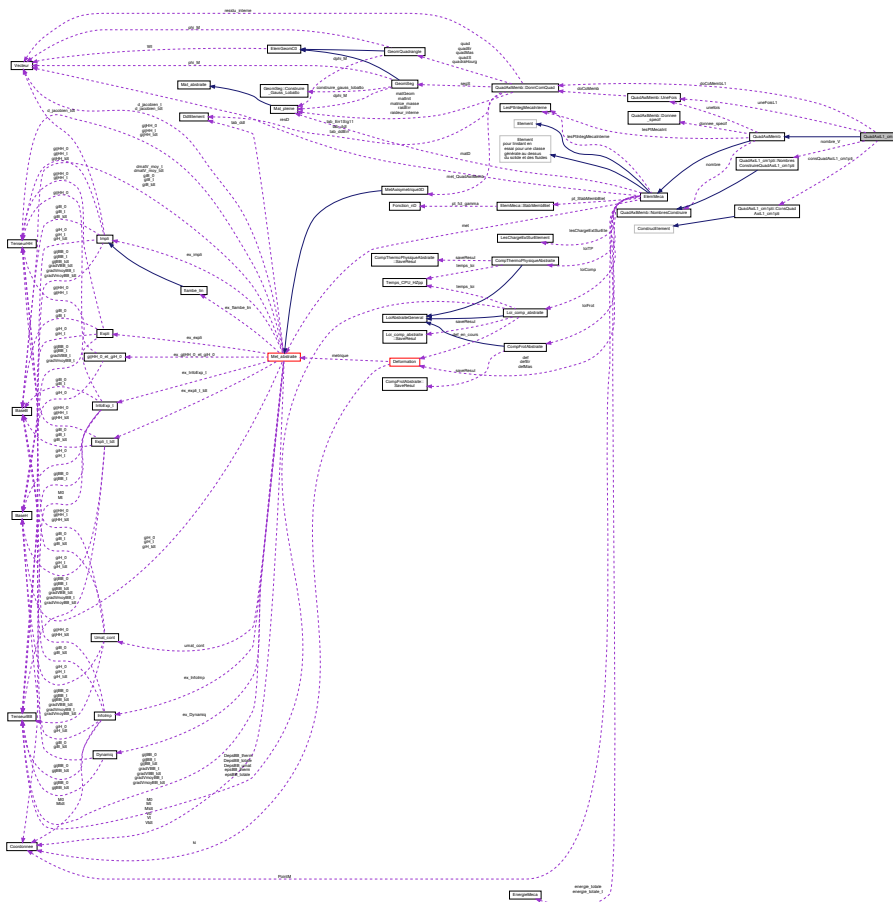
## 6.670 Référence de la classe QuadAxiL1\_cm1pti

Graphe d'héritage de QuadAxiL1\_cm1pti:





Graphe de collaboration de QuadAxiL1\_cm1pti:



## Classes

- class [ConsQuadAxiL1\\_cm1pti](#)
- class [NombresConstruireQuadAxiL1\\_cm1pti](#)

## Fonctions membres publiques

- [QuadAxiL1\\_cm1pti](#) (int num\_mail, int num\_id)
- [QuadAxiL1\\_cm1pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [QuadAxiL1\\_cm1pti](#) (const [QuadAxiL1\\_cm1pti](#) &quad)
- [Element](#) \* [Nevez\\_copie](#) () const
- [QuadAxiL1\\_cm1pti](#) & [operator=](#) ([QuadAxiL1\\_cm1pti](#) &quad)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadAxiMemb::DonnComQuad](#) \* [doCoMembL1](#) = NULL
- static [QuadAxiMemb::UneFois](#) [uneFoisL1](#)
- static [NombresConstruireQuadAxiL1\\_cm1pti](#) [nombre\\_V](#)
- static [ConsQuadAxiL1\\_cm1pti](#) [consQuadAxiL1\\_cm1pti](#)

## Membres hérités additionnels

### 6.670.1 Documentation des fonctions membres

#### 6.670.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadAxiL1_cmlpti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.670.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadAxiL1_cmlpti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

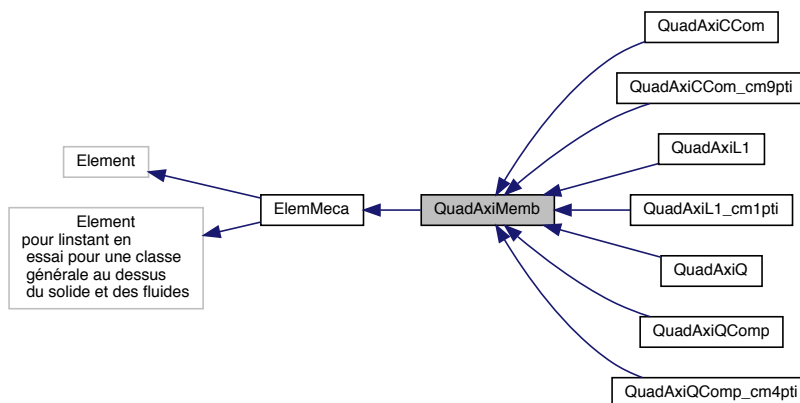
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

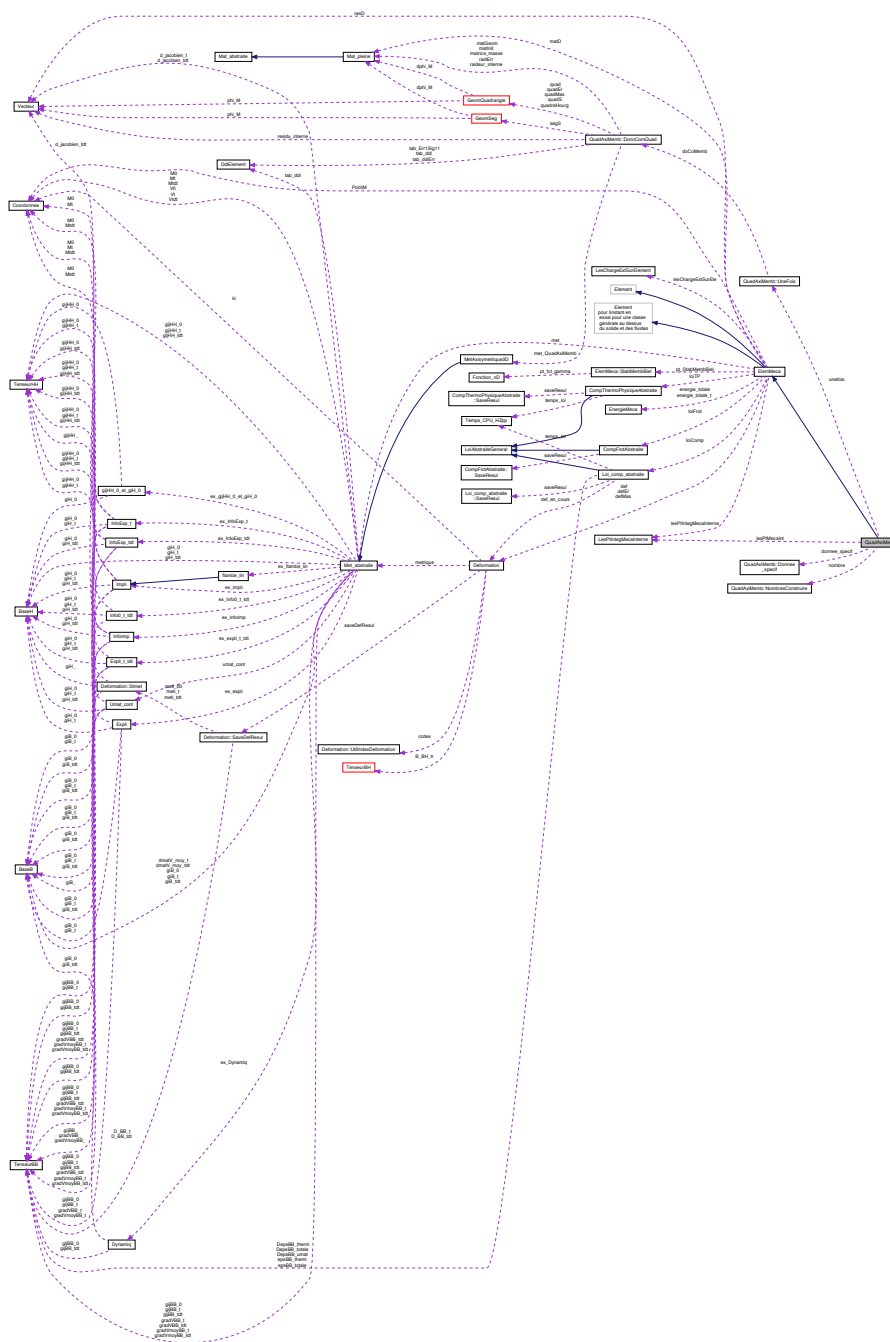
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1\_cmlpti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiL1\_cmlpti.cc

## 6.671 Référence de la classe QuadAxiMemb

Graphe d'héritage de QuadAxiMemb:



Graphe de collaboration de QuadAxiMemb:



## Classes

- class [DonnComQuad](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)
- class [UneFois](#)

## Fonctions membres publiques

- **QuadAxiMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")

- **QuadAxiMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, const [Tableau](#)< [Noeud](#) \* > &tab, string info="")
- **QuadAxiMemb** (const [QuadAxiMemb](#) &quad)
- **QuadAxiMemb** & **operator=** (const [QuadAxiMemb](#) &quad)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- **ElemGeomC0** & **ElementGeometrique** () const
- const **ElemGeomC0** & **ElementGeometrique\_const** () const
- **Coordonnee** & **Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComplet** ()
- Element \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- Element \* **Complet\_Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- bool **SurfExiste** (int ns) const
- bool **AreteExiste** (int na) const
- const **DdlElement** & **TableauDdl** () const
- Element::ResRaid **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- **DdlElement** & **Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- ResRaid **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_lineique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_lineique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)

- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_lineique\\_Suiv\\_E\\_t](#) (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_lineique\\_Suiv\\_E\\_tdt](#) (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_pression\\_E\\_t](#) (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_pression\\_E\\_tdt](#) (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_hydrostatique\\_E\\_t](#) (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- [Vecteur SM\\_charge\\_hydrostatique\\_E\\_tdt](#) (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- [Vecteur SM\\_charge\\_hydrodynamique\\_E\\_t](#) ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_hydrodynamique\\_E\\_tdt](#) ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Tableau](#)< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- void **ConstTabDdl** ()

## Fonctions membres protégées

- [QuadAxiMemb::DonnComQuad](#) \* **Init** ([Donnee\\_specif](#) donnee\_specif=[Donnee\\_specif](#)()), bool sans\_init\_↔ noeud=false)
- void **Destruction** ()
- int **Dim\_sig\_eps** () const

## Attributs protégés

- [UneFois](#) \* **unefois**
- [Donnee\\_specif](#) **donnee\_specif**
- [LesPtIntegMecalInterne](#) **lesPtMecalnt**
- [NombresConstruire](#) \* **nombre**

## Membres hérités additionnels

### 6.671.1 Documentation des fonctions membres

#### 6.671.1.1 Active\_ddl\_Sigma()

void [QuadAxiMemb::Active\\_ddl\\_Sigma](#) ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

#### 6.671.1.2 Active\_premier\_ddl\_Sigma()

void [QuadAxiMemb::Active\\_premier\\_ddl\\_Sigma](#) ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

### 6.671.1.3 ContraintesAbsolues()

```
bool QuadAxiMemb::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.671.1.4 Dim\_sig\_eps()

```
int QuadAxiMemb::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.671.1.5 ErreurElement()

```
void QuadAxiMemb::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

### 6.671.1.6 Inactive\_ddl\_Sigma()

```
void QuadAxiMemb::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.671.1.7 LectureContraintes()

```
void QuadAxiMemb::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.671.1.8 Long\_arrete\_mini\_sur\_c()

```
double QuadAxiMemb::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.671.1.9 Plus\_ddl\_Sigma()

```
void QuadAxiMemb::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.671.1.10 Tableau\_de\_Sig1()

```
DdlElement & QuadAxiMemb::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

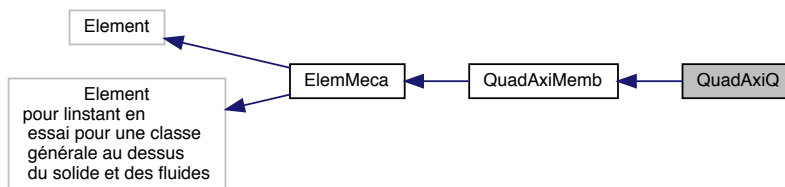
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

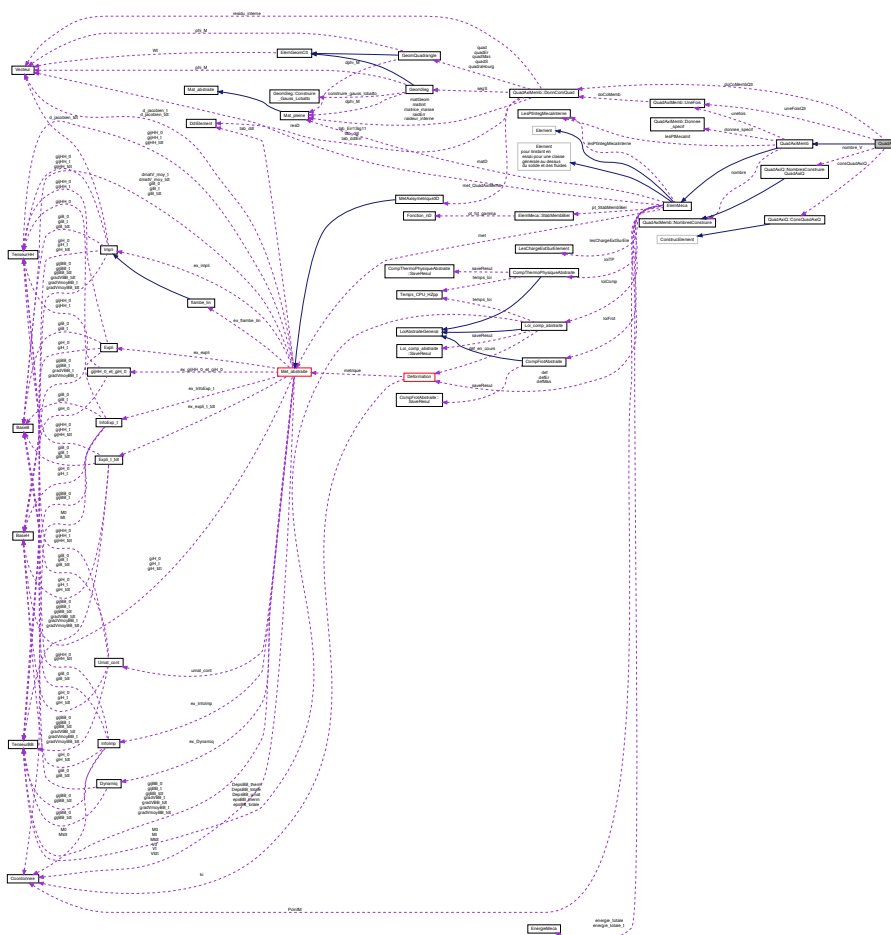
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiMemb.cc

## 6.672 Référence de la classe QuadAxiQ

Graphe d'héritage de QuadAxiQ:



Graphe de collaboration de QuadAxiQ:



### Classes

- class [ConsQuadAxiQ](#)
- class [NombresConstruireQuadAxiQ](#)

### Fonctions membres publiques

- [QuadAxiQ](#) (int num\_mail, int num\_id)

- `QuadAxiQ` (int num\_mail, int num\_id, const `Tableau< Noeud * >` &tab)
- `QuadAxiQ` (const `QuadAxiQ` &quad)
- `Element * Nevez_copie` () const
- `QuadAxiQ & operator=` (`QuadAxiQ` &quad)
- void `AfficheVarDual` (ofstream &sort, `Tableau< string >` &nom)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `QuadAxiMemb::DonnComQuad * doCoMembQ3` = NULL
- static `QuadAxiMemb::UneFois uneFoisQ3`
- static `NombresConstruireQuadAxiQ nombre_V`
- static `ConsQuadAxiQ consQuadAxiQ`

## Membres hérités additionnels

### 6.672.1 Documentation des fonctions membres

#### 6.672.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadAxiQ::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.672.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadAxiQ::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

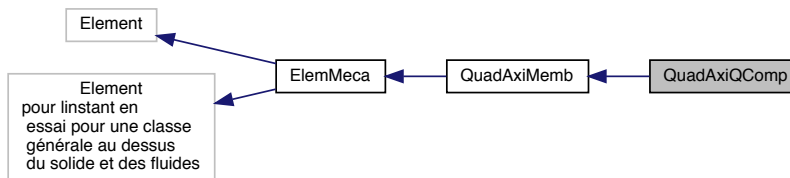
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxi↵  
Q.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxi↵  
Q.cc

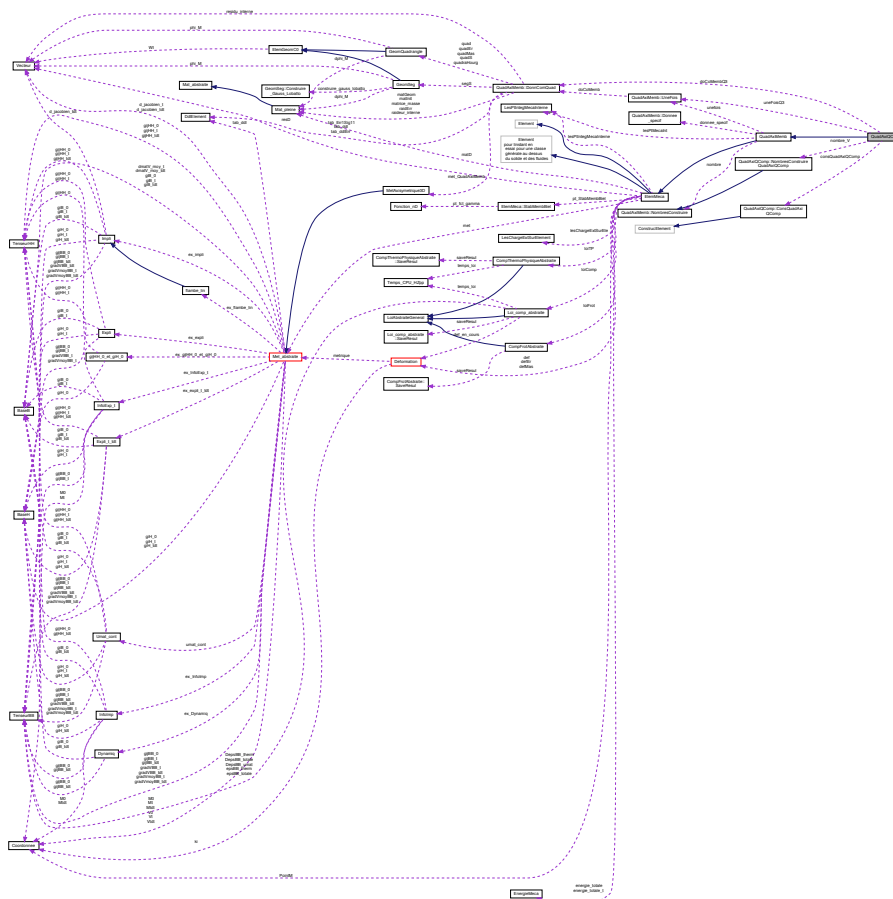


## 6.673 Référence de la classe QuadAxiQComp

Graphe d'héritage de QuadAxiQComp:



Graphe de collaboration de QuadAxiQComp:



### Classes

- class [ConsQuadAxiQComp](#)
- class [NombresConstruireQuadAxiQComp](#)

### Fonctions membres publiques

- [QuadAxiQComp](#) (int num\_mail, int num\_id)
- [QuadAxiQComp](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [QuadAxiQComp](#) (const [QuadAxiQComp](#) &quad)

- `Element * Nevez_copie () const`
- `QuadAxiQComp & operator= (QuadAxiQComp &quad)`
- `void AfficheVarDual (ofstream &sort, Tableau< string > &nom)`

### Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- `static QuadAxiMemb::DonnComQuad * doCoMembQ3 = NULL`
- `static QuadAxiMemb::UneFois uneFoisQ3`
- `static NombresConstruireQuadAxiQComp nombre_V`
- `static ConsQuadAxiQComp consQuadAxiQComp`

### Membres hérités additionnels

#### 6.673.1 Documentation des fonctions membres

##### 6.673.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadAxiQComp::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.673.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadAxiQComp::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

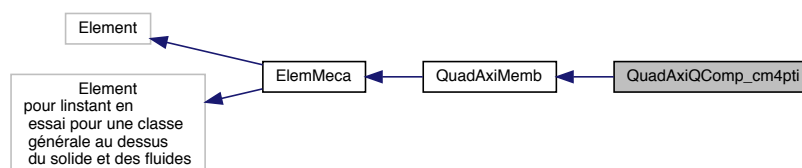
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

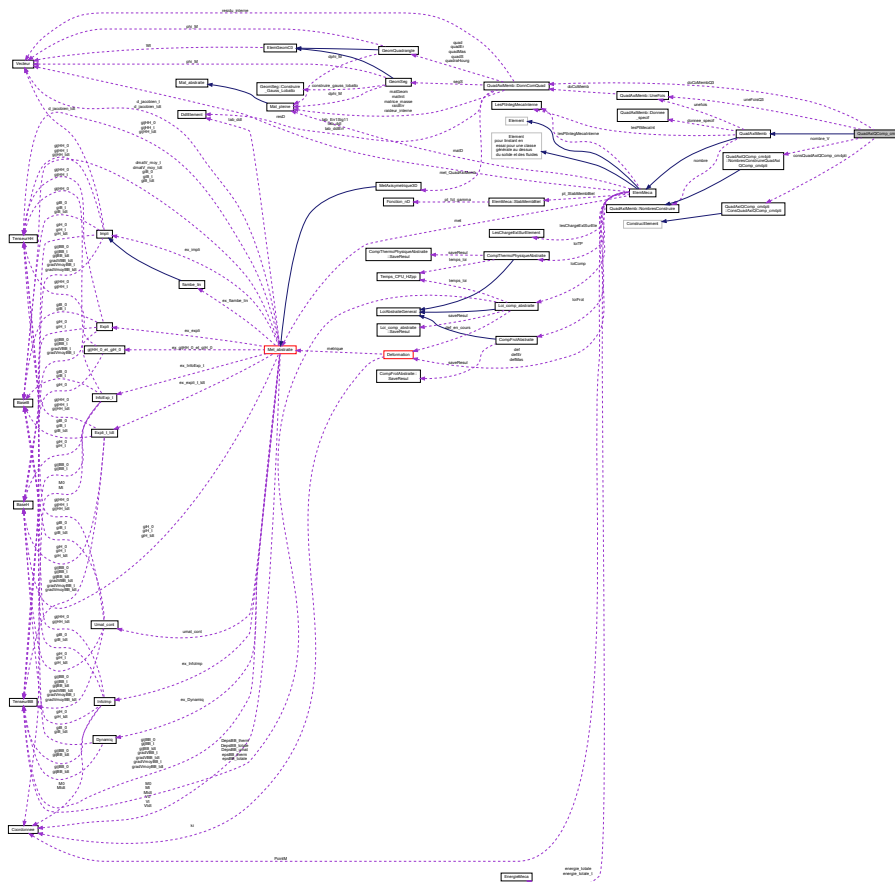
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiQComp.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad_asisymetrie/QuadAxiQComp.cc`

## 6.674 Référence de la classe QuadAxiQComp\_cm4pti

Grphe d'héritage de QuadAxiQComp\_cm4pti:



Graphe de collaboration de QuadAxiQComp\_cm4pti:



## Classes

- class [ConsQuadAxiQComp\\_cm4pti](#)
- class [NombresConstruireQuadAxiQComp\\_cm4pti](#)

## Fonctions membres publiques

- [QuadAxiQComp\\_cm4pti](#) (int num\_mail, int num\_id)
- [QuadAxiQComp\\_cm4pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [QuadAxiQComp\\_cm4pti](#) (const [QuadAxiQComp\\_cm4pti](#) &quad)
- [Element](#) \* [Nevez\\_copie](#) () const
- [QuadAxiQComp\\_cm4pti](#) & [operator=](#) ([QuadAxiQComp\\_cm4pti](#) &quad)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadAxiMemb::DonnComQuad](#) \* [doCoMembQ3](#) = NULL
- static [QuadAxiMemb::UneFois](#) [uneFoisQ3](#)
- static [NombresConstruireQuadAxiQComp\\_cm4pti](#) [nombre\\_V](#)
- static [ConsQuadAxiQComp\\_cm4pti](#) [consQuadAxiQComp\\_cm4pti](#)

## Membres hérités additionnels

### 6.674.1 Documentation des fonctions membres

#### 6.674.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadAxiQComp_cm4pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.674.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadAxiQComp_cm4pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

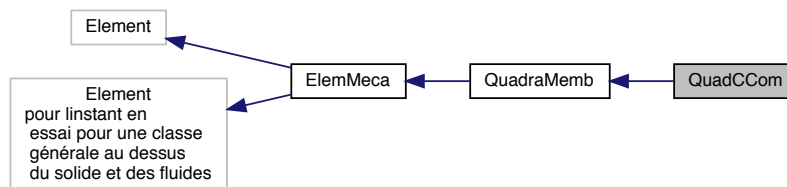
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

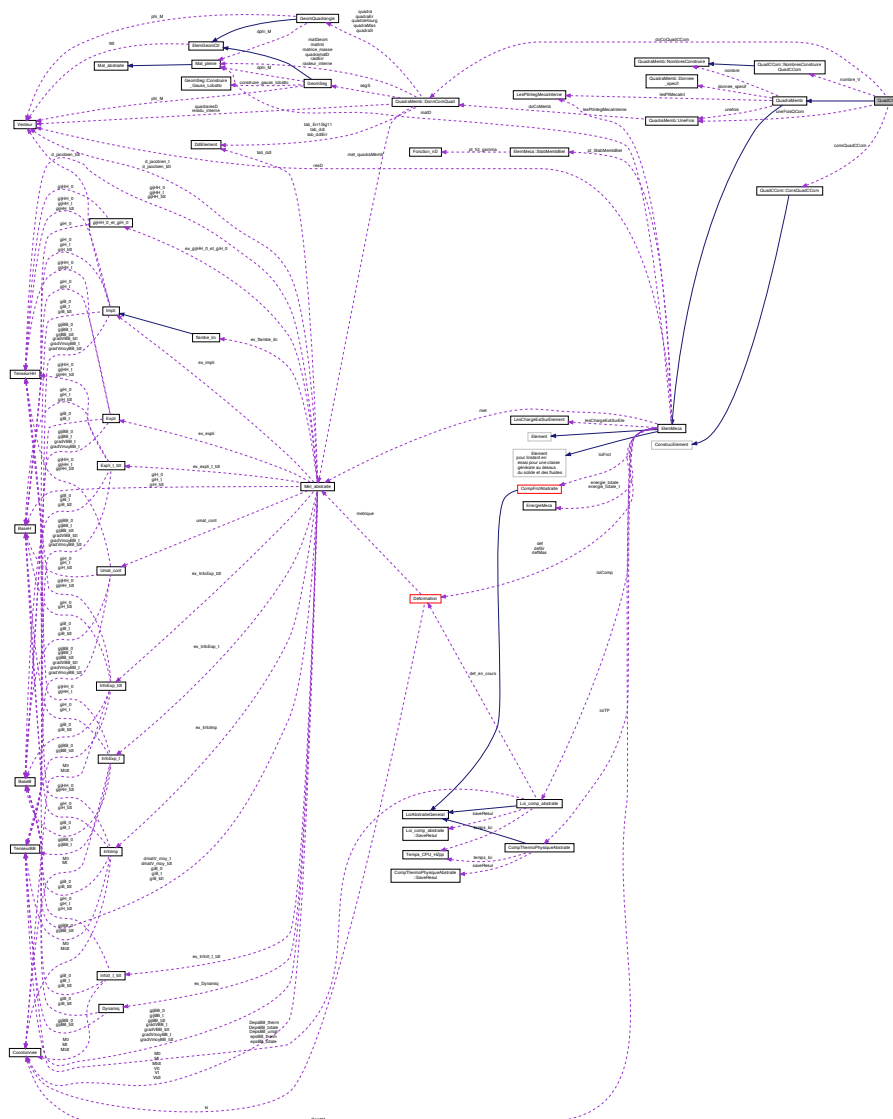
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQComp\_cm4pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiQComp\_cm4pti.cc

## 6.675 Référence de la classe QuadCCom

Graphe d'héritage de QuadCCom:



Graphe de collaboration de QuadCCom:



## Classes

- class [ConsQuadCCom](#)
- class [NombresConstruireQuadCCom](#)

## Fonctions membres publiques

- **QuadCCom** (double epaiss, int num\_mail=0, int num\_id=-3)
- **QuadCCom** (int num\_mail, int num\_id)
- **QuadCCom** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **QuadCCom** (const [QuadCCom](#) &quadra)
- [Element](#) \* **Nevez\_copie** () const
- [QuadCCom](#) & **operator=** ([QuadCCom](#) &quadra)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadraMemb::DonnComQuad](#) \* **doCoQuadCCom** = NULL
- static [QuadraMemb::UneFois](#) **uneFoisQCom**
- static [NombresConstruireQuadCCom](#) **nombre\_V**
- static [ConsQuadCCom](#) **consQuadCCom**

## Membres hérités additionnels

### 6.675.1 Documentation des fonctions membres

#### 6.675.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadCCom::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.675.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadCCom::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

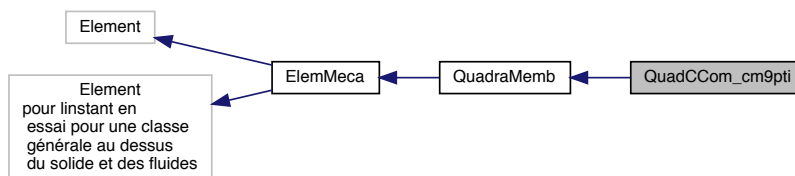
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

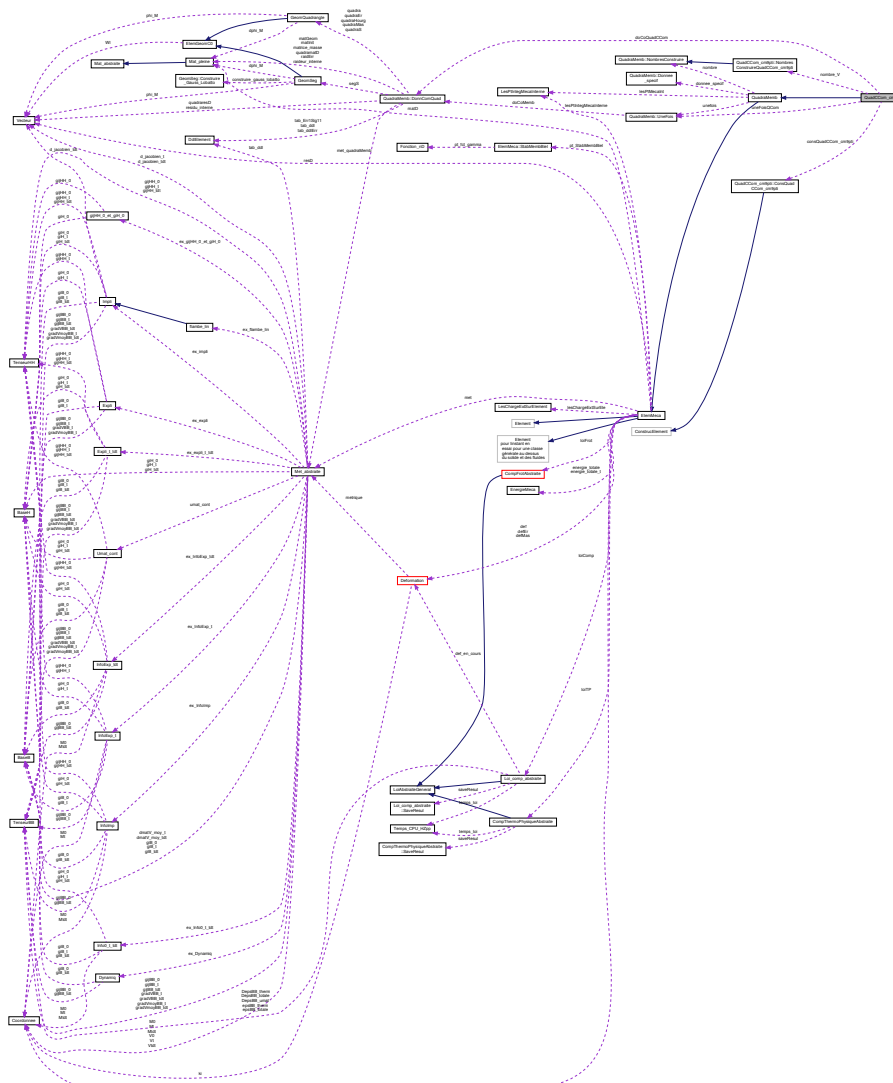
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom.cc

## 6.676 Référence de la classe QuadCCom\_cm9pti

Graphe d'héritage de QuadCCom\_cm9pti:



Graphe de collaboration de QuadCCom\_cm9pti:



## Classes

- class [ConsQuadCCom\\_cm9pti](#)
- class [NombresConstruireQuadCCom\\_cm9pti](#)

## Fonctions membres publiques

- **QuadCCom\_cm9pti** (double epaiss, int num\_mail=0, int num\_id=-3)
- **QuadCCom\_cm9pti** (int num\_mail, int num\_id)
- **QuadCCom\_cm9pti** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **QuadCCom\_cm9pti** (const [QuadCCom\\_cm9pti](#) &quadra)
- [Element](#) \* **Nevez\_copie** () const
- [QuadCCom\\_cm9pti](#) & **operator=** ([QuadCCom\\_cm9pti](#) &quadra)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadraMemb::DonnComQuad](#) \* **doCoQuadCCom** = NULL
- static [QuadraMemb::UneFois](#) **uneFoisQCom**
- static [NombresConstruireQuadCCom\\_cm9pti](#) **nombre\_V**
- static [ConsQuadCCom\\_cm9pti](#) **consQuadCCom\_cm9pti**

## Membres hérités additionnels

### 6.676.1 Documentation des fonctions membres

#### 6.676.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadCCom_cm9pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.676.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadCCom_cm9pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

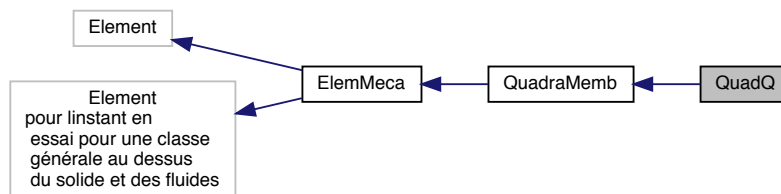
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom\_↔cm9pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadCCom\_↔cm9pti.cc

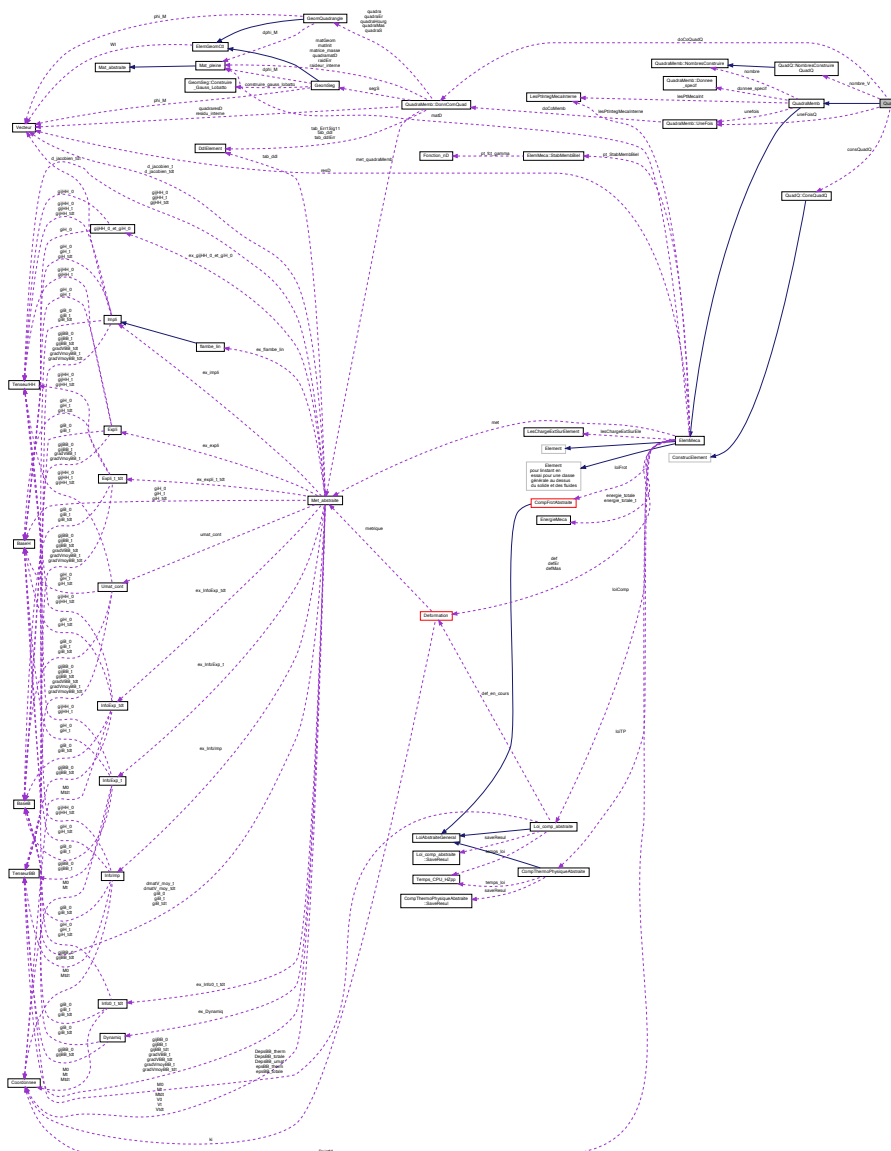
## 6.677 Référence de la classe QuadQ

Grappe d'héritage de QuadQ:





Graphe de collaboration de QuadQ:



## Classes

- class [ConsQuadQ](#)
- class [NombresConstruireQuadQ](#)

## Fonctions membres publiques

- **QuadQ** (double epais, int num\_mail=0, int num\_id=-3)
- **QuadQ** (int num\_mail, int num\_id)
- **QuadQ** (double epais, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **QuadQ** (const [QuadQ](#) &quadra)
- [Element](#) \* **Nevez\_copie** () const
- [QuadQ](#) & **operator=** ([QuadQ](#) &quadra)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadraMemb::DonnComQuad](#) \* **doCoQuadQ** = NULL
- static [QuadraMemb::UneFois](#) **uneFoisQ**
- static [NombresConstruireQuadQ](#) **nombre\_V**
- static [ConsQuadQ](#) **consQuadQ**

## Membres hérités additionnels

### 6.677.1 Documentation des fonctions membres

#### 6.677.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadQ::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.677.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadQ::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

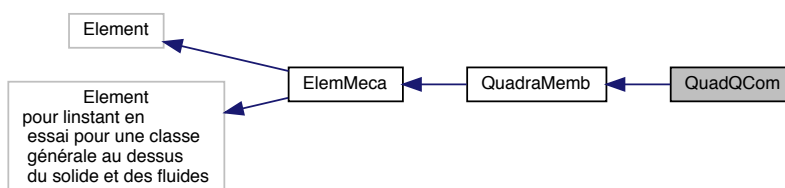
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

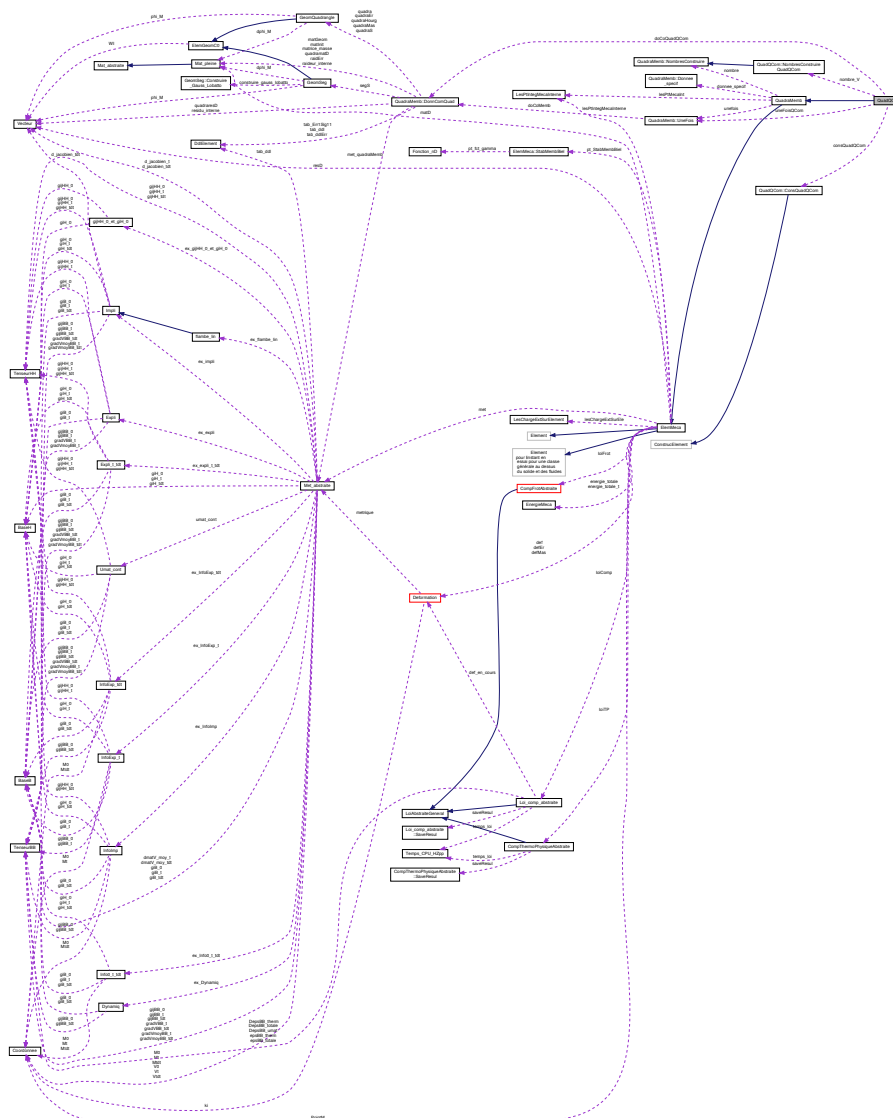
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQ.cc

## 6.678 Référence de la classe QuadQCom

Graphe d'héritage de QuadQCom:



Graphe de collaboration de QuadQCom:



## Classes

- class [ConsQuadQCom](#)
- class [NombresConstruireQuadQCom](#)

## Fonctions membres publiques

- **QuadQCom** (double epaiss, int num\_mail=0, int num\_id=-3)
- **QuadQCom** (int num\_mail, int num\_id)
- **QuadQCom** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **QuadQCom** (const [QuadQCom](#) &quadra)
- [Element](#) \* **Nevez\_copie** () const
- [QuadQCom](#) & **operator=** ([QuadQCom](#) &quadra)
- virtual list< [Noeud](#) \* > **Construct\_from\_imcomplet** (const [Element](#) &elem, list< [DeuxEntiers](#) > &li\_↵ bornes, int nbnt)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

— `ElFrontiere * new_frontiere_surf` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)

### Attributs protégés statiques

— static `QuadraMemb::DonnComQuad * doCoQuadQCom` = NULL  
 — static `QuadraMemb::UneFois uneFoisQCom`  
 — static `NombresConstruireQuadQCom nombre_V`  
 — static `ConsQuadQCom consQuadQCom`

### Membres hérités additionnels

#### 6.678.1 Documentation des fonctions membres

##### 6.678.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadQCom::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.678.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadQCom::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

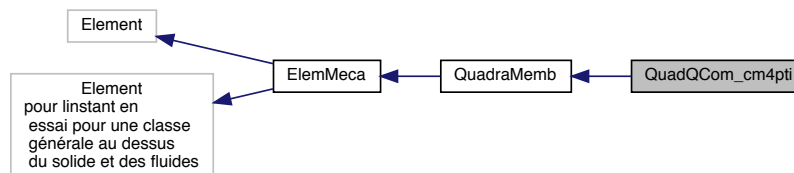
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

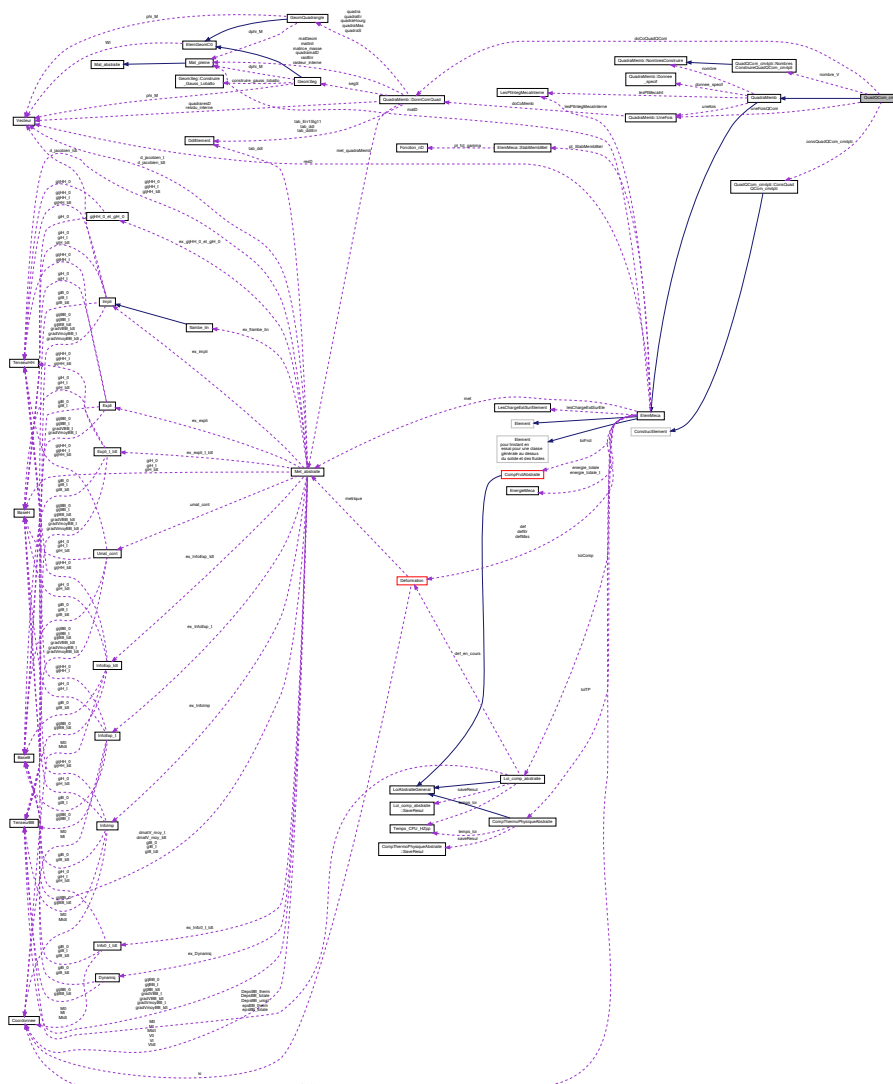
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom.cc

## 6.679 Référence de la classe QuadQCom\_cm4pti

Grappe d'héritage de QuadQCom\_cm4pti:



Graphe de collaboration de QuadQCom\_cm4pti:



## Classes

- class [ConsQuadQCom\\_cm4pti](#)
- class [NombresConstruireQuadQCom\\_cm4pti](#)

## Fonctions membres publiques

- [QuadQCom\\_cm4pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [QuadQCom\\_cm4pti](#) (int num\_mail, int num\_id)
- [QuadQCom\\_cm4pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [QuadQCom\\_cm4pti](#) (const [QuadQCom\\_cm4pti](#) &quadra)
- Element \* [Nevez\\_copie](#) () const
- [QuadQCom\\_cm4pti](#) & [operator=](#) ([QuadQCom\\_cm4pti](#) &quadra)
- virtual list< [Noeud](#) \* > [Construct\\_from\\_imcomplet](#) (const Element &elem, list< [DeuxEntiers](#) > &li\_↔ bornes, int nbnt)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [QuadraMemb::DonnComQuad](#) \* **doCoQuadQCom** = NULL
- static [QuadraMemb::UneFois](#) **uneFoisQCom**
- static [NombresConstruireQuadQCom\\_cm4pti](#) **nombre\_V**
- static [ConsQuadQCom\\_cm4pti](#) **consQuadQCom\_cm4pti**

## Membres hérités additionnels

### 6.679.1 Documentation des fonctions membres

#### 6.679.1.1 new\_frontiere\_lin()

```
ElFrontiere * QuadQCom_cm4pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.679.1.2 new\_frontiere\_surf()

```
ElFrontiere * QuadQCom_cm4pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

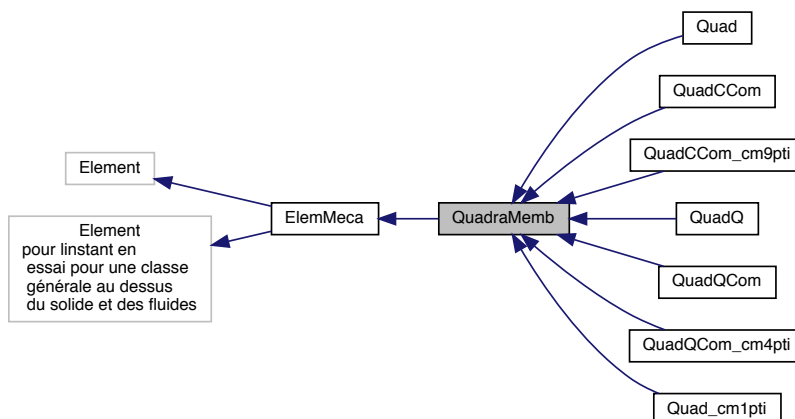
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

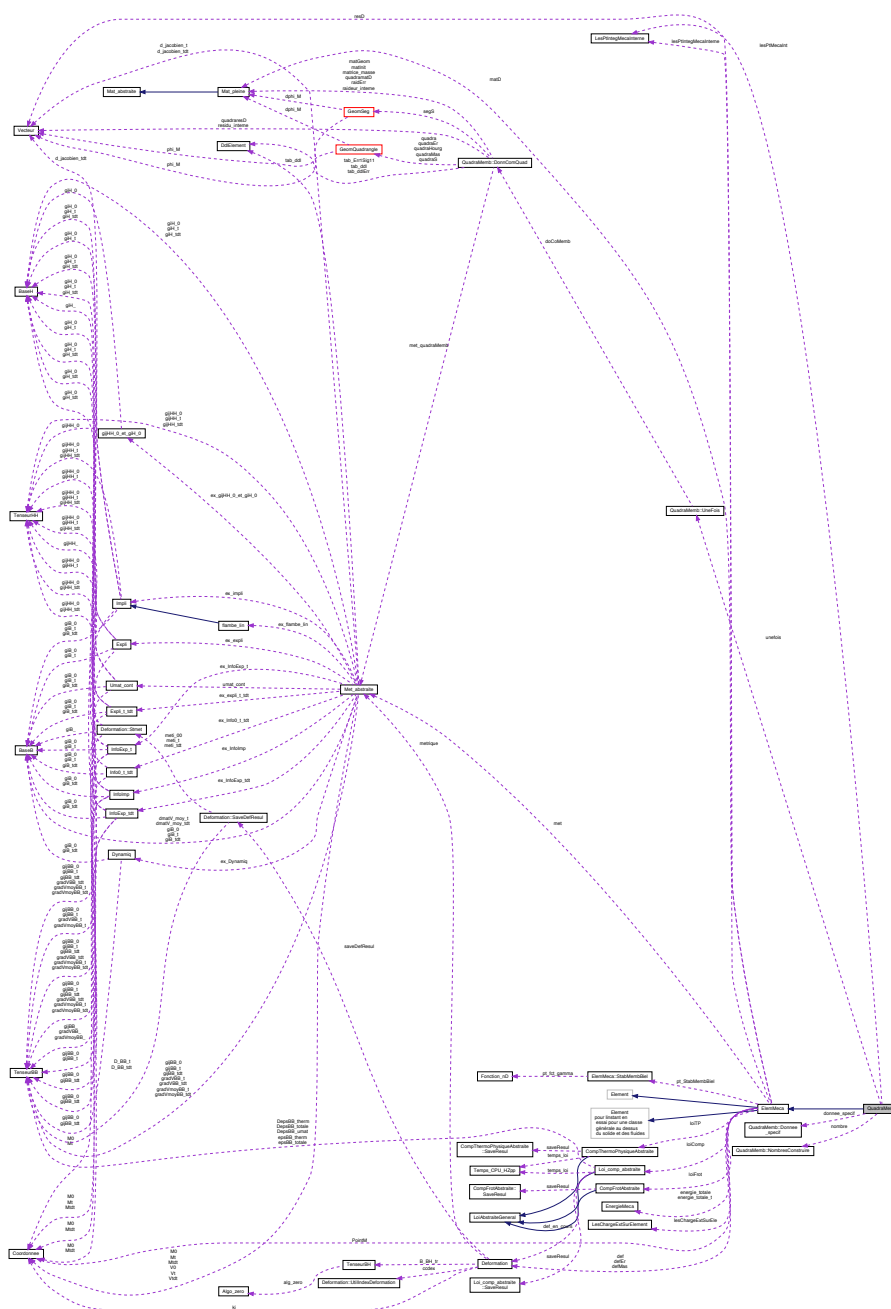
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom\_↔cm4pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadQCom\_↔cm4pti.cc

## 6.680 Référence de la classe QuadraMemb

Graphe d'héritage de QuadraMemb:



Graphe de collaboration de QuadraMemb:



## Classes

- class [DonnComQuad](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)
- class [UneFois](#)

## Fonctions membres publiques

- **QuadraMemb** (int num\_mail, int num\_id, [Enum\\_interp](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")
- **QuadraMemb** (int num\_mail, int num\_id, [Enum\\_interp](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, const [Tableau](#)< [Noeud](#) \* > &tab, string info="")

- **QuadraMemb** (const [QuadraMemb](#) &quadra)
- **QuadraMemb & operator=** ([QuadraMemb](#) &quadra)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- **ElemGeomC0 & ElementGeometrique** () const
- const **ElemGeomC0 & ElementGeometrique\_const** () const
- **Coordonnee & Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- virtual double **Epaisseur** ([Enum\\_dure](#) enu, const [Coordonnee](#) &)
- virtual double **EpaisseurMoyenne** ([Enum\\_dure](#) enu)
- **List\_io**< [TypeQuelconque](#) > **Les\_types\_particuliers\_interne**s (bool absolue) const
- void **Grandeur\_particuliere** (bool absolue, **List\_io**< [TypeQuelconque](#) > &litQ, int iteg)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComple**t ()
- Element \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- Element \* **Comple**t\_Hourglass ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- double **H** (int ni, [Enum\\_dure](#) enu=TEMPS\_tdt)
- double **H\_moy** ([Enum\\_dure](#) enu)
- bool **SurfExiste** (int ns) const
- bool **AreteExiste** (int na) const
- const **DdlElement & TableauDdl** () const
- Element::ResRaid **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const **List\_io**< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, **List\_io**< [TypeQuelconque](#) > &enu, int iteg)
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- **DdlElement & Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual const **DeuxCoordonnees & Boite\_encombre\_element** ([Enum\\_dure](#) temps)
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- ResRaid **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)



- Vecteur **SM\_charge\_lineique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E\_t** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E\_tdt** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_presUniDir\_E\_t** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_presUniDir\_E\_tdt** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_presUniDir\_I** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrostatique\_E\_t** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_hydrostatique\_E\_tdt** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Tableau< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- void **ConstTabDdl** ()

### Fonctions membres protégées

- [QuadraMemb::DonnComQuad](#) \* **Init** ([Donnee\\_specif](#) donnee\_specif=[Donnee\\_specif](#)(), bool sans\_init\_↵ noeud=false)
- void **Destruction** ()
- int **Dim\_sig\_eps** () const

### Attributs protégés

- [UneFois](#) \* **unefois**
- [Donnee\\_specif](#) **donnee\_specif**
- [LesPtIntegMecalInterne](#) **lesPtMecalnt**
- [NombresConstruire](#) \* **nombre**

### Membres hérités additionnels

#### 6.680.1 Documentation des fonctions membres

### 6.680.1.1 Active\_ddl\_Sigma()

```
void QuadraMemb::Active_ddl_Sigma ( ) [inline], [virtual]
Implémente ElemMeca.
```

### 6.680.1.2 Active\_premier\_ddl\_Sigma()

```
void QuadraMemb::Active_premier_ddl_Sigma ( ) [inline], [virtual]
Implémente ElemMeca.
```

### 6.680.1.3 ContraintesAbsolues()

```
bool QuadraMemb::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
Implémente ElemMeca.
```

### 6.680.1.4 Dim\_sig\_eps()

```
int QuadraMemb::Dim_sig_eps ( ) const [inline], [protected], [virtual]
Implémente ElemMeca.
```

### 6.680.1.5 EpaisseurMoyenne()

```
virtual double QuadraMemb::EpaisseurMoyenne (
    Enum_dure enu ) [inline], [virtual]
Réimplémentée à partir de ElemMeca.
```

### 6.680.1.6 Epaisseurs()

```
virtual double QuadraMemb::Epaisseurs (
    Enum_dure enu,
    const Coordonnee & ) [inline], [virtual]
Réimplémentée à partir de ElemMeca.
```

### 6.680.1.7 ErreurElement()

```
void QuadraMemb::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en  
Réimplémentée à partir de [ElemMeca](#).

### 6.680.1.8 Inactive\_ddl\_Sigma()

```
void QuadraMemb::Inactive_ddl_Sigma ( ) [inline], [virtual]
Implémente ElemMeca.
```

**6.680.1.9 LectureContraintes()**

```
void QuadraMemb::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

**6.680.1.10 Les\_types\_particuliers\_internes()**

```
List_io< TypeQuelconque > QuadraMemb::Les_types_particuliers_internes (
    bool absolue ) const [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

**6.680.1.11 Long\_arrete\_mini\_sur\_c()**

```
double QuadraMemb::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

**6.680.1.12 Plus\_ddl\_Sigma()**

```
void QuadraMemb::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

**6.680.1.13 Tableau\_de\_Sig1()**

```
DdlElement & QuadraMemb::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.cc

**6.681 Référence de la classe Quartic****Fonctions membres publiques**

- void **Change** (int debug, bool iteratee)
- double **errors** (double a, double b, double c, double d, double rts[4], double rterr[4], int nrts)
- double **acos3** (double x)
- double **curoot** (double x)
- int **quadratic** (double b, double c, double rts[4])
- int **cubic** (double p, double q, double r, double v3[4])
- void **cubnewton** (double p, double q, double r, int n3, double v3[4])
- int **quartic** (double a, double b, double c, double d, double rts[4])
- int **descartes** (double a, double b, double c, double d, double rts[4])
- int **ferrari** (double a, double b, double c, double d, double rts[4])
- int **neumark** (double a, double b, double c, double d, double rts[4])
- int **yafracid** (double a, double b, double c, double d, double rts[4])
- int **chris** (double a, double b, double c, double d, double rts[4])

**Attributs protégés**

- double **d3o8**
- double **d3o256**
- double **doub0**
- double **doub1**

- double **doub2**
- double **doub3**
- double **doub4**
- double **doub5**
- double **doub6**
- double **doub8**
- double **doub9**
- double **doub12**
- double **doub16**
- double **doub24**
- double **doub27**
- double **doub64**
- double **doubmax**
- double **doubmin**
- double **doubtol**
- double **inv2**
- double **inv3**
- double **inv4**
- double **inv8**
- double **inv16**
- double **inv32**
- double **inv64**
- double **inv128**
- double **qrts** [4][3]
- double **rt3**
- double **rterc** [4]
- double **rterd** [4]
- double **rterf** [4]
- double **rtern** [4]
- double **rterq** [4]
- double **rtery** [4]
- double **worst3** [3]
- int **debug**
- bool **iterate**
- int **j3**
- int **n1**
- int **n2**
- int **n3**
- int **n4** [3]
- int **nqud** [NCASES]
- int **ncub** [NCASES]
- int **nchr** [NCASES]
- int **ndes** [NCASES]
- int **nfer** [NCASES]
- int **nneu** [NCASES]
- int **nyac** [NCASES]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/externe/Racine.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/externe/Racine.cc

## 6.682 Référence de la classe `quatre_string_un_entier`

cas de 4 string et un entier

```
#include <Basiques.h>
```

### Fonctions membres publiques

- **quatre\_string\_un\_entier** (const string &n1, const string &n2, const string &n3, const string &n4, const int &nn)
- **quatre\_string\_un\_entier** (const [quatre\\_string\\_un\\_entier](#) &de)
- **quatre\_string\_un\_entier & operator=** (const [quatre\\_string\\_un\\_entier](#) &de)
- **istream & LectXML\_quatre\_string\_un\_entier** (istream &ent)

- ostream & **EcritXML\_quatre\_string\_un\_entier** (ostream &sort)
- bool **operator==** (const quatre\_string\_un\_entier &a) const
- bool **operator!=** (const quatre\_string\_un\_entier &a) const
- void **SchemaXML\_quatre\_string\_un\_entier** (ofstream &sort, const Enum\_IO\_XML enu) const
- bool **operator>** (const quatre\_string\_un\_entier &a) const
- bool **operator>=** (const quatre\_string\_un\_entier &a) const
- bool **operator<** (const quatre\_string\_un\_entier &a) const
- bool **operator<=** (const quatre\_string\_un\_entier &a) const

### Attributs publics

- string **nom1**
- string **nom2**
- string **nom3**
- string **nom4**
- int **n**

### Attributs publics statiques

- static short int **impre\_schem\_XML** =0

### Amis

- istream & **operator>>** (istream &ent, quatre\_string\_un\_entier &de)
- ostream & **operator<<** (ostream &sort, const quatre\_string\_un\_entier &de)

#### 6.682.1 Description détaillée

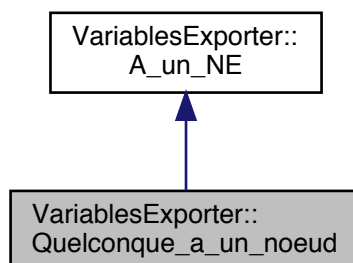
cas de 4 string et un entier

La documentation de cette classe a été générée à partir du fichier suivant :

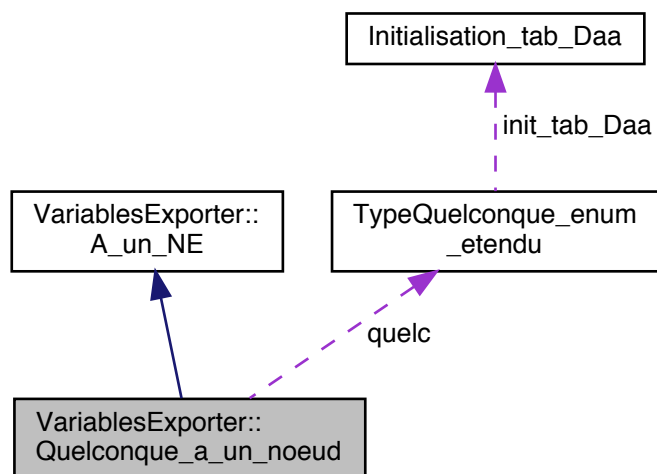
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.cc

## 6.683 Référence de la classe VariablesExporter::Quelconque\_a\_un\_noeud

Graphe d'héritage de VariablesExporter::Quelconque\_a\_un\_noeud:



Graphe de collaboration de VariablesExporter::Quelconque\_a\_un\_noeud:



## Fonctions membres publiques

- **Quelconque\_a\_un\_noeud** ([TypeQuelconque\\_enum\\_etendu](#) e, string ref\_noeud, string nom\_mail\_, string nom\_var\_, int num\_ord)
- **Quelconque\_a\_un\_noeud** (const [Quelconque\\_a\\_un\\_noeud](#) &a)
- **Quelconque\_a\_un\_noeud & operator=** (const [Quelconque\\_a\\_un\\_noeud](#) &a)
- bool **operator==** (const [Quelconque\\_a\\_un\\_noeud](#) &a) const
- bool **operator!=** (const [Quelconque\\_a\\_un\\_noeud](#) &a) const
- bool **operator<** (const [Quelconque\\_a\\_un\\_noeud](#) &a) const
- bool **operator<=** (const [Quelconque\\_a\\_un\\_noeud](#) &a) const
- bool **operator>** (const [Quelconque\\_a\\_un\\_noeud](#) &a) const
- bool **operator>=** (const [Quelconque\\_a\\_un\\_noeud](#) &a) const
- void **Affiche** ()
- const [TypeQuelconque\\_enum\\_etendu](#) & **Quelc\_const** () const
- [TypeQuelconque\\_enum\\_etendu](#) & **Quelc** ()
- const int & **Num\_ordre\_const** () const
- int & **Num\_ordre** ()

## Attributs protégés

- [TypeQuelconque\\_enum\\_etendu](#) **quelc**
- int **num\_ordre**

## Amis

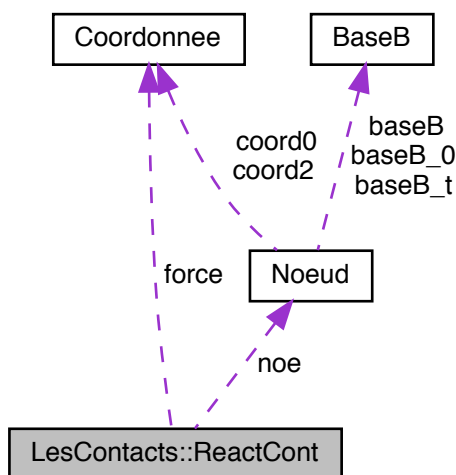
- istream & **operator>>** (istream &, [Quelconque\\_a\\_un\\_noeud](#) &)
- ostream & **operator<<** (ostream &, const [Quelconque\\_a\\_un\\_noeud](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

## 6.684 Référence de la classe LesContacts::ReactCont

Graphe de collaboration de LesContacts::ReactCont:



### Fonctions membres publiques

- `ReactCont` (`Noeud *no`, `const Coordonnee &forc`)
- `ReactCont` (`Noeud *no`, `const Coordonnee &forc`, `Tableau< Noeud * > tN`, `const Tableau< Coordonnee > &tFor`)
- `ReactCont` (`const ReactCont &a`)
- `ReactCont &operator=` (`const ReactCont &a`)
- `bool operator==` (`const ReactCont &a`)
- `bool operator!=` (`const ReactCont &a`)

### Attributs publics

- `Noeud * noe`
- `Coordonnee force`
- `Tableau< Noeud * > tabNoeud`
- `Tableau< Coordonnee > tabForce`

### Amis

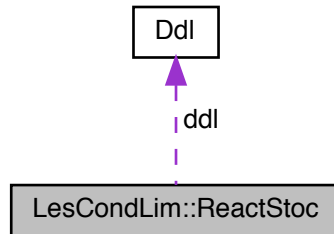
- `istream &operator>>` (`istream &`, `ReactCont &`)
- `ostream &operator<<` (`ostream &`, `const ReactCont &`)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/LesContacts.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/LesContacts.cc`

## 6.685 Référence de la classe LesCondLim::ReactStoc

Grappe de collaboration de LesCondLim::ReactStoc:



### Fonctions membres publiques

- bool **operator==** (const [ReactStoc](#) &a) const
- bool **operator!=** (const [ReactStoc](#) &a) const
- bool **operator<** (const [ReactStoc](#) &a) const
- bool **operator>** (const [ReactStoc](#) &a) const

### Attributs publics

- int **numMail**
- int **numNoeud**
- [Ddl](#) **ddl**
- int **casAss**

### Amis

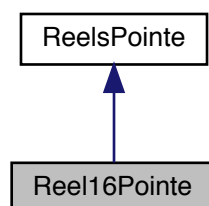
- `istream &` **operator>>** (`istream &`, [ReactStoc](#) &)
- `ostream &` **operator<<** (`ostream &`, const [ReactStoc](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesCondLim.h`

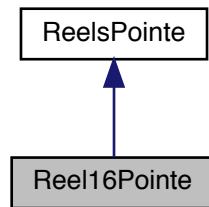
## 6.686 Référence de la classe Reel16Pointe

Grappe d'héritage de Reel16Pointe:





Graphe de collaboration de Reel16Pointe:



### Attributs publics

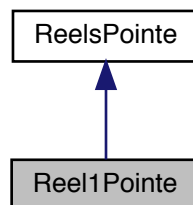
- listdouble16Iter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

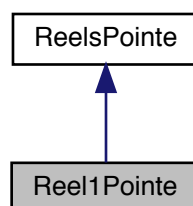
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.687 Référence de la classe Reel1Pointe

Graphe d'héritage de Reel1Pointe:



Graphe de collaboration de Reel1Pointe:



### Attributs publics

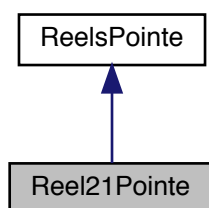
- listdouble1Iter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

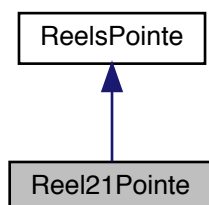
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.688 Référence de la classe Reel21Pointe

Graphe d'héritage de Reel21Pointe:



Graphe de collaboration de Reel21Pointe:



### Attributs publics

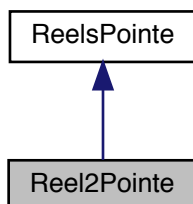
- listdouble21Iter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

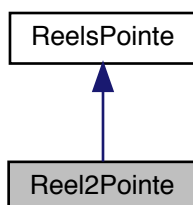
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.689 Référence de la classe Reel2Pointe

Graphe d'héritage de Reel2Pointe:



Graphe de collaboration de Reel2Pointe:



### Attributs publics

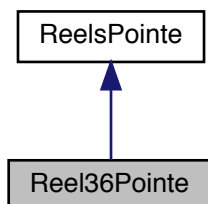
- listdouble2lter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

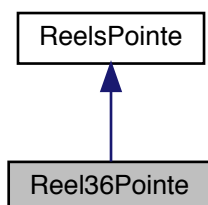
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

## 6.690 Référence de la classe Reel36Pointe

Graphe d'héritage de Reel36Pointe:



Graphe de collaboration de Reel36Pointe:



### Attributs publics

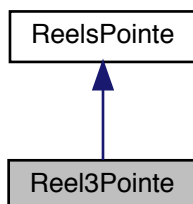
- listdouble36lter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

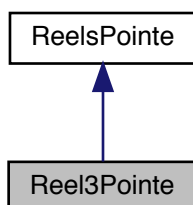
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

## 6.691 Référence de la classe Reel3Pointe

Graphe d'héritage de Reel3Pointe:



Graphe de collaboration de Reel3Pointe:



### Attributs publics

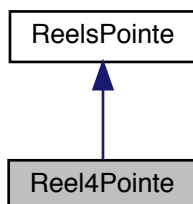
- listdouble3lter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

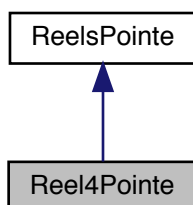
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

## 6.692 Référence de la classe Reel4Pointe

Graphe d'héritage de Reel4Pointe:



Graphe de collaboration de Reel4Pointe:



### Attributs publics

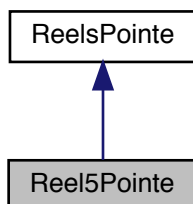
- listdouble4lter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

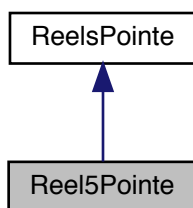
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

## 6.693 Référence de la classe Reel5Pointe

Graphe d'héritage de Reel5Pointe:



Graphe de collaboration de Reel5Pointe:



### Attributs publics

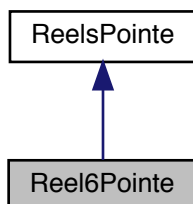
- listdouble5lter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

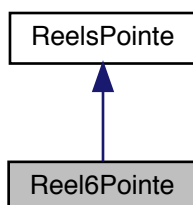
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

## 6.694 Référence de la classe Reel6Pointe

Graphe d'héritage de Reel6Pointe:



Graphe de collaboration de Reel6Pointe:



### Attributs publics

- listdouble6lter **ipointe**

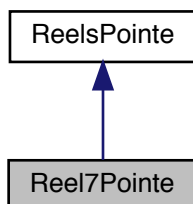
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

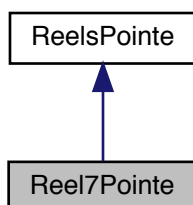


## 6.695 Référence de la classe Reel7Pointe

Graphe d'héritage de Reel7Pointe:



Graphe de collaboration de Reel7Pointe:



### Attributs publics

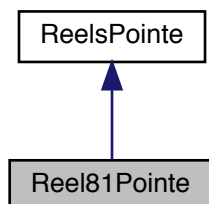
- listdouble7lter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

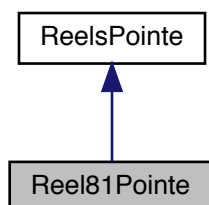
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

## 6.696 Référence de la classe Reel81Pointe

Graphe d'héritage de Reel81Pointe:



Graphe de collaboration de Reel81Pointe:



### Attributs publics

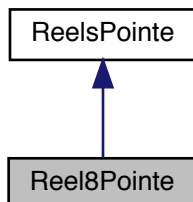
- listdouble81Iter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

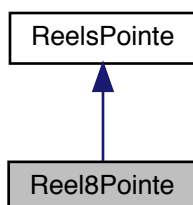
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

## 6.697 Référence de la classe Reel8Pointe

Graphe d'héritage de Reel8Pointe:



Graphe de collaboration de Reel8Pointe:



### Attributs publics

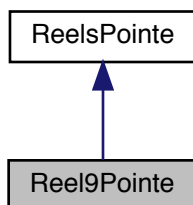
- listdouble8lter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

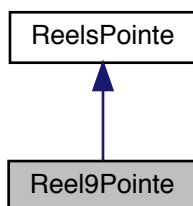
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PiTabRel.h

## 6.698 Référence de la classe Reel9Pointe

Graphe d'héritage de Reel9Pointe:



Graphe de collaboration de Reel9Pointe:



### Attributs publics

- listdouble9lter **ipointe**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.699 Référence de la classe Reels1

### Attributs publics

- double **donnees**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.700 Référence de la classe Reels16

### Attributs publics

- double **donnees** [16]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.701 Référence de la classe Reels2

### Attributs publics

- double **donnees** [2]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.702 Référence de la classe Reels21

### Attributs publics

- double **donnees** [21]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.703 Référence de la classe Reels3

### Attributs publics

- double **donnees** [3]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.704 Référence de la classe Reels36

### Attributs publics

- double **donnees** [36]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.705 Référence de la classe Reels4

### Attributs publics

- double **donnees** [4]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.706 Référence de la classe Reels5

### Attributs publics

- double **donnees** [5]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.707 Référence de la classe Reels6

### Attributs publics

- double **donnees** [6]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.708 Référence de la classe Reels7

### Attributs publics

- double **donnees** [7]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.709 Référence de la classe Reels8

### Attributs publics

- double **donnees** [8]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.710 Référence de la classe Reels81

### Attributs publics

- double **donnees** [81]

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.711 Référence de la classe Reels9

### Attributs publics

- double **donnees** [9]

La documentation de cette classe a été générée à partir du fichier suivant :

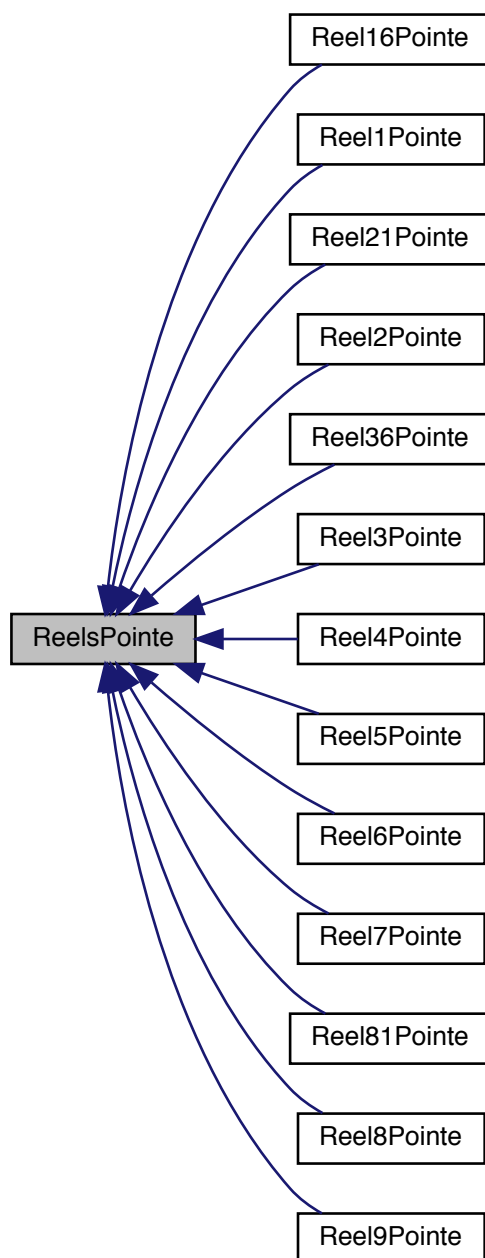
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.712 Référence de la classe ReelsPointe

[ReelsPointe](#) classe virtuelle de laquelle dérive les classe contenant un pointeur de réel.

```
#include <PtTabRel.h>
```

Graphe d'héritage de ReelsPointe:



### 6.712.1 Description détaillée

[ReelsPointe](#) classe virtuelle de laquelle dérive les classe contenant un pointeur de réel.

Auteur

Gérard Rio

## Version

1.0

## Date

23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

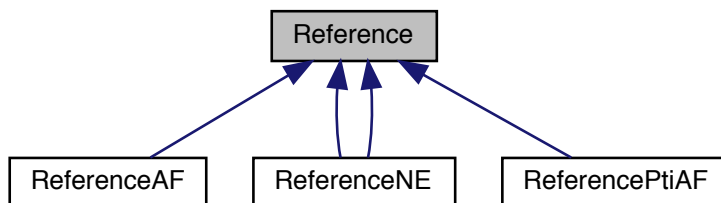
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/PtTabRel.h

## 6.713 Référence de la classe Reference

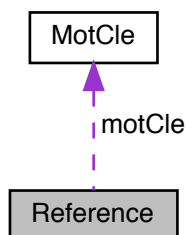
class [Reference](#): classe général virtuel des références. Les classes dérivées permettent de définir précisément les différentes références : de noeuds, d'élément, d'aretes, de faces etc.. Une instance de la classe s'identifie a partir d'un nom

```
#include <Reference.h>
```

Graphe d'héritage de Reference:



Graphe de collaboration de Reference:



### Fonctions membres publiques

- **Reference** (string nom="rien\_actuellement")
- **Reference** (int nbmaille, int indic)
- **Reference** (string nom, int nbmaille, int indic)
- **Reference** (const [Reference](#) &ref)
- virtual [Reference](#) \* **Nevez\_Ref\_copie** () const =0
- virtual [Reference](#) & **operator=** (const [Reference](#) &ref)



- string **Nom** () const
- int **Indic** () const
- int **Nbmaille** () const
- void **Change\_Nbmaille** (int nv\_nm)
- void **Change\_nom** (string nouveau\_nom)
- virtual void **Supprime\_doublons\_internes** ()=0
- virtual void **Affiche** () const
- virtual void **Affiche\_dans\_lis** (ofstream &sort) const =0
- virtual bool **LectureReference** (**UtilLecture** &entreePrinc)=0
- virtual void **Info\_commande\_Ref** (int nbMaxi, **UtilLecture** \*entreePrinc, int cas)=0
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)=0
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)=0

### Fonctions membres protégées

- void **Lect\_int\_base\_info** (ifstream &ent)
- void **Ecrit\_int\_base\_info** (ofstream &sort)

### Attributs protégés

- string **nom\_ref**
- int **nbmaille**
- int **indic**

### Attributs protégés statiques

- static **MotCle** **motCle**  
                   ----- *variables statiques* -----

#### 6.713.1 Description détaillée

class **Reference**: classe général virtuel des références. Les classes dérivées permettent de définir précisément les différentes références : de noeuds, d'élément, d'aretes, de faces etc.. Une instance de la classe s'identifie a partir d'un nom

##### Auteur

Gérard Rio

##### Version

1.0

##### Date

06/02/00

La documentation de cette classe a été générée à partir du fichier suivant :

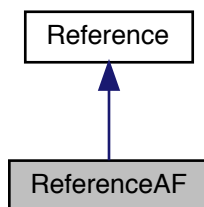
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/Reference.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/Reference\_static.cc

## 6.714 Référence de la classe ReferenceAF

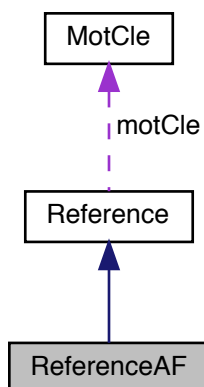
Def, stockage et manipulation des références pour les faces et les arêtes.

```
#include <ReferenceAF.h>
```

Graphe d'héritage de ReferenceAF:



Graphe de collaboration de ReferenceAF:



## Fonctions membres publiques

- **ReferenceAF** (string nom="rien\_actuellement")
- **ReferenceAF** (int nbmaille, int indic)
- **ReferenceAF** (const [Tableau](#)< int > &tabelem, const [Tableau](#)< int > &tab, int nbmaille, int indic, string nom="rien\_actuellement")
- **ReferenceAF** (const list< int > &list\_elem, const list< int > &list\_num, int nbmaille, int indic, string nom="rien\_actuellement")
- **ReferenceAF** (string nom, int nbmaille, int indic)
- **ReferenceAF** (const [ReferenceAF](#) &ref)
- [Reference](#) \* **Nevez\_Ref\_copie** () const
- [Reference](#) & **operator=** (const [Reference](#) &ref)
- const [Tableau](#)< int > & **Tab\_Elem** () const
- const [Tableau](#)< int > & **Tab\_FA** () const
- int **NumeroElem** (int i) const
- int **NumeroFA** (int i) const
- int **Taille** () const
- void **Change\_num\_AF\_dans\_ref** (int i, int nbe, int nbAF)
- void **Change\_tab\_num** (const [Tableau](#)< int > &tabele, const [Tableau](#)< int > &tab)
- bool **Contient\_AF** (int nbe, int nbAF) const

- virtual void [Supprime\\_doublons\\_internes](#) ()
- void [Affiche](#) () const
- void [Affiche\\_dans\\_lis](#) (ofstream &sort) const
- bool [LectureReference](#) (UtilLecture &entreePrinc)
- void [Info\\_commande\\_Ref](#) (int nbMaxi, UtilLecture \*entreePrinc, int cas)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)

### Attributs protégés

- [Tableau](#)< int > **tab\_Elem**
- [Tableau](#)< int > **tab\_FA**

### Membres hérités additionnels

#### 6.714.1 Description détaillée

Def, stockage et manipulation des références pour les faces et les arêtes.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

#### 6.714.2 Documentation des fonctions membres

##### 6.714.2.1 Affiche()

```
void ReferenceAF::Affiche ( ) const [virtual]
```

Réimplémentée à partir de [Reference](#).

##### 6.714.2.2 Affiche\_dans\_lis()

```
void ReferenceAF::Affiche_dans_lis (
    ofstream & sort ) const [virtual]
```

Implémente [Reference](#).

##### 6.714.2.3 Ecriture\_base\_info()

```
void ReferenceAF::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Reference](#).

##### 6.714.2.4 Info\_commande\_Ref()

```
void ReferenceAF::Info_commande_Ref (
    int nbMaxi,
    UtilLecture * entreePrinc,
    int cas ) [virtual]
```

Implémente [Reference](#).

### 6.714.2.5 Lecture\_base\_info()

```
void ReferenceAF::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Reference](#).

### 6.714.2.6 LectureReference()

```
bool ReferenceAF::LectureReference (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Reference](#).

### 6.714.2.7 Nevez\_Ref\_copie()

```
Reference * ReferenceAF::Nevez_Ref_copie ( ) const [inline], [virtual]
```

Implémente [Reference](#).

### 6.714.2.8 operator=()

```
Reference & ReferenceAF::operator= (
    const Reference & ref ) [virtual]
```

Réimplémentée à partir de [Reference](#).

### 6.714.2.9 Supprime\_doublons\_internes()

```
virtual void ReferenceAF::Supprime_doublons_internes ( ) [virtual]
```

Implémente [Reference](#).

La documentation de cette classe a été générée à partir du fichier suivant :

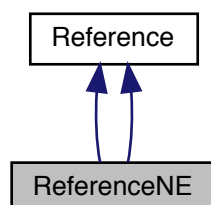
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferenceAF.h

## 6.715 Référence de la classe ReferenceNE

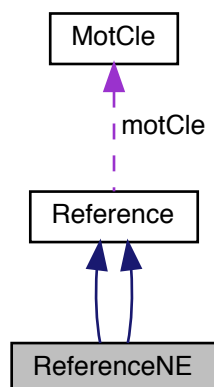
Def, stockage et manipulation des références pour les noeuds et les éléments.

```
#include <ReferenceNE.h>
```

Graphe d'héritage de ReferenceNE:



Graphe de collaboration de ReferenceNE:



## Fonctions membres publiques

- **ReferenceNE** (string nom="rien\_actuellement")
- **ReferenceNE** (const [Tableau](#)< int > &tab, int nbmaille, int indic, string nom="rien\_actuellement")
- **ReferenceNE** (string nom, int nbmaille, int indic)
- **ReferenceNE** (const [ReferenceNE](#) &ref)
- [Reference](#) & **operator=** (const [Reference](#) &ref)
- [Tableau](#)< int > & **Tab\_num** ()
- string **Nom** () const
- int **Indic** () const
- int **Nbmaille** () const
- int & **Numero** (int i)
- int **Numero** (int i) const
- int **Taille** () const
- void **Change\_nom** (string nouveau\_nom)
- void **Affiche** () const
- **ReferenceNE** (string nom="rien\_actuellement")
- **ReferenceNE** (int nbmaille, int indic)
- **ReferenceNE** (const [Tableau](#)< int > &tab, int nbmaille, int indic, string nom="rien\_actuellement")
- **ReferenceNE** (const list< int > &list\_entiers, int nbmaille, int indic, string nom="rien\_actuellement")
- **ReferenceNE** (string nom, int nbmaille, int indic)
- **ReferenceNE** (const [ReferenceNE](#) &ref)
- [Reference](#) \* **Nevez\_Ref\_copie** () const
- [Reference](#) & **operator=** (const [Reference](#) &ref)
- const [Tableau](#)< int > & **Tab\_num** () const
- int **Numero** (int i) const
- int **Taille** () const
- void **Change\_num\_dans\_ref** (int i, int nv\_num)
- void **Change\_tab\_num** (const [Tableau](#)< int > &tab)
- virtual void **Supprime\_doublons\_internes** ()
- void **Affiche** () const
- void **Affiche\_dans\_lis** (ofstream &sort) const
- bool **LectureReference** ([UtilLecture](#) &entreePrinc)
- void **Info\_commande\_Ref** (int nbMaxi, [UtilLecture](#) \*entreePrinc, int cas)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

## Attributs protégés

- string **nom\_ref**
- int **nbmaille**
- int **indic**
- [Tableau](#)< int > **tab\_num**

## Membres hérités additionnels

### 6.715.1 Description détaillée

Def, stockage et manipulation des références pour les noeuds et les éléments.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

06/02/00

### 6.715.2 Documentation des fonctions membres

#### 6.715.2.1 Affiche() [1/2]

```
void ReferenceNE::Affiche ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Reference](#).

#### 6.715.2.2 Affiche() [2/2]

```
void ReferenceNE::Affiche ( ) const [virtual]
```

Réimplémentée à partir de [Reference](#).

#### 6.715.2.3 Affiche\_dans\_lis()

```
void ReferenceNE::Affiche_dans_lis (
    ofstream & sort ) const [virtual]
```

Implémente [Reference](#).

#### 6.715.2.4 Ecriture\_base\_info()

```
void ReferenceNE::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Reference](#).

#### 6.715.2.5 Info\_commande\_Ref()

```
void ReferenceNE::Info_commande_Ref (
    int nbMaxi,
    UtilLecture * entreePrinc,
    int cas ) [virtual]
```

Implémente [Reference](#).

### 6.715.2.6 Lecture\_base\_info()

```
void ReferenceNE::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Reference](#).

### 6.715.2.7 LectureReference()

```
bool ReferenceNE::LectureReference (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Reference](#).

### 6.715.2.8 Nevez\_Ref\_copie()

```
Reference * ReferenceNE::Nevez_Ref_copie ( ) const [inline], [virtual]
```

Implémente [Reference](#).

### 6.715.2.9 operator=() [1/2]

```
Reference & ReferenceNE::operator= (
    const Reference & ref ) [inline], [virtual]
```

Réimplémentée à partir de [Reference](#).

### 6.715.2.10 operator=() [2/2]

```
Reference & ReferenceNE::operator= (
    const Reference & ref ) [virtual]
```

Réimplémentée à partir de [Reference](#).

### 6.715.2.11 Supprime\_doublons\_internes()

```
virtual void ReferenceNE::Supprime_doublons_internes ( ) [virtual]
```

Implémente [Reference](#).

La documentation de cette classe a été générée à partir du fichier suivant :

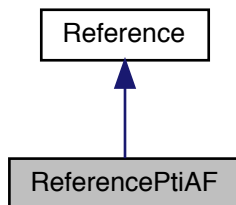
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferenceFA.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferenceNE.h

## 6.716 Référence de la classe ReferencePtiAF

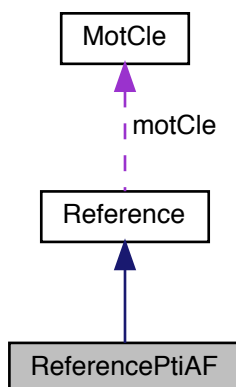
Def, stockage et manipulation des références pour les pti de faces et d'arêtes.

```
#include <ReferencePtiAF.h>
```

Grphe d'héritage de ReferencePtiAF:



Grphe de collaboration de ReferencePtiAF:



## Fonctions membres publiques

- **ReferencePtiAF** (string nom="rien\_actuellement")
- **ReferencePtiAF** (int nbmaille, int indic)
- **ReferencePtiAF** (const [Tableau](#)< int > &tabelem, const [Tableau](#)< int > &tabAF, const [Tableau](#)< int > &tabPti, int nbmaille, int indic, string nom="rien\_actuellement")
- **ReferencePtiAF** (const list< int > &list\_elem, const list< int > &list\_numAF, const list< int > &list\_num\_pti, int nbmaille, int indic, string nom="rien\_actuellement")
- **ReferencePtiAF** (string nom, int nbmaille, int indic)
- **ReferencePtiAF** (const [ReferencePtiAF](#) &ref)
- [Reference](#) \* **Nevez\_Ref\_copie** () const
- [Reference](#) & **operator=** (const [Reference](#) &ref)
- const [Tableau](#)< int > & **Tab\_Elem** () const
- const [Tableau](#)< int > & **Tab\_FA** () const
- const [Tableau](#)< int > & **Tab\_Pti** () const
- int **NumeroElem** (int i) const
- int **NumeroFA** (int i) const
- int **NumeroPti** (int i) const
- int **Taille** () const
- void **Change\_num\_PtiAF\_dans\_ref** (int i, int nbe, int nbAF, int nbPti)



- void **Change\_tab\_num** (const [Tableau](#)< int > &tabele, const [Tableau](#)< int > &tabAF, const [Tableau](#)< int > &tabPti)
- bool **Contient\_PtiAF** (int nbe, int nbAF, int nbPti) const
- virtual void **Supprime\_doublons\_internes** ()
- void **Affiche** () const
- void **Affiche\_dans\_lis** (ofstream &sort) const
- bool **LectureReference** ([UtilLecture](#) &entreePrinc)
- void **Info\_commande\_Ref** (int nbMaxi, [UtilLecture](#) \*entreePrinc, int cas)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

### Attributs protégés

- [Tableau](#)< int > **tab\_Elem**
- [Tableau](#)< int > **tab\_FA**
- [Tableau](#)< int > **tab\_pti**

### Membres hérités additionnels

#### 6.716.1 Description détaillée

Def, stockage et manipulation des références pour les pti de faces et d'arêtes.

##### Auteur

Gérard Rio

##### Version

1.0

##### Date

22/04/2020

#### 6.716.2 Documentation des fonctions membres

##### 6.716.2.1 Affiche()

```
void ReferencePtiAF::Affiche ( ) const [virtual]
```

Réimplémentée à partir de [Reference](#).

##### 6.716.2.2 Affiche\_dans\_lis()

```
void ReferencePtiAF::Affiche_dans_lis (
    ofstream & sort ) const [virtual]
```

Implémente [Reference](#).

##### 6.716.2.3 Ecriture\_base\_info()

```
void ReferencePtiAF::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Reference](#).

#### 6.716.2.4 Info\_commande\_Ref()

```
void ReferencePtiAF::Info_commande_Ref (
    int nbMaxi,
    UtilLecture * entreePrinc,
    int cas ) [virtual]
```

Implémente [Reference](#).

#### 6.716.2.5 Lecture\_base\_info()

```
void ReferencePtiAF::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Reference](#).

#### 6.716.2.6 LectureReference()

```
bool ReferencePtiAF::LectureReference (
    UtilLecture & entreePrinc ) [virtual]
```

Implémente [Reference](#).

#### 6.716.2.7 Nevez\_Ref\_copie()

```
Reference * ReferencePtiAF::Nevez_Ref_copie ( ) const [inline], [virtual]
```

Implémente [Reference](#).

#### 6.716.2.8 operator=()

```
Reference & ReferencePtiAF::operator= (
    const Reference & ref ) [virtual]
```

Réimplémentée à partir de [Reference](#).

#### 6.716.2.9 Supprime\_doublons\_internes()

```
virtual void ReferencePtiAF::Supprime_doublons_internes ( ) [virtual]
```

Implémente [Reference](#).

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/References/ReferencePtiAF.h

## 6.717 Référence de la classe LesLoisDeComp::RefLoi

### Fonctions membres publiques

- [RefLoi](#) (string \*nom, string &s1, string &s2)
- [RefLoi](#) (const [RefLoi](#) &a)
- [RefLoi](#) & [operator=](#) (const [RefLoi](#) &a)
- void [Change\\_nom\\_maillage](#) (const string &nouveau)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas) const

### Attributs publics

- string \* [nom\\_maillage](#)
- string [st1](#)
- string [st2](#)

## Amis

- `istream & operator>>` (`istream &`, [LesLoisDeComp::RefLoi](#) &)
- `ostream & operator<<` (`ostream &`, const [LesLoisDeComp::RefLoi](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LesLoisDeComp.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/LesLoisDeComp.cc`

## 6.718 Référence de la classe Resultats

### Fonctions membres publiques

- **Resultats** ([UtilLecture](#) \*entreePrinc)
- void **Lecture** ([LesReferences](#) \*lesRef)
- void **Affiche** () const
- bool **Copie** ()
- void **Info\_commande\_Resultats** ()
- void **SortieInfo** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMail, [LesReferences](#) \*lesRef, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lescontacts)
- [LesValVecPropres](#) & **VeaPropre** ()
- void **Lect\_result\_base\_info** (`ifstream &ent`, const int cas)
- void **Ecri\_result\_base\_info** (`ofstream &sort`, const int cas)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Resultats.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Resultats.cc`

## 6.719 Référence de la classe Rgb

### Fonctions membres publiques

- **Rgb** (float, float, float)

## Amis

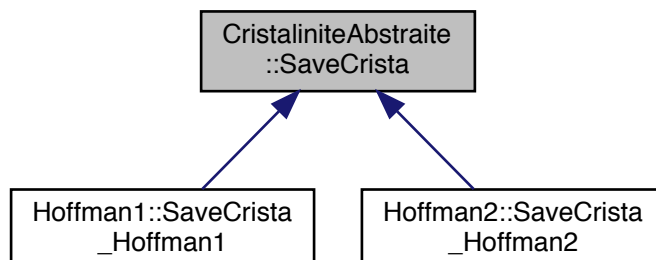
- `istream & operator>>` (`istream &i`, [Rgb](#) &c)
- `ostream & operator<<` (`ostream &o`, const [Rgb](#) &c)
- int **operator!=** ([Rgb](#) c1, [Rgb](#) c2)
- [Rgb](#) **operator+** ([Rgb](#) c1, [Rgb](#) c2)
- [Rgb](#) **operator\*** (float f, [Rgb](#) c)
- [Rgb](#) **hexaToRgb** (const char \*)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext_visu/rgb.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext_visu/rgb.cc`

## 6.720 Référence de la classe CristaliniteAbstraite::SaveCrista

Graphe d'héritage de CristaliniteAbstraite::SaveCrista:



### Fonctions membres publiques

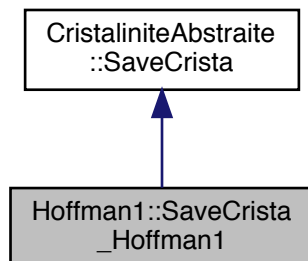
- virtual [SaveCrista](#) \* **Nevez\_SaveCrista** () const =0
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)=0
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)=0
- virtual void **TdtversT** ()=0
- virtual void **TversTdt** ()=0
- virtual void **Affiche** ()=0
- virtual void **Affiche** (ofstream &sort)=0
- virtual [SaveCrista](#) & **operator=** (const [SaveCrista](#) &a)=0

La documentation de cette classe a été générée à partir du fichier suivant :

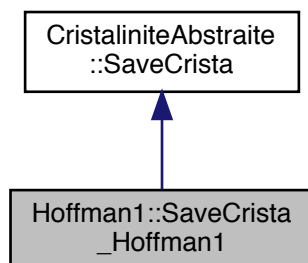
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/CristaliniteAbstraite.h

## 6.721 Référence de la classe Hoffman1::SaveCrista\_Hoffman1

Graphe d'héritage de Hoffman1::SaveCrista\_Hoffman1:



Grappe de collaboration de Hoffman1::SaveCrista\_Hoffman1:



## Fonctions membres publiques

- `SaveCrista_Hoffman1` (const double &I\_Kcinetique\_t, const double &I\_Kcinetique)
- `SaveCrista_Hoffman1` (const [SaveCrista\\_Hoffman1](#) &sav)
- `SaveCrista * Nevez_SaveCrista` () const
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `Affiche` ()
- void `Affiche` (ofstream &sort)
- void `TdtversT` ()
- void `TversTdt` ()
- `SaveCrista & operator=` (const [SaveCrista](#) &a)

## Attributs publics

- double `I_Kcinetique_t`
- double `I_Kcinetique`

### 6.721.1 Documentation des fonctions membres

#### 6.721.1.1 Affiche() [1/2]

```
void Hoffman1::SaveCrista_Hoffman1::Affiche ( ) [virtual]
Implémente CristaliniteAbstraite::SaveCrista.
```

#### 6.721.1.2 Affiche() [2/2]

```
void Hoffman1::SaveCrista_Hoffman1::Affiche (
    ofstream & sort ) [virtual]
Implémente CristaliniteAbstraite::SaveCrista.
```

#### 6.721.1.3 Ecriture\_base\_info()

```
void Hoffman1::SaveCrista_Hoffman1::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
Implémente CristaliniteAbstraite::SaveCrista.
```

**6.721.1.4 Lecture\_base\_info()**

```
void Hoffman1::SaveCrista_Hoffman1::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.721.1.5 Nevez\_SaveCrista()**

```
SaveCrista * Hoffman1::SaveCrista_Hoffman1::Nevez_SaveCrista ( ) const [inline], [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.721.1.6 operator=()**

```
CristaliniteAbstraite::SaveCrista & Hoffman1::SaveCrista_Hoffman1::operator= (
    const SaveCrista & a ) [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.721.1.7 TdtversT()**

```
void Hoffman1::SaveCrista_Hoffman1::TdtversT ( ) [inline], [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.721.1.8 TversTdt()**

```
void Hoffman1::SaveCrista_Hoffman1::TversTdt ( ) [inline], [virtual]
```

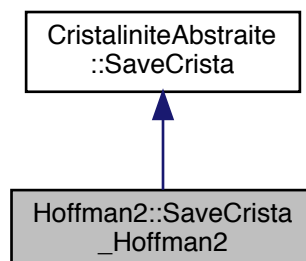
Implémente [CristaliniteAbstraite::SaveCrista](#).

La documentation de cette classe a été générée à partir du fichier suivant :

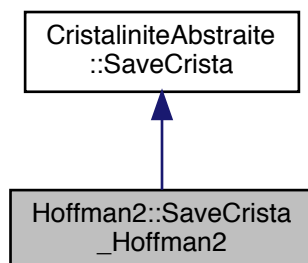
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Hoffman1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Hoffman1.cc

**6.722 Référence de la classe Hoffman2::SaveCrista\_Hoffman2**

Graphe d'héritage de Hoffman2::SaveCrista\_Hoffman2:



Graph de collaboration de Hoffman2::SaveCrista\_Hoffman2:



## Fonctions membres publiques

- `SaveCrista_Hoffman2` (const double &I\_Kcinetique\_t, const double &I\_Kcinetique)
- `SaveCrista_Hoffman2` (const [SaveCrista\\_Hoffman2](#) &sav)
- `SaveCrista * Nevez_SaveCrista` () const
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `Affiche` ()
- void `Affiche` (ofstream &sort)
- void `TdtversT` ()
- void `TversTdt` ()
- `SaveCrista & operator=` (const [SaveCrista](#) &a)

## Attributs publics

- double `I_Kcinetique_t`
- double `I_Kcinetique`

### 6.722.1 Documentation des fonctions membres

#### 6.722.1.1 Affiche() [1/2]

```
void Hoffman2::SaveCrista_Hoffman2::Affiche ( ) [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

#### 6.722.1.2 Affiche() [2/2]

```
void Hoffman2::SaveCrista_Hoffman2::Affiche (
    ofstream & sort ) [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

#### 6.722.1.3 Ecriture\_base\_info()

```
void Hoffman2::SaveCrista_Hoffman2::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.722.1.4 Lecture\_base\_info()**

```
void Hoffman2::SaveCrista_Hoffman2::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.722.1.5 Nevez\_SaveCrista()**

```
SaveCrista * Hoffman2::SaveCrista_Hoffman2::Nevez_SaveCrista ( ) const [inline], [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.722.1.6 operator=()**

```
CristaliniteAbstraite::SaveCrista & Hoffman2::SaveCrista_Hoffman2::operator= (
    const SaveCrista & a ) [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.722.1.7 TdtversT()**

```
void Hoffman2::SaveCrista_Hoffman2::TdtversT ( ) [inline], [virtual]
```

Implémente [CristaliniteAbstraite::SaveCrista](#).

**6.722.1.8 TversTdt()**

```
void Hoffman2::SaveCrista_Hoffman2::TversTdt ( ) [inline], [virtual]
```

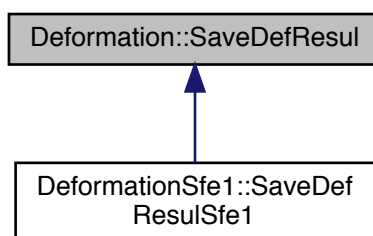
Implémente [CristaliniteAbstraite::SaveCrista](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Hoffman2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Taux\_crista/Hoffman2.cc

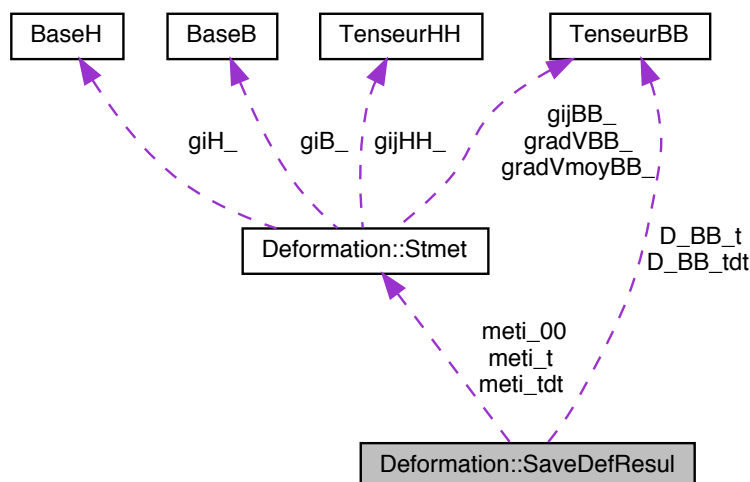
**6.723 Référence de la classe Deformation::SaveDefResul**

Graphe d'héritage de Deformation::SaveDefResul:





Graph de collaboration de Deformation::SaveDefResul:



## Fonctions membres publiques

- **SaveDefResul** (const [Met\\_abstraite](#) &meta)
- **SaveDefResul** (const [SaveDefResul](#) &a)
- virtual **SaveDefResul & operator=** (const [SaveDefResul](#) &a)
- virtual void **Affiche** ()
- virtual void **Affiche** (ofstream &sort)
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- virtual void **TdtversT** ()
- virtual void **TversTdt** ()
- virtual void **MiseAJourGrandeurs\_a\_0** (const [Met\\_abstraite](#) \*metrique)
- virtual void **MiseAJourGrandeurs\_a\_tdt** (const [Met\\_abstraite](#) \*metrique, const [TenseurBB](#) &Deps\_BB)
- const [Deformation::Stmet](#) & **Meti\_00** () const
- const [Deformation::Stmet](#) & **Meti\_t** () const
- const [Deformation::Stmet](#) & **Meti\_tdt** () const
- [Enum\\_type\\_stocke\\_deformation](#) **Type\_stocke** () const

## Attributs protégés

- [Deformation::Stmet](#) **meti\_00**
- [Deformation::Stmet](#) **meti\_t**
- [Deformation::Stmet](#) **meti\_tdt**
- [TenseurBB](#) \* **D\_BB\_t**
- [TenseurBB](#) \* **D\_BB\_tdt**
- [Enum\\_type\\_stocke\\_deformation](#) **enu\_type**

## Amis

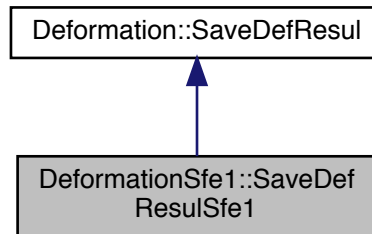
- class **Deformation**

La documentation de cette classe a été générée à partir du fichier suivant :

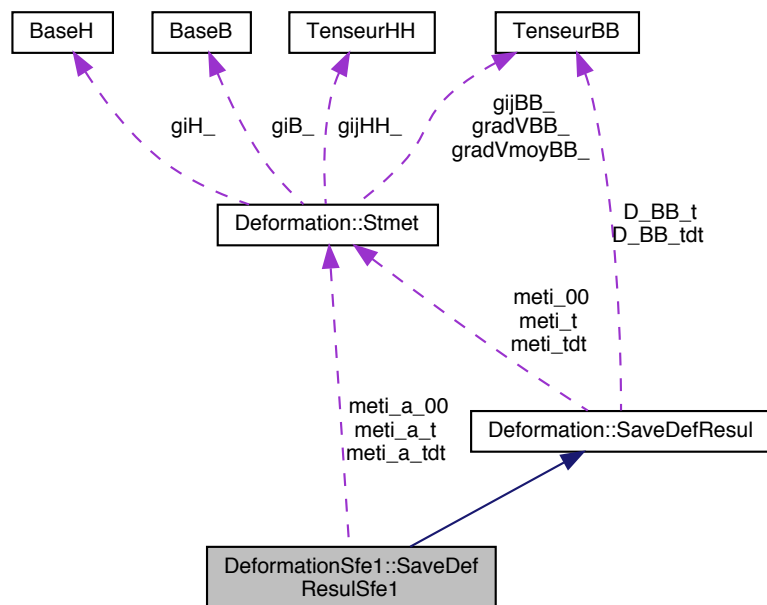
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation.↵  
h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation↵  
\_stockage.cc

## 6.724 Référence de la classe DeformationSfe1::SaveDefResulSfe1

Grphe d'héritage de DeformationSfe1::SaveDefResulSfe1:



Grphe de collaboration de DeformationSfe1::SaveDefResulSfe1:



### Fonctions membres publiques

- **SaveDefResulSfe1** (const [Met\\_abstraite](#) &meta)
- **SaveDefResulSfe1** (const [SaveDefResulSfe1](#) &a)
- virtual [SaveDefResul](#) & **operator=** (const [SaveDefResul](#) &a)
- virtual void [Affiche](#) ()
- virtual void [Affiche](#) (ofstream &sort)
- virtual void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- virtual void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- virtual void [TdtversT](#) ()
- virtual void [TversTdt](#) ()

- virtual void `MiseAJourGrandeurs_a_0` (const `Met_abstraite` \*metrique)
- virtual void `MiseAJourGrandeurs_a_tdt` (const `Met_abstraite` \*metrique, const `TenseurBB` &`Depes_BB`)
- const `Deformation::Stmet` & `Meti_a_00` () const
- const `Deformation::Stmet` & `Meti_a_t` () const
- const `Deformation::Stmet` & `Meti_a_tdt` () const

### Attributs protégés

- `Deformation::Stmet` `meti_a_00`
- `Deformation::Stmet` `meti_a_t`
- `Deformation::Stmet` `meti_a_tdt`

### Amis

- class `Deformation`
- class `DeformationSfe1`

## 6.724.1 Documentation des fonctions membres

### 6.724.1.1 `Affiche()` [1/2]

void `DeformationSfe1::SaveDefResulSfe1::Affiche` ( ) [virtual]  
Réimplémentée à partir de `Deformation::SaveDefResul`.

### 6.724.1.2 `Affiche()` [2/2]

void `DeformationSfe1::SaveDefResulSfe1::Affiche` (  
    ofstream & *sort* ) [virtual]  
Réimplémentée à partir de `Deformation::SaveDefResul`.

### 6.724.1.3 `Ecriture_base_info()`

void `DeformationSfe1::SaveDefResulSfe1::Ecriture_base_info` (  
    ofstream & *sort*,  
    const int *cas* ) [virtual]  
Réimplémentée à partir de `Deformation::SaveDefResul`.

### 6.724.1.4 `Lecture_base_info()`

void `DeformationSfe1::SaveDefResulSfe1::Lecture_base_info` (  
    ifstream & *ent*,  
    const int *cas* ) [virtual]  
Réimplémentée à partir de `Deformation::SaveDefResul`.

### 6.724.1.5 `MiseAJourGrandeurs_a_0()`

void `DeformationSfe1::SaveDefResulSfe1::MiseAJourGrandeurs_a_0` (  
    const `Met_abstraite` \* *metrique* ) [virtual]  
Réimplémentée à partir de `Deformation::SaveDefResul`.

**6.724.1.6 MiseAJourGrandeurs\_a\_tdt()**

```
void DeformationSfel::SaveDefResulSfel::MiseAJourGrandeurs_a_tdt (
    const Met_abstraite * metrique,
    const TenseurBB & Deps_BB ) [virtual]
```

Réimplémentée à partir de [Deformation::SaveDefResul](#).

**6.724.1.7 operator=()**

```
Deformation::SaveDefResul & DeformationSfel::SaveDefResulSfel::operator= (
    const SaveDefResul & a ) [virtual]
```

Réimplémentée à partir de [Deformation::SaveDefResul](#).

**6.724.1.8 TdtversT()**

```
virtual void DeformationSfel::SaveDefResulSfel::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Deformation::SaveDefResul](#).

**6.724.1.9 TversTdt()**

```
virtual void DeformationSfel::SaveDefResulSfel::TversTdt ( ) [inline], [virtual]
```

Réimplémentée à partir de [Deformation::SaveDefResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/DeformationSfe1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation↵  
Sfe1\_stockage.cc

**6.725 Référence de la classe CompFrotAbstraite::SaveResul****Fonctions membres publiques**

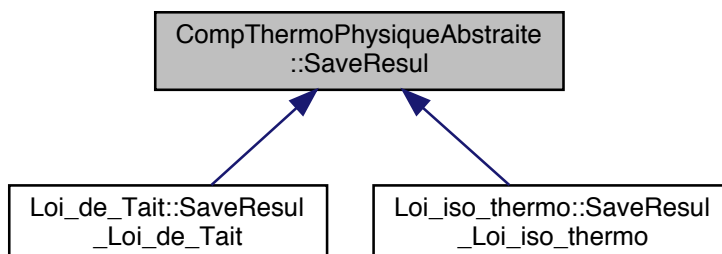
- virtual [SaveResul](#) \* **Nevez\_SaveResul** () const =0
- virtual [SaveResul](#) & **operator=** (const [SaveResul](#) &a)=0
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)=0
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)=0
- virtual void **TdtversT** ()
- virtual void **TversTdt** ()
- virtual void **ChBase\_des\_grandeurs** (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)=0

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/CompFrotAbstraite.h

## 6.726 Référence de la classe CompThermoPhysiqueAbstraite::SaveResul

Graphe d'héritage de CompThermoPhysiqueAbstraite::SaveResul:



### Fonctions membres publiques

- virtual [SaveResul](#) \* **Nevez\_SaveResul** () const =0
- virtual [SaveResul](#) & **operator=** (const [SaveResul](#) &)=0
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)=0
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)=0
- virtual void **TdtversT** ()=0
- virtual void **TversTdt** ()=0
- virtual void **ChBase\_des\_grandeurs** (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)=0

### Attributs publics

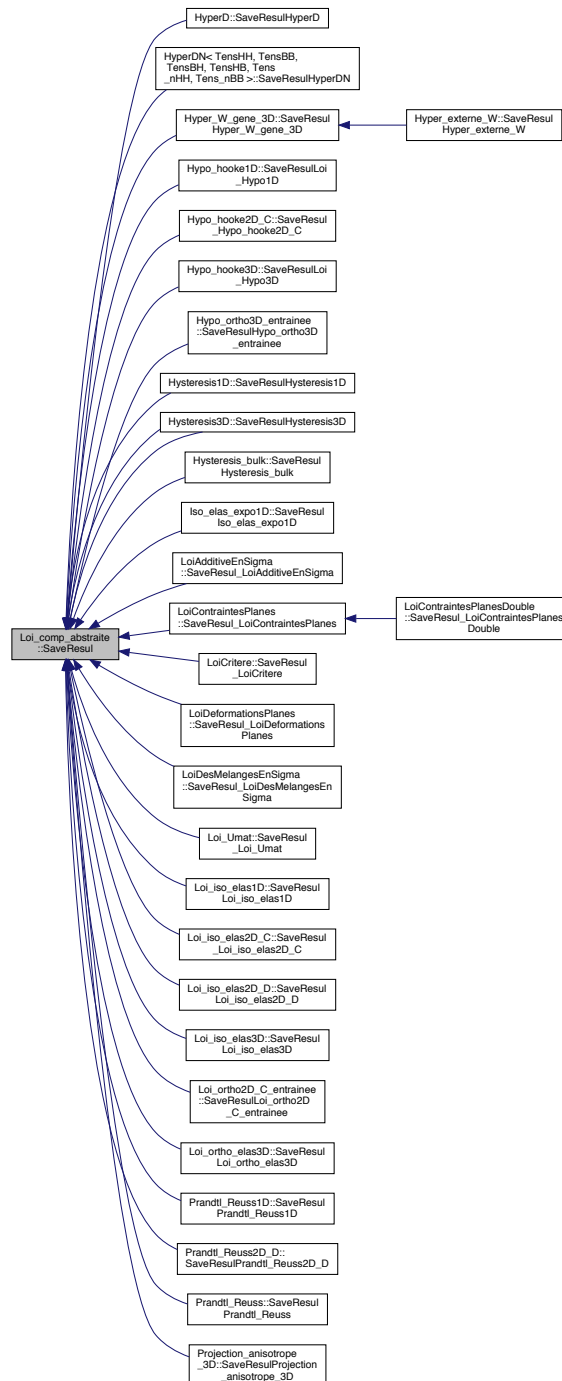
- double **temperature\_0**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/CompThermoPhysiqueAbstraite.h

## 6.727 Référence de la classe Loi\_comp\_abstraite::SaveResul

Graphe d'héritage de Loi\_comp\_abstraite::SaveResul:



### Fonctions membres publiques

- virtual [SaveResul](#) \* **Nevez\_SaveResul** () const =0
- virtual [SaveResul](#) & **operator=** (const [SaveResul](#) &)=0
- virtual void **Lecture\_base\_info** (ifstream &ent, const int cas)=0
- virtual void **Ecriture\_base\_info** (ofstream &sort, const int cas)=0
- virtual void **TdtversT** ()

- virtual void `TversTdt` ()
- virtual void `Affiche` () const =0
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)=0
- virtual `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau`< `Coordonnee` > &tab\_↵  
coor, const `Loi_comp_abstraite` \*loi)=0
- virtual int `TestCompleT` () const
- virtual double `Deformation_plastique` ()
- virtual const map< `EnumTypeQuelconque`, `TypeQuelconque`, std::less< `EnumTypeQuelconque` > > \*  
`Map_type_quelconque` () const

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Loi\_comp\_abstraite.h

## 6.728 Référence de la classe `Loi_comp_abstraite::SaveResul_C`

### Fonctions membres publiques

- `SaveResul_C` (const `SaveResul_C` &sav)
- virtual `SaveResul_C * Nevez_SaveResul_C` ()
- virtual void `Lecture_base_info` (ifstream &ent, const int)
- virtual void `Ecriture_base_info` (ofstream &sort, const int)
- virtual void `TdtversT` ()
- virtual void `TversTdt` ()

### Attributs publics

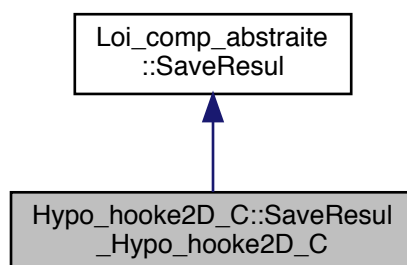
- bool `actif`
- bool `actif_t`

La documentation de cette classe a été générée à partir du fichier suivant :

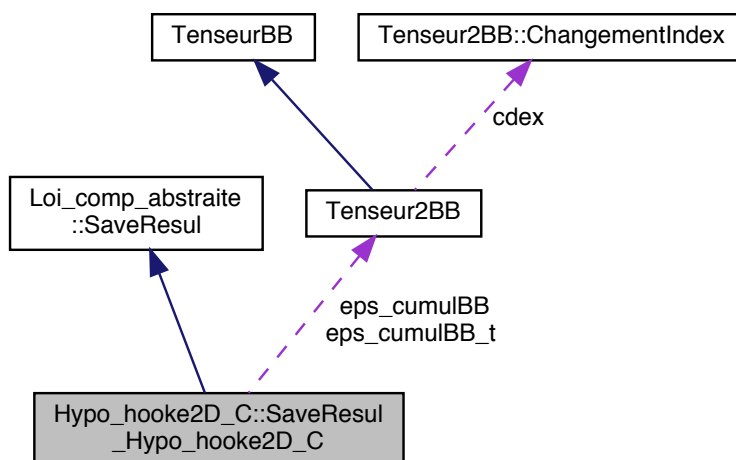
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Loi\_comp\_abstraite.h

## 6.729 Référence de la classe `Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C`

Grappe d'héritage de `Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C`:



Grphe de collaboration de `Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C`:



## Fonctions membres publiques

- `SaveResul_Hypo_hooke2D_C` (`SaveResul *l_des_SaveResul`)
- `SaveResul_Hypo_hooke2D_C` (const `SaveResul_Hypo_hooke2D_C &sav`)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul &a`)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine &beta`, const `Mat_pleine &gamma`)
- `SaveResul * Complete_SaveResul` (const `BlocGen &bloc`, const `Tableau< Coordonnee > &tab_coor`, const `Loi_comp_abstraite *loi`)
- double `Deformation_plastique` ()

## Attributs publics

- double `Kc`
- double `Kc_t`
- double `mu`
- double `mu_t`
- double `eps33`
- double `eps33_t`
- `Tenseur2BB eps_cumulBB`
- `Tenseur2BB eps_cumulBB_t`

## 6.729.1 Documentation des fonctions membres

### 6.729.1.1 Affiche()

void `Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::Affiche` () const [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.



### 6.729.1.2 ChBase\_des\_grandeurs()

```
void Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.729.1.3 Complete\_SaveResul()

```
SaveResul * Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.729.1.4 Deformation\_plastique()

```
double Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::Deformation_plastique ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.729.1.5 Ecriture\_base\_info()

```
void Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.729.1.6 Lecture\_base\_info()

```
void Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.729.1.7 Nevez\_SaveResul()

```
SaveResul * Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.729.1.8 operator=()

```
virtual SaveResul & Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::operator= (
    const SaveResul & a ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.729.1.9 TdtversT()

```
void Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.729.1.10 TversTdt()

```
void Hypo_hooke2D_C::SaveResul_Hypo_hooke2D_C::TversTdt ( ) [inline], [virtual]
```

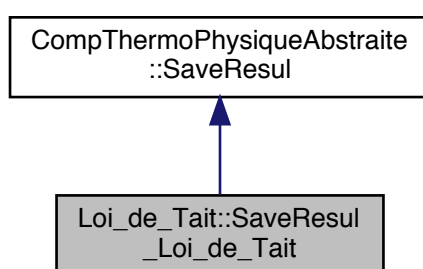
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

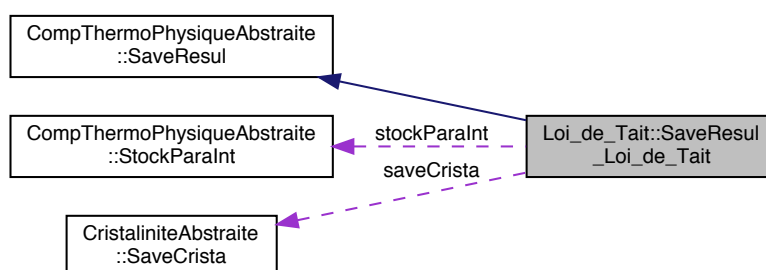
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke2D\_C.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke2D\_C.cc

## 6.730 Référence de la classe Loi\_de\_Tait::SaveResul\_Loi\_de\_Tait

Graphe d'héritage de Loi\_de\_Tait::SaveResul\_Loi\_de\_Tait:



Graphe de collaboration de Loi\_de\_Tait::SaveResul\_Loi\_de\_Tait:



## Fonctions membres publiques

- **SaveResul\_Loi\_de\_Tait** ([CompThermoPhysiqueAbstraite::StockParalnt](#) \*stock, [CristaliniteAbstraite::SaveCrista](#) \*sCrista)
- **SaveResul\_Loi\_de\_Tait** (const [SaveResul\\_Loi\\_de\\_Tait](#) &sav)
- [SaveResul](#) \* **Nevez\_SaveResul** () const
- virtual [SaveResul](#) & **operator=** (const [SaveResul](#) &a)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **Affiche** ()
- virtual void **ChBase\_des\_grandeurs** (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- [SaveResul](#) \* **Complete\_SaveResul** (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [CompThermoPhysiqueAbstraite](#) \*loi)

- void **Affiche** (ofstream &sort)
- void **TdtversT** ()
- void **TversTdt** ()

### Attributs publics

- `CompThermoPhysiqueAbstraite::StockParalnt` \* **stockParalnt**
- `CristaliniteAbstraite::SaveCrista` \* **saveCrista**

## 6.730.1 Documentation des fonctions membres

### 6.730.1.1 ChBase\_des\_grandeurs()

```
virtual void Loi_de_Tait::SaveResul_Loi_de_Tait::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.730.1.2 Ecriture\_base\_info()

```
void Loi_de_Tait::SaveResul_Loi_de_Tait::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.730.1.3 Lecture\_base\_info()

```
void Loi_de_Tait::SaveResul_Loi_de_Tait::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.730.1.4 Nevez\_SaveResul()

```
SaveResul * Loi_de_Tait::SaveResul_Loi_de_Tait::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.730.1.5 operator=()

```
CompThermoPhysiqueAbstraite::SaveResul & Loi_de_Tait::SaveResul_Loi_de_Tait::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.730.1.6 TdtversT()

```
void Loi_de_Tait::SaveResul_Loi_de_Tait::TdtversT ( ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.730.1.7 TversTdt()

```
void Loi_de_Tait::SaveResul_Loi_de_Tait::TversTdt ( ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

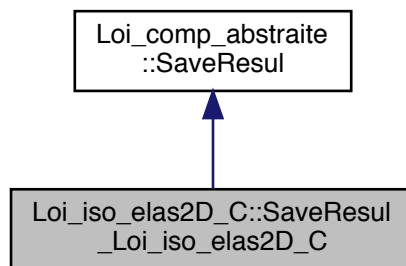
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi\_de\_Tait.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi\_de\_Tait.cc

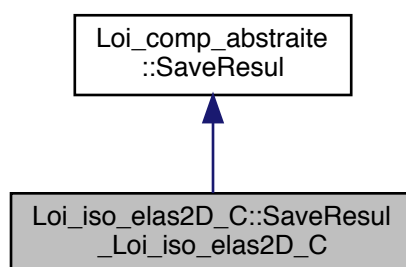
## 6.731 Référence de la classe

### Loi\_iso\_elas2D\_C::SaveResul\_Loi\_iso\_elas2D\_C

Graphe d'héritage de Loi\_iso\_elas2D\_C::SaveResul\_Loi\_iso\_elas2D\_C:



Graphe de collaboration de Loi\_iso\_elas2D\_C::SaveResul\_Loi\_iso\_elas2D\_C:



## Fonctions membres publiques

- [SaveResul\\_Loi\\_iso\\_elas2D\\_C](#) ([SaveResul](#) \*l\_des\_SaveResul)
- [SaveResul\\_Loi\\_iso\\_elas2D\\_C](#) (const [SaveResul\\_Loi\\_iso\\_elas2D\\_C](#) &sav)
- [SaveResul](#) \* [Nevez\\_SaveResul](#) () const
- virtual [SaveResul](#) & [operator=](#) (const [SaveResul](#) &a)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [TdtversT](#) ()

- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau`< `Coordonnee` > &tab\_coor, const `Loi_comp_abstraite` \*loi)
- double `Deformation_plastique` ()
- const map< `EnumTypeQuelconque`, `TypeQuelconque`, std::less< `EnumTypeQuelconque` > > \* `Map_type_quelconque` () const

## Attributs publics

- double `E`
- double `E_t`
- double `nu`
- double `nu_t`
- double `eps33`
- double `eps33_t`
- map< `EnumTypeQuelconque`, `TypeQuelconque`, std::less< `EnumTypeQuelconque` > > `map_type_← quelconque`

## 6.731.1 Documentation des fonctions membres

### 6.731.1.1 `Affiche()`

void `Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::Affiche` () const [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

### 6.731.1.2 `ChBase_des_grandeurs()`

virtual void `Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::ChBase_des_grandeurs` (  
     const `Mat_pleine` & *beta*,  
     const `Mat_pleine` & *gamma* ) [inline], [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

### 6.731.1.3 `Complete_SaveResul()`

`SaveResul * Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::Complete_SaveResul` (  
     const `BlocGen` & *bloc*,  
     const `Tableau`< `Coordonnee` > & *tab\_coor*,  
     const `Loi_comp_abstraite` \* *loi* ) [inline], [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

### 6.731.1.4 `Deformation_plastique()`

double `Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::Deformation_plastique` () [virtual]  
 Réimplémentée à partir de `Loi_comp_abstraite::SaveResul`.

### 6.731.1.5 `Ecriture_base_info()`

void `Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::Ecriture_base_info` (  
     ofstream & *sort*,  
     const int *cas* ) [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

### 6.731.1.6 Lecture\_base\_info()

```
void Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.731.1.7 Map\_type\_quelconque()

```
const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * Loi_iso_↔
elas2D_C::SaveResul_Loi_iso_elas2D_C::Map_type_quelconque ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.731.1.8 Nevez\_SaveResul()

```
SaveResul * Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.731.1.9 operator=()

```
virtual SaveResul & Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::operator= (
    const SaveResul & a ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.731.1.10 TdtversT()

```
void Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.731.1.11 TversTdt()

```
void Loi_iso_elas2D_C::SaveResul_Loi_iso_elas2D_C::TversTdt ( ) [inline], [virtual]
```

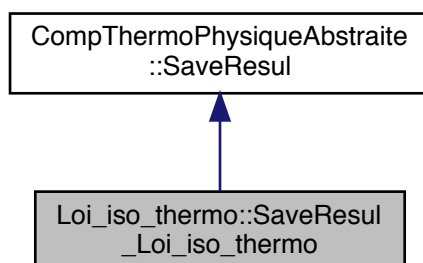
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

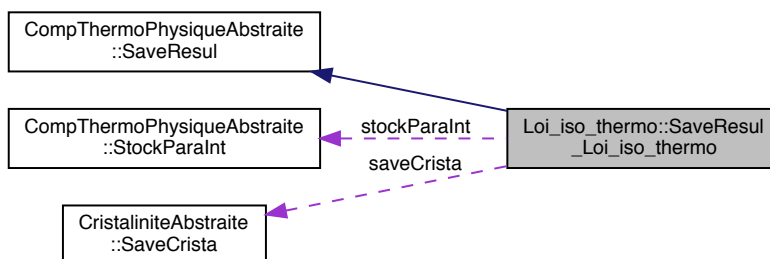
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas2D\_C.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas2D\_C.cc

## 6.732 Référence de la classe Loi\_iso\_thermo::SaveResul\_Loi\_iso\_thermo

Graphe d'héritage de Loi\_iso\_thermo::SaveResul\_Loi\_iso\_thermo:



Graphe de collaboration de Loi\_iso\_thermo::SaveResul\_Loi\_iso\_thermo:



### Fonctions membres publiques

- `SaveResul_Loi_iso_thermo` (`CompThermoPhysiqueAbstraite::StockParalnt *stock`, `CristaliniteAbstraite::SaveCrista *sCrista`)
- `SaveResul_Loi_iso_thermo` (`const SaveResul_Loi_iso_thermo &sav`)
- `SaveResul * Nevez_SaveResul` () const
- `virtual SaveResul & operator=` (`const SaveResul &a`)
- `void Lecture_base_info` (`ifstream &ent`, `const int cas`)
- `void Ecriture_base_info` (`ofstream &sort`, `const int cas`)
- `void Affiche` ()
- `virtual void ChBase_des_grandeurs` (`const Mat_pleine &beta`, `const Mat_pleine &gamma`)
- `SaveResul * Complete_SaveResul` (`const BlocGen &bloc`, `const Tableau< Coordonnee > &tab_coor`, `const CompThermoPhysiqueAbstraite *loi`)
- `void Affiche` (`ofstream &sort`)
- `void TdtversT` ()
- `void TversTdt` ()

### Attributs publics

- `CompThermoPhysiqueAbstraite::StockParalnt * stockParalnt`
- `CristaliniteAbstraite::SaveCrista * saveCrista`

## 6.732.1 Documentation des fonctions membres

### 6.732.1.1 ChBase\_des\_grandeurs()

```
virtual void Loi_iso_thermo::SaveResul_Loi_iso_thermo::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.732.1.2 Ecriture\_base\_info()

```
void Loi_iso_thermo::SaveResul_Loi_iso_thermo::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.732.1.3 Lecture\_base\_info()

```
void Loi_iso_thermo::SaveResul_Loi_iso_thermo::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.732.1.4 Nevez\_SaveResul()

```
SaveResul * Loi_iso_thermo::SaveResul_Loi_iso_thermo::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.732.1.5 operator=()

```
CompThermoPhysiqueAbstraite::SaveResul & Loi_iso_thermo::SaveResul_Loi_iso_thermo::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.732.1.6 TdtversT()

```
void Loi_iso_thermo::SaveResul_Loi_iso_thermo::TdtversT ( ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

### 6.732.1.7 TversTdt()

```
void Loi_iso_thermo::SaveResul_Loi_iso_thermo::TversTdt ( ) [virtual]
```

Implémente [CompThermoPhysiqueAbstraite::SaveResul](#).

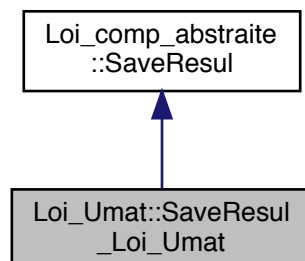
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi\_iso\_thermo.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/Loi\_iso\_thermo.cc

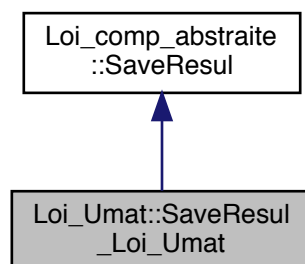


## 6.733 Référence de la classe Loi\_Umat::SaveResul\_Loi\_Umat

Graphe d'héritage de Loi\_Umat::SaveResul\_Loi\_Umat:



Graphe de collaboration de Loi\_Umat::SaveResul\_Loi\_Umat:



### Fonctions membres publiques

- `SaveResul_Loi_Umat` (list< [SaveResul](#) \* > &l\_des\_SaveResul, list< [TenseurHH](#) \* > &l\_siHH, list< [TenseurHH](#) \* > &l\_siHH\_t)
- `SaveResul_Loi_Umat` (const [SaveResul\\_Loi\\_Umat](#) &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const [SaveResul](#) &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- `SaveResul * Complete_SaveResul` (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)

### Attributs publics

- [SaveResul](#) \* `save_pour_loi_ext`
- double `hsurh0`
- double `h_tsurh0`

## 6.733.1 Documentation des fonctions membres

### 6.733.1.1 Affiche()

```
void Loi_Umat::SaveResul_Loi_Umat::Affiche ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.733.1.2 ChBase\_des\_grandeurs()

```
virtual void Loi_Umat::SaveResul_Loi_Umat::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.733.1.3 Complete\_SaveResul()

```
SaveResul * Loi_Umat::SaveResul_Loi_Umat::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.733.1.4 Ecriture\_base\_info()

```
void Loi_Umat::SaveResul_Loi_Umat::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.733.1.5 Lecture\_base\_info()

```
void Loi_Umat::SaveResul_Loi_Umat::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.733.1.6 Nevez\_SaveResul()

```
SaveResul * Loi_Umat::SaveResul_Loi_Umat::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.733.1.7 operator=()

```
Loi_comp_abstraite::SaveResul & Loi_Umat::SaveResul_Loi_Umat::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.733.1.8 TdtversT()

```
void Loi_Umat::SaveResul_Loi_Umat::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.733.1.9 TversTdt()

```
void Loi_Umat::SaveResul_Loi_Umat::TversTdt ( ) [virtual]
```

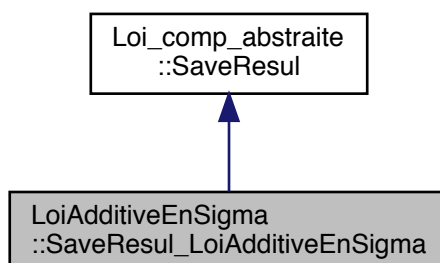
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

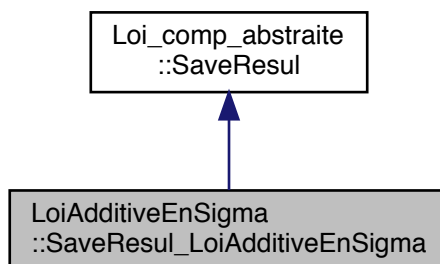
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loi\_Umat/Loi\_Umat.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loi\_Umat/Loi\_Umat.cc

## 6.734 Référence de la classe LoiAdditiveEnSigma::SaveResul\_LoiAdditiveEnSigma

Grappe d'héritage de LoiAdditiveEnSigma::SaveResul\_LoiAdditiveEnSigma:



Grappe de collaboration de LoiAdditiveEnSigma::SaveResul\_LoiAdditiveEnSigma:



### Fonctions membres publiques

- **SaveResul\_LoiAdditiveEnSigma** (list< [SaveResul](#) \* > &l\_des\_SaveResul, list< [TenseurHH](#) \* > &l\_si↔HH, list< [TenseurHH](#) \* > &l\_siHH\_t, list< [EnergieMeca](#) > &l\_energ, list< [EnergieMeca](#) > &l\_energ\_t, bool avec\_ponderation)
- **SaveResul\_LoiAdditiveEnSigma** (const [SaveResul\\_LoiAdditiveEnSigma](#) &sav)
- **SaveResul** \* **Nevez\_SaveResul** () const

- virtual [SaveResul](#) & operator= (const [SaveResul](#) &a)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [TdtversT](#) ()
- void [TversTdt](#) ()
- void [Affiche](#) () const
- virtual void [ChBase\\_des\\_grandeurs](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- [SaveResul](#) \* [Complete\\_SaveResul](#) (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- double [Deformation\\_plastique](#) ()

## Attributs publics

- list< [SaveResul](#) \* > [liste\\_des\\_SaveResul](#)
- list< [TenseurHH](#) \* > [I\\_sigoHH](#)
- list< [TenseurHH](#) \* > [I\\_sigoHH\\_t](#)
- list< [EnergieMeca](#) > [I\\_energ](#)
- list< [EnergieMeca](#) > [I\\_energ\\_t](#)
- list< double > [f\\_ponder](#)
- list< double > [f\\_ponder\\_t](#)

## 6.734.1 Documentation des fonctions membres

### 6.734.1.1 Affiche()

void [LoiAdditiveEnSigma::SaveResul\\_LoiAdditiveEnSigma::Affiche](#) ( ) const [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.2 ChBase\_des\_grandeurs()

void [LoiAdditiveEnSigma::SaveResul\\_LoiAdditiveEnSigma::ChBase\\_des\\_grandeurs](#) (   
     const [Mat\\_pleine](#) & beta,   
     const [Mat\\_pleine](#) & gamma ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.3 Complete\_SaveResul()

[Loi\\_comp\\_abstraite::SaveResul](#) \* [LoiAdditiveEnSigma::SaveResul\\_LoiAdditiveEnSigma::Complete](#) ←  
[SaveResul](#) (   
     const [BlocGen](#) & bloc,   
     const [Tableau](#)< [Coordonnee](#) > & tab\_coor,   
     const [Loi\\_comp\\_abstraite](#) \* loi ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.4 Deformation\_plastique()

double [LoiAdditiveEnSigma::SaveResul\\_LoiAdditiveEnSigma::Deformation\\_plastique](#) ( ) [virtual]  
 Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.5 Ecriture\_base\_info()

void [LoiAdditiveEnSigma::SaveResul\\_LoiAdditiveEnSigma::Ecriture\\_base\\_info](#) (   
     ofstream & sort,   
     const int cas ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.6 Lecture\_base\_info()

```
void LoiAdditiveEnSigma::SaveResul_LoiAdditiveEnSigma::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.7 Nevez\_SaveResul()

```
SaveResul * LoiAdditiveEnSigma::SaveResul_LoiAdditiveEnSigma::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.8 operator=()

```
Loi_comp_abstraite::SaveResul & LoiAdditiveEnSigma::SaveResul_LoiAdditiveEnSigma::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.9 TdtversT()

```
void LoiAdditiveEnSigma::SaveResul_LoiAdditiveEnSigma::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.734.1.10 TversTdt()

```
void LoiAdditiveEnSigma::SaveResul_LoiAdditiveEnSigma::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

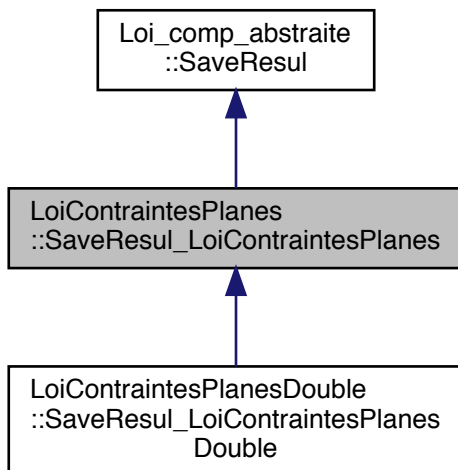
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiAdditiveEn↔Sigma.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiAdditiveEn↔Sigma.cc

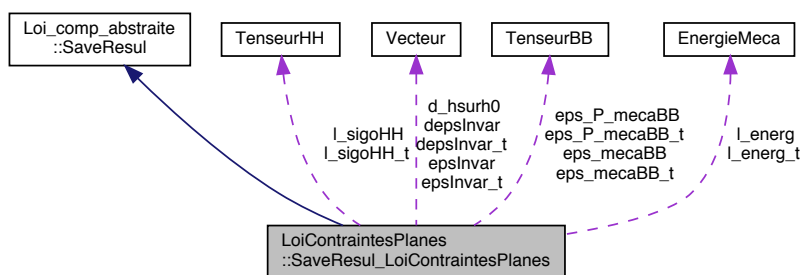
## 6.735 Référence de la classe

### LoiContraintesPlanes::SaveResul\_LoiContraintesPlanes

Grphe d'héritage de LoiContraintesPlanes::SaveResul\_LoiContraintesPlanes:



Grphe de collaboration de LoiContraintesPlanes::SaveResul\_LoiContraintesPlanes:



### Fonctions membres publiques

- **SaveResul\_LoiContraintesPlanes** (**SaveResul** \*l\_des\_SaveResul)
- **SaveResul\_LoiContraintesPlanes** (const **SaveResul\_LoiContraintesPlanes** &sav)
- **SaveResul** \* **Nevez\_SaveResul** () const
- virtual **SaveResul** & **operator=** (const **SaveResul** &a)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **TdtversT** ()
- void **TversTdt** ()
- void **Affiche** () const
- virtual void **ChBase\_des\_grandeurs** (const **Mat\_pleine** &beta, const **Mat\_pleine** &gamma)
- **SaveResul** \* **Complete\_SaveResul** (const **BlocGen** &bloc, const **Tableau**< **Coordonnee** > &tab\_coor, const **Loi\_comp\_abstraite** \*loi)

- `double` `Deformation_plastique` ()
- `void` `Transfert_var_h` (const `SaveResul_LoiContraintesPlanes` &sav)
- `void` `Creation_def_mecanique` ()

### Attributs publics

- `SaveResul` \* `le_SaveResul`
- `TenseurHH` \* `I_sigoHH`
- `TenseurHH` \* `I_sigoHH_t`
- `Vecteur` `epsInvar`
- `Vecteur` `epsInvar_t`
- `Vecteur` `depsInvar`
- `Vecteur` `depsInvar_t`
- `Tableau`< `double` > `def_equi`
- `Tableau`< `double` > `def_equi_t`
- `TenseurBB` \* `eps_mecaBB`
- `TenseurBB` \* `eps_mecaBB_t`
- `TenseurBB` \* `eps_P_mecaBB`
- `TenseurBB` \* `eps_P_mecaBB_t`
- `Met_abstraite::Umat_cont` `met_meca`
- `EnergieMeca` `I_energ`
- `EnergieMeca` `I_energ_t`
- `double` `hsurh0`
- `double` `h_tsurh0`
- `Vecteur` `d_hsurh0`
- `Tableau`< `double` > `indicateurs_resolution`
- `Tableau`< `double` > `indicateurs_resolution_t`

## 6.735.1 Documentation des fonctions membres

### 6.735.1.1 Affiche()

`void` `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::Affiche` ( ) const [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

### 6.735.1.2 ChBase\_des\_grandeurs()

`void` `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::ChBase_des_grandeurs` (   
     const `Mat_pleine` & `beta`,   
     const `Mat_pleine` & `gamma` ) [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

### 6.735.1.3 Complete\_SaveResul()

`Loi_comp_abstraite::SaveResul` \* `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::Complete↔_SaveResul` (   
     const `BlocGen` & `bloc`,   
     const `Tableau`< `Coordonnee` > & `tab_coor`,   
     const `Loi_comp_abstraite` \* `loi` ) [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

### 6.735.1.4 Deformation\_plastique()

`double` `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::Deformation_plastique` ( ) [virtual]  
 Réimplémentée à partir de `Loi_comp_abstraite::SaveResul`.

### 6.735.1.5 Ecriture\_base\_info()

```
void LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.735.1.6 Lecture\_base\_info()

```
void LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.735.1.7 Nevez\_SaveResul()

```
SaveResul * LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::Nevez_SaveResul ( ) const
[inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.735.1.8 operator=()

```
Loi_comp_abstraite::SaveResul & LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::operator=
(
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.735.1.9 TdtversT()

```
void LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.735.1.10 TversTdt()

```
void LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

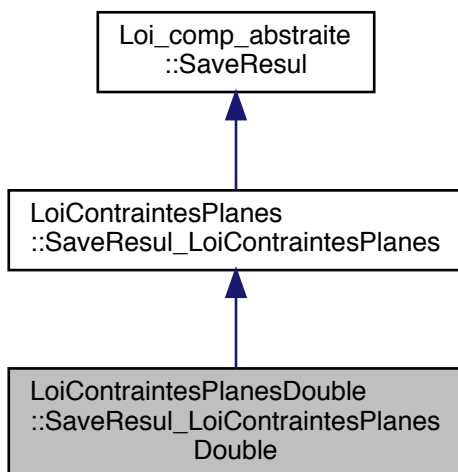
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔  
Planes.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔  
Planes.cc

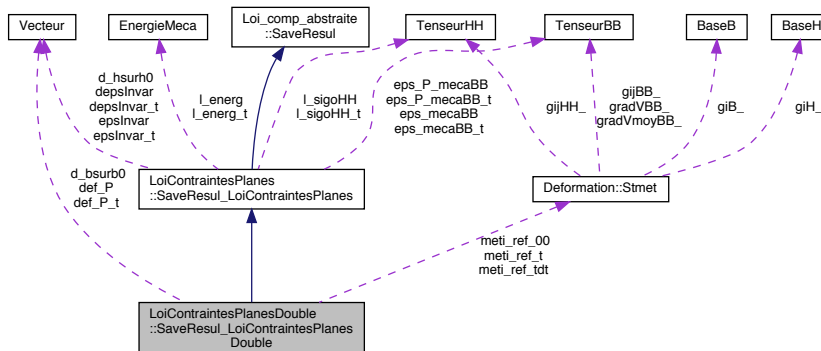


## 6.736 Référence de la classe LoiContraintesPlanesDouble::SaveResul\_LoiContraintesPlanesDouble

Graphe d'héritage de LoiContraintesPlanesDouble::SaveResul\_LoiContraintesPlanesDouble:



Graphe de collaboration de LoiContraintesPlanesDouble::SaveResul\_LoiContraintesPlanesDouble:



### Fonctions membres publiques

- **SaveResul\_LoiContraintesPlanesDouble** ([SaveResul](#) \*l\_des\_SaveResul)
- **SaveResul\_LoiContraintesPlanesDouble** (const [SaveResul\\_LoiContraintesPlanesDouble](#) &sav)
- [SaveResul](#) \* [Nevez\\_SaveResul](#) () const
- virtual [SaveResul](#) & operator= (const [SaveResul](#) &a)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [TdtversT](#) ()
- void [TversTdt](#) ()
- void [Affiche](#) () const
- virtual void [ChBase\\_des\\_grandeurs](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)

- `SaveResul * Complete_SaveResul` (const `BlocGen` &\_bloc, const `Tableau< Coordonnee >` &tab\_coor, const `Loi_comp_abstraite *loi`)
- `double Deformation_plastique` ()
- `void Transfert_var_hetb` (const `SaveResul_LoiContraintesPlanesDouble` &sav)
- `void Affectation_metriques_locale` (`Deformation::Stmet` &met\_ref\_00, `Deformation::Stmet` &met\_ref\_t, `Deformation::Stmet` &met\_ref\_tdt)
- `void Zero_var_largeur` ()

## Attributs publics

- `Deformation::Stmet meti_ref_00`
- `Deformation::Stmet meti_ref_t`
- `Deformation::Stmet meti_ref_tdt`
- `Vecteur def_P`
- `Vecteur def_P_t`
- `double bsurb0`
- `double b_tsurb0`
- `Vecteur d_bsurb0`

## 6.736.1 Documentation des fonctions membres

### 6.736.1.1 Affiche()

`void LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::Affiche ( ) const [virtual]`  
 Réimplémentée à partir de `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes`.

### 6.736.1.2 ChBase\_des\_grandeurs()

`void LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::ChBase_des_grandeurs ( const Mat_pleine & beta, const Mat_pleine & gamma ) [virtual]`

Réimplémentée à partir de `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes`.

### 6.736.1.3 Complete\_SaveResul()

`Loi_comp_abstraite::SaveResul * LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::Complete_SaveResul ( const BlocGen & bloc, const Tableau< Coordonnee > & tab_coor, const Loi_comp_abstraite * loi ) [virtual]`

Réimplémentée à partir de `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes`.

### 6.736.1.4 Deformation\_plastique()

`double LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::Deformation_plastique ( ) [virtual]`

Réimplémentée à partir de `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes`.

### 6.736.1.5 Ecriture\_base\_info()

`void LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::Ecriture_base_info ( ofstream & sort, const int cas ) [virtual]`

Réimplémentée à partir de `LoiContraintesPlanes::SaveResul_LoiContraintesPlanes`.

**6.736.1.6 Lecture\_base\_info()**

```
void LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Réimplémentée à partir de [LoiContraintesPlanes::SaveResul\\_LoiContraintesPlanes](#).

**6.736.1.7 Nevez\_SaveResul()**

```
SaveResul * LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::Nevez_SaveResul
( ) const [inline], [virtual]
```

Réimplémentée à partir de [LoiContraintesPlanes::SaveResul\\_LoiContraintesPlanes](#).

**6.736.1.8 operator=()**

```
Loi_comp_abstraite::SaveResul & LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanes↔
Double::operator= (
    const SaveResul & a ) [virtual]
```

Réimplémentée à partir de [LoiContraintesPlanes::SaveResul\\_LoiContraintesPlanes](#).

**6.736.1.9 TdtversT()**

```
void LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [LoiContraintesPlanes::SaveResul\\_LoiContraintesPlanes](#).

**6.736.1.10 TversTdt()**

```
void LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble::TversTdt ( ) [virtual]
```

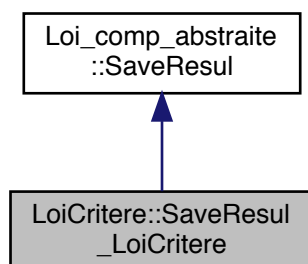
Réimplémentée à partir de [LoiContraintesPlanes::SaveResul\\_LoiContraintesPlanes](#).

La documentation de cette classe a été générée à partir du fichier suivant :

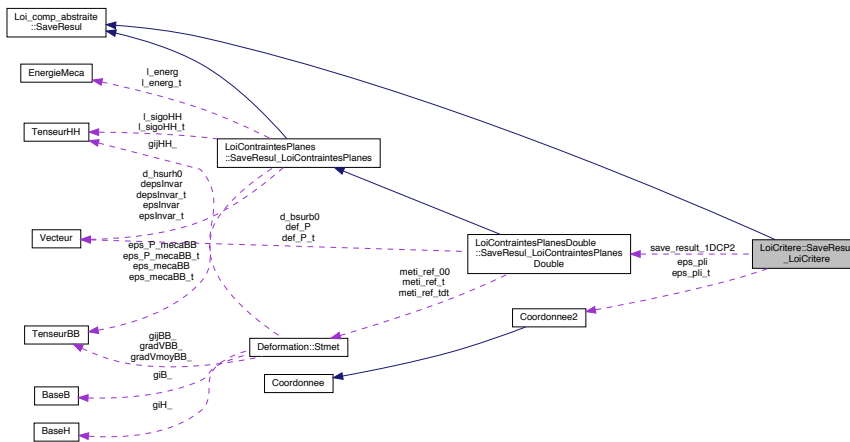
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔PlanesDouble.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiContraintes↔PlanesDouble.cc

**6.737 Référence de la classe LoiCritere::SaveResul\_LoiCritere**

Graphe d'héritage de LoiCritere::SaveResul\_LoiCritere:



Graphe de collaboration de `LoiCritere::SaveResul_LoiCritere`:



## Fonctions membres publiques

- `SaveResul_LoiCritere` (list< `SaveResul` \* > &I\_des\_SaveResul, list< `TenseurHH` \* > &I\_siHH, list< `TenseurHH` \* > &I\_siHH\_t, list< `EnergieMeca` > &I\_energ, list< `EnergieMeca` > &I\_energ\_t, bool avec\_↔ ponderation, `Enum_Critere_Loi` type\_crite)
- `SaveResul_LoiCritere` (const `SaveResul_LoiCritere` &sav)
- `SaveResul` \* `Nevez_SaveResul` () const
- virtual `SaveResul` & `operator=` (const `SaveResul` &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- `SaveResul` \* `Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau`< `Coordonnee` > &tab\_coor, const `Loi_comp_abstraite` \*loi)
- double `Deformation_plastique` ()
- const map< `EnumTypeQuelconque`, `TypeQuelconque`, std::less< `EnumTypeQuelconque` > > \* `Map_type_quelconque` () const

## Attributs publics

- list< `SaveResul` \* > `liste_des_SaveResul`
- list< `TenseurHH` \* > `I_sigoHH`
- list< `TenseurHH` \* > `I_sigoHH_t`
- list< `EnergieMeca` > `I_energ`
- list< `EnergieMeca` > `I_energ_t`
- list< double > `f_ponder`
- list< double > `f_ponder_t`
- `Tableau`< `Coordonnee` > \* `V_P_sig`
- `Tableau`< `Coordonnee` > \* `V_P_sig_t`
- `Coordonnee2` `eps_pli`
- `Coordonnee2` `eps_pli_t`
- `LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble` \* `save_result_1DCP2`
- `Enum_Critere_Loi` `le_type_critere`
- double `niveau_declenchement_critere`
- int `cas_cal_plis`
- int `cas_cal_plis_t`
- map< `EnumTypeQuelconque`, `TypeQuelconque`, std::less< `EnumTypeQuelconque` > > `map_type_↔ quelconque`

## 6.737.1 Documentation des fonctions membres

### 6.737.1.1 Affiche()

```
void LoiCritere::SaveResul_LoiCritere::Affiche ( ) const [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.2 ChBase\_des\_grandeurs()

```
void LoiCritere::SaveResul_LoiCritere::ChBase_des_grandeurs (
```

```
    const Mat_pleine & beta,
```

```
    const Mat_pleine & gamma ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.3 Complete\_SaveResul()

```
Loi_comp_abstraite::SaveResul * LoiCritere::SaveResul_LoiCritere::Complete_SaveResul (
```

```
    const BlocGen & bloc,
```

```
    const Tableau< Coordonnee > & tab_coor,
```

```
    const Loi_comp_abstraite * loi ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.4 Deformation\_plastique()

```
double LoiCritere::SaveResul_LoiCritere::Deformation_plastique ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.5 Ecriture\_base\_info()

```
void LoiCritere::SaveResul_LoiCritere::Ecriture_base_info (
```

```
    ofstream & sort,
```

```
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.6 Lecture\_base\_info()

```
void LoiCritere::SaveResul_LoiCritere::Lecture_base_info (
```

```
    ifstream & ent,
```

```
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.7 Map\_type\_quelconque()

```
const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * Loi↔
```

```
Critere::SaveResul_LoiCritere::Map_type_quelconque ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.8 Nevez\_SaveResul()

```
SaveResul * LoiCritere::SaveResul_LoiCritere::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.9 operator=()

```
Loi_comp_abstraite::SaveResul & LoiCritere::SaveResul_LoiCritere::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.10 TdtversT()

```
void LoiCritere::SaveResul_LoiCritere::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.737.1.11 TversTdt()

```
void LoiCritere::SaveResul_LoiCritere::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

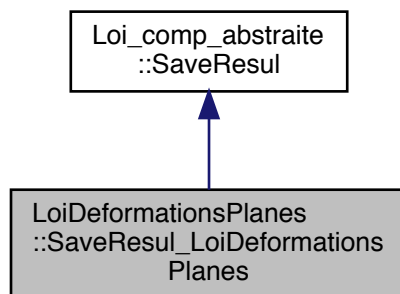
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiCritere.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiCritere.cc

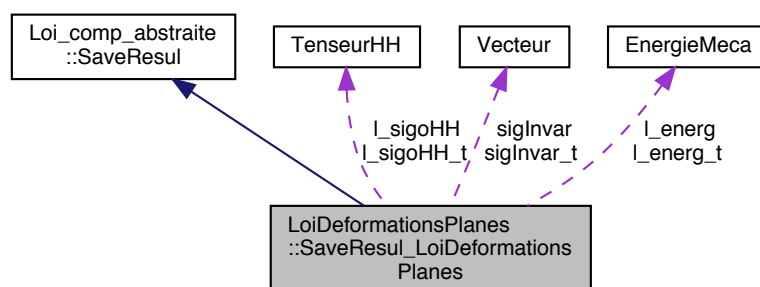
## 6.738 Référence de la classe

### LoiDeformationsPlanes::SaveResul\_LoiDeformationsPlanes

Graphe d'héritage de LoiDeformationsPlanes::SaveResul\_LoiDeformationsPlanes:



Graph de collaboration de LoiDeformationsPlanes::SaveResul\_LoiDeformationsPlanes:



## Fonctions membres publiques

- `SaveResul_LoiDeformationsPlanes` (`SaveResul *l_des_SaveResul`)
- `SaveResul_LoiDeformationsPlanes` (`const SaveResul_LoiDeformationsPlanes &sav`)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (`const SaveResul &a`)
- void `Lecture_base_info` (`ifstream &ent, const int cas`)
- void `Ecriture_base_info` (`ofstream &sort, const int cas`)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (`const Mat_pleine &beta, const Mat_pleine &gamma`)
- `SaveResul * Complete_SaveResul` (`const BlocGen &bloc, const Tableau< Coordonnee > &tab_coor, const Loi_comp_abstraite *loi`)
- double `Deformation_plastique` ()

## Attributs publics

- `SaveResul * le_SaveResul`
- `TenseurHH * I_sigoHH`
- `TenseurHH * I_sigoHH_t`
- `Vecteur sigInvar`
- `Vecteur sigInvar_t`
- `EnergieMeca I_energ`
- `EnergieMeca I_energ_t`

## 6.738.1 Documentation des fonctions membres

### 6.738.1.1 Affiche()

void LoiDeformationsPlanes::SaveResul\_LoiDeformationsPlanes::Affiche () const [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.738.1.2 ChBase\_des\_grandeurs()

void LoiDeformationsPlanes::SaveResul\_LoiDeformationsPlanes::ChBase\_des\_grandeurs (   
     const `Mat_pleine` & `beta`,   
     const `Mat_pleine` & `gamma` ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.738.1.3 Complete\_SaveResul()**

```
Loi_comp_abstraite::SaveResul * LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes::↔
Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.738.1.4 Deformation\_plastique()**

```
double LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes::Deformation_plastique ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.738.1.5 Ecriture\_base\_info()**

```
void LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.738.1.6 Lecture\_base\_info()**

```
void LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.738.1.7 Nevez\_SaveResul()**

```
SaveResul * LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes::Nevez_SaveResul ( ) const
[inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.738.1.8 operator=()**

```
Loi_comp_abstraite::SaveResul & LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes::operator=
(
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.738.1.9 TdtversT()**

```
void LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.738.1.10 TversTdt()**

```
void LoiDeformationsPlanes::SaveResul_LoiDeformationsPlanes::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

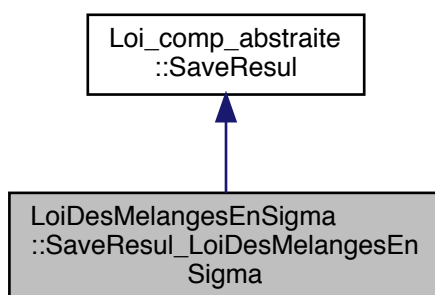
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiDeformations↔  
Planes.h



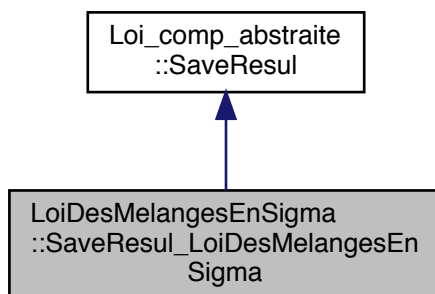
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/lois\_combinees/LoiDeformations←  
Planes.cc

## 6.739 Référence de la classe LoiDesMelangesEnSigma::SaveResul\_LoiDesMelangesEnSigma

Graphe d'héritage de LoiDesMelangesEnSigma::SaveResul\_LoiDesMelangesEnSigma:



Graphe de collaboration de LoiDesMelangesEnSigma::SaveResul\_LoiDesMelangesEnSigma:



### Fonctions membres publiques

- **SaveResul\_LoiDesMelangesEnSigma** (list< [SaveResul](#) \* > &l\_des\_SaveResul, list< [TenseurHH](#) \* > &l\_siHH, list< [TenseurHH](#) \* > &l\_siHH\_t, list< [TenseurHH](#) \* > &J\_siHH, list< [TenseurHH](#) \* > &J\_siHH\_t, list< [EnergieMeca](#) > &l\_energ, list< [EnergieMeca](#) > &l\_energ\_t, int type\_evol\_proportion)
- **SaveResul\_LoiDesMelangesEnSigma** (const [SaveResul\\_LoiDesMelangesEnSigma](#) &sav)
- [SaveResul](#) \* **Nevez\_SaveResul** () const
- virtual [SaveResul](#) & **operator=** (const [SaveResul](#) &a)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- void **TdtversT** ()
- void **TversTdt** ()

- void [Affiche](#) () const
- virtual void [ChBase\\_des\\_grandeurs](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- [SaveResul](#) \* [Complete\\_SaveResul](#) (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- double [Deformation\\_plastique](#) ()

## Attributs publics

- list< [SaveResul](#) \* > [liste\\_des\\_SaveResul](#)
- list< [TenseurHH](#) \* > [I\\_sigoHH](#)
- list< [TenseurHH](#) \* > [I\\_sigoHH\\_t](#)
- list< [TenseurHH](#) \* > [J\\_sigoHH](#)
- list< [TenseurHH](#) \* > [J\\_sigoHH\\_t](#)
- list< [EnergieMeca](#) > [I\\_energ](#)
- list< [EnergieMeca](#) > [I\\_energ\\_t](#)
- double [proportion](#)
- double [proportion\\_t](#)
- int [type\\_evolution\\_proportion](#)
- bool [deja\\_actif\\_sur\\_iter1](#)
- bool [deja\\_actif\\_sur\\_iter2](#)

## 6.739.1 Documentation des fonctions membres

### 6.739.1.1 Affiche()

void [LoiDesMelangesEnSigma::SaveResul\\_LoiDesMelangesEnSigma::Affiche](#) () const [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.2 ChBase\_des\_grandeurs()

void [LoiDesMelangesEnSigma::SaveResul\\_LoiDesMelangesEnSigma::ChBase\\_des\\_grandeurs](#) (  
     const [Mat\\_pleine](#) & *beta*,  
     const [Mat\\_pleine](#) & *gamma* ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.3 Complete\_SaveResul()

[Loi\\_comp\\_abstraite::SaveResul](#) \* [LoiDesMelangesEnSigma::SaveResul\\_LoiDesMelangesEnSigma::↔](#)  
[Complete\\_SaveResul](#) (  
     const [BlocGen](#) & *bloc*,  
     const [Tableau](#)< [Coordonnee](#) > & *tab\_coor*,  
     const [Loi\\_comp\\_abstraite](#) \* *loi* ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.4 Deformation\_plastique()

double [LoiDesMelangesEnSigma::SaveResul\\_LoiDesMelangesEnSigma::Deformation\\_plastique](#) () [virtual]  
 Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.5 Ecriture\_base\_info()

void [LoiDesMelangesEnSigma::SaveResul\\_LoiDesMelangesEnSigma::Ecriture\\_base\\_info](#) (  
     ofstream & *sort*,  
     const int *cas* ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.6 `Lecture_base_info()`

```
void LoiDesMelangesEnSigma::SaveResul_LoiDesMelangesEnSigma::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.7 `Nevez_SaveResul()`

```
SaveResul * LoiDesMelangesEnSigma::SaveResul_LoiDesMelangesEnSigma::Nevez_SaveResul ( ) const
[inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.8 `operator=()`

```
Loi_comp_abstraite::SaveResul & LoiDesMelangesEnSigma::SaveResul_LoiDesMelangesEnSigma::operator=
(
```

```
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.9 `TdtversT()`

```
void LoiDesMelangesEnSigma::SaveResul_LoiDesMelangesEnSigma::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.739.1.10 `TversTdt()`

```
void LoiDesMelangesEnSigma::SaveResul_LoiDesMelangesEnSigma::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

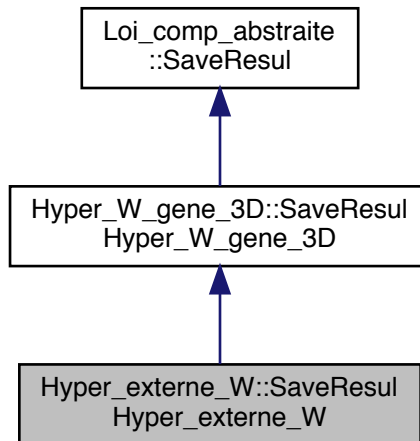
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiDesMelangesEnSigma.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/loiscombinees/LoiDesMelangesEnSigma.cc`

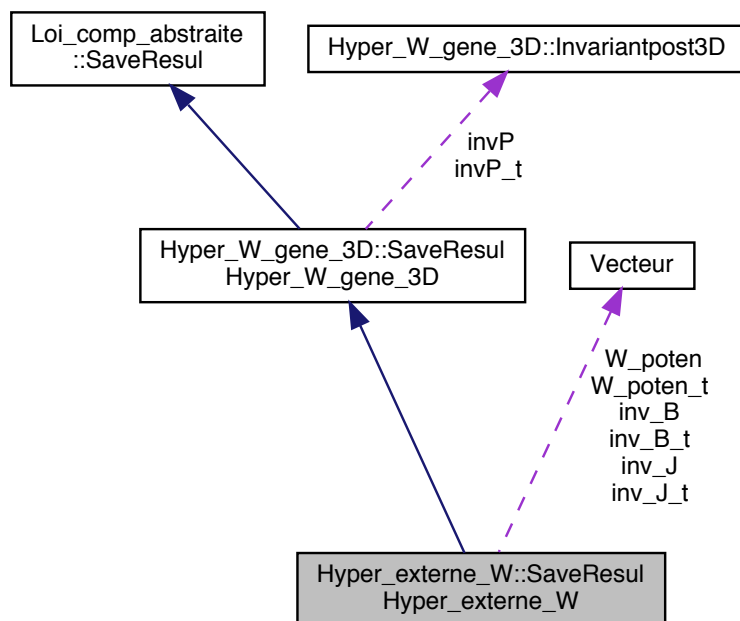
## 6.740 Référence de la classe

**Hyper\_externe\_W::SaveResulHyper\_externe\_W**

Graphe d'héritage de Hyper\_externe\_W::SaveResulHyper\_externe\_W:



Graphe de collaboration de Hyper\_externe\_W::SaveResulHyper\_externe\_W:



## Fonctions membres publiques

- `SaveResulHyper_externer_W` (int sortie\_post)
- `SaveResulHyper_externer_W` (const `SaveResulHyper_externer_W` &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- virtual `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau`< `Coordonnee` > &tab\_coor, const `Loi_comp_abstraite` \*loi)

## Attributs publics

- double `module_compressibilite`
- double `module_compressibilite_t`
- `Vecteur * W_poten`
- `Vecteur * W_poten_t`
- `Vecteur * inv_B`
- `Vecteur * inv_B_t`
- `Vecteur * inv_J`
- `Vecteur * inv_J_t`

### 6.740.1 Documentation des fonctions membres

#### 6.740.1.1 `Affiche()`

void `Hyper_externer_W::SaveResulHyper_externer_W::Affiche` ( ) const [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

#### 6.740.1.2 `ChBase_des_grandeurs()`

virtual void `Hyper_externer_W::SaveResulHyper_externer_W::ChBase_des_grandeurs` ( const `Mat_pleine` & beta, const `Mat_pleine` & gamma ) [inline], [virtual]  
 Réimplémentée à partir de `Hyper_W_gene_3D::SaveResulHyper_W_gene_3D`.

#### 6.740.1.3 `Complete_SaveResul()`

virtual `SaveResul * Hyper_externer_W::SaveResulHyper_externer_W::Complete_SaveResul` ( const `BlocGen` & bloc, const `Tableau`< `Coordonnee` > & tab\_coor, const `Loi_comp_abstraite` \* loi ) [inline], [virtual]  
 Implémente `Loi_comp_abstraite::SaveResul`.

#### 6.740.1.4 `Ecriture_base_info()`

void `Hyper_externer_W::SaveResulHyper_externer_W::Ecriture_base_info` ( ofstream & sort, const int cas ) [virtual]  
 Réimplémentée à partir de `Hyper_W_gene_3D::SaveResulHyper_W_gene_3D`.

### 6.740.1.5 Lecture\_base\_info()

```
void Hyper_externe_W::SaveResulHyper_externe_W::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Réimplémentée à partir de [Hyper\\_W\\_gene\\_3D::SaveResulHyper\\_W\\_gene\\_3D](#).

### 6.740.1.6 Nevez\_SaveResul()

```
SaveResul * Hyper_externe_W::SaveResulHyper_externe_W::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.740.1.7 operator=()

```
Loi_comp_abstraite::SaveResul & Hyper_externe_W::SaveResulHyper_externe_W::operator= (
    const SaveResul & a ) [virtual]
```

Réimplémentée à partir de [Hyper\\_W\\_gene\\_3D::SaveResulHyper\\_W\\_gene\\_3D](#).

### 6.740.1.8 TdtversT()

```
void Hyper_externe_W::SaveResulHyper_externe_W::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.740.1.9 TversTdt()

```
void Hyper_externe_W::SaveResulHyper_externe_W::TversTdt ( ) [virtual]
```

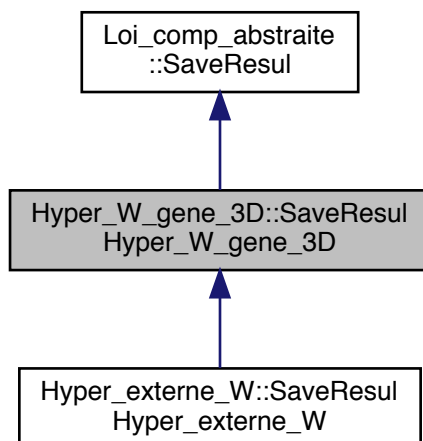
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

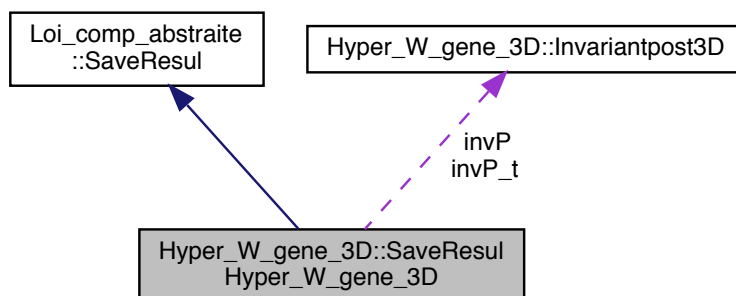
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_externe\_↔  
W.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_externe\_↔  
W.cc

## 6.741 Référence de la classe Hyper\_W\_gene\_3D::SaveResulHyper\_W\_gene\_3D

Graphe d'héritage de Hyper\_W\_gene\_3D::SaveResulHyper\_W\_gene\_3D:



Graphe de collaboration de Hyper\_W\_gene\_3D::SaveResulHyper\_W\_gene\_3D:



### Fonctions membres publiques

- `SaveResulHyper_W_gene_3D` (int sortie\_post)
- `SaveResulHyper_W_gene_3D` (const `SaveResulHyper_W_gene_3D` &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- virtual void `Lecture_base_info` (ifstream &, const int)
- virtual void `Ecriture_base_info` (ofstream &, const int)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)

- `SaveResul * Complete_SaveResul` (const `BlocGen` &`bloc`, const `Tableau`< `Coordonnee` > &`tab_coor`, const `Loi_comp_abstraite` \*`loi`)

## Attributs publics

- `Hyper_W_gene_3D::Invariantpost3D` \* `invP`
- `Hyper_W_gene_3D::Invariantpost3D` \* `invP_t`

## 6.741.1 Documentation des fonctions membres

### 6.741.1.1 Affiche()

`void Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::Affiche ( ) const [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.741.1.2 ChBase\_des\_grandeurs()

`virtual void Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::ChBase_des_grandeurs (`  
     const `Mat_pleine` & `beta`,  
     const `Mat_pleine` & `gamma` ) [inline], [virtual]

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.741.1.3 Complete\_SaveResul()

`SaveResul * Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::Complete_SaveResul (`  
     const `BlocGen` & `bloc`,  
     const `Tableau`< `Coordonnee` > & `tab_coor`,  
     const `Loi_comp_abstraite` \* `loi` ) [inline], [virtual]

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.741.1.4 Ecriture\_base\_info()

`void Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::Ecriture_base_info (`  
     ofstream & `sort`,  
     const int ) [virtual]

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.741.1.5 Lecture\_base\_info()

`void Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::Lecture_base_info (`  
     ifstream & `ent`,  
     const int ) [virtual]

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.741.1.6 Nevez\_SaveResul()

`SaveResul * Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::Nevez_SaveResul ( ) const [inline],`  
 [virtual]

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).



### 6.741.1.7 operator=()

```
Loi_comp_abstraite::SaveResul & Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.741.1.8 TdtversT()

```
void Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.741.1.9 TversTdt()

```
void Hyper_W_gene_3D::SaveResulHyper_W_gene_3D::TversTdt ( ) [inline], [virtual]
```

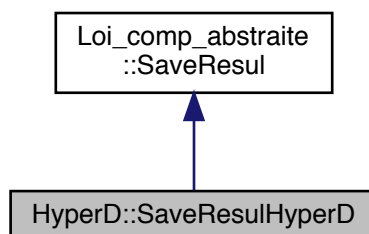
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

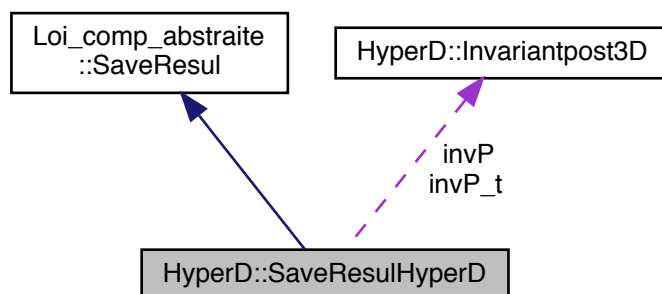
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_W\_gene\_3↔D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/Hyper\_W\_gene\_3↔D.cc

## 6.742 Référence de la classe HyperD::SaveResulHyperD

Graphe d'héritage de HyperD::SaveResulHyperD:



Graphe de collaboration de HyperD::SaveResulHyperD:



## Fonctions membres publiques

- **SaveResulHyperD** (int sortie\_post)
- **SaveResulHyperD** (const [SaveResulHyperD](#) &sav)
- [SaveResul](#) \* [Nevez\\_SaveResul](#) () const
- virtual [SaveResul](#) & [operator=](#) (const [SaveResul](#) &a)
- virtual void [Lecture\\_base\\_info](#) (ifstream &, const int)
- virtual void [Ecriture\\_base\\_info](#) (ofstream &, const int)
- void [TdtversT](#) ()
- void [TversTdt](#) ()
- void [Affiche](#) () const
- virtual void [ChBase\\_des\\_grandeurs](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- [SaveResul](#) \* [Complete\\_SaveResul](#) (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- const map< [EnumTypeQuelconque](#), [TypeQuelconque](#), std::less< [EnumTypeQuelconque](#) > > \* [Map\\_type\\_quelconque](#) () const

## Attributs publics

- [HyperD::Invariantpost3D](#) \* [invP](#)
- [HyperD::Invariantpost3D](#) \* [invP\\_t](#)
- map< [EnumTypeQuelconque](#), [TypeQuelconque](#), std::less< [EnumTypeQuelconque](#) > > [map\\_type\\_quelconque](#)

## 6.742.1 Documentation des fonctions membres

### 6.742.1.1 Affiche()

```
void HyperD::SaveResulHyperD::Affiche ( ) const [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.742.1.2 ChBase\_des\_grandeurs()

```
virtual void HyperD::SaveResulHyperD::ChBase_des_grandeurs (
    const Mat\_pleine & beta,
    const Mat\_pleine & gamma ) [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.742.1.3 Complete\_SaveResul()

```
SaveResul * HyperD::SaveResulHyperD::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.742.1.4 Ecriture\_base\_info()

```
void HyperD::SaveResulHyperD::Ecriture_base_info (
    ofstream & sort,
    const int ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.742.1.5 Lecture\_base\_info()

```
void HyperD::SaveResulHyperD::Lecture_base_info (
    ifstream & ent,
    const int ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.742.1.6 Map\_type\_quelconque()

```
const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * HyperD::↔
SaveResulHyperD::Map_type_quelconque ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.742.1.7 Nevez\_SaveResul()

```
SaveResul * HyperD::SaveResulHyperD::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.742.1.8 operator=()

```
Loi_comp_abstraite::SaveResul & HyperD::SaveResulHyperD::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.742.1.9 TdtversT()

```
void HyperD::SaveResulHyperD::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.742.1.10 TversTdt()

```
void HyperD::SaveResulHyperD::TversTdt ( ) [virtual]
```

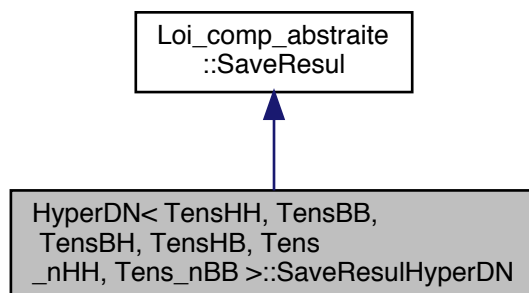
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

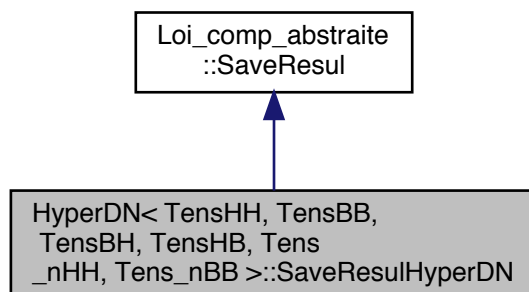
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperD.cc

## 6.743 Référence de la classe `HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN`

Graphe d'héritage de `HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN`:



Graphe de collaboration de `HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN`:



### Fonctions membres publiques

- `SaveResulHyperDN` (const double &jacob)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- virtual void `Lecture_base_info` (ifstream &ent, const int cas)
- virtual void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau< Coordonnee >` &tab\_coor, const `Loi_comp_abstraite *loi`)

### Attributs publics

- double `jacobien_0`

## 6.743.1 Documentation des fonctions membres

### 6.743.1.1 Affiche()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class Tens_nBB >
void HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN::Affiche
( ) const [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.743.1.2 ChBase\_des\_grandeurs()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class Tens_nBB >
virtual void HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN↔
::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.743.1.3 Complete\_SaveResul()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class Tens_nBB >
SaveResul * HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN↔
::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.743.1.4 Ecriture\_base\_info()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class Tens_nBB >
virtual void HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN↔
::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.743.1.5 Lecture\_base\_info()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class Tens_nBB >
virtual void HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN↔
::Lecture_base_info (
    ifstream & ent,
    const int cas ) [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.743.1.6 Nevez\_SaveResul()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class
Tens_nBB >
SaveResul * HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResulHyperDN↔
::Nevez_SaveResul ( ) const [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.743.1.7 operator=()

```
template<class TensHH , class TensBB , class TensBH , class TensHB , class Tens_nHH , class
Tens_nBB >
virtual SaveResul & HyperDN< TensHH, TensBB, TensBH, TensHB, Tens_nHH, Tens_nBB >::SaveResul↔
HyperDN::operator= (
    const SaveResul & a ) [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

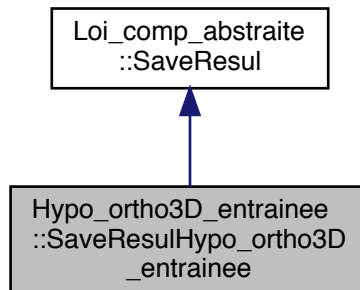
La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/HyperDN.h

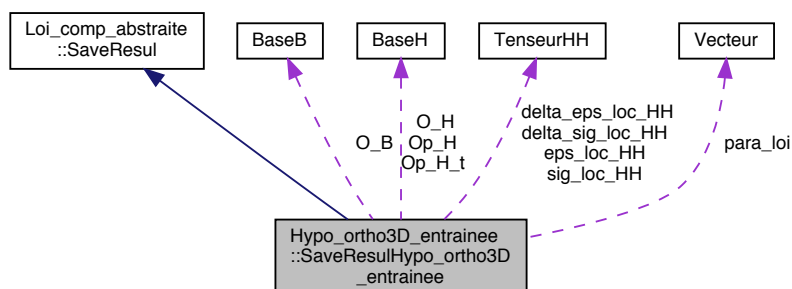
## 6.744 Référence de la classe

### Hypo\_ortho3D\_entrainee::SaveResulHypo\_ortho3D\_entrainee

Graphe d'héritage de Hypo\_ortho3D\_entrainee::SaveResulHypo\_ortho3D\_entrainee:



Graphe de collaboration de Hypo\_ortho3D\_entrainee::SaveResulHypo\_ortho3D\_entrainee:



## Fonctions membres publiques

- `SaveResulHypo_ortho3D_entrainee` (const int type\_transport=0)
- `SaveResulHypo_ortho3D_entrainee` (const `SaveResulHypo_ortho3D_entrainee` &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau`< `Coordonnee` > &tab\_coor, const `Loi_comp_abstraite` \*loi)
- void `Init_debut_calcul` ()

## Attributs publics

- `BaseB` \* `O_B`
- `BaseH` \* `O_H`
- `BaseH` `Op_H`
- `BaseH` `Op_H_t`
- `TenseurHH` \* `eps_loc_HH`
- `TenseurHH` \* `sig_loc_HH`
- `TenseurHH` \* `delta_eps_loc_HH`
- `TenseurHH` \* `delta_sig_loc_HH`
- `Vecteur` \* `para_loi`

### 6.744.1 Documentation des fonctions membres

#### 6.744.1.1 Affiche()

```
void Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee::Affiche () const [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

#### 6.744.1.2 ChBase\_des\_grandeurs()

```
void Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

**6.744.1.3 Complete\_SaveResul()**

```
Loi_comp_abstraite::SaveResul * Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee::↔
Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.744.1.4 Ecriture\_base\_info()**

```
void Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.744.1.5 Lecture\_base\_info()**

```
void Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.744.1.6 Nevez\_SaveResul()**

```
SaveResul * Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee::Nevez_SaveResul ( ) const
[inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.744.1.7 operator=()**

```
Loi_comp_abstraite::SaveResul & Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee↔
::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.744.1.8 TdtversT()**

```
void Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.744.1.9 TversTdt()**

```
void Hypo_ortho3D_entrainee::SaveResulHypo_ortho3D_entrainee::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

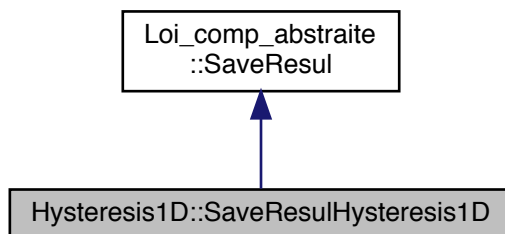
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Hypo\_ortho3D\_entrainee.↔h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Hypo\_ortho3D\_entrainee.↔cc

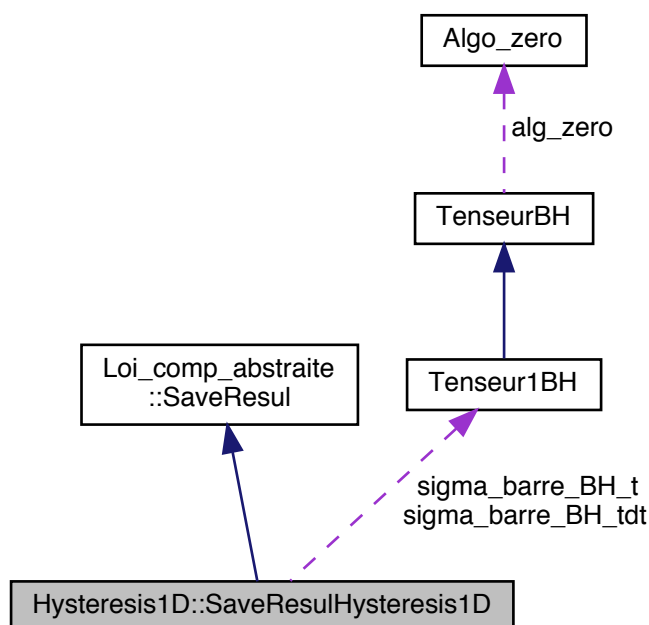


## 6.745 Référence de la classe Hysteresis1D::SaveResulHysteresis1D

Grappe d'héritage de Hysteresis1D::SaveResulHysteresis1D:



Grappe de collaboration de Hysteresis1D::SaveResulHysteresis1D:



### Fonctions membres publiques

- **SaveResulHysteresis1D** (const [SaveResulHysteresis1D](#) &sav)
- [SaveResul](#) \* [Nevez\\_SaveResul](#) () const
- virtual [SaveResul](#) & [operator=](#) (const [SaveResul](#) &a)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [TdtversT](#) ()
- void [TversTdt](#) ()

- void [Affiche](#) () const
- virtual void [ChBase\\_des\\_grandeurs](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- [SaveResul](#) \* [Complete\\_SaveResul](#) (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- void [Init\\_debut\\_calcul](#) ()

### Attributs publics

- [Tenseur1BH](#) [sigma\\_barre\\_BH\\_t](#)
- [Tenseur1BH](#) [sigma\\_barre\\_BH\\_tdt](#)
- double [fonction\\_aide\\_t](#)
- double [fonction\\_aide\\_tdt](#)
- double [wprime\\_t](#)
- double [wprime\\_tdt](#)
- [List\\_io](#)< double >::iterator [ip2](#)
- int [modif](#)
- [List\\_io](#)< [Tenseur1BH](#) > [sigma\\_barre\\_BH\\_R\\_t\\_a\\_tdt](#)
- int [nb\\_coincidence](#)
- [List\\_io](#)< double > [fct\\_aide\\_t\\_a\\_tdt](#)
- [List\\_io](#)< bool > [indic\\_coin](#)
- [List\\_io](#)< [Tenseur1BH](#) > [sigma\\_barre\\_BH\\_R](#)
- [List\\_io](#)< double > [fct\\_aide](#)
- [Tableau](#)< double > [indicateurs\\_resolution](#)
- [Tableau](#)< double > [indicateurs\\_resolution\\_t](#)

## 6.745.1 Documentation des fonctions membres

### 6.745.1.1 Affiche()

```
void Hysteresis1D::SaveResulHysteresis1D::Affiche () const [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.745.1.2 ChBase\_des\_grandeurs()

```
void Hysteresis1D::SaveResulHysteresis1D::ChBase_des_grandeurs (
    const Mat\_pleine & beta,
    const Mat\_pleine & gamma ) [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.745.1.3 Complete\_SaveResul()

```
SaveResul * Hysteresis1D::SaveResulHysteresis1D::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi\_comp\_abstraite * loi ) [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.745.1.4 Ecriture\_base\_info()

```
void Hysteresis1D::SaveResulHysteresis1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.745.1.5 Lecture\_base\_info()

```
void Hysteresis1D::SaveResulHysteresis1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.745.1.6 Nevez\_SaveResul()

```
SaveResul * Hysteresis1D::SaveResulHysteresis1D::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.745.1.7 operator=()

```
Loi_comp_abstraite::SaveResul & Hysteresis1D::SaveResulHysteresis1D::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.745.1.8 TdtversT()

```
void Hysteresis1D::SaveResulHysteresis1D::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.745.1.9 TversTdt()

```
void Hysteresis1D::SaveResulHysteresis1D::TversTdt ( ) [virtual]
```

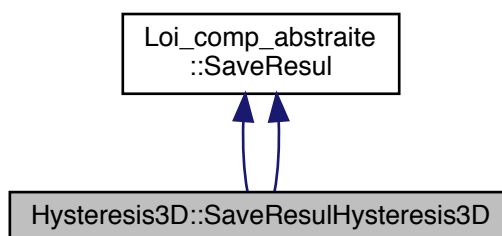
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

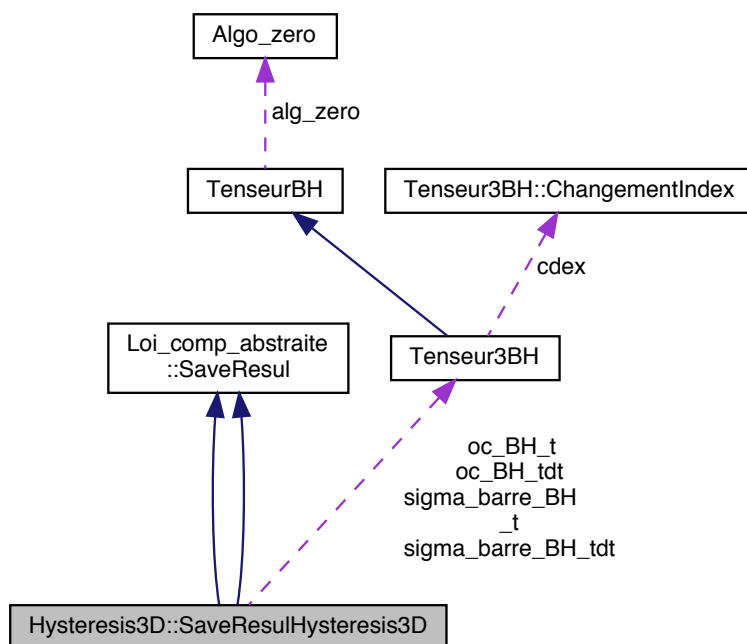
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis1D.cc

## 6.746 Référence de la classe Hysteresis3D::SaveResulHysteresis3D

Graphe d'héritage de Hysteresis3D::SaveResulHysteresis3D:



Grappe de collaboration de Hysteresis3D::SaveResulHysteresis3D:



## Fonctions membres publiques

- **SaveResulHysteresis3D** (const [SaveResulHysteresis3D](#) &sav)
- [SaveResul](#) \* [Nevez\\_SaveResul](#) () const
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [TdtversT](#) ()
- void [TversTdt](#) ()
- void [Init\\_debut\\_calcul](#) ()
- **SaveResulHysteresis3D** (const [SaveResulHysteresis3D](#) &sav)
- [SaveResul](#) \* [Nevez\\_SaveResul](#) () const
- virtual [SaveResul](#) & [operator=](#) (const [SaveResul](#) &a)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [TdtversT](#) ()
- void [TversTdt](#) ()
- void [Affiche](#) () const
- virtual void [ChBase\\_des\\_grandeurs](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- [SaveResul](#) \* [Complete\\_SaveResul](#) (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- void [Init\\_debut\\_calcul](#) ()
- void [Verif\\_centre\\_reference](#) (const int &permet\_affichage)

## Attributs publics

- [Tenseur3BH](#) [sigma\\_barre\\_BH\\_t](#)
- [Tenseur3BH](#) [sigma\\_barre\\_BH\\_tdt](#)
- double [fonction\\_aide\\_t](#)
- double [fonction\\_aide\\_tdt](#)
- int [wBase\\_t](#)
- int [wBase\\_tdt](#)

- [List\\_io](#)< double >::iterator **ip2**
- int **modif**
- [List\\_io](#)< [Tenseur3BH](#) > **sigma\_barre\_BH\_R\_t\_a\_tdt**
- int **nb\_coincidence**
- [List\\_io](#)< double > **fct\_aide\_t\_a\_tdt**
- [List\\_io](#)< bool > **indic\_coin**
- [List\\_io](#)< [Tenseur3BH](#) > **sigma\_barre\_BH\_R**
- [List\\_io](#)< double > **fct\_aide**
- [Tenseur3BH](#) **oc\_BH\_t**
- [Tenseur3BH](#) **oc\_BH\_tdt**
- double **def\_equi\_at**
- double **def\_equi\_atdt**
- [List\\_io](#)< [Tenseur3BH](#) > **oc\_BH\_t\_a\_tdt**
- [List\\_io](#)< double > **def\_equi\_t\_a\_tdt**
- [List\\_io](#)< [Tenseur3BH](#) > **oc\_BH\_R**
- [List\\_io](#)< double > **def\_equi**
- [List\\_io](#)< [Tenseur3BH](#) > **sigma\_barre\_BH\_R\_atrans**
- [List\\_io](#)< [Tenseur3BH](#) > **oc\_BH\_R\_atrans**
- [Tableau](#)< double > **indicateurs\_resolution**
- [Tableau](#)< double > **indicateurs\_resolution\_t**

## 6.746.1 Documentation des fonctions membres

### 6.746.1.1 Affiche()

void Hysteresis3D::SaveResulHysteresis3D::Affiche ( ) const [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.746.1.2 ChBase\_des\_grandeurs()

void Hysteresis3D::SaveResulHysteresis3D::ChBase\_des\_grandeurs (   
     const [Mat\\_pleine](#) & *beta*,  
     const [Mat\\_pleine](#) & *gamma* ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.746.1.3 Complete\_SaveResul()

[SaveResul](#) \* Hysteresis3D::SaveResulHysteresis3D::Complete\_SaveResul (   
     const [BlocGen](#) & *bloc*,  
     const [Tableau](#)< [Coordonnee](#) > & *tab\_coor*,  
     const [Loi\\_comp\\_abstraite](#) \* *loi* ) [inline], [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.746.1.4 Ecriture\_base\_info() [1/2]

void Hysteresis3D::SaveResulHysteresis3D::Ecriture\_base\_info (   
     ofstream & *sort*,  
     const int *cas* ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.746.1.5 Ecriture\_base\_info() [2/2]

void Hysteresis3D::SaveResulHysteresis3D::Ecriture\_base\_info (   
     ofstream & *sort*,  
     const int *cas* ) [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.6 Lecture\_base\_info()** [1/2]

```
void Hysteresis3D::SaveResulHysteresis3D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.7 Lecture\_base\_info()** [2/2]

```
void Hysteresis3D::SaveResulHysteresis3D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.8 Nevez\_SaveResul()** [1/2]

```
SaveResul * Hysteresis3D::SaveResulHysteresis3D::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.9 Nevez\_SaveResul()** [2/2]

```
SaveResul * Hysteresis3D::SaveResulHysteresis3D::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.10 operator=()**

```
Loi_comp_abstraite::SaveResul & Hysteresis3D::SaveResulHysteresis3D::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.11 TdtversT()** [1/2]

```
void Hysteresis3D::SaveResulHysteresis3D::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.12 TdtversT()** [2/2]

```
void Hysteresis3D::SaveResulHysteresis3D::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.13 TversTdt()** [1/2]

```
void Hysteresis3D::SaveResulHysteresis3D::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.746.1.14 TversTdt()** [2/2]

```
void Hysteresis3D::SaveResulHysteresis3D::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

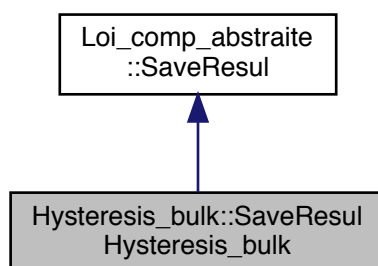
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Copie\_de\_Hysteresis3↔D.h

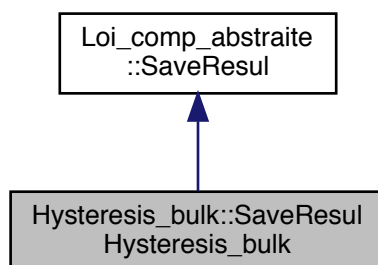
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis3D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Copie_de_Hysteresis3D.cc`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis3D.cc`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis3D_2.cc`

## 6.747 Référence de la classe `Hysteresis_bulk::SaveResulHysteresis_bulk`

Graphe d'héritage de `Hysteresis_bulk::SaveResulHysteresis_bulk`:



Graphe de collaboration de `Hysteresis_bulk::SaveResulHysteresis_bulk`:



### Fonctions membres publiques

- `SaveResulHysteresis_bulk` (const `SaveResulHysteresis_bulk` &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)

- `SaveResul * Complete_SaveResul (const BlocGen &bloc, const Tableau< Coordonnee > &tab_coor, const Loi_comp_abstraite *loi)`
- `void Init_debut_calcul ()`
- `const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * Map_type_quelconque () const`

### Attributs publics

- `double MPr_t`
- `double MPr_tdt`
- `double fonction_aide_t`
- `double fonction_aide_tdt`
- `double wprime_t`
- `double wprime_tdt`
- `List_io< double >::iterator ip2`
- `int modif`
- `List_io< double > MPr_R_t_a_tdt`
- `int nb_coincidence`
- `List_io< double > fct_aide_t_a_tdt`
- `List_io< int > indic_coin`
- `List_io< double > MPr_R`
- `List_io< double > fct_aide`
- `Tableau< double > indicateurs_resolution`
- `Tableau< double > indicateurs_resolution_t`
- `map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > map_type_↔ quelconque`

## 6.747.1 Documentation des fonctions membres

### 6.747.1.1 Affiche()

`void Hysteresis_bulk::SaveResulHysteresis_bulk::Affiche ( ) const [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.747.1.2 ChBase\_des\_grandeurs()

`virtual void Hysteresis_bulk::SaveResulHysteresis_bulk::ChBase_des_grandeurs ( const Mat_pleine & beta, const Mat_pleine & gamma ) [inline], [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.747.1.3 Complete\_SaveResul()

`SaveResul * Hysteresis_bulk::SaveResulHysteresis_bulk::Complete_SaveResul ( const BlocGen & bloc, const Tableau< Coordonnee > & tab_coor, const Loi_comp_abstraite * loi ) [inline], [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.747.1.4 Ecriture\_base\_info()

`void Hysteresis_bulk::SaveResulHysteresis_bulk::Ecriture_base_info ( ofstream & sort, const int cas ) [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).



### 6.747.1.5 Lecture\_base\_info()

```
void Hysteresis_bulk::SaveResulHysteresis_bulk::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.747.1.6 Map\_type\_quelconque()

```
const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * Hysteresis←
_bulk::SaveResulHysteresis_bulk::Map_type_quelconque ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.747.1.7 Nevez\_SaveResul()

```
SaveResul * Hysteresis_bulk::SaveResulHysteresis_bulk::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.747.1.8 operator=()

```
Loi_comp_abstraite::SaveResul & Hysteresis_bulk::SaveResulHysteresis_bulk::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.747.1.9 TdtversT()

```
void Hysteresis_bulk::SaveResulHysteresis_bulk::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.747.1.10 TversTdt()

```
void Hysteresis_bulk::SaveResulHysteresis_bulk::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

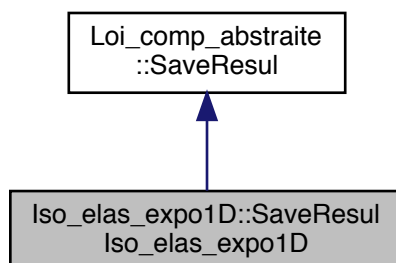
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis\_bulk.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/hysteresis/Hysteresis\_bulk.cc

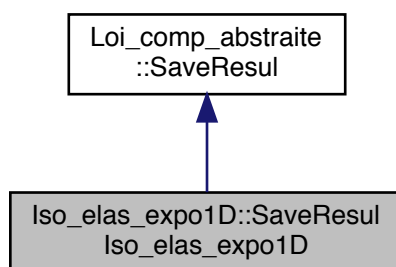
## 6.748 Référence de la classe

### Iso\_elas\_expo1D::SaveResulIso\_elas\_expo1D

Graphe d'héritage de Iso\_elas\_expo1D::SaveResulIso\_elas\_expo1D:



Graphe de collaboration de Iso\_elas\_expo1D::SaveResulIso\_elas\_expo1D:



### Fonctions membres publiques

- `SaveResulIso_elas_expo1D` (const `SaveResulIso_elas_expo1D` &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- virtual `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau`< `Coordonnee` > &tab\_coor, const `Loi_comp_abstraite` \*loi)
- const map< `EnumTypeQuelconque`, `TypeQuelconque`, std::less< `EnumTypeQuelconque` > > \* `Map_type_quelconque` () const

### Attributs publics

- double `E`

- double **E\_t**
- double **nu**
- double **nu\_t**
- map< [EnumTypeQuelconque](#), [TypeQuelconque](#), std::less< [EnumTypeQuelconque](#) > > **map\_type\_↔**  
[quelconque](#)

## 6.748.1 Documentation des fonctions membres

### 6.748.1.1 Affiche()

```
void Iso_elas_expo1D::SaveResulIso_elas_expo1D::Affiche ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.748.1.2 ChBase\_des\_grandeurs()

```
virtual void Iso_elas_expo1D::SaveResulIso_elas_expo1D::ChBase_des_grandeurs (
    const Mat\_pleine & beta,
    const Mat\_pleine & gamma ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.748.1.3 Complete\_SaveResul()

```
virtual SaveResul * Iso_elas_expo1D::SaveResulIso_elas_expo1D::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi\_comp\_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.748.1.4 Ecriture\_base\_info()

```
void Iso_elas_expo1D::SaveResulIso_elas_expo1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.748.1.5 Lecture\_base\_info()

```
void Iso_elas_expo1D::SaveResulIso_elas_expo1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.748.1.6 Map\_type\_quelconque()

```
const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * Iso_elas↔  
_expo1D::SaveResulIso_elas_expo1D::Map_type_quelconque ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.748.1.7 Nevez\_SaveResul()**

```
SaveResul * Iso_elas_exp01D::SaveResulIso_elas_exp01D::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.748.1.8 operator=()**

```
virtual SaveResul & Iso_elas_exp01D::SaveResulIso_elas_exp01D::operator= (
    const SaveResul & a ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.748.1.9 TdtversT()**

```
void Iso_elas_exp01D::SaveResulIso_elas_exp01D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.748.1.10 TversTdt()**

```
void Iso_elas_exp01D::SaveResulIso_elas_exp01D::TversTdt ( ) [inline], [virtual]
```

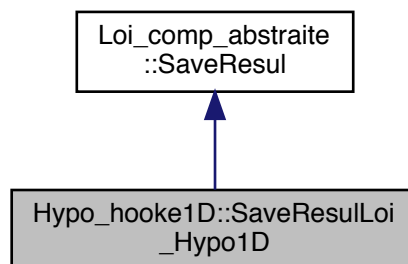
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

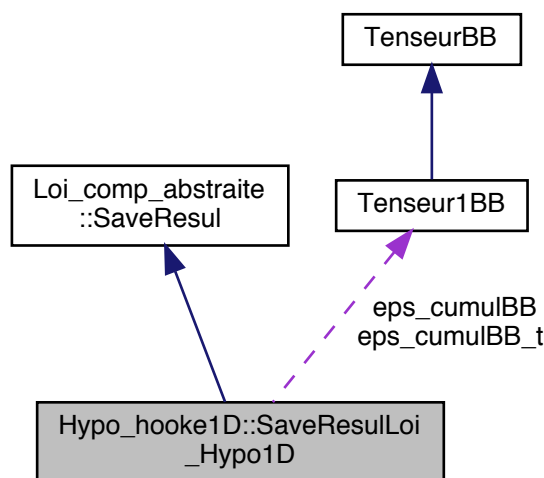
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_nonlinear/Iso\_elas\_exp01↔  
D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_nonlinear/Iso\_elas\_exp01↔  
D.cc

**6.749 Référence de la classe Hypo\_hooke1D::SaveResulLoi\_Hypo1D**

Grphe d'héritage de Hypo\_hooke1D::SaveResulLoi\_Hypo1D:



Graphe de collaboration de Hypo\_hooke1D::SaveResulLoi\_Hypo1D:



## Fonctions membres publiques

- `SaveResulLoi_Hypo1D` (`SaveResul *`)
- `SaveResulLoi_Hypo1D` (`const SaveResulLoi_Hypo1D &sav`)
- `SaveResul * Nevez_SaveResul` (`() const`)
- `virtual SaveResul & operator=` (`const SaveResul &a`)
- `void Lecture_base_info` (`ifstream &ent, const int cas`)
- `void Ecriture_base_info` (`ofstream &sort, const int cas`)
- `void TdtversT` (`()`)
- `void TversTdt` (`()`)
- `void Affiche` (`() const`)
- `virtual void ChBase_des_grandeurs` (`const Mat_pleine &beta, const Mat_pleine &gamma`)
- `virtual SaveResul * Complete_SaveResul` (`const BlocGen &bloc, const Tableau< Coordonnee > &tab_coor, const Loi_comp_abstraite *loi`)

## Attributs publics

- `double Kc`
- `double Kc_t`
- `double f`
- `double f_t`
- `double eps33`
- `double eps22`
- `double eps33_t`
- `double eps22_t`
- `Tenseur1BB eps_cumulBB`
- `Tenseur1BB eps_cumulBB_t`

### 6.749.1 Documentation des fonctions membres

#### 6.749.1.1 Affiche()

`void Hypo_hooke1D::SaveResulLoi_Hypo1D::Affiche ( ) const [inline], [virtual]`  
 Implémente `Loi_comp_abstraite::SaveResul`.

**6.749.1.2 ChBase\_des\_grandeurs()**

```
void Hypo_hooke1D::SaveResulLoi_Hypo1D::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.749.1.3 Complete\_SaveResul()**

```
virtual SaveResul * Hypo_hooke1D::SaveResulLoi_Hypo1D::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.749.1.4 Ecriture\_base\_info()**

```
void Hypo_hooke1D::SaveResulLoi_Hypo1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.749.1.5 Lecture\_base\_info()**

```
void Hypo_hooke1D::SaveResulLoi_Hypo1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.749.1.6 Nevez\_SaveResul()**

```
SaveResul * Hypo_hooke1D::SaveResulLoi_Hypo1D::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.749.1.7 operator=()**

```
virtual SaveResul & Hypo_hooke1D::SaveResulLoi_Hypo1D::operator= (
    const SaveResul & a ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.749.1.8 TdtversT()**

```
void Hypo_hooke1D::SaveResulLoi_Hypo1D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.749.1.9 TversTdt()**

```
void Hypo_hooke1D::SaveResulLoi_Hypo1D::TversTdt ( ) [inline], [virtual]
```

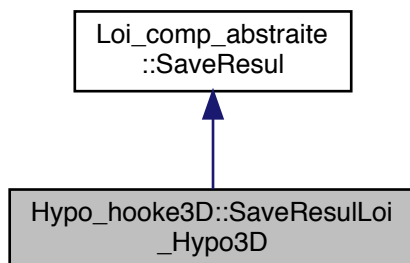
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

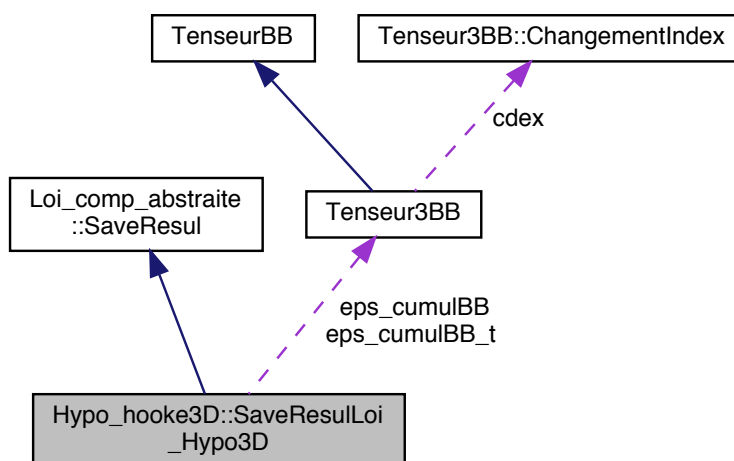
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo\_elastique/Hypo\_hooke1D.cc

## 6.750 Référence de la classe Hypo\_hooke3D::SaveResulLoi\_Hypo3D

Graphe d'héritage de Hypo\_hooke3D::SaveResulLoi\_Hypo3D:



Graphe de collaboration de Hypo\_hooke3D::SaveResulLoi\_Hypo3D:



### Fonctions membres publiques

- `SaveResulLoi_Hypo3D (SaveResul *l_des_SaveResul)`
- `SaveResulLoi_Hypo3D (const SaveResulLoi_Hypo3D &sav)`
- `SaveResul * Nevez_SaveResul () const`
- `virtual SaveResul & operator= (const SaveResul &a)`
- `void Lecture_base_info (ifstream &ent, const int cas)`
- `void Ecriture_base_info (ofstream &sort, const int cas)`
- `void TdtversT ()`
- `void TversTdt ()`
- `void Affiche () const`
- `virtual void ChBase_des_grandeurs (const Mat_pleine &beta, const Mat_pleine &gamma)`
- `virtual SaveResul * Complete_SaveResul (const BlocGen &bloc, const Tableau< Coordonnee > &tab_coor, const Loi_comp_abstraite *loi)`

## Attributs publics

- double **Kc**
- double **Kc\_t**
- double **mu**
- double **mu\_t**
- [Tenseur3BB](#) **eps\_cumulBB**
- [Tenseur3BB](#) **eps\_cumulBB\_t**

## 6.750.1 Documentation des fonctions membres

### 6.750.1.1 Affiche()

`void Hypo_hooke3D::SaveResulLoi_Hypo3D::Affiche ( ) const [inline], [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.750.1.2 ChBase\_des\_grandeurs()

`void Hypo_hooke3D::SaveResulLoi_Hypo3D::ChBase_des_grandeurs (`  
     `const Mat\_pleine & beta,`  
     `const Mat\_pleine & gamma ) [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.750.1.3 Complete\_SaveResul()

`virtual SaveResul * Hypo_hooke3D::SaveResulLoi_Hypo3D::Complete_SaveResul (`  
     `const BlocGen & bloc,`  
     `const Tableau< Coordonnee > & tab_coor,`  
     `const Loi\_comp\_abstraite * loi ) [inline], [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.750.1.4 Ecriture\_base\_info()

`void Hypo_hooke3D::SaveResulLoi_Hypo3D::Ecriture_base_info (`  
     `ofstream & sort,`  
     `const int cas ) [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.750.1.5 Lecture\_base\_info()

`void Hypo_hooke3D::SaveResulLoi_Hypo3D::Lecture_base_info (`  
     `ifstream & ent,`  
     `const int cas ) [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.750.1.6 Nevez\_SaveResul()

`SaveResul * Hypo_hooke3D::SaveResulLoi_Hypo3D::Nevez_SaveResul ( ) const [inline], [virtual]`  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).



### 6.750.1.7 `operator=()`

```
virtual SaveResul & Hypo_hooke3D::SaveResulLoi_Hypo3D::operator= (  
    const SaveResul & a ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.750.1.8 `TdtversT()`

```
void Hypo_hooke3D::SaveResulLoi_Hypo3D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.750.1.9 `TversTdt()`

```
void Hypo_hooke3D::SaveResulLoi_Hypo3D::TversTdt ( ) [inline], [virtual]
```

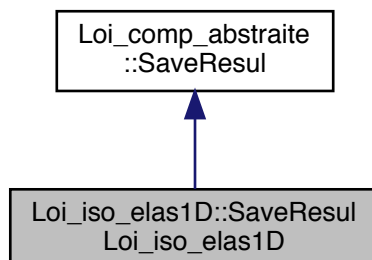
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

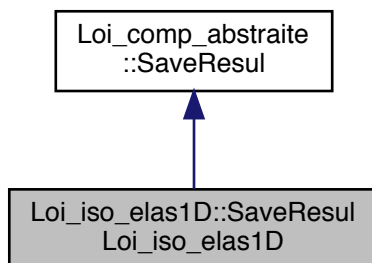
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo_elastique/Hypo_hooke3D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hypo_elastique/Hypo_hooke3D.cc`

## 6.751 Référence de la classe `Loi_iso_elas1D::SaveResulLoi_iso_elas1D`

Graphes d'héritage de `Loi_iso_elas1D::SaveResulLoi_iso_elas1D`:



Graphe de collaboration de `Loi_iso_elas1D::SaveResulLoi_iso_elas1D`:



## Fonctions membres publiques

- `SaveResulLoi_iso_elas1D` (const [SaveResulLoi\\_iso\\_elas1D](#) &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const [SaveResul](#) &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- virtual `SaveResul * Complete_SaveResul` (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- const map< [EnumTypeQuelconque](#), [TypeQuelconque](#), std::less< [EnumTypeQuelconque](#) > > \* `Map_type_quelconque` () const

## Attributs publics

- double `E`
- double `E_t`
- double `nu`
- double `nu_t`
- map< [EnumTypeQuelconque](#), [TypeQuelconque](#), std::less< [EnumTypeQuelconque](#) > > `map_type_↔ quelconque`

## 6.751.1 Documentation des fonctions membres

### 6.751.1.1 Affiche()

void `Loi_iso_elas1D::SaveResulLoi_iso_elas1D::Affiche` () const [inline], [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.2 ChBase\_des\_grandeurs()

virtual void `Loi_iso_elas1D::SaveResulLoi_iso_elas1D::ChBase_des_grandeurs` (  
     const [Mat\\_pleine](#) & beta,  
     const [Mat\\_pleine](#) & gamma ) [inline], [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.3 Complete\_SaveResul()

```
virtual SaveResul * Loi_iso_elas1D::SaveResulLoi_iso_elas1D::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.4 Ecriture\_base\_info()

```
void Loi_iso_elas1D::SaveResulLoi_iso_elas1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.5 Lecture\_base\_info()

```
void Loi_iso_elas1D::SaveResulLoi_iso_elas1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.6 Map\_type\_quelconque()

```
const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * Loi_iso_↔
elas1D::SaveResulLoi_iso_elas1D::Map_type_quelconque ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.7 Nevez\_SaveResul()

```
SaveResul * Loi_iso_elas1D::SaveResulLoi_iso_elas1D::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.8 operator=()

```
virtual SaveResul & Loi_iso_elas1D::SaveResulLoi_iso_elas1D::operator= (
    const SaveResul & a ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.9 TdtversT()

```
void Loi_iso_elas1D::SaveResulLoi_iso_elas1D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.751.1.10 TversTdt()

```
void Loi_iso_elas1D::SaveResulLoi_iso_elas1D::TversTdt ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

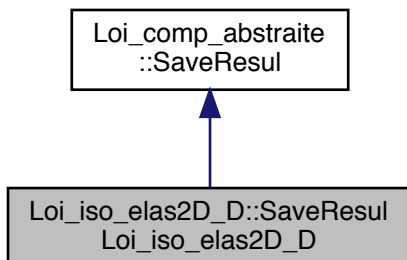
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas1D.cc

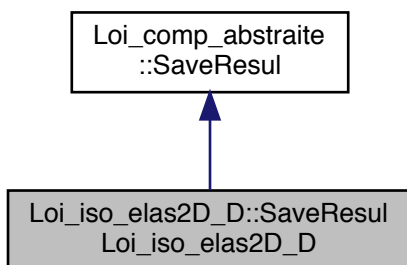
## 6.752 Référence de la classe

### Loi\_iso\_elas2D\_D::SaveResulLoi\_iso\_elas2D\_D

Grphe d'héritage de Loi\_iso\_elas2D\_D::SaveResulLoi\_iso\_elas2D\_D:



Grphe de collaboration de Loi\_iso\_elas2D\_D::SaveResulLoi\_iso\_elas2D\_D:



### Fonctions membres publiques

- `SaveResulLoi_iso_elas2D_D` (const `SaveResulLoi_iso_elas2D_D` &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- virtual `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau`< `Coordonnee` > &tab\_coor, const `Loi_comp_abstraite` \*loi)
- const `map`< `EnumTypeQuelconque`, `TypeQuelconque`, `std::less`< `EnumTypeQuelconque` > > \* `Map_type_quelconque` () const

### Attributs publics

- double `E`

- double `E_t`
- double `nu`
- double `nu_t`
- `map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > map_type_↔ quelconque`

## 6.752.1 Documentation des fonctions membres

### 6.752.1.1 Affiche()

```
void Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::Affiche ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.752.1.2 ChBase\_des\_grandeurs()

```
virtual void Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.752.1.3 Complete\_SaveResul()

```
virtual SaveResul * Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.752.1.4 Ecriture\_base\_info()

```
void Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.752.1.5 Lecture\_base\_info()

```
void Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.752.1.6 Map\_type\_quelconque()

```
const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * Loi_iso_↔
elas2D_D::SaveResulLoi_iso_elas2D_D::Map_type_quelconque ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.752.1.7 Nevez\_SaveResul()**

```
SaveResul * Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.752.1.8 operator=()**

```
virtual SaveResul & Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::operator= (
    const SaveResul & a ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.752.1.9 TdtversT()**

```
void Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.752.1.10 TversTdt()**

```
void Loi_iso_elas2D_D::SaveResulLoi_iso_elas2D_D::TversTdt ( ) [inline], [virtual]
```

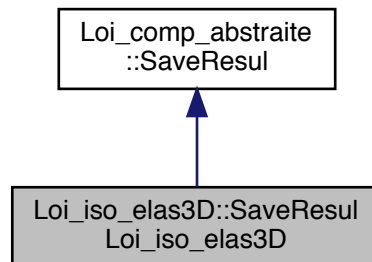
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

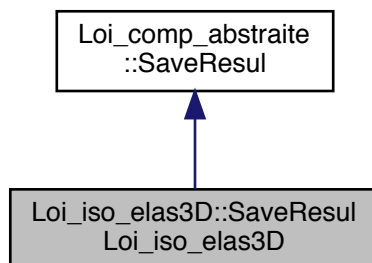
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas2D\_D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas2D\_D.cc

**6.753 Référence de la classe Loi\_iso\_elas3D::SaveResulLoi\_iso\_elas3D**

Graphe d'héritage de Loi\_iso\_elas3D::SaveResulLoi\_iso\_elas3D:



Graphe de collaboration de `Loi_iso_elas3D::SaveResulLoi_iso_elas3D`:



## Fonctions membres publiques

- `SaveResulLoi_iso_elas3D` (const [SaveResulLoi\\_iso\\_elas3D](#) &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const [SaveResul](#) &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- virtual `SaveResul * Complete_SaveResul` (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- const map< [EnumTypeQuelconque](#), [TypeQuelconque](#), std::less< [EnumTypeQuelconque](#) > > \* `Map_type_quelconque` () const

## Attributs publics

- double `E`
- double `E_t`
- double `nu`
- double `nu_t`
- map< [EnumTypeQuelconque](#), [TypeQuelconque](#), std::less< [EnumTypeQuelconque](#) > > `map_type_↔ quelconque`

## 6.753.1 Documentation des fonctions membres

### 6.753.1.1 Affiche()

void `Loi_iso_elas3D::SaveResulLoi_iso_elas3D::Affiche` () const [inline], [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.753.1.2 ChBase\_des\_grandeurs()

virtual void `Loi_iso_elas3D::SaveResulLoi_iso_elas3D::ChBase_des_grandeurs` (  
     const [Mat\\_pleine](#) & beta,  
     const [Mat\\_pleine](#) & gamma ) [inline], [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.753.1.3 Complete\_SaveResul()**

```
virtual SaveResul * Loi_iso_elas3D::SaveResulLoi_iso_elas3D::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.753.1.4 Ecriture\_base\_info()**

```
void Loi_iso_elas3D::SaveResulLoi_iso_elas3D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.753.1.5 Lecture\_base\_info()**

```
void Loi_iso_elas3D::SaveResulLoi_iso_elas3D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.753.1.6 Map\_type\_quelconque()**

```
const map< EnumTypeQuelconque, TypeQuelconque, std::less< EnumTypeQuelconque > > * Loi_iso_↔
elas3D::SaveResulLoi_iso_elas3D::Map_type_quelconque ( ) const [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.753.1.7 Nevez\_SaveResul()**

```
SaveResul * Loi_iso_elas3D::SaveResulLoi_iso_elas3D::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.753.1.8 operator=()**

```
virtual SaveResul & Loi_iso_elas3D::SaveResulLoi_iso_elas3D::operator= (
    const SaveResul & a ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

**6.753.1.9 TdtversT()**

```
void Loi_iso_elas3D::SaveResulLoi_iso_elas3D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

**6.753.1.10 TversTdt()**

```
void Loi_iso_elas3D::SaveResulLoi_iso_elas3D::TversTdt ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

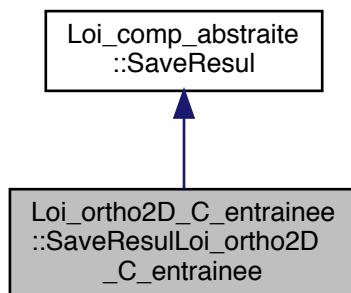
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/iso\_elas\_hooke/Loi\_iso\_elas3D.cc

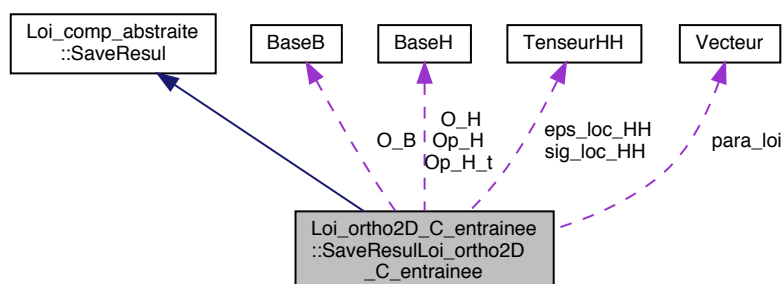


## 6.754 Référence de la classe Loi\_ortho2D\_C\_entrainee::SaveResulLoi\_ortho2D\_C\_entrainee

Grappe d'héritage de Loi\_ortho2D\_C\_entrainee::SaveResulLoi\_ortho2D\_C\_entrainee:



Grappe de collaboration de Loi\_ortho2D\_C\_entrainee::SaveResulLoi\_ortho2D\_C\_entrainee:



### Fonctions membres publiques

- `SaveResulLoi_ortho2D_C_entrainee` (const int type\_transport=0)
- `SaveResulLoi_ortho2D_C_entrainee` (const [SaveResulLoi\\_ortho2D\\_C\\_entrainee](#) &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const [SaveResul](#) &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- `SaveResul * Complete_SaveResul` (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- void `Init_debut_calcul` ()

### Attributs publics

- `BaseB * O_B`

- [BaseH](#) \* [O\\_H](#)
- [BaseH](#) [Op\\_H](#)
- [BaseH](#) [Op\\_H\\_t](#)
- [TenseurHH](#) \* [eps\\_loc\\_HH](#)
- [TenseurHH](#) \* [sig\\_loc\\_HH](#)
- [Vecteur](#) \* [para\\_loi](#)
- double [eps33](#)
- double [eps33\\_t](#)

## 6.754.1 Documentation des fonctions membres

### 6.754.1.1 Affiche()

```
void Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee::Affiche ( ) const [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.754.1.2 ChBase\_des\_grandeurs()

```
void Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee::ChBase_des_grandeurs (
    const Mat\_pleine & beta,
    const Mat\_pleine & gamma ) [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.754.1.3 Complete\_SaveResul()

```
Loi\_comp\_abstraite::SaveResul * Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee::↔
Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi\_comp\_abstraite * loi ) [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.754.1.4 Ecriture\_base\_info()

```
void Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.754.1.5 Lecture\_base\_info()

```
void Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.754.1.6 Nevez\_SaveResul()

```
SaveResul * Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee::Nevez_SaveResul ( )
const [inline], [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.754.1.7 `operator=()`

```
Loi_comp_abstraite::SaveResul & Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee↔  
::operator= (   
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.754.1.8 `TdtversT()`

```
void Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.754.1.9 `TversTdt()`

```
void Loi_ortho2D_C_entrainee::SaveResulLoi_ortho2D_C_entrainee::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

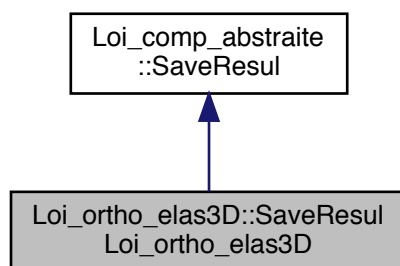
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi_ortho2D_C_↔entrainee.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi_ortho2D_C_↔entrainee.cc`

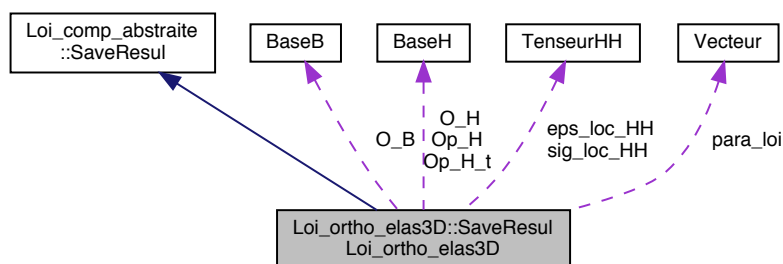
## 6.755 Référence de la classe

### `Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D`

Graphes d'héritage de `Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D`:



Graphe de collaboration de `Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D`:



## Fonctions membres publiques

- `SaveResulLoi_ortho_elas3D` (const int type\_transport=0)
- `SaveResulLoi_ortho_elas3D` (const `SaveResulLoi_ortho_elas3D` &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau`< `Coordonnee` > &tab\_coor, const `Loi_comp_abstraite` \*loi)
- void `Init_debut_calcul` ()

## Attributs publics

- `BaseB` \* `O_B`
- `BaseH` \* `O_H`
- `BaseH` `Op_H`
- `BaseH` `Op_H_t`
- `TenseurHH` \* `eps_loc_HH`
- `TenseurHH` \* `sig_loc_HH`
- `Vecteur` \* `para_loi`

### 6.755.1 Documentation des fonctions membres

#### 6.755.1.1 Affiche()

```
void Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::Affiche ( ) const [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

#### 6.755.1.2 ChBase\_des\_grandeurs()

```
void Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
Implémente Loi\_comp\_abstraite::SaveResul.
```

### 6.755.1.3 Complete\_SaveResul()

```
Loi_comp_abstraite::SaveResul * Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.755.1.4 Ecriture\_base\_info()

```
void Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.755.1.5 Lecture\_base\_info()

```
void Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.755.1.6 Nevez\_SaveResul()

```
SaveResul * Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.755.1.7 operator=()

```
Loi_comp_abstraite::SaveResul & Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.755.1.8 TdtversT()

```
void Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.755.1.9 TversTdt()

```
void Loi_ortho_elas3D::SaveResulLoi_ortho_elas3D::TversTdt ( ) [virtual]
```

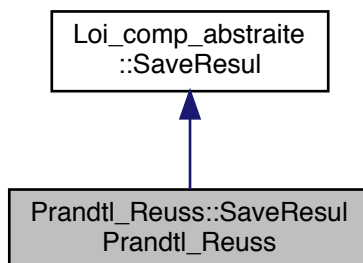
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

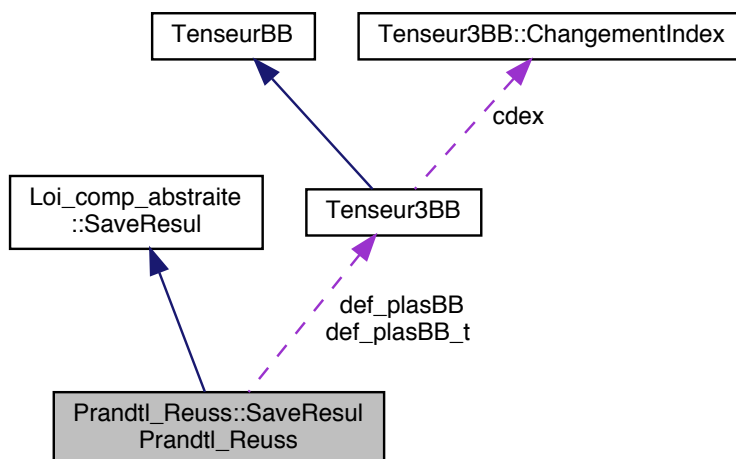
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi_ortho3D_entraine.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Loi_ortho3D_entraine.cc`

## 6.756 Référence de la classe Prandtl\_Reuss::SaveResulPrandtl\_Reuss

Graphe d'héritage de Prandtl\_Reuss::SaveResulPrandtl\_Reuss:



Graphe de collaboration de Prandtl\_Reuss::SaveResulPrandtl\_Reuss:



### Fonctions membres publiques

- **SaveResulPrandtl\_Reuss** (const [SaveResulPrandtl\\_Reuss](#) &sav)
- [SaveResul](#) \* [Nevez\\_SaveResul](#) () const
- virtual [SaveResul](#) & [operator=](#) (const [SaveResul](#) &a)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [TdtversT](#) ()
- void [TversTdt](#) ()
- void [Affiche](#) () const
- virtual void [ChBase\\_des\\_grandeurs](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- [SaveResul](#) \* [Complete\\_SaveResul](#) (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- double [Deformation\\_plastique](#) ()

## Attributs publics

- double `epsilon_barre`
- [Tenseur3BB](#) `def_plasBB`
- double `epsilon_barre_t`
- [Tenseur3BB](#) `def_plasBB_t`

## 6.756.1 Documentation des fonctions membres

### 6.756.1.1 Affiche()

```
void Prandtl_Reuss::SaveResulPrandtl_Reuss::Affiche ( ) const [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.2 ChBase\_des\_grandeurs()

```
virtual void Prandtl_Reuss::SaveResulPrandtl_Reuss::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.3 Complete\_SaveResul()

```
SaveResul * Prandtl_Reuss::SaveResulPrandtl_Reuss::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.4 Deformation\_plastique()

```
double Prandtl_Reuss::SaveResulPrandtl_Reuss::Deformation_plastique ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.5 Ecriture\_base\_info()

```
void Prandtl_Reuss::SaveResulPrandtl_Reuss::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.6 Lecture\_base\_info()

```
void Prandtl_Reuss::SaveResulPrandtl_Reuss::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.7 Nevez\_SaveResul()

```
SaveResul * Prandtl_Reuss::SaveResulPrandtl_Reuss::Nevez_SaveResul ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.8 operator=()

```
Loi_comp_abstraite::SaveResul & Prandtl_Reuss::SaveResulPrandtl_Reuss::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.9 TdtversT()

```
void Prandtl_Reuss::SaveResulPrandtl_Reuss::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.756.1.10 TversTdt()

```
void Prandtl_Reuss::SaveResulPrandtl_Reuss::TversTdt ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

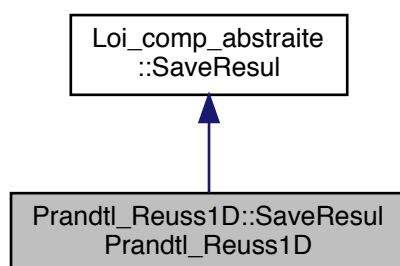
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss.cc

## 6.757 Référence de la classe

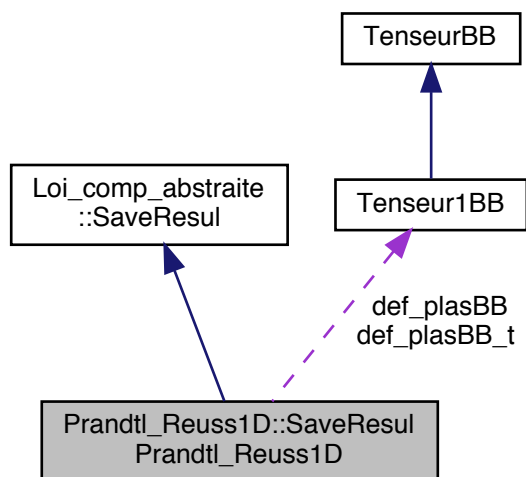
### Prandtl\_Reuss1D::SaveResulPrandtl\_Reuss1D

Graphpe d'héritage de Prandtl\_Reuss1D::SaveResulPrandtl\_Reuss1D:





Grphe de collaboration de Prandtl\_Reuss1D::SaveResulPrandtl\_Reuss1D:



## Fonctions membres publiques

- `SaveResulPrandtl_Reuss1D` (const `SaveResulPrandtl_Reuss1D` &sav)
- `SaveResul * Nevez_SaveResul` () const
- virtual `SaveResul & operator=` (const `SaveResul` &a)
- void `Lecture_base_info` (ifstream &ent, const int cas)
- void `Ecriture_base_info` (ofstream &sort, const int cas)
- void `TdtversT` ()
- void `TversTdt` ()
- void `Affiche` () const
- virtual void `ChBase_des_grandeurs` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- `SaveResul * Complete_SaveResul` (const `BlocGen` &bloc, const `Tableau< Coordonnee >` &tab\_coor, const `Loi_comp_abstraite *loi`)
- double `Deformation_plastique` ()

## Attributs publics

- double `epsilon_barre`
- `Tenseur1BB def_plasBB`
- double `epsilon_barre_t`
- `Tenseur1BB def_plasBB_t`

### 6.757.1 Documentation des fonctions membres

#### 6.757.1.1 Affiche()

```
void Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::Affiche ( ) const [virtual]
```

Implémente `Loi_comp_abstraite::SaveResul`.

### 6.757.1.2 ChBase\_des\_grandeurs()

```
virtual void Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.757.1.3 Complete\_SaveResul()

```
SaveResul * Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.757.1.4 Deformation\_plastique()

```
double Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::Deformation_plastique ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.757.1.5 Ecriture\_base\_info()

```
void Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.757.1.6 Lecture\_base\_info()

```
void Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.757.1.7 Nevez\_SaveResul()

```
SaveResul * Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.757.1.8 operator=()

```
Loi_comp_abstraite::SaveResul & Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.757.1.9 TdtversT()

```
void Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

## 6.757.1.10 TversTdt()

```
void Prandtl_Reuss1D::SaveResulPrandtl_Reuss1D::TversTdt ( ) [inline], [virtual]
```

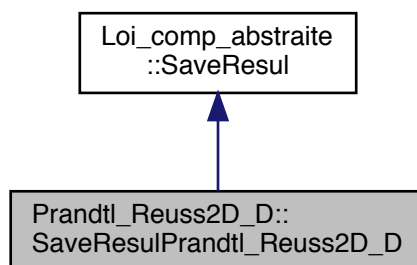
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

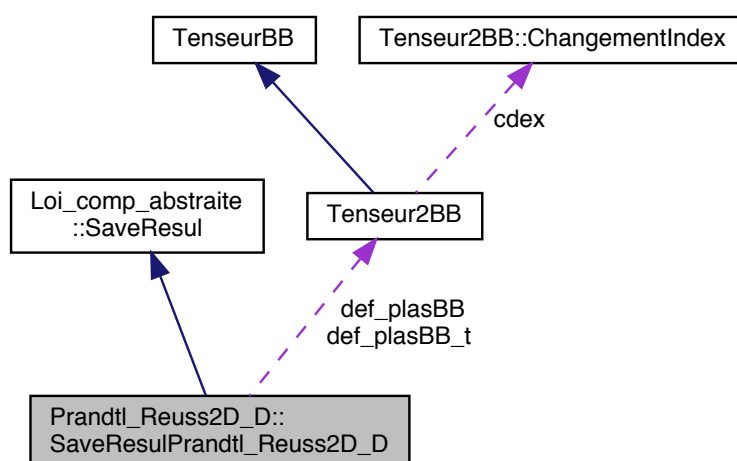
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss1D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl\_Reuss1D.cc

## 6.758 Référence de la classe Prandtl\_Reuss2D\_D::SaveResulPrandtl\_Reuss2D\_D

Graphe d'héritage de Prandtl\_Reuss2D\_D::SaveResulPrandtl\_Reuss2D\_D:



Graphe de collaboration de Prandtl\_Reuss2D\_D::SaveResulPrandtl\_Reuss2D\_D:



### Fonctions membres publiques

- `SaveResulPrandtl_Reuss2D_D` (const [SaveResulPrandtl\\_Reuss2D\\_D](#) &sav)

- [SaveResul](#) \* [Nevez\\_SaveResul](#) () const
- virtual [SaveResul](#) & [operator=](#) (const [SaveResul](#) &a)
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
- void [TdtversT](#) ()
- void [TversTdt](#) ()
- void [Affiche](#) () const
- virtual void [ChBase\\_des\\_grandeurs](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- [SaveResul](#) \* [Complete\\_SaveResul](#) (const [BlocGen](#) &bloc, const [Tableau](#)< [Coordonnee](#) > &tab\_coor, const [Loi\\_comp\\_abstraite](#) \*loi)
- double [Deformation\\_plastique](#) ()

## Attributs publics

- double [epsilon\\_barre](#)
- [Tenseur2BB](#) [def\\_plasBB](#)
- double [epsilon\\_barre\\_t](#)
- [Tenseur2BB](#) [def\\_plasBB\\_t](#)

## 6.758.1 Documentation des fonctions membres

### 6.758.1.1 Affiche()

void [Prandtl\\_Reuss2D\\_D::SaveResulPrandtl\\_Reuss2D\\_D::Affiche](#) () const [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.2 ChBase\_des\_grandeurs()

virtual void [Prandtl\\_Reuss2D\\_D::SaveResulPrandtl\\_Reuss2D\\_D::ChBase\\_des\\_grandeurs](#) (  
     const [Mat\\_pleine](#) & *beta*,  
     const [Mat\\_pleine](#) & *gamma* ) [inline], [virtual]  
 Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.3 Complete\_SaveResul()

[SaveResul](#) \* [Prandtl\\_Reuss2D\\_D::SaveResulPrandtl\\_Reuss2D\\_D::Complete\\_SaveResul](#) (  
     const [BlocGen](#) & *bloc*,  
     const [Tableau](#)< [Coordonnee](#) > & *tab\_coor*,  
     const [Loi\\_comp\\_abstraite](#) \* *loi* ) [inline], [virtual]

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.4 Deformation\_plastique()

double [Prandtl\\_Reuss2D\\_D::SaveResulPrandtl\\_Reuss2D\\_D::Deformation\\_plastique](#) () [inline],  
 [virtual]

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.5 Ecriture\_base\_info()

void [Prandtl\\_Reuss2D\\_D::SaveResulPrandtl\\_Reuss2D\\_D::Ecriture\\_base\\_info](#) (  
     ofstream & *sort*,  
     const int *cas* ) [virtual]

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.6 `Lecture_base_info()`

```
void Prandtl_Reuss2D_D::SaveResulPrandtl_Reuss2D_D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.7 `Nevez_SaveResul()`

```
SaveResul * Prandtl_Reuss2D_D::SaveResulPrandtl_Reuss2D_D::Nevez_SaveResul ( ) const [inline],
[virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.8 `operator=()`

```
Loi_comp_abstraite::SaveResul & Prandtl_Reuss2D_D::SaveResulPrandtl_Reuss2D_D::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.9 `TdtversT()`

```
void Prandtl_Reuss2D_D::SaveResulPrandtl_Reuss2D_D::TdtversT ( ) [inline], [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.758.1.10 `TversTdt()`

```
void Prandtl_Reuss2D_D::SaveResulPrandtl_Reuss2D_D::TversTdt ( ) [inline], [virtual]
```

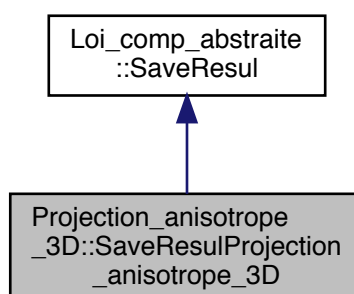
Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

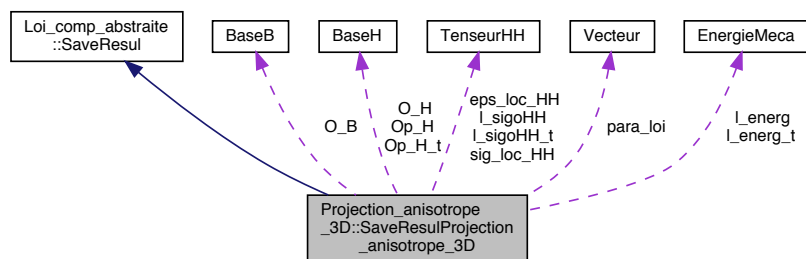
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl_Reuss2D_D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/plasticite/Prandtl_Reuss2D_D.cc`

## 6.759 Référence de la classe `Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D`

Graphe d'héritage de `Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D`:



Graphe de collaboration de `Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D`:



## Fonctions membres publiques

- `SaveResulProjection_anisotrope_3D` (`SaveResul *l_SaveResul`, `TenseurHH *l_siHH`, `TenseurHH *l_siHH_t`, `EnergieMeca l_energ_`, `EnergieMeca l_energ_t`, `bool avec_ponderation`)
- `SaveResulProjection_anisotrope_3D` (`const SaveResulProjection_anisotrope_3D &sav`)
- `SaveResul * Nevez_SaveResul` () `const`
- `virtual SaveResul & operator=` (`const SaveResul &a`)
- `void Lecture_base_info` (`ifstream &ent`, `const int cas`)
- `void Ecriture_base_info` (`ofstream &sort`, `const int cas`)
- `void TdtversT` ()
- `void TversTdt` ()
- `void Affiche` () `const`
- `virtual void ChBase_des_grandeurs` (`const Mat_pleine &beta`, `const Mat_pleine &gamma`)
- `SaveResul * Complete_SaveResul` (`const BlocGen &bloc`, `const Tableau< Coordonnee > &tab_coor`, `const Loi_comp_abstraite *loi`)
- `void Init_debut_calcul` ()

## Attributs publics

- `BaseB * O_B`
- `BaseH * O_H`
- `BaseH Op_H`
- `BaseH Op_H_t`
- `TenseurHH * eps_loc_HH`
- `TenseurHH * sig_loc_HH`
- `Vecteur * para_loi`
- `SaveResul * SaveResul_interne`
- `TenseurHH * l_sigoHH`
- `TenseurHH * l_sigoHH_t`
- `EnergieMeca l_energ`
- `EnergieMeca l_energ_t`
- `double f_ponder`
- `double f_ponder_t`

## 6.759.1 Documentation des fonctions membres

### 6.759.1.1 Affiche()

`void Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D::Affiche ( ) const [virtual]`  
 Implémente `Loi_comp_abstraite::SaveResul`.

### 6.759.1.2 ChBase\_des\_grandeurs()

```
void Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D::ChBase_des_grandeurs (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.759.1.3 Complete\_SaveResul()

```
Loi_comp_abstraite::SaveResul * Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D::↔
Complete_SaveResul (
    const BlocGen & bloc,
    const Tableau< Coordonnee > & tab_coor,
    const Loi_comp_abstraite * loi ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.759.1.4 Ecriture\_base\_info()

```
void Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.759.1.5 Lecture\_base\_info()

```
void Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.759.1.6 Nevez\_SaveResul()

```
SaveResul * Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D::Nevez_SaveResul ( )
const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.759.1.7 operator=()

```
Loi_comp_abstraite::SaveResul & Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D↔
::operator= (
    const SaveResul & a ) [virtual]
```

Implémente [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.759.1.8 TdtversT()

```
void Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D::TdtversT ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

### 6.759.1.9 TversTdt()

```
void Projection_anisotrope_3D::SaveResulProjection_anisotrope_3D::TversTdt ( ) [virtual]
```

Réimplémentée à partir de [Loi\\_comp\\_abstraite::SaveResul](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Projection\\_anisotrope\\_↔3D.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/anisotropie/Projection\\_anisotrope\\_↔3D.cc](#)

## 6.760 Référence de la classe Sect

Conteneur très basique pour les sections.

```
#include <Section.h>
```

### Fonctions membres publiques

- **Sect** (const double ep0, const double ept, const double eptdt)
- **Sect** (const [Sect](#) &a)
- [Sect](#) & **operator=** (const [Sect](#) &a)

### Attributs publics

- double **section0**
- double **section\_t**
- double **section\_tdt**

### Amis

- `istream & operator>>` (istream &ent, [Sect](#) &de)
- `ostream & operator<<` (ostream &sort, const [Sect](#) &de)

#### 6.760.1 Description détaillée

Conteneur très basique pour les sections.

Auteur

Gérard Rio

Version

1.0



Date

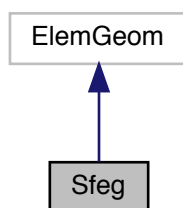
06/03/2023

La documentation de cette classe a été générée à partir du fichier suivant :

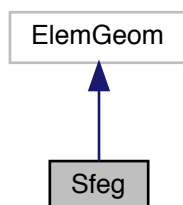
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Section.h

## 6.761 Référence de la classe Sfeg

Graphe d'héritage de Sfeg:



Graphe de collaboration de Sfeg:



### Fonctions membres publiques

- **Sfeg** (int nbi=1, int nbne=3)
- **Sfeg** (Sfeg &a)
- **Vecteur Phi** (Coordonnee &M)
- **Mat\_pleine Dphi** (Coordonnee &M)
- bool **Interieur** (Coordonnee &M)

### Fonctions membres protégées

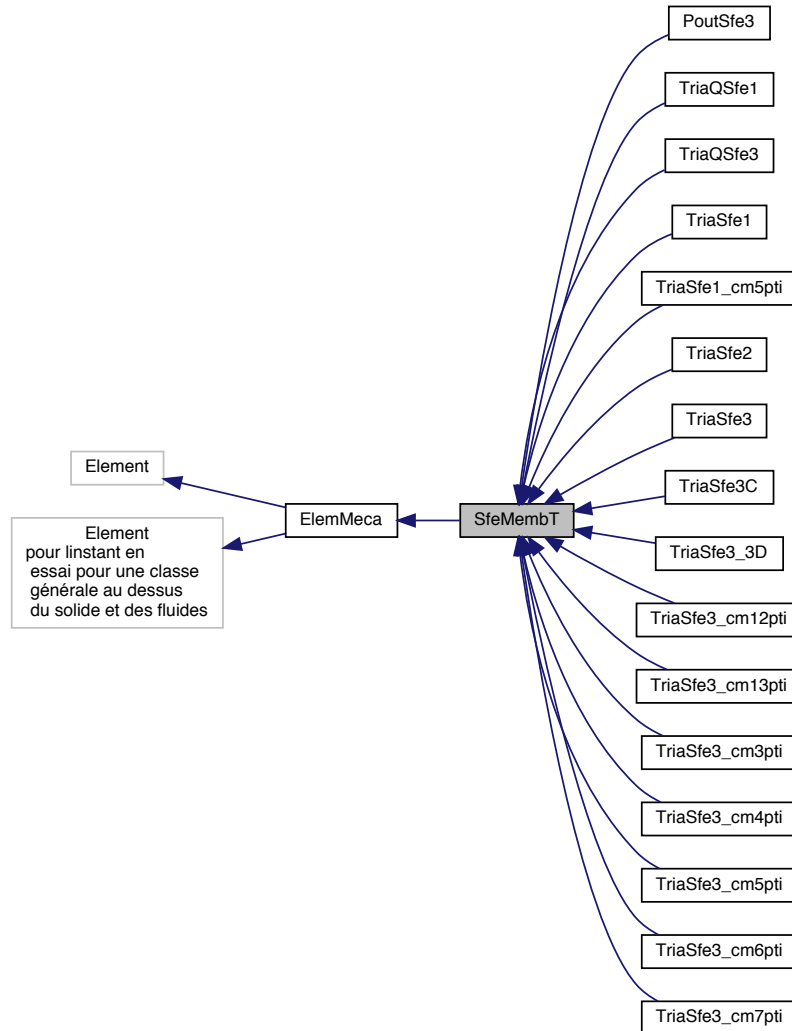
- double & **KSI** (int i)
- double & **ETA** (int i)

La documentation de cette classe a été générée à partir du fichier suivant :

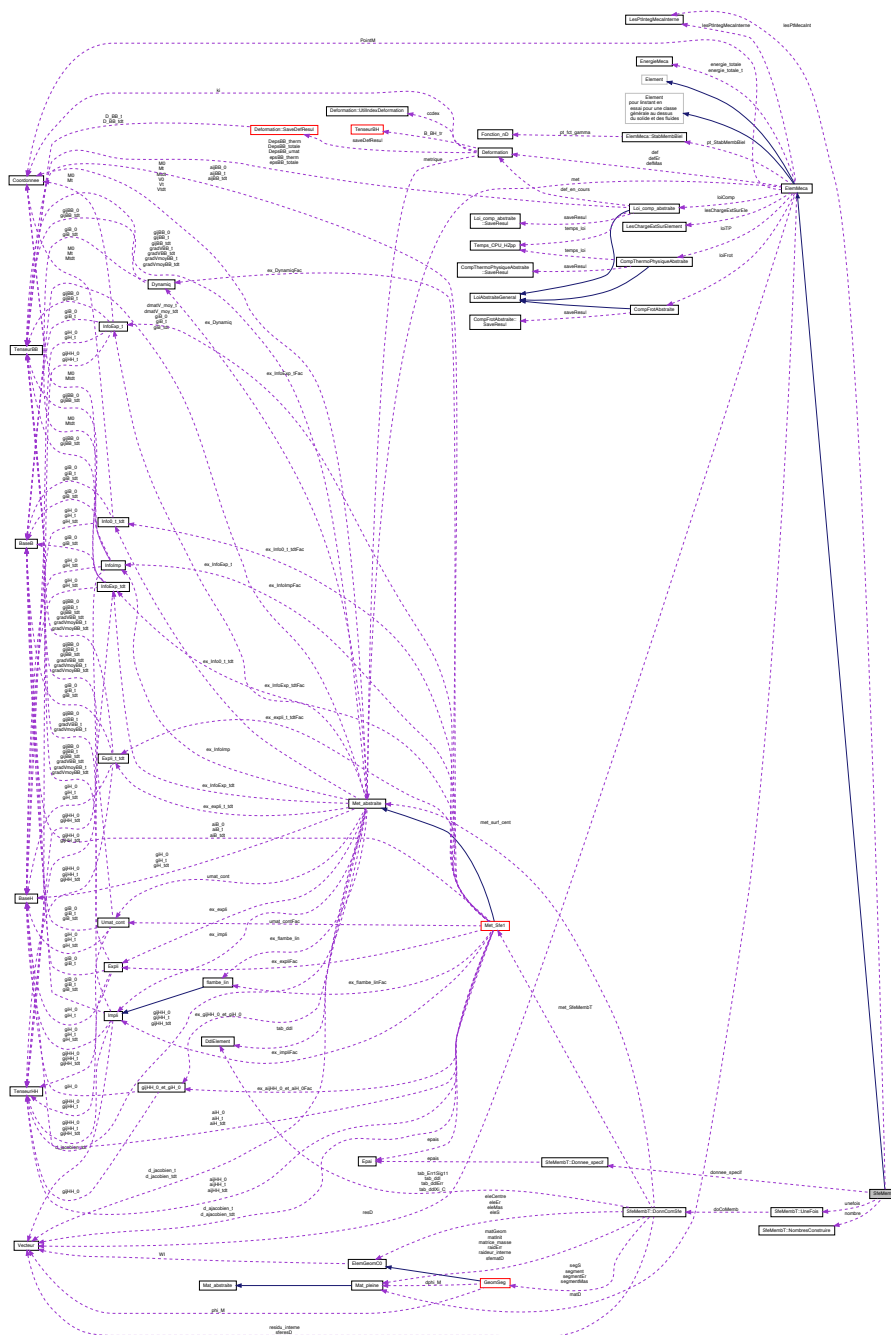
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/Sfeg.h

## 6.762 Référence de la classe SfeMembT

Graphe d'héritage de SfeMembT:



Graphe de collaboration de SfeMembT:



## Classes

- class [DonnComSfe](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)
- class [UneFois](#)

## Fonctions membres publiques

- [SfeMembT](#) (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")
- [SfeMembT](#) (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, const [Tableau](#)< [Noeud](#) \* > &tab, string info="")

- **SfeMembT** (const [SfeMembT](#) &sfe)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- [ElemGeomC0](#) & [ElementGeometrique](#) () const
- const [ElemGeomC0](#) & [ElementGeometrique\\_const](#) () const
- [Coordonnee](#) & **Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- virtual double **Epaisseurs** ([Enum\\_dure](#) enu, const [Coordonnee](#) &)
- virtual double **EpaisseurMoyenne** ([Enum\\_dure](#) enu)
- int **Test\_courbure\_anormale3** ([Enum\\_dure](#) enu, double inf\_normale)
- [List\\_io](#)< [TypeQuelconque](#) > **Les\_types\_particuliers\_internes** (bool absolue) const
- void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &litQ, int iteg)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- int **Nb\_pt\_int\_surf** ()
- int **Nb\_pt\_int\_epai** ()
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- void **DefCondLim** (const [EnumTypeCL](#) &arTypeCL, const [Coordonnee3](#) &vpIa, const int &nb\_ar)
- void **DefCondLim** (const [Tableau](#)< [EnumTypeCL](#) > &arTypeCL, const [Tableau](#)< [Coordonnee3](#) > &vpIa)
- int **TestComple** ()
- [Element](#) \* **Comple** **Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- [Element](#) \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- double **H** ([Enum\\_dure](#) enu=TEMPS\_tdt)
- virtual double **KSI** (int i)=0
- bool **SurfExiste** (int ns) const
- bool **AreteExiste** (int na) const
- const [DdlElement](#) & [TableauDdl](#) () const
- [Element](#)::ResRaid **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- [Element](#)::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- [Element](#)::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#), [List\\_io](#)< [TypeQuelconque](#) > &, int)
- virtual int **CalculNormale\_noeud** ([Enum\\_dure](#) temps, const [Noeud](#) &noe, [Coordonnee](#) &coor)
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- virtual int **NbPtIntegSurface** ([Enum\\_ddl](#)) const
- virtual int **NbPtIntegEpaiss** ([Enum\\_ddl](#)) const
- virtual [ElemGeomC0](#) \* **ElementGeometrieSurface** ([Enum\\_ddl](#)) const
- virtual [ElemGeomC0](#) \* **ElementGeometrieEpaiss** ([Enum\\_ddl](#)) const
- [DdlElement](#) & [Tableau\\_de\\_Sig1](#) () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual const [DeuxCoordonnees](#) & **Boite\_encombre\_element** ([Enum\\_dure](#) temps)
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)

- ResRaid **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- Vecteur **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E\_t** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E\_tdt** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_presUniDir\_E\_t** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_presUniDir\_E\_tdt** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_presUniDir\_I** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrostatique\_E\_t** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_hydrostatique\_E\_tdt** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- virtual int **NbPtInteg** (Enum\_ddl enu) const
- [Tableau](#)< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- void **ConstTabDdl** ()

### Fonctions membres protégées

- [SfeMembT::DonnComSfe](#) \* **Init** ([ElemGeomC0](#) \*eleCentre, [ElemGeomC0](#) \*eleEr, [ElemGeomC0](#) \*eleS, [ElemGeomC0](#) \*eleMas, int type\_calcul\_jacobien=1, [Donnee\\_specif](#) donnee\_specif=[Donnee\\_specif](#)(), bool sans\_init\_noeud=false)
- void **Destruction** ()
- int **Dim\_sig\_eps** () const

### Attributs protégés

- [UneFois](#) \* **unefois**
- [Donnee\\_specif](#) **donnee\_specif**
- [LesPtIntegMecalInterne](#) **lesPtMecalnt**

- [Tableau< EnuTypeCL > \\* areteTypeCL](#)
- [Tableau< Coordonnee3 > \\* vplan](#)
- [Tableau< Noeud \\* > t\\_N\\_centre](#)
- [NombresConstruire \\* nombre](#)

## Membres hérités additionnels

### 6.762.1 Documentation des fonctions membres

#### 6.762.1.1 Active\_ddl\_Sigma()

```
void SfeMembT::Active_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.762.1.2 Active\_premier\_ddl\_Sigma()

```
void SfeMembT::Active_premier_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.762.1.3 CalculNormale\_noeud()

```
int SfeMembT::CalculNormale_noeud (
    Enum_dure temps,
    const Noeud & noe,
    Coordonnee & coor ) [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.762.1.4 ContraintesAbsolues()

```
bool SfeMembT::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

#### 6.762.1.5 Dim\_sig\_eps()

```
int SfeMembT::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.762.1.6 EpaisseurMoyenne()

```
virtual double SfeMembT::EpaisseurMoyenne (
    Enum_dure enu ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

#### 6.762.1.7 Epaisseurs()

```
virtual double SfeMembT::Epaisseurs (
    Enum_dure enu,
    const Coordonnee & ) [inline], [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.762.1.8 ErreurElement()

```
void SfeMembT::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en Réimplémentée à partir de [ElemMeca](#).

### 6.762.1.9 Inactive\_ddl\_Sigma()

```
void SfeMembT::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.762.1.10 LectureContraintes()

```
void SfeMembT::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.762.1.11 Les\_types\_particuliers\_internes()

```
List_io< TypeQuelconque > SfeMembT::Les_types_particuliers_internes (
    bool absolue ) const [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

### 6.762.1.12 Long\_arrete\_mini\_sur\_c()

```
double SfeMembT::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.762.1.13 Plus\_ddl\_Sigma()

```
void SfeMembT::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.762.1.14 Tableau\_de\_Sig1()

```
DdlElement & SfeMembT::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT2.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT3.cc

## 6.763 Référence de la classe Front::Signature\_Front

### Attributs publics

- int numelem

— int **numMail**

La documentation de cette classe a été générée à partir du fichier suivant :

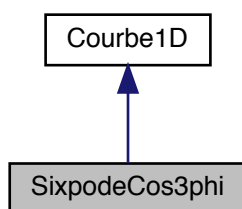
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Geometrie/Frontiere/Front.h

## 6.764 Référence de la classe SixpodeCos3phi

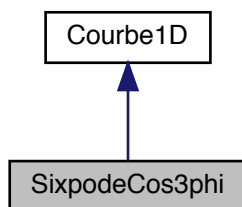
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = 1./(1.+gamma*cos(3*phi)^2)^n$ .

```
#include <SixpodeCos3phi.h>
```

Graphe d'héritage de SixpodeCos3phi:



Graphe de collaboration de SixpodeCos3phi:



### Fonctions membres publiques

- **SixpodeCos3phi** (string nom="")
- **SixpodeCos3phi** (const [SixpodeCos3phi](#) &Co)
- **SixpodeCos3phi** (const [Courbe1D](#) &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Compleet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)



- ramène la valeur*
- [Courbe1D::ValDer Valeur\\_Et\\_derivee](#) (double x)
- ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)
- ramène la dérivée*
- [Courbe1D::ValDer2 Valeur\\_Et\\_der12](#) (double x)
- ramène la valeur et les dérivées première et seconde en paramètre*
- double [Der\\_sec](#) (double x)
- ramène la dérivée seconde*
- [Courbe1D::Valbool Valeur\\_stricte](#) (double x)
- ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène la valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)
- ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant*
- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)
  - lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)
- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)
  - cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)
  - sortie du schemaXML: en fonction de enu

### Attributs protégés

- double **xn**
- double **gamma**

### Membres hérités additionnels

#### 6.764.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = 1./(1.+gamma*cos(3*phi)^2)^n$ .

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x) = 1./(1.+gamma*cos(3*phi)^2)^n$  ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

#### 6.764.2 Documentation des fonctions membres

##### 6.764.2.1 Affiche()

```
void SixpodeCos3phi::Affiche ( ) const [virtual]
affichage de la courbe
Implémente Courbe1D.
```

### 6.764.2.2 Complet\_courbe()

```
bool SixpodeCos3phi::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

### 6.764.2.3 Der\_sec()

```
double SixpodeCos3phi::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde  
Implémente [Courbe1D](#).

### 6.764.2.4 Derivee()

```
double SixpodeCos3phi::Derivee (
    double x ) [virtual]
```

ramène la dérivée  
Implémente [Courbe1D](#).

### 6.764.2.5 Ecriture\_base\_info()

```
void SixpodeCos3phi::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.764.2.6 Info\_commande\_Courbes1D()

```
void SixpodeCos3phi::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande  
Implémente [Courbe1D](#).

### 6.764.2.7 LectDonnParticulieres\_courbes()

```
void SixpodeCos3phi::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.  
Implémente [Courbe1D](#).

### 6.764.2.8 Lecture\_base\_info()

```
void SixpodeCos3phi::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)  
Implémente [Courbe1D](#).

### 6.764.2.9 SchemaXML\_Courbes1D()

```
void SixpodeCos3phi::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

### 6.764.2.10 Valeur()

```
double SixpodeCos3phi::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

### 6.764.2.11 Valeur\_Et\_der12()

```
CourbelD::ValDer2 SixpodeCos3phi::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

### 6.764.2.12 Valeur\_Et\_derivee()

```
CourbelD::ValDer SixpodeCos3phi::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

### 6.764.2.13 Valeur\_Et\_derivee\_stricte()

```
CourbelD::ValDerbool SixpodeCos3phi::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

### 6.764.2.14 Valeur\_stricte()

```
CourbelD::Valbool SixpodeCos3phi::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

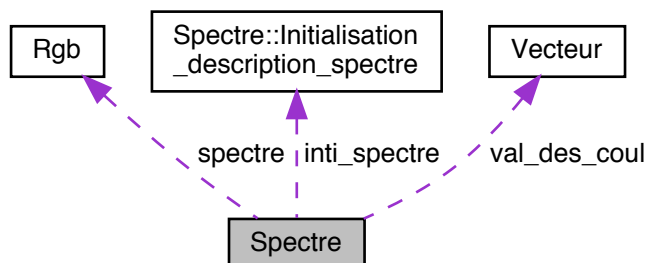
Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/SixpodeCos3phi.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/SixpodeCos3phi.cc

## 6.765 Référence de la classe Spectre

Graphe de collaboration de Spectre:



### Classes

— class [Initialisation\\_description\\_spectre](#)

### Types publics

— enum `EnumType_spectre` {  
**SPECTRE\_STANDART** =1 , **SPECTRE\_ARCENCIEL** , **SPECTRE\_THERMIQUE** , **SPECTRE\_MAXIMUM** ,  
**SPECTRE\_NOIRBLANCN** , **SPECTRE\_GEO** , **SPECTRE\_PHASER** }

### Fonctions membres publiques

— `char * Nom_Type_Spectre` (const `Spectre::EnumType_spectre` id\_TypeSpectre)  
— `EnumType_spectre Id_nom_TypeSpectre` (const `char *nom_TypeSpectre`)  
— `Spectre` (`EnumType_spectre` type\_spect, double val\_min=0., double val\_max=1.)  
— `Spectre` (const [Spectre](#) &spect)  
— void `Change_min_max` (const double &val\_min, const double &val\_max)  
— void `Change_spectre_courant` (`EnumType_spectre` nv\_spectre)  
— double `Valeur_maxi` () const  
— double `Valeur_mini` () const  
— `EnumType_spectre TypeSpectreEnCours` () const  
— int `NbCouleurBaseSpectre` () const  
— `Rgb CouleurVal` (const double &val) const

### Fonctions membres publiques statiques

— static const `Tableau< string > &Description_spectres_disponibles` ()

### Attributs protégés

— `Tableau< Rgb > spectre_courant`  
— `EnumType_spectre type_spectre`  
— `Vecteur val_des_coul`  
— double `delta_val`

### Attributs protégés statiques

— static `Rgb spectre` [7][8]  
— static `Tableau< string > description_spectre`  
— static `Initialisation\_description\_spectre inti_spectre`

## Amis

- class `Initialisation_description_spectre`

## 6.765.1 Documentation des données membres

### 6.765.1.1 spectre

`Rgb` `Spectre::spectre` [static], [protected]

#### Valeur initiale :

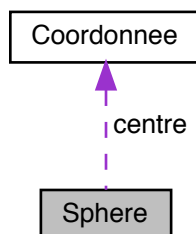
```
= {
    {Bleu, Cyan, Vert, Jaune, Rouge},
    {Violet, Bleu, Cyan, Vert, Jaune, Rouge},
    {Noir, Bleu, Magenta, Rouge, Jaune, Blanc},
    {Noir, Mag, Bleu2, Cyan2, Vert2, Jaune, Rouge, Blanc},
    {Noir, Blanc},
    {Noir, Bleu, Cyan, Vert, Jaune, Rouge, Blanc, Noir},
    {Bleu3, Cyan, Vert, Jaune, Rouge}}
```

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext_visu/spectre.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Ext_visu/spectre.cc`

## 6.766 Référence de la classe Sphere

Graphe de collaboration de Sphere:



## Fonctions membres publiques

- `Sphere` (const `Coordonnee` &B, const double &r)
- `Sphere` (int dim)
- `Sphere` (const `Sphere` &a)
- `Sphere` & `operator=` (const `Sphere` &P)
- const `Coordonnee` & `CentreSphere` () const
- double `RayonSphere` () const
- void `Change_centre` (const `Coordonnee` &B)
- void `Change_rayon` (const double &r)
- void `change_donnees` (const `Coordonnee` &B, const double &r)
- int `Intersection` (const `Droite` &D, `Coordonnee` &M1, `Coordonnee` &M2)
- double `Distance_a_sphere` (const `Coordonnee` &M) const
- bool `Dedans` (const `Coordonnee` &M) const
- `Coordonnee` `Projete` (const `Coordonnee` &M, bool &projection\_ok) const

## Attributs protégés

- [Coordonnee](#) **centre**
- **double** [rayon](#)

## Amis

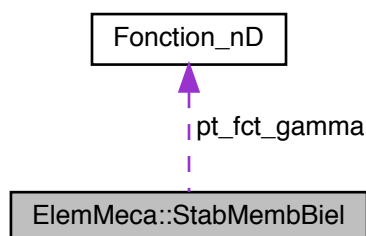
- `istream & operator>>` (`istream &`, [Sphere](#) &)
- `ostream & operator<<` (`ostream &`, `const` [Sphere](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Sphere.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/contact/Sphere.cc`

## 6.767 Référence de la classe ElemMeca::StabMembBiel

Graphe de collaboration de ElemMeca::StabMembBiel:



## Fonctions membres publiques

- **StabMembBiel** (`int` nbnoe)
- **StabMembBiel** (`double` v, `string` \*s)
- **StabMembBiel** (`const` [StabMembBiel](#) &a)
- [StabMembBiel](#) & **operator=** (`const` [StabMembBiel](#) &a)
- [Enum\\_StabMembraneBiel](#) & **Type\_stabMembrane** ()
- `bool` & **Aa\_calculer** ()
- `double` & **Valgamma** ()
- [Fonction\\_nD](#) \* **Pt\_fct\_gamma** ()
- `void` **Change\_pt\_fct\_gamma** ([Fonction\\_nD](#) \*fct)
- `const double` & **Beta** () `const`
- `void` **Change\_beta** (`double` bet)
- `const double` & **F\_mini** () `const`
- `void` **Change\_f\_mini** (`double` f)
- `const double` & **D\_maxi** () `const`
- `void` **Change\_d\_maxi** (`double` d)
- `double` & **FF\_StabMembBiel** (`int` i)
- `double` & **FF\_StabMembBiel\_t** (`int` i)
- `double` & **EE\_StabMembBiel** (`int` i)
- `double` & **EE\_StabMembBiel\_t** (`int` i)
- `double` **EE\_total\_StabMembBiel** () `const`
- `double` **EE\_total\_StabMembBiel\_t** () `const`
- `double` & **Stab\_ref** ()
- `string` \* **Nom\_fctnD** ()
- `void` **TdtversT** ()
- `void` **TversTdt** ()
- `void` **Change\_nb\_pti** (`int` nbnoe)
- `int` **Taille** () `const`

## Attributs protégés

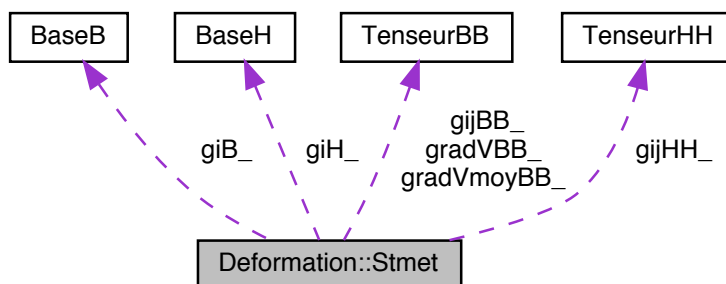
- Enum\_StabMembraneBiel type\_stabMembrane
- bool a\_calculer
- double gamma
- Fonction\_nD \* pt\_fct\_gamma
- double stab\_ref
- double beta
- double f\_mini
- double d\_maxi
- Tableau< double > F\_StabMembBiel
- Tableau< double > F\_StabMembBiel\_t
- Tableau< double > E\_StabMembBiel
- Tableau< double > E\_StabMembBiel\_t
- string \* nom\_fctnD

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemMeca.cc

## 6.768 Référence de la classe Deformation::Stmet

Grphe de collaboration de Deformation::Stmet:



## Fonctions membres publiques

- Stmet (int dim\_base, int nbvec\_base)
- Stmet (const Stmet &ex)
- Stmet & operator= (const Stmet &ex)
- void Affiche ()

## Attributs publics

- BaseB \* giB\_
- BaseH \* giH\_
- TenseurBB \* gijBB\_
- TenseurHH \* gijHH\_
- TenseurBB \* gradVmoyBB\_
- TenseurBB \* gradVBB\_
- double \* jacobien\_

## Amis

- `istream & operator>>` (`istream &`, [Stmet &](#))
- `ostream & operator<<` (`ostream &`, `const` [Stmet &](#))

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Deformation.↔h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation_gene/Deformation↔_stockage.cc`

## 6.769 Référence de la classe `CompThermoPhysiqueAbstraite::StockParalnt`

### Fonctions membres publiques

- `StockParalnt` (`double press`, `double temper`)
- `StockParalnt` (`const` [StockParalnt &a](#))

### Attributs publics

- `double pression`
- `double temperature`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/CompThermoPhysiqueAbstraite.h`

## 6.770 Référence de la classe `String_et_entier`

cas d'un string et un entier

```
#include <Basiques.h>
```

### Fonctions membres publiques

- `String_et_entier` (`const string &nomm`, `int nn`)
- `String_et_entier` (`const` [String\\_et\\_entier &de](#))
- `String_et_entier & operator=` (`const` [String\\_et\\_entier &de](#))
- `istream & LectXML_String_et_entier` (`istream &ent`)
- `ostream & EcritXML_String_et_entier` (`ostream &sort`)
- `bool operator==` (`const` [String\\_et\\_entier &a](#)) `const`
- `bool operator!=` (`const` [String\\_et\\_entier &a](#)) `const`
- `void SchemaXML_String_et_entier` (`ofstream &sort`, `const` [Enum\\_IO\\_XML enu](#)) `const`

### Attributs publics

- `string nom`
- `int n`

### Attributs publics statiques

- `static short int impre_schem_XML =0`

## Amis

- `istream & operator>>` (`istream &ent`, [String\\_et\\_entier &de](#))
- `ostream & operator<<` (`ostream &sort`, `const` [String\\_et\\_entier &de](#))



### 6.770.1 Description détaillée

cas d'un string et un entier

La documentation de cette classe a été générée à partir du fichier suivant :

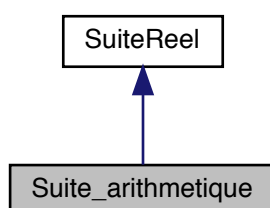
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.cc

## 6.771 Référence de la classe Suite\_arithmetique

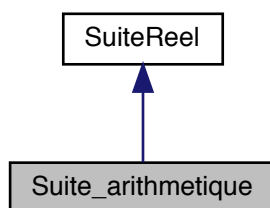
Classe permettant des calculs relatifs à des suites arithmétique:  $u_n = q U_{(n-1)}$

```
#include <Suite_arithmetique.h>
```

Graphe d'héritage de Suite\_arithmetique:



Graphe de collaboration de Suite\_arithmetique:



### Fonctions membres publiques

- Suite\_arithmetique (const Suite\_arithmetique &Co)
- void Affiche () const
- double U\_n (int n)
- double Somme\_Suite (int n)
- void Def\_suite (double amplitude, int n)

### Attributs protégés

- double U\_0
- double p

### 6.771.1 Description détaillée

Classe permettant des calculs relatifs à des suites arithmétique:  $u_n = q U_{(n-1)}$

### 6.771.2 Documentation des fonctions membres

#### 6.771.2.1 Affiche()

```
void Suite_arithmetique::Affiche ( ) const [virtual]
```

Implémente [SuiteReel](#).

#### 6.771.2.2 Def\_suite()

```
void Suite_arithmetique::Def_suite (
    double amplitude,
    int n ) [virtual]
```

Implémente [SuiteReel](#).

#### 6.771.2.3 Somme\_Suite()

```
double Suite_arithmetique::Somme_Suite (
    int n ) [virtual]
```

Implémente [SuiteReel](#).

#### 6.771.2.4 U\_n()

```
double Suite_arithmetique::U_n (
    int n ) [virtual]
```

Implémente [SuiteReel](#).

La documentation de cette classe a été générée à partir du fichier suivant :

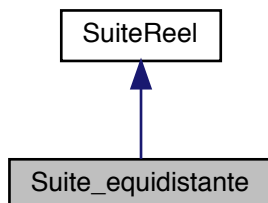
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite\_arithmetique.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite\_arithmetique.cc

## 6.772 Référence de la classe Suite\_equidistante

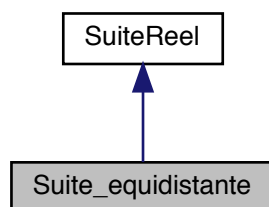
[Suite\\_equidistante](#): Classe permettant des calculs relatifs à des suites equidistante:  $u_n = U_{(n-1)}$

```
#include <Suite_equidistante.h>
```

Graphe d'héritage de Suite\_equidistante:



Graphe de collaboration de Suite\_equidistante:



## Fonctions membres publiques

- **Suite\_equidistante** (const [Suite\\_equidistante](#) &Co)
- void [Affiche](#) () const
- double [U\\_n](#) (int)
- double [Somme\\_Suite](#) (int n)
- void [Def\\_suite](#) (double amplitude, int n)

## Attributs protégés

- double **U\_0**

### 6.772.1 Description détaillée

[Suite\\_equidistante](#): Classe permettant des calculs relatifs à des suites equidistante:  $u_n=U_{(n-1)}$

### 6.772.2 Documentation des fonctions membres

#### 6.772.2.1 Affiche()

```
void Suite_equidistante::Affiche ( ) const [virtual]
Implémente SuiteReel.
```

#### 6.772.2.2 Def\_suite()

```
void Suite_equidistante::Def_suite (
    double amplitude,
    int n ) [virtual]
Implémente SuiteReel.
```

#### 6.772.2.3 Somme\_Suite()

```
double Suite_equidistante::Somme_Suite (
    int n ) [virtual]
Implémente SuiteReel.
```

### 6.772.2.4 U\_n()

```
double Suite_equidistante::U_n (
    int ) [inline], [virtual]
```

Implémente [SuiteReel](#).

La documentation de cette classe a été générée à partir du fichier suivant :

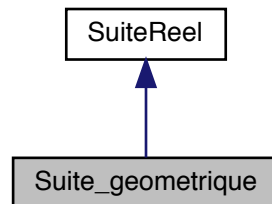
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite\_equidistante.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite\_equidistante.cc

## 6.773 Référence de la classe Suite\_geometrique

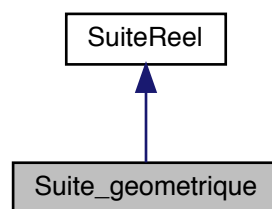
[Suite\\_geometrique](#): cas d'une suite géométrique.

```
#include <Suite_geometrique.h>
```

Grphe d'héritage de Suite\_geometrique:



Grphe de collaboration de Suite\_geometrique:



### Fonctions membres publiques

- [Suite\\_geometrique](#) (const [Suite\\_geometrique](#) &Co)
- void [Affiche](#) () const
- double [U\\_n](#) (int n)
- double [Somme\\_Suite](#) (int n)
- void [Def\\_suite](#) (double amplitude, int n)

### Fonctions membres protégées

- double [PUISSn](#) (double a, const int n)

## Attributs protégés

- double `U_0`
- double `p`

### 6.773.1 Description détaillée

[Suite\\_geometrique](#): cas d'une suite géométrique.

### 6.773.2 Documentation des fonctions membres

#### 6.773.2.1 Affiche()

```
void Suite_geometrique::Affiche ( ) const [virtual]
```

Implémente [SuiteReel](#).

#### 6.773.2.2 Def\_suite()

```
void Suite_geometrique::Def_suite (
    double amplitude,
    int n ) [virtual]
```

Implémente [SuiteReel](#).

#### 6.773.2.3 Somme\_Suite()

```
double Suite_geometrique::Somme_Suite (
    int n ) [virtual]
```

Implémente [SuiteReel](#).

#### 6.773.2.4 U\_n()

```
double Suite_geometrique::U_n (
    int n ) [virtual]
```

Implémente [SuiteReel](#).

La documentation de cette classe a été générée à partir du fichier suivant :

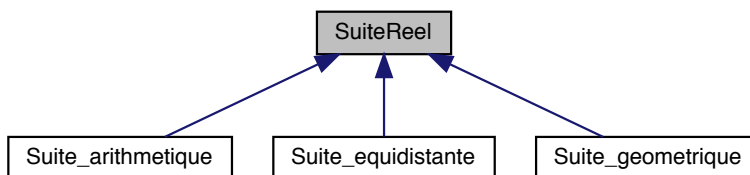
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite_geometrique.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/Suite_geometrique.cc`

## 6.774 Référence de la classe SuiteReel

classe d'interface (virtuelle pure) pour la création et l'utilisation de suite

```
#include <SuiteReel.h>
```

Graphe d'héritage de SuiteReel:



## Fonctions membres publiques

- **SuiteReel** ([Enum\\_Suite](#) typ=SUITE\_NON\_DEFINIE)
- **SuiteReel** (const [SuiteReel](#) &Co)
- virtual void **Affiche** () const =0
- virtual double **U\_n** (int n)=0
- virtual double **Somme\_Suite** (int n)=0
- [Enum\\_Suite](#) **Type\_Suite** () const
- virtual void **Def\_suite** (double amplitude, int n)=0

## Attributs protégés

- [Enum\\_Suite](#) typeSuite

### 6.774.1 Description détaillée

classe d'interface (virtuelle pure) pour la création et l'utilisation de suite

La documentation de cette classe a été générée à partir du fichier suivant :

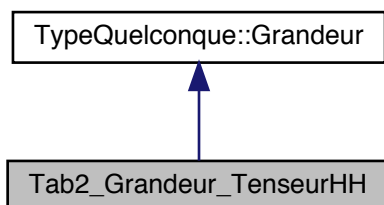
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/suites/SuiteReel.h

### 6.775 Référence de la classe Tab2\_Grandeur\_TenseurHH

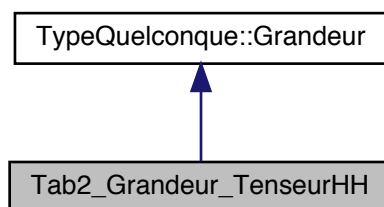
grandeur un tableau à 2 dim de [TenseurHH](#): `TypeQuelconque::Grandeur::Tab2_Grandeur_TenseurHH`

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Tab2\_Grandeur\_TenseurHH:



Graphe de collaboration de Tab2\_Grandeur\_TenseurHH:



## Fonctions membres publiques

- `Tab2_Grandeur_TenseurHH` (`TenseurHH` &tens, int nb1, int nb2)
- `Tab2_Grandeur_TenseurHH` (istream &ent)
- `Tab2_Grandeur_TenseurHH` (const `Tab2_Grandeur_TenseurHH` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Tab2_Grandeur_TenseurHH` &a)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur` &c)
- void `operator*=` (double val)
- void `operator/=` (double val)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcar) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnuTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- `TenseurHH & operator()` (int i, int j) const
- const `Grandeur_TenseurHH & Grandeur_TenseurHH_associee` (int i, int j) const
- void `Change_taille` (int n1, int n2)
- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()

## Attributs protégés

- `Tableau2< Grandeur_TenseurHH * > tabTens`

## Amis

- istream & `operator>>` (istream &ent, `Tab2_Grandeur_TenseurHH` &a)
- ostream & `operator<<` (ostream &sort, const `Tab2_Grandeur_TenseurHH` &a)

### 6.775.1 Description détaillée

grandeur un tableau à 2 dim de `TenseurHH`: `TypeQuelconque::Grandeur::Tab2_Grandeur_TenseurHH`

### 6.775.2 Documentation des fonctions membres

### 6.775.2.1 Affectation\_numerique()

```
void Tab2_Grandeur_TenseurHH::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.775.2.2 Change\_repere()

```
void Tab2_Grandeur_TenseurHH::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.775.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab2_Grandeur_TenseurHH::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.775.2.4 Grandeur\_brut()

```
void Tab2_Grandeur_TenseurHH::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.775.2.5 GrandeurNumOrdre()

```
double Tab2_Grandeur_TenseurHH::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.775.2.6 InitParDefaut()

```
void Tab2_Grandeur_TenseurHH::InitParDefaut ( ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.775.2.7 Lecture\_grandeur()

```
istream & Tab2_Grandeur_TenseurHH::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.775.2.8 NbMaxiNumeroOrdre()

```
int Tab2_Grandeur_TenseurHH::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.775.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Tab2_Grandeur_TenseurHH::New_idem_grandeur ( ) const [inline],
[virtual]
```



Implémente [TypeQuelconque::Grandeur](#).

#### 6.775.2.10 operator\*=( )

```
void Tab2_Grandeur_TenseurHH::operator*= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.775.2.11 operator/=( )

```
void Tab2_Grandeur_TenseurHH::operator/= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.775.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Tab2_Grandeur_TenseurHH::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.775.2.13 Type\_grandeurAssocie()

```
EnumTypeGrandeur Tab2_Grandeur_TenseurHH::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.775.2.14 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Tab2_Grandeur_TenseurHH::Type_structure_grandeurAssocie ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

## 6.776 Référence de l'union Tab\_car\_double\_int

### Attributs publics

- char tampon [928]
- double x [116]
- int n [232]

La documentation de cette union a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/UmatAbaqus.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/UmatAbaqus\_c.c

## 6.777 Référence de l'union Tab\_car\_double\_int\_1

définition d'une union qui lie les réels, les entiers et les caractères

```
#include <Fonction_externe_nD.h>
```

## Attributs publics

- char **tampon** [928]
- double **x** [116]
- int **n** [232]

### 6.777.1 Description détaillée

définition d'une union qui lie les réels, les entiers et les caractères

La documentation de cette union a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Fonction\_externer\_nD.h

## 6.778 Référence de l'union Tab\_car\_et\_double

### Attributs publics

- char **tampon** [21]
- double **truc** [2]

La documentation de cette union a été générée à partir du fichier suivant :

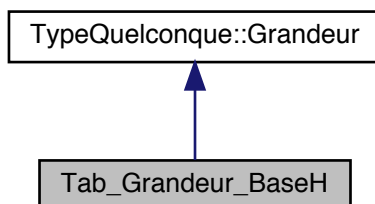
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/UmatAbaqus\_c.c

## 6.779 Référence de la classe Tab\_Grandeur\_BaseH

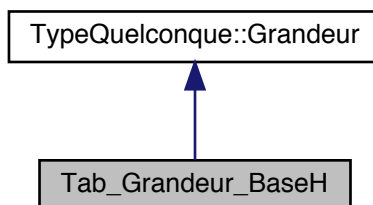
grandeur un tableau de [Grandeur\\_BaseH](#): TypeQuelconque::Grandeur::Tab\_Grandeur\_BaseH tous les coordonnées doivent avoir la même dimension !! pour la routine GrandeurNumOrdre

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Tab\_Grandeur\_BaseH:



Graphe de collaboration de Tab\_Grandeur\_BaseH:



## Fonctions membres publiques

- [Tab\\_Grandeur\\_BaseH](#) ([Grandeur\\_BaseH](#) &baseH, int nb)
- [Tab\\_Grandeur\\_BaseH](#) (istream &ent)
- [Tab\\_Grandeur\\_BaseH](#) (const [Tab\\_Grandeur\\_BaseH](#) &a)
- [TypeQuelconque::Grandeur](#) \* [New\\_idem\\_grandeur](#) () const
- void [EcriturePourLectureAvecCreation](#) (ostream &sort)
- [Grandeur](#) & [operator=](#) (const [Grandeur](#) &a)
- [Grandeur](#) & [operator=](#) (const [Tab\\_Grandeur\\_BaseH](#) &a)
- void [Affectation\\_numerique](#) (const [TypeQuelconque::Grandeur](#) &a)
- void [operator+=](#) (const [Grandeur](#) &c)
- void [operator-=](#) (const [Grandeur](#) &c)
- void [operator\\*=](#) (double val)
- void [operator/=](#) (double val)
- double [GrandeurNumOrdre](#) (int num) const
- int [NbMaxiNumeroOrdre](#) () const
- void [Grandeur\\_brut](#) (ostream &sort, int nbcarr) const
- [EnumTypeGrandeur](#) [Type\\_grandeurAssocie](#) () const
- [EnumType2Niveau](#) [Type\\_structure\\_grandeurAssocie](#) () const
- [EnumTypeQuelParticulier](#) [Type\\_enumGrandeurParticuliere](#) () const
- [Grandeur\\_BaseH](#) & [operator\(\)](#) (int i) const
- const [Grandeur\\_BaseH](#) & [Grandeur\\_Grandeur\\_BaseH\\_associee](#) (int i) const
- int [Taille](#) () const
- void [Change\\_taille](#) (int n)
- void [Change\\_repere](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & [Lecture\\_grandeur](#) (istream &ent)
- virtual ostream & [Ecriture\\_grandeur](#) (ostream &sort) const
- void [InitParDefaut](#) ()

## Attributs protégés

- [Tableau](#)< [Grandeur\\_BaseH](#) > [tabBaseH](#)

## Amis

- istream & [operator](#)>> (istream &ent, [Tab\\_Grandeur\\_BaseH](#) &a)
- ostream & [operator](#)<< (ostream &sort, const [Tab\\_Grandeur\\_BaseH](#) &a)

### 6.779.1 Description détaillée

grandeur un tableau de [Grandeur\\_BaseH](#): [TypeQuelconque::Grandeur::Tab\\_Grandeur\\_BaseH](#) tous les coordonnees doivent avoir la même dimension !! pour la routine [GrandeurNumOrdre](#)

## 6.779.2 Documentation des fonctions membres

### 6.779.2.1 Affectation\_numerique()

```
void Tab_Grandeur_BaseH::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.779.2.2 Change\_repere()

```
void Tab_Grandeur_BaseH::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.779.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_BaseH::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.779.2.4 Grandeur\_brut()

```
void Tab_Grandeur_BaseH::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.779.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_BaseH::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.779.2.6 InitParDefaut()

```
void Tab_Grandeur_BaseH::InitParDefaut ( ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.779.2.7 Lecture\_grandeur()

```
istream & Tab_Grandeur_BaseH::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.779.2.8 NbMaxiNumeroOrdre()

```
int Tab_Grandeur_BaseH::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.779.2.9 New\_idem\_grandeur()

`TypeQuelconque::Grandeur * Tab_Grandeur_BaseH::New_idem_grandeur ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.779.2.10 operator\*=( )

`void Tab_Grandeur_BaseH::operator*= ( double val ) [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.779.2.11 operator/=( )

`void Tab_Grandeur_BaseH::operator/= ( double val ) [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.779.2.12 Type\_enumGrandeurParticuliere()

`EnumTypeQuelParticulier Tab_Grandeur_BaseH::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.779.2.13 Type\_grandeurAssocie()

`EnumTypeGrandeur Tab_Grandeur_BaseH::Type_grandeurAssocie ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.779.2.14 Type\_structure\_grandeurAssocie()

`EnumType2Niveau Tab_Grandeur_BaseH::Type_structure_grandeurAssocie ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

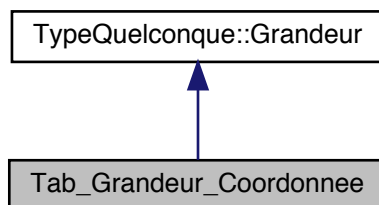
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier_2.cc`

## 6.780 Référence de la classe Tab\_Grandeur\_Coordonnee

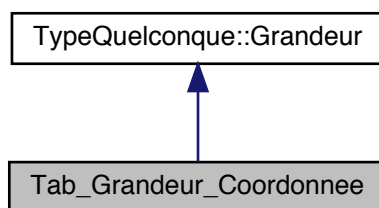
grandeur un tableau de `Coordonnee`: `TypeQuelconque::Grandeur::Tab_Grandeur_CoordonneeHH` tous les coordonnées doivent avoir la même dimension!! pour la routine `GrandeurNumOrdre`

```
#include <TypeQuelconqueParticulier.h>
```

Grappe d'héritage de Tab\_Grandeur\_Coordonnee:



Grappe de collaboration de Tab\_Grandeur\_Coordonnee:



## Fonctions membres publiques

- **Tab\_Grandeur\_Coordonnee** ([Coordonnee](#) &coor, int nb)
- **Tab\_Grandeur\_Coordonnee** (istream &ent)
- **Tab\_Grandeur\_Coordonnee** (const [Tab\\_Grandeur\\_Coordonnee](#) &a)
- [TypeQuelconque::Grandeur](#) \* **New\_idem\_grandeur** () const
- void **EcriturePourLectureAvecCreation** (ostream &sort)
- [Grandeur](#) & **operator=** (const [Grandeur](#) &a)
- [Grandeur](#) & **operator=** (const [Tab\\_Grandeur\\_Coordonnee](#) &a)
- void **Affectation\_numerique** (const [TypeQuelconque::Grandeur](#) &a)
- void **operator+=** (const [Grandeur](#) &c)
- void **operator-=** (const [Grandeur](#) &c)
- void **operator\*=** (double val)
- void **operator/=** (double val)
- double **GrandeurNumOrdre** (int num) const
- int **NbMaxiNumeroOrdre** () const
- void **Grandeur\_brut** (ostream &sort, int nbcarr) const
- [EnumTypeGrandeur](#) **Type\_grandeurAssocie** () const
- [EnumType2Niveau](#) **Type\_structure\_grandeurAssocie** () const
- [EnumTypeQuelParticulier](#) **Type\_enumGrandeurParticuliere** () const
- [Coordonnee](#) & **operator()** (int i) const
- const [Grandeur\\_cooronnee](#) & **Grandeur\_Coordonnee\_associee** (int i) const
- int **Taille** () const
- void **Change\_taille** (int n)
- void **Change\_repere** (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & **Lecture\_grandeur** (istream &ent)
- virtual ostream & **Ecriture\_grandeur** (ostream &sort) const
- void **InitParDefaut** ()

## Attributs protégés

- `Tableau< Grandeur_cooronnee * > tabCoor`

## Amis

- `istream & operator>>` (`istream &ent`, `Tab_Grandeur_Coordonnee &a`)
- `ostream & operator<<` (`ostream &sort`, `const Tab_Grandeur_Coordonnee &a`)

### 6.780.1 Description détaillée

grandeur un tableau de `Coordonnee`: `TypeQuelconque::Grandeur::Tab_Grandeur_CoordonneeHH` tous les coordonnées doivent avoir la même dimension !! pour la routine `GrandeurNumOrdre`

### 6.780.2 Documentation des fonctions membres

#### 6.780.2.1 Affectation\_numerique()

```
void Tab_Grandeur_Coordonnee::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.780.2.2 Change\_repere()

```
void Tab_Grandeur_Coordonnee::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.780.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_Coordonnee::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.780.2.4 Grandeur\_brut()

```
void Tab_Grandeur_Coordonnee::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.780.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_Coordonnee::GrandeurNumOrdre (
    int num ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.780.2.6 InitParDefaut()

```
void Tab_Grandeur_Coordonnee::InitParDefaut ( ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.780.2.7 Lecture\_grandeur()

```
istream & Tab_Grandeur_Coordonnee::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.780.2.8 NbMaxiNumeroOrdre()

```
int Tab_Grandeur_Coordonnee::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.780.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Tab_Grandeur_Coordonnee::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.780.2.10 operator\*=( )

```
void Tab_Grandeur_Coordonnee::operator*= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.780.2.11 operator/=( )

```
void Tab_Grandeur_Coordonnee::operator/= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.780.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Tab_Grandeur_Coordonnee::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.780.2.13 Type\_grandeurAssocie()

```
EnumTypeGrandeur Tab_Grandeur_Coordonnee::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.780.2.14 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Tab_Grandeur_Coordonnee::Type_structure_grandeurAssocie ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

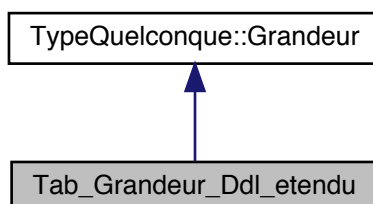


## 6.781 Référence de la classe Tab\_Grandeur\_Ddl\_etendu

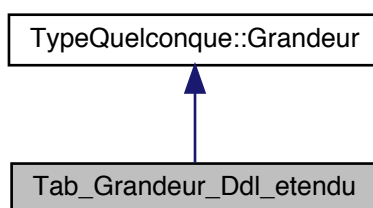
grandeur un tableau de [Coordonnee](#): TypeQuelconque::Grandeur::Tab\_Grandeur\_CoordonneeHH tous les coordonnées doivent avoir la même dimension!! pour la routine GrandeurNumOrdre

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Tab\_Grandeur\_Ddl\_etendu:



Graphe de collaboration de Tab\_Grandeur\_Ddl\_etendu:



### Fonctions membres publiques

- [Tab\\_Grandeur\\_Ddl\\_etendu](#) ([Grandeur\\_Ddl\\_etendu](#) &coor, int nb)
- [Tab\\_Grandeur\\_Ddl\\_etendu](#) (istream &ent)
- [Tab\\_Grandeur\\_Ddl\\_etendu](#) (const [Tab\\_Grandeur\\_Ddl\\_etendu](#) &a)
- [TypeQuelconque::Grandeur](#) \* [New\\_idem\\_grandeur](#) () const
- void [EcriturePourLectureAvecCreation](#) (ostream &sort)
- [Grandeur](#) & **operator=** (const [Grandeur](#) &a)
- [Grandeur](#) & **operator=** (const [Tab\\_Grandeur\\_Ddl\\_etendu](#) &a)
- void [Affectation\\_numerique](#) (const [TypeQuelconque::Grandeur](#) &a)
- void **operator+=** (const [Grandeur](#) &c)
- void **operator-=** (const [Grandeur](#) &c)
- void **operator\*=** (double val)
- void **operator/=** (double val)
- double [GrandeurNumOrdre](#) (int num) const
- int [NbMaxiNumeroOrdre](#) () const
- void [Grandeur\\_brut](#) (ostream &sort, int nbcar) const
- [EnumTypeGrandeur](#) [Type\\_grandeurAssocie](#) () const
- [EnumType2Niveau](#) [Type\\_structure\\_grandeurAssocie](#) () const
- [EnuTypeQuelParticulier](#) [Type\\_enumGrandeurParticuliere](#) () const
- [Ddl\\_etendu](#) & **operator()** (int i) const

- const [Grandeur\\_Ddl\\_etendu](#) & [Grandeur\\_Ddl\\_etendu\\_associee](#) (int i) const
- int [Taille](#) () const
- void [Change\\_taille](#) (int n)
- void [Change\\_repere](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & [Lecture\\_grandeur](#) (istream &ent)
- virtual ostream & [Ecriture\\_grandeur](#) (ostream &sort) const
- void [InitParDefaut](#) ()

### Attributs protégés

- [Tableau](#)< [Grandeur\\_Ddl\\_etendu](#) \* > [tabDdlEte](#)

### Amis

- istream & [operator](#)>> (istream &ent, [Tab\\_Grandeur\\_Ddl\\_etendu](#) &a)
- ostream & [operator](#)<< (ostream &sort, const [Tab\\_Grandeur\\_Ddl\\_etendu](#) &a)

### 6.781.1 Description détaillée

grandeur un tableau de [Coordonnee](#): `TypeQuelconque::Grandeur::Tab_Grandeur_CoordonneeHH` tous les coordonnées doivent avoir la même dimension !! pour la routine `GrandeurNumOrdre`

### 6.781.2 Documentation des fonctions membres

#### 6.781.2.1 Affectation\_numerique()

```
void Tab_Grandeur_Ddl_etendu::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.781.2.2 Change\_repere()

```
void Tab_Grandeur_Ddl_etendu::Change_repere (
    const Mat\_pleine & beta,
    const Mat\_pleine & gamma ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.781.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_Ddl_etendu::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.781.2.4 Grandeur\_brut()

```
void Tab_Grandeur_Ddl_etendu::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_Ddl_etendu::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.6 InitParDefaut()

```
void Tab_Grandeur_Ddl_etendu::InitParDefaut ( ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.7 Lecture\_grandeur()

```
istream & Tab_Grandeur_Ddl_etendu::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.8 NbMaxiNumeroOrdre()

```
int Tab_Grandeur_Ddl_etendu::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Tab_Grandeur_Ddl_etendu::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.10 operator\*=( )

```
void Tab_Grandeur_Ddl_etendu::operator*= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.11 operator/=( )

```
void Tab_Grandeur_Ddl_etendu::operator/= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Tab_Grandeur_Ddl_etendu::Type_enumGrandeurParticuliere ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.13 Type\_grandeurAssocie()

```
EnumTypeGrandeur Tab_Grandeur_Ddl_etendu::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.781.2.14 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Tab_Grandeur_Ddl_etendu::Type_structure_grandeurAssocie ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

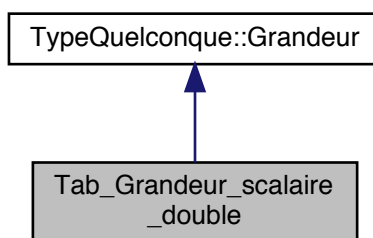
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

## 6.782 Référence de la classe Tab\_Grandeur\_scalaire\_double

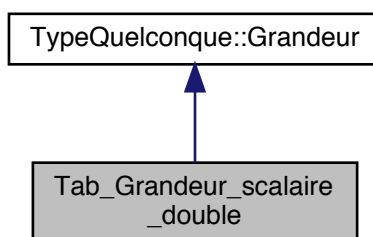
grandeur tableau de scalaires réel: [TypeQuelconque::Grandeur::Tab\\_Grandeur\\_scalaire\\_double](#)

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de [Tab\\_Grandeur\\_scalaire\\_double](#):



Graphe de collaboration de [Tab\\_Grandeur\\_scalaire\\_double](#):



### Fonctions membres publiques

- [Tab\\_Grandeur\\_scalaire\\_double](#) (const [Tableau](#)< double > &tab\_va)
- [Tab\\_Grandeur\\_scalaire\\_double](#) (istream &ent)
- [Tab\\_Grandeur\\_scalaire\\_double](#) (const [Tab\\_Grandeur\\_scalaire\\_double](#) &a)
- [TypeQuelconque::Grandeur](#) \* [New\\_idem\\_grandeur](#) () const
- void [EcriturePourLectureAvecCreation](#) (ostream &sort)
- [Grandeur](#) & [operator=](#) (const [Grandeur](#) &a)
- [Grandeur](#) & [operator=](#) (const [Tableau](#)< double > &b)
- void [Affectation\\_numerique](#) (const [TypeQuelconque::Grandeur](#) &a)

- void `operator+=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur` &c)
- void `operator*=` (double x)
- void `operator/=` (double x)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur` `Type_grandeurAssocie` () const
- `EnumType2Niveau` `Type_structure_grandeurAssocie` () const
- `EnuTypeQuelParticulier` `Type_enumGrandeurParticuliere` () const
- `Grandeur_scalaire_double` & `Grandeur_scalaire_associee` (int i)
- int `Taille` () const
- void `Change_taille` (int n)
- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()
- `Tableau`< `Grandeur_scalaire_double` > & `ConteneurTabDouble` ()
- double & `operator()` (const int i)

### Attributs protégés

- `Tableau`< `Grandeur_scalaire_double` > `tab_val`

### Amis

- istream & `operator>>` (istream &ent, `Tab_Grandeur_scalaire_double` &a)
- ostream & `operator<<` (ostream &sort, const `Tab_Grandeur_scalaire_double` &a)

## 6.782.1 Description détaillée

grandeur tableau de scalaires réel: `TypeQuelconque::Grandeur::Tab_Grandeur_scalaire_double`

## 6.782.2 Documentation des fonctions membres

### 6.782.2.1 `Affectation_numerique()`

```
void Tab_Grandeur_scalaire_double::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.782.2.2 `Change_repere()`

```
void Tab_Grandeur_scalaire_double::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.782.2.3 `Ecriture_grandeur()`

```
virtual ostream & Tab_Grandeur_scalaire_double::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.782.2.4 Grandeur\_brut()

```
void Tab_Grandeur_scalaire_double::Grandeur_brut (
    ostream & sort,
    int nbcars ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.782.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_scalaire_double::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.782.2.6 InitParDefaut()

```
void Tab_Grandeur_scalaire_double::InitParDefaut ( ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.782.2.7 Lecture\_grandeur()

```
istream & Tab_Grandeur_scalaire_double::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.782.2.8 NbMaxiNumeroOrdre()

```
int Tab_Grandeur_scalaire_double::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.782.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Tab_Grandeur_scalaire_double::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.782.2.10 operator\*=( )

```
void Tab_Grandeur_scalaire_double::operator*=(
    double x ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.782.2.11 operator/=( )

```
void Tab_Grandeur_scalaire_double::operator/=(
    double x ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.782.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Tab_Grandeur_scalaire_double::Type_enumGrandeurParticuliere ( ) const
[inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.782.2.13 Type\_grandeurAssocie()

`EnumTypeGrandeur` Tab\_Grandeur\_scalaire\_double::Type\_grandeurAssocie ( ) const [inline], [virtual]  
Implémente `TypeQuelconque::Grandeur`.

### 6.782.2.14 Type\_structure\_grandeurAssocie()

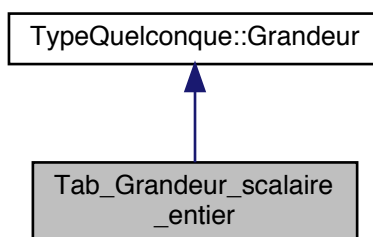
`EnumType2Niveau` Tab\_Grandeur\_scalaire\_double::Type\_structure\_grandeurAssocie ( ) const [inline], [virtual]  
Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

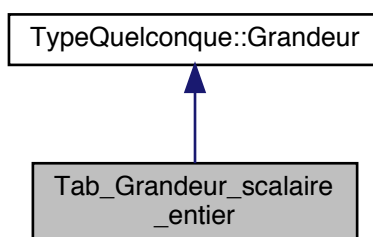
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

## 6.783 Référence de la classe Tab\_Grandeur\_scalaire\_entier

grandeur tableau de scalaires entière: `TypeQuelconque::Grandeur::Tab_Grandeur_scalaire_entiere`  
`#include <TypeQuelconqueParticulier.h>`  
Graphe d'héritage de `Tab_Grandeur_scalaire_entier`:



Graphe de collaboration de `Tab_Grandeur_scalaire_entier`:



## Fonctions membres publiques

- `Tab_Grandeur_scalaire_entier` (const `Tableau< int >` &tab\_va)
- `Tab_Grandeur_scalaire_entier` (istream &ent)
- `Tab_Grandeur_scalaire_entier` (const `Tab_Grandeur_scalaire_entier` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Tableau< int >` &b)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur` &c)
- void `operator*=` (double x)
- void `operator/=` (double x)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnumTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- `Grandeur_scalaire_entier & Grandeur_scalaire_associe` (int i)
- int `Taille` () const
- void `Change_taille` (int n)
- void `Change_repere` (const `Mat_pleine` &, const `Mat_pleine` &)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()
- `Tableau< Grandeur_scalaire_entier > & ConteneurTabEntier` ()
- int & `operator()` (const int i)

## Attributs protégés

- `Tableau< Grandeur_scalaire_entier > tab_val`

## Amis

- istream & `operator>>` (istream &ent, `Tab_Grandeur_scalaire_entier` &a)
- ostream & `operator<<` (ostream &sort, const `Tab_Grandeur_scalaire_entier` &a)

### 6.783.1 Description détaillée

grandeur tableau de scalaires entière: `TypeQuelconque::Grandeur::Tab_Grandeur_scalaire_entiere`

### 6.783.2 Documentation des fonctions membres

#### 6.783.2.1 Affectation\_numerique()

```
void Tab_Grandeur_scalaire_entier::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.783.2.2 Change\_repere()

```
void Tab_Grandeur_scalaire_entier::Change_repere (
    const Mat_pleine & ,
    const Mat_pleine & ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.



### 6.783.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_scalaire_entier::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.783.2.4 Grandeur\_brut()

```
void Tab_Grandeur_scalaire_entier::Grandeur_brut (
    ostream & sort,
    int nbcars ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.783.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_scalaire_entier::GrandeurNumOrdre (
    int num ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.783.2.6 InitParDefaut()

```
void Tab_Grandeur_scalaire_entier::InitParDefaut ( ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.783.2.7 Lecture\_grandeur()

```
istream & Tab_Grandeur_scalaire_entier::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.783.2.8 NbMaxiNumeroOrdre()

```
int Tab_Grandeur_scalaire_entier::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.783.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Tab_Grandeur_scalaire_entier::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.783.2.10 operator\*=( )

```
void Tab_Grandeur_scalaire_entier::operator*= (
    double x ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.783.2.11 operator/=( )

```
void Tab_Grandeur_scalaire_entier::operator/= (
    double x ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

#### 6.783.2.12 Type\_enumGrandeurParticuliere()

`EnumTypeQuelParticulier` Tab\_Grandeur\_scalaire\_entier::Type\_enumGrandeurParticuliere ( ) const [inline], [virtual]

Implémente `TypeQuelconque::Grandeur`.

#### 6.783.2.13 Type\_grandeurAssocie()

`EnumTypeGrandeur` Tab\_Grandeur\_scalaire\_entier::Type\_grandeurAssocie ( ) const [inline], [virtual]

Implémente `TypeQuelconque::Grandeur`.

#### 6.783.2.14 Type\_structure\_grandeurAssocie()

`EnumType2Niveau` Tab\_Grandeur\_scalaire\_entier::Type\_structure\_grandeurAssocie ( ) const [inline], [virtual]

Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

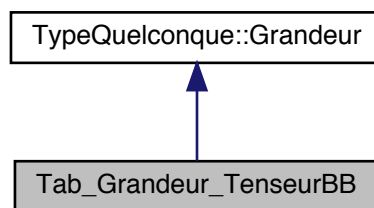
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

### 6.784 Référence de la classe Tab\_Grandeur\_TenseurBB

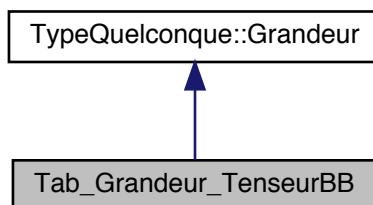
grandeur un tableau de `TenseurBB`: `TypeQuelconque::Grandeur::Tab_Grandeur_TenseurBB` | tous les tenseurs doivent avoir la même dimension!! pour la routine `GrandeurNumOrdre`

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de `Tab_Grandeur_TenseurBB`:



Graphe de collaboration de Tab\_Grandeur\_TenseurBB:



## Fonctions membres publiques

- [Tab\\_Grandeur\\_TenseurBB](#) ([TenseurBB](#) &tens, int nb)
- [Tab\\_Grandeur\\_TenseurBB](#) (istream &ent)
- [Tab\\_Grandeur\\_TenseurBB](#) (const [Tab\\_Grandeur\\_TenseurBB](#) &a)
- [TypeQuelconque::Grandeur](#) \* [New\\_idem\\_grandeur](#) () const
- void [EcriturePourLectureAvecCreation](#) (ostream &sort)
- [Grandeur](#) & [operator=](#) (const [Grandeur](#) &a)
- [Grandeur](#) & [operator=](#) (const [Tab\\_Grandeur\\_TenseurBB](#) &a)
- void [Affectation\\_numerique](#) (const [TypeQuelconque::Grandeur](#) &a)
- void [operator+=](#) (const [Grandeur](#) &c)
- void [operator-=](#) (const [Grandeur](#) &c)
- void [operator\\*=](#) (double val)
- void [operator/=](#) (double val)
- double [GrandeurNumOrdre](#) (int num) const
- int [NbMaxiNumeroOrdre](#) () const
- void [Grandeur\\_brut](#) (ostream &sort, int nbcar) const
- [EnumTypeGrandeur](#) [Type\\_grandeurAssocie](#) () const
- [EnumType2Niveau](#) [Type\\_structure\\_grandeurAssocie](#) () const
- [EnuTypeQuelParticulier](#) [Type\\_enumGrandeurParticuliere](#) () const
- [TenseurBB](#) & [operator\(\)](#) (int i) const
- const [Grandeur\\_TenseurBB](#) & [Grandeur\\_TenseurBB\\_associee](#) (int i) const
- int [Taille](#) () const
- void [Change\\_taille](#) (int n)
- void [Change\\_repere](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & [Lecture\\_grandeur](#) (istream &ent)
- virtual ostream & [Ecriture\\_grandeur](#) (ostream &sort) const
- void [InitParDefaut](#) ()

## Attributs protégés

- [Tableau](#)< [Grandeur\\_TenseurBB](#) \* > [tabTens](#)

## Amis

- istream & [operator>>](#) (istream &ent, [Tab\\_Grandeur\\_TenseurBB](#) &a)
- ostream & [operator<<](#) (ostream &sort, const [Tab\\_Grandeur\\_TenseurBB](#) &a)

### 6.784.1 Description détaillée

grandeur un tableau de [TenseurBB](#): [TypeQuelconque::Grandeur::Tab\\_Grandeur\\_TenseurBB](#) | tous les tenseurs doivent avoir la même dimension!! pour la routine [GrandeurNumOrdre](#)

## 6.784.2 Documentation des fonctions membres

### 6.784.2.1 Affectation\_numerique()

```
void Tab_Grandeur_TenseurBB::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.784.2.2 Change\_repere()

```
void Tab_Grandeur_TenseurBB::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.784.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_TenseurBB::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.784.2.4 Grandeur\_brut()

```
void Tab_Grandeur_TenseurBB::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.784.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_TenseurBB::GrandeurNumOrdre (
    int num ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.784.2.6 InitParDefaut()

```
void Tab_Grandeur_TenseurBB::InitParDefaut ( ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.784.2.7 Lecture\_grandeur()

```
istream & Tab_Grandeur_TenseurBB::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.784.2.8 NbMaxiNumeroOrdre()

```
int Tab_Grandeur_TenseurBB::NbMaxiNumeroOrdre ( ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.784.2.9 New\_idem\_grandeur()

`TypeQuelconque::Grandeur * Tab_Grandeur_TenseurBB::New_idem_grandeur ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.784.2.10 operator\*=( )

`void Tab_Grandeur_TenseurBB::operator*=(  
double val ) [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.784.2.11 operator/=( )

`void Tab_Grandeur_TenseurBB::operator/=(  
double val ) [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.784.2.12 Type\_enumGrandeurParticuliere()

`EnumTypeQuelParticulier Tab_Grandeur_TenseurBB::Type_enumGrandeurParticuliere ( ) const [inline],  
[virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.784.2.13 Type\_grandeurAssocie()

`EnumTypeGrandeur Tab_Grandeur_TenseurBB::Type_grandeurAssocie ( ) const [inline], [virtual]`  
Implémente `TypeQuelconque::Grandeur`.

### 6.784.2.14 Type\_structure\_grandeurAssocie()

`EnumType2Niveau Tab_Grandeur_TenseurBB::Type_structure_grandeurAssocie ( ) const [inline],  
[virtual]`  
Implémente `TypeQuelconque::Grandeur`.

La documentation de cette classe a été générée à partir du fichier suivant :

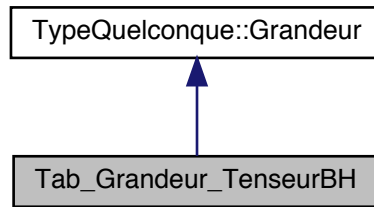
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

## 6.785 Référence de la classe Tab\_Grandeur\_TenseurBH

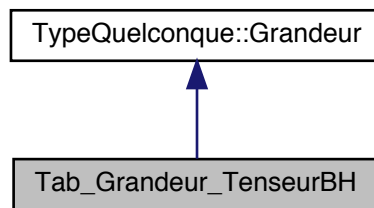
grandeur un tableau de `TenseurBH`: `TypeQuelconque::Grandeur::Tab_Grandeur_TenseurBH` | tous les tenseurs doivent avoir la même dimension!! pour la routine `GrandeurNumOrdre`

```
#include <TypeQuelconqueParticulier.h>
```

Grappe d'héritage de Tab\_Grandeur\_TenseurBH:



Grappe de collaboration de Tab\_Grandeur\_TenseurBH:



## Fonctions membres publiques

- **Tab\_Grandeur\_TenseurBH** ([TenseurBH](#) &tens, int nb)
- **Tab\_Grandeur\_TenseurBH** (istream &ent)
- **Tab\_Grandeur\_TenseurBH** (const [Tab\\_Grandeur\\_TenseurBH](#) &a)
- [TypeQuelconque::Grandeur](#) \* [New\\_idem\\_grandeur](#) () const
- void **EcriturePourLectureAvecCreation** (ostream &sort)
- [Grandeur](#) & **operator=** (const [Grandeur](#) &a)
- [Grandeur](#) & **operator=** (const [Tab\\_Grandeur\\_TenseurBH](#) &a)
- void [Affectation\\_numerique](#) (const [TypeQuelconque::Grandeur](#) &a)
- void **operator+=** (const [Grandeur](#) &c)
- void **operator-=** (const [Grandeur](#) &c)
- void **operator\*=** (double val)
- void **operator/=** (double val)
- double [GrandeurNumOrdre](#) (int num) const
- int [NbMaxiNumeroOrdre](#) () const
- void [Grandeur\\_brut](#) (ostream &sort, int nbcar) const
- [EnumTypeGrandeur](#) [Type\\_grandeurAssocie](#) () const
- [EnumType2Niveau](#) [Type\\_structure\\_grandeurAssocie](#) () const
- [EnumTypeQuelParticulier](#) [Type\\_enumGrandeurParticuliere](#) () const
- [TenseurBH](#) & **operator()** (int i) const
- const [Grandeur\\_TenseurBH](#) & [Grandeur\\_TenseurBH\\_associee](#) (int i) const
- int [Taille](#) () const
- void **Change\_taille** (int n)
- void [Change\\_repere](#) (const [Mat\\_pleine](#) &beta, const [Mat\\_pleine](#) &gamma)
- istream & [Lecture\\_grandeur](#) (istream &ent)
- virtual ostream & [Ecriture\\_grandeur](#) (ostream &sort) const
- void [InitParDefaut](#) ()

## Attributs protégés

- `Tableau< Grandeur_TenseurBH * >` `tabTens`

## Amis

- `istream & operator>>` (`istream &ent`, `Tab_Grandeur_TenseurBH &a`)
- `ostream & operator<<` (`ostream &sort`, `const Tab_Grandeur_TenseurBH &a`)

### 6.785.1 Description détaillée

grandeur un tableau de `TenseurBH`: `TypeQuelconque::Grandeur::Tab_Grandeur_TenseurBH` | tous les tenseurs doivent avoir la même dimension!! pour la routine `GrandeurNumOrdre`

### 6.785.2 Documentation des fonctions membres

#### 6.785.2.1 Affectation\_numerique()

```
void Tab_Grandeur_TenseurBH::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.785.2.2 Change\_repere()

```
void Tab_Grandeur_TenseurBH::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.785.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_TenseurBH::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.785.2.4 Grandeur\_brut()

```
void Tab_Grandeur_TenseurBH::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.785.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_TenseurBH::GrandeurNumOrdre (
    int num ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.785.2.6 InitParDefaut()

```
void Tab_Grandeur_TenseurBH::InitParDefaut ( ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.785.2.7 Lecture\_grandeur()**

```
istream & Tab_Grandeur_TenseurBH::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.785.2.8 NbMaxiNumeroOrdre()**

```
int Tab_Grandeur_TenseurBH::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.785.2.9 New\_idem\_grandeur()**

```
TypeQuelconque::Grandeur * Tab_Grandeur_TenseurBH::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.785.2.10 operator\*=( )**

```
void Tab_Grandeur_TenseurBH::operator*= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.785.2.11 operator/=( )**

```
void Tab_Grandeur_TenseurBH::operator/= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.785.2.12 Type\_enumGrandeurParticuliere()**

```
EnumTypeQuelParticulier Tab_Grandeur_TenseurBH::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.785.2.13 Type\_grandeurAssocie()**

```
EnumTypeGrandeur Tab_Grandeur_TenseurBH::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.785.2.14 Type\_structure\_grandeurAssocie()**

```
EnumType2Niveau Tab_Grandeur_TenseurBH::Type_structure_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

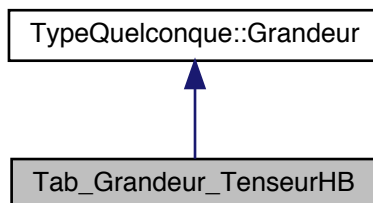
**6.786 Référence de la classe Tab\_Grandeur\_TenseurHB**

grandeur un tableau de [TenseurHB](#): TypeQuelconque::Grandeur::Tab\_Grandeur\_TenseurHB tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre

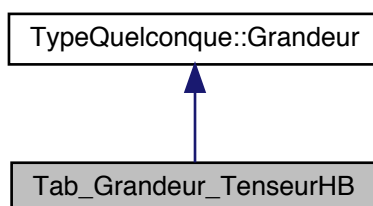


```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Tab\_Grandeur\_TenseurHB:



Graphe de collaboration de Tab\_Grandeur\_TenseurHB:



## Fonctions membres publiques

- `Tab_Grandeur_TenseurHB (TenseurHB &tens, int nb)`
- `Tab_Grandeur_TenseurHB (istream &ent)`
- `Tab_Grandeur_TenseurHB (const Tab_Grandeur_TenseurHB &a)`
- `TypeQuelconque::Grandeur * New_idem_grandeur () const`
- `void EcriturePourLectureAvecCreation (ostream &sort)`
- `Grandeur & operator= (const Grandeur &a)`
- `Grandeur & operator= (const Tab_Grandeur_TenseurHB &a)`
- `void Affectation_numerique (const TypeQuelconque::Grandeur &a)`
- `void operator+= (const Grandeur &c)`
- `void operator-= (const Grandeur &c)`
- `void operator*= (double val)`
- `void operator/= (double val)`
- `double GrandeurNumOrdre (int num) const`
- `int NbMaxiNumeroOrdre () const`
- `void Grandeur_brut (ostream &sort, int nbcar) const`
- `EnumTypeGrandeur Type_grandeurAssocie () const`
- `EnumType2Niveau Type_structure_grandeurAssocie () const`
- `EnuTypeQuelParticulier Type_enumGrandeurParticuliere () const`
- `TenseurHB & operator() (int i) const`
- `const Grandeur_TenseurHB & Grandeur_TenseurHB_associee (int i) const`
- `int Taille () const`
- `void Change_taille (int n)`
- `void Change_repere (const Mat_pleine &beta, const Mat_pleine &gamma)`
- `istream & Lecture_grandeur (istream &ent)`
- `virtual ostream & Ecriture_grandeur (ostream &sort) const`
- `void InitParDefaut ()`

## Attributs protégés

- `Tableau< Grandeur_TenseurHB * > tabTens`

## Amis

- `istream & operator>>` (`istream &ent`, `Tab_Grandeur_TenseurHB &a`)
- `ostream & operator<<` (`ostream &sort`, `const Tab_Grandeur_TenseurHB &a`)

### 6.786.1 Description détaillée

grandeur un tableau de `TenseurHB`: `TypeQuelconque::Grandeur::Tab_Grandeur_TenseurHB` tous les tenseurs doivent avoir la même dimension!! pour la routine `GrandeurNumOrdre`

### 6.786.2 Documentation des fonctions membres

#### 6.786.2.1 Affectation\_numerique()

```
void Tab_Grandeur_TenseurHB::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.786.2.2 Change\_repere()

```
void Tab_Grandeur_TenseurHB::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.786.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_TenseurHB::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.786.2.4 Grandeur\_brut()

```
void Tab_Grandeur_TenseurHB::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.786.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_TenseurHB::GrandeurNumOrdre (
    int num ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.786.2.6 InitParDefaut()

```
void Tab_Grandeur_TenseurHB::InitParDefaut ( ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.786.2.7 Lecture\_grandeur()**

```
istream & Tab_Grandeur_TenseurHB::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.786.2.8 NbMaxiNumeroOrdre()**

```
int Tab_Grandeur_TenseurHB::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.786.2.9 New\_idem\_grandeur()**

```
TypeQuelconque::Grandeur * Tab_Grandeur_TenseurHB::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.786.2.10 operator\*=( )**

```
void Tab_Grandeur_TenseurHB::operator*= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.786.2.11 operator/=( )**

```
void Tab_Grandeur_TenseurHB::operator/= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.786.2.12 Type\_enumGrandeurParticuliere()**

```
EnumTypeQuelParticulier Tab_Grandeur_TenseurHB::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.786.2.13 Type\_grandeurAssocie()**

```
EnumTypeGrandeur Tab_Grandeur_TenseurHB::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.786.2.14 Type\_structure\_grandeurAssocie()**

```
EnumType2Niveau Tab_Grandeur_TenseurHB::Type_structure_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

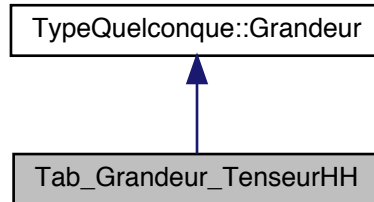
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

**6.787 Référence de la classe Tab\_Grandeur\_TenseurHH**

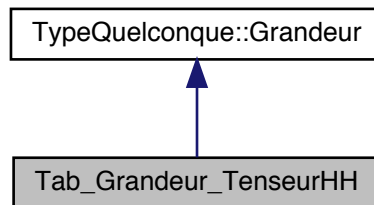
grandeur un tableau de [TenseurHH](#): `TypeQuelconque::Grandeur::Tab_Grandeur_TenseurHH` | tous les tenseurs doivent avoir la même dimension !! pour la routine `GrandeurNumOrdre`

```
#include <TypeQuelconqueParticulier.h>
```

Grphe d'hritage de Tab\_Grandeur\_TenseurHH:



Grphe de collaboration de Tab\_Grandeur\_TenseurHH:



## Fonctions membres publiques

- `Tab_Grandeur_TenseurHH (TenseurHH &tens, int nb)`
- `Tab_Grandeur_TenseurHH (istream &ent)`
- `Tab_Grandeur_TenseurHH (const Tab_Grandeur_TenseurHH &a)`
- `TypeQuelconque::Grandeur * New_idem_grandeur () const`
- `void EcriturePourLectureAvecCreation (ostream &sort)`
- `Grandeur & operator= (const Grandeur &a)`
- `Grandeur & operator= (const Tab_Grandeur_TenseurHH &a)`
- `void Affectation_numerique (const TypeQuelconque::Grandeur &a)`
- `void operator+= (const Grandeur &c)`
- `void operator-= (const Grandeur &c)`
- `void operator*= (double val)`
- `void operator/= (double val)`
- `double GrandeurNumOrdre (int num) const`
- `int NbMaxiNumeroOrdre () const`
- `void Grandeur_brut (ostream &sort, int nbcar) const`
- `EnumTypeGrandeur Type_grandeurAssocie () const`
- `EnumType2Niveau Type_structure_grandeurAssocie () const`
- `EnumTypeQuelParticulier Type_enumGrandeurParticuliere () const`
- `TenseurHH & operator() (int i) const`
- `const Grandeur_TenseurHH & Grandeur_TenseurHH_associee (int i) const`
- `int Taille () const`
- `void Change_taille (int n)`
- `void Change_repere (const Mat_pleine &beta, const Mat_pleine &gamma)`
- `istream & Lecture_grandeur (istream &ent)`
- `virtual ostream & Ecriture_grandeur (ostream &sort) const`
- `void InitParDefaut ()`

## Attributs protégés

- `Tableau< Grandeur_TenseurHH * > tabTens`

## Amis

- `istream & operator>>` (`istream &ent`, `Tab_Grandeur_TenseurHH &a`)
- `ostream & operator<<` (`ostream &sort`, `const Tab_Grandeur_TenseurHH &a`)

### 6.787.1 Description détaillée

grandeur un tableau de `TenseurHH`: `TypeQuelconque::Grandeur::Tab_Grandeur_TenseurHH` | tous les tenseurs doivent avoir la même dimension!! pour la routine `GrandeurNumOrdre`

### 6.787.2 Documentation des fonctions membres

#### 6.787.2.1 Affectation\_numerique()

```
void Tab_Grandeur_TenseurHH::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.787.2.2 Change\_repere()

```
void Tab_Grandeur_TenseurHH::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.787.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_TenseurHH::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.787.2.4 Grandeur\_brut()

```
void Tab_Grandeur_TenseurHH::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.787.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_TenseurHH::GrandeurNumOrdre (
    int num ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.787.2.6 InitParDefaut()

```
void Tab_Grandeur_TenseurHH::InitParDefaut ( ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

**6.787.2.7 Lecture\_grandeur()**

```
istream & Tab_Grandeur_TenseurHH::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.787.2.8 NbMaxiNumeroOrdre()**

```
int Tab_Grandeur_TenseurHH::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.787.2.9 New\_idem\_grandeur()**

```
TypeQuelconque::Grandeur * Tab_Grandeur_TenseurHH::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.787.2.10 operator\*=( )**

```
void Tab_Grandeur_TenseurHH::operator*= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.787.2.11 operator/=( )**

```
void Tab_Grandeur_TenseurHH::operator/= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.787.2.12 Type\_enumGrandeurParticuliere()**

```
EnumTypeQuelParticulier Tab_Grandeur_TenseurHH::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.787.2.13 Type\_grandeurAssocie()**

```
EnumTypeGrandeur Tab_Grandeur_TenseurHH::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

**6.787.2.14 Type\_structure\_grandeurAssocie()**

```
EnumType2Niveau Tab_Grandeur_TenseurHH::Type_structure_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

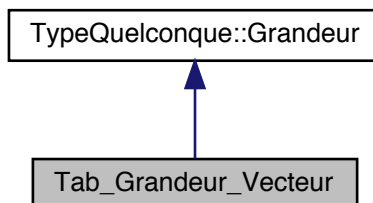
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.cc

**6.788 Référence de la classe Tab\_Grandeur\_Vecteur**

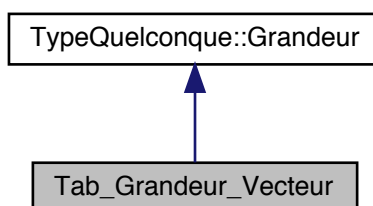
grandeur un tableau de [Vecteur](#): `TypeQuelconque::Grandeur::Tab_Grandeur_Vecteur` | tous les Vecteurs doivent avoir la même dimension !! pour la routine `GrandeurNumOrdre` ?? à voir

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Tab\_Grandeur\_Vecteur:



Graphe de collaboration de Tab\_Grandeur\_Vecteur:



## Fonctions membres publiques

- `Tab_Grandeur_Vecteur` (`Vecteur` &vect, int nb)
- `Tab_Grandeur_Vecteur` (istream &ent)
- `Tab_Grandeur_Vecteur` (const `Tab_Grandeur_Vecteur` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Tab_Grandeur_Vecteur` &a)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur` &c)
- void `operator*=` (double val)
- void `operator/=` (double val)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcar) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnuTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- `Vecteur & operator()` (int i) const
- const `Grandeur_Vecteur & Grandeur_Vecteur_associee` (int i) const
- int `Taille` () const
- void `Change_taille` (int n)
- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()

## Attributs protégés

- `Tableau< Grandeur_Vecteur * > tabVe`

## Amis

- `istream & operator>>` (`istream &ent`, `Tab_Grandeur_Vecteur &a`)
- `ostream & operator<<` (`ostream &sort`, `const Tab_Grandeur_Vecteur &a`)

### 6.788.1 Description détaillée

grandeur un tableau de `Vecteur`: `TypeQuelconque::Grandeur::Tab_Grandeur_Vecteur` | tous les Vecteurs doivent avoir la même dimension !! pour la routine `GrandeurNumOrdre` ?? à voir

### 6.788.2 Documentation des fonctions membres

#### 6.788.2.1 Affectation\_numerique()

```
void Tab_Grandeur_Vecteur::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.788.2.2 Change\_repere()

```
void Tab_Grandeur_Vecteur::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.788.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_Vecteur::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.788.2.4 Grandeur\_brut()

```
void Tab_Grandeur_Vecteur::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.788.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_Vecteur::GrandeurNumOrdre (
    int num ) const [virtual]
Implémente TypeQuelconque::Grandeur.
```

#### 6.788.2.6 InitParDefaut()

```
void Tab_Grandeur_Vecteur::InitParDefaut ( ) [virtual]
Implémente TypeQuelconque::Grandeur.
```



### 6.788.2.7 Lecture\_grandeur()

```
istream & Tab_Grandeur_Vecteur::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.788.2.8 NbMaxiNumeroOrdre()

```
int Tab_Grandeur_Vecteur::NbMaxiNumeroOrdre ( ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.788.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Tab_Grandeur_Vecteur::New_idem_grandeur ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.788.2.10 operator\*=( )

```
void Tab_Grandeur_Vecteur::operator*= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.788.2.11 operator/=( )

```
void Tab_Grandeur_Vecteur::operator/= (
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.788.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Tab_Grandeur_Vecteur::Type_enumGrandeurParticuliere ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.788.2.13 Type\_grandeurAssocie()

```
EnumTypeGrandeur Tab_Grandeur_Vecteur::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.788.2.14 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Tab_Grandeur_Vecteur::Type_structure_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

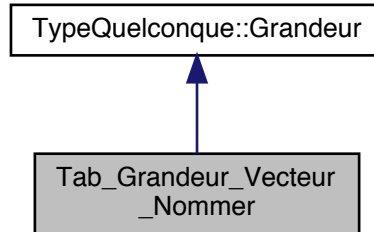
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_3.cc

## 6.789 Référence de la classe Tab\_Grandeur\_Vecteur\_Nommer

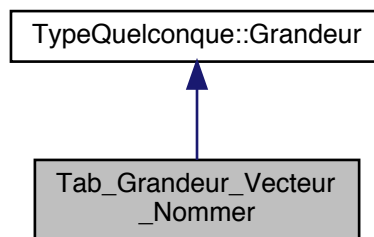
grandeur un tableau de [Grandeur\\_Vecteur\\_Nommer](#): `TypeQuelconque::Grandeur::Tab_Grandeur_Grandeur_Vecteur_Nommer`

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de Tab\_Grandeur\_Vecteur\_Nommer:



Graphe de collaboration de Tab\_Grandeur\_Vecteur\_Nommer:



## Fonctions membres publiques

- `Tab_Grandeur_Vecteur_Nommer` (`Grandeur_Vecteur_Nommer` &grno, int nb)
- `Tab_Grandeur_Vecteur_Nommer` (istream &ent)
- `Tab_Grandeur_Vecteur_Nommer` (const `Tab_Grandeur_Vecteur_Nommer` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `Grandeur & operator=` (const `Grandeur` &a)
- `Grandeur & operator=` (const `Tab_Grandeur_Vecteur_Nommer` &a)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur` &c)
- void `operator*=` (double val)
- void `operator/=` (double val)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnuTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- `Grandeur_Vecteur_Nommer & operator()` (int i) const
- const `Grandeur_Vecteur_Nommer & Grandeur_Vecteur_Nommer_associee` (int i) const
- int `Taille` () const
- void `Change_taille` (int n)

- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()

### Attributs protégés

- `Tableau`< `Grandeur_Vecteur_Nommer` \* > `tabVN`

### Amis

- istream & `operator`>> (istream &ent, `Tab_Grandeur_Vecteur_Nommer` &a)
- ostream & `operator`<< (ostream &sort, const `Tab_Grandeur_Vecteur_Nommer` &a)

## 6.789.1 Description détaillée

grandeur un tableau de `Grandeur_Vecteur_Nommer`: `TypeQuelconque::Grandeur::Tab_Grandeur_Grandeur_Vecteur_Nommer`

## 6.789.2 Documentation des fonctions membres

### 6.789.2.1 Affectation\_numerique()

```
void Tab_Grandeur_Vecteur_Nommer::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.789.2.2 Change\_repere()

```
void Tab_Grandeur_Vecteur_Nommer::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.789.2.3 Ecriture\_grandeur()

```
virtual ostream & Tab_Grandeur_Vecteur_Nommer::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.789.2.4 Grandeur\_brut()

```
void Tab_Grandeur_Vecteur_Nommer::Grandeur_brut (
    ostream & sort,
    int nbcarr ) const [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.789.2.5 GrandeurNumOrdre()

```
double Tab_Grandeur_Vecteur_Nommer::GrandeurNumOrdre (
    int num ) const [inline], [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.789.2.6 InitParDefault()

```
void Tab_Grandeur_Vecteur_Nommer::InitParDefault ( ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.789.2.7 Lecture\_grandeur()

```
istream & Tab_Grandeur_Vecteur_Nommer::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.789.2.8 NbMaxiNumeroOrdre()

```
int Tab_Grandeur_Vecteur_Nommer::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.789.2.9 New\_idem\_grandeur()

```
TypeQuelconque::Grandeur * Tab_Grandeur_Vecteur_Nommer::New_idem_grandeur ( ) const [inline],
[virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.789.2.10 operator\*=( )

```
void Tab_Grandeur_Vecteur_Nommer::operator*= (
    double val ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.789.2.11 operator/=( )

```
void Tab_Grandeur_Vecteur_Nommer::operator/= (
    double val ) [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.789.2.12 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Tab_Grandeur_Vecteur_Nommer::Type_enumGrandeurParticuliere ( ) const
[inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.789.2.13 Type\_grandeurAssocie()

```
EnumTypeGrandeur Tab_Grandeur_Vecteur_Nommer::Type_grandeurAssocie ( ) const [inline], [virtual]
Implémente TypeQuelconque::Grandeur.
```

### 6.789.2.14 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Tab_Grandeur_Vecteur_Nommer::Type_structure_grandeurAssocie ( ) const [inline],
[virtual]
Implémente TypeQuelconque::Grandeur.
```

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

## 6.790 Référence de la classe Table33

### Attributs publics

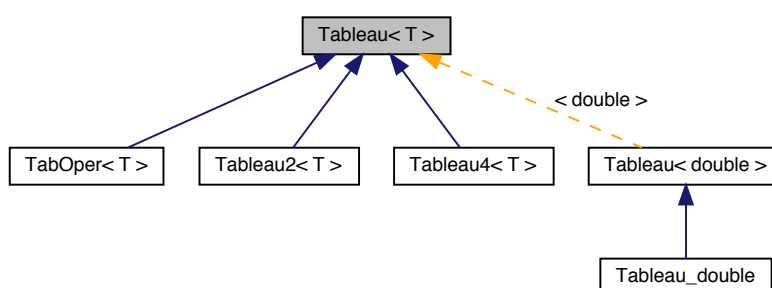
— double **A** [3][3]

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/utilDebug.h

## 6.791 Référence du modèle de la classe Tableau< T >

Grappe d'héritage de Tableau< T > :



### Fonctions membres publiques

- **Tableau** (int nb)
- **Tableau** (int nb, T val)
- **Tableau** (int nb, T \*tab)
- **Tableau** (const [Tableau](#)< T > &tab)
- int **Taille** () const
- T & **operator()** (int i) const
- const T **val\_const** (int i) const
- const T & **ref\_const** (int i) const
- int **operator!=** (const [Tableau](#)< T > &tab) const
- int **operator==** (const [Tableau](#)< T > &tab) const
- [Tableau](#)< T > & **operator=** (const [Tableau](#)< T > &tab)
- void **Change\_taille** (int n)
- void **Change\_taille** (int n, const T &tb)
- void **Enleve** (int i)
- void **Enleve\_val** (T val)
- void **Libere** ()
- void **Init\_from\_list** (const list< T > &liste)
- void **Init\_list** (list< T > &liste)
- istream & **Entree** (istream &)
- istream & **Entree\_sans\_redim** (istream &)
- [Tableau](#)< T > \* **New\_en\_lecture\_si\_taille\_superieur\_a\_lire** (istream &)
- ostream & **Sortir** (ostream &) const
- ostream & **Sortir\_sansRet** (ostream &) const
- int **Contient** (const T &e) const
- bool **Inita** (const T &e)

## Attributs protégés

- int **taille**
- T \* **t**

## Amis

- ostream & **operator**<< (ostream &sort, const [Tableau](#)< T > &tab)
- void **Affiche\_premier\_lem** (ostream &sort, const [Tableau](#)< T > &tab)

## 6.791.1 Documentation des fonctions membres

### 6.791.1.1 Change\_taille()

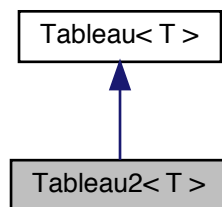
```
template<class T >
void Tableau< T >::Change_taille (
    int n ) [inline]
(*ptr2++)=defaut;
```

La documentation de cette classe a été générée à partir du fichier suivant :

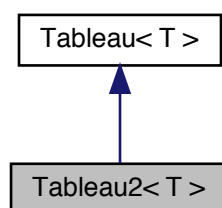
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau\_T.h

## 6.792 Référence du modèle de la classe [Tableau2](#)< T >

Grappe d'héritage de [Tableau2](#)< T > :



Grappe de collaboration de [Tableau2](#)< T > :



## Fonctions membres publiques

- `Tableau2` (const int nb)
- `Tableau2` (const int nb1, const int nb2)
- `Tableau2` (const int nb, const T &val)
- `Tableau2` (const int nb1, const int nb2, const T &val)
- `Tableau2` (const `Tableau2< T >` &tab)
- int `Taille1` () const
- int `Taille2` () const
- int `Taille` () const
- T & `operator()` (int i, int j)
- T `operator()` (int i, int j) const
- int `operator!=` (const `Tableau2< T >` &tab) const
- int `operator==` (const `Tableau2< T >` &tab) const
- `Tableau2< T >` & `operator=` (const `Tableau2< T >` &tab)
- void `Change_taille` (int n)
- void `Change_taille` (int n, const T &tb)
- void `Change_taille` (int n1, int n2)
- void `Change_taille` (int n1, int n2, const T &tb)
- void `Libere` ()

## Attributs protégés

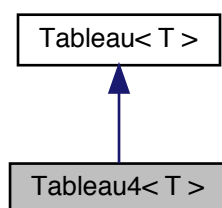
- int `taille1`
- int `taille2`

La documentation de cette classe a été générée à partir du fichier suivant :

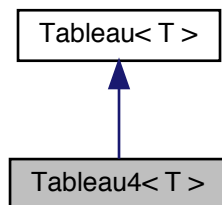
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau2_T.h`

## 6.793 Référence du modèle de la classe `Tableau4< T >`

Graphe d'héritage de `Tableau4< T >` :



Graphe de collaboration de `Tableau4< T >`:



### Fonctions membres publiques

- **Tableau4** (const int nb)
- **Tableau4** (const int nb1, const int nb2, const int nb3, const int nb4)
- **Tableau4** (const int nb, const T &val)
- **Tableau4** (const int nb1, const int nb2, const int nb3, const int nb4, const T &val)
- **Tableau4** (const [Tableau4](#)< T > &tab)
- int **Taille1** () const
- int **Taille2** () const
- int **Taille3** () const
- int **Taille4** () const
- int **Taille** () const
- T & **operator()** (int i, int j, int k, int l)
- T **operator()** (int i, int j, int k, int l) const
- int **operator!=** (const [Tableau4](#)< T > &tab) const
- int **operator==** (const [Tableau4](#)< T > &tab) const
- [Tableau4](#)< T > & **operator=** (const [Tableau4](#)< T > &tab)
- void **Change\_taille** (int n)
- void **Change\_taille** (int n1, int n2, int n3, int n4)
- void **Change\_taille** (int n, T &tb)
- void **Libere** ()

### Attributs protégés

- int **taille1**
- int **taille2**
- int **taille3**
- int **taille4**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau4\_T.h

## 6.794 Référence de la classe `Tableau_3D`

### Fonctions membres publiques

- **Tableau\_3D** (int d1, int d2, int d3, double val=0.0)
- **Tableau\_3D** (const [Tableau\\_3D](#) &tab)
- void **Affiche** () const
- int **Taille1** () const
- int **Taille2** () const
- int **Taille3** () const
- double & **operator()** (int i, int j, int k)
- void **Initialise** (double val=0.0)
- void **Libere** ()
- [Tableau\\_3D](#) & **operator=** (const [Tableau\\_3D](#) &tab)



### Attributs protégés

- int `dim1`
- int `dim2`
- int `dim3`
- double \* `t`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_3D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_3D.cc`

## 6.795 Référence de la classe `Tableau_4D`

### Fonctions membres publiques

- `Tableau_4D` (int d1, int d2, int d3, int d4, double val=0.0)
- `Tableau_4D` (const `Tableau_4D` &tab)
- void `Affiche` ()
- int `Taille1` ()
- int `Taille2` ()
- int `Taille3` ()
- int `Taille4` ()
- double & `operator()` (int i, int j, int k, int l)
- void `Initialise` (double val=0.0)
- void `Libere` ()
- `Tableau_4D` & `operator=` (const `Tableau_4D` &tab)

### Attributs protégés

- int `dim1`
- int `dim2`
- int `dim3`
- int `dim4`
- double \* `t`

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_4D.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_4D.cc`

## 6.796 Référence de la classe `Tableau_5D`

### Fonctions membres publiques

- `Tableau_5D` (int d1, int d2, int d3, int d4, int d5, double val=0.0)
- `Tableau_5D` (const `Tableau_5D` &tab)
- void `Affiche` ()
- int `Taille1` ()
- int `Taille2` ()
- int `Taille3` ()
- int `Taille4` ()
- int `Taille5` ()
- double & `operator()` (int i, int j, int k, int l, int m)
- void `Initialise` (double val=0.0)
- void `Libere` ()
- `Tableau_5D` & `operator=` (const `Tableau_5D` &tab)

## Attributs protégés

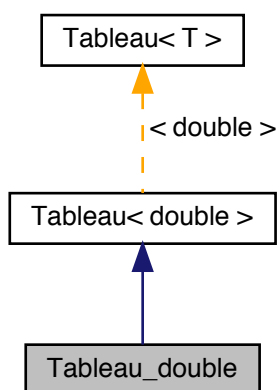
- int **dim1**
- int **dim2**
- int **dim3**
- int **dim4**
- int **dim5**
- double \* **t**

La documentation de cette classe a été générée à partir du fichier suivant :

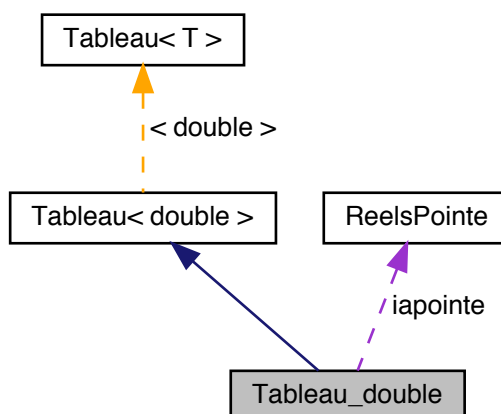
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau\_5D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau\_5D.cc

## 6.797 Référence de la classe Tableau\_double

Graphe d'héritage de Tableau\_double:



Graphe de collaboration de Tableau\_double:



## Fonctions membres publiques

- `Tableau_double` (int nb)
- `Tableau_double` (int nb, double val)
- `Tableau_double` (int nb, double \*tab)
- `Tableau_double` (const `Tableau_double` &tab)
- `Tableau_double` & `operator=` (const `Tableau_double` &tab)
- void `Change_taille` (int n)
- void `Enleve` (int i)
- void `Enleve_val` (double val)
- void `Libere` ()

## Attributs protégés

- `ReelsPointe * iapointe`

La documentation de cette classe a été générée à partir du fichier suivant :

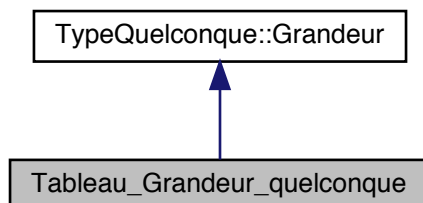
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_double.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/Tableau_double.cc`

## 6.798 Référence de la classe `Tableau_Grandeur_quelconque`

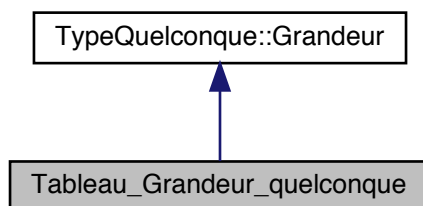
un tableau de grandeur quelconque

```
#include <TypeQuelconqueParticulier.h>
```

Graphe d'héritage de `Tableau_Grandeur_quelconque`:



Graphe de collaboration de `Tableau_Grandeur_quelconque`:



## Fonctions membres publiques

- `Tableau_Grandeur_quelconque` (const `TypeQuelconque::Grandeur` &tg, int nb)
- `Tableau_Grandeur_quelconque` (const `Tableau< TypeQuelconque::Grandeur * >` &tab)
- `Tableau_Grandeur_quelconque` (const `Tableau< TypeQuelconque >` &tab)
- `Tableau_Grandeur_quelconque` (const `Tableau_Grandeur_quelconque` &a)
- `TypeQuelconque::Grandeur * New_idem_grandeur` () const
- void `EcriturePourLectureAvecCreation` (ostream &sort)
- `TypeQuelconque::Grandeur & operator=` (const `TypeQuelconque::Grandeur` &a)
- void `Affectation_numerique` (const `TypeQuelconque::Grandeur` &a)
- void `operator+=` (const `Grandeur` &c)
- void `operator-=` (const `Grandeur` &c)
- void `operator*=` (double val)
- void `operator/=` (double val)
- double `GrandeurNumOrdre` (int num) const
- int `NbMaxiNumeroOrdre` () const
- void `Grandeur_brut` (ostream &sort, int nbcarr) const
- `EnumTypeGrandeur Type_grandeurAssocie` () const
- `EnumType2Niveau Type_structure_grandeurAssocie` () const
- `EnumTypeQuelParticulier Type_enumGrandeurParticuliere` () const
- `TypeQuelconque::Grandeur & operator()` (int i) const
- const `TypeQuelconque::Grandeur & ConteneurGrandeurQuelconque` (int i) const
- int `Taille` () const
- void `Change_taille` (int n)
- void `Change_repere` (const `Mat_pleine` &beta, const `Mat_pleine` &gamma)
- istream & `Lecture_grandeur` (istream &ent)
- virtual ostream & `Ecriture_grandeur` (ostream &sort) const
- void `InitParDefaut` ()

## Attributs protégés

- `Tableau< TypeQuelconque::Grandeur * >` `tab_grandeur`

## Amis

- istream & `operator>>` (istream &ent, `Tableau_Grandeur_quelconque` &a)
- ostream & `operator<<` (ostream &sort, const `Tableau_Grandeur_quelconque` &a)

### 6.798.1 Description détaillée

un tableau de grandeur quelconque

### 6.798.2 Documentation des fonctions membres

#### 6.798.2.1 Affectation\_numerique()

```
void Tableau_Grandeur_quelconque::Affectation_numerique (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

#### 6.798.2.2 Change\_repere()

```
void Tableau_Grandeur_quelconque::Change_repere (
    const Mat_pleine & beta,
    const Mat_pleine & gamma ) [virtual]
```

Implémente `TypeQuelconque::Grandeur`.

### 6.798.2.3 `Ecriture_grandeur()`

```
virtual ostream & Tableau_Grandeur_quelconque::Ecriture_grandeur (
    ostream & sort ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.4 `Grandeur_brut()`

```
void Tableau_Grandeur_quelconque::Grandeur_brut (
    ostream & sort,
    int nbcars ) const [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.5 `GrandeurNumOrdre()`

```
double Tableau_Grandeur_quelconque::GrandeurNumOrdre (
    int num ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.6 `InitParDefaut()`

```
void Tableau_Grandeur_quelconque::InitParDefaut ( ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.7 `Lecture_grandeur()`

```
istream & Tableau_Grandeur_quelconque::Lecture_grandeur (
    istream & ent ) [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.8 `NbMaxiNumeroOrdre()`

```
int Tableau_Grandeur_quelconque::NbMaxiNumeroOrdre ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.9 `New_idem_grandeur()`

```
TypeQuelconque::Grandeur * Tableau_Grandeur_quelconque::New_idem_grandeur ( ) const [inline],
[virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.10 `operator*=( )`

```
void Tableau_Grandeur_quelconque::operator*=(
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.11 `operator/=( )`

```
void Tableau_Grandeur_quelconque::operator/=(
    double val ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.12 operator=()

```
TypeQuelconque::Grandeur & Tableau_Grandeur_quelconque::operator= (
    const TypeQuelconque::Grandeur & a ) [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.13 Type\_enumGrandeurParticuliere()

```
EnumTypeQuelParticulier Tableau_Grandeur_quelconque::Type_enumGrandeurParticuliere ( ) const
[inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.14 Type\_grandeurAssocie()

```
EnumTypeGrandeur Tableau_Grandeur_quelconque::Type_grandeurAssocie ( ) const [inline], [virtual]
```

Implémente [TypeQuelconque::Grandeur](#).

### 6.798.2.15 Type\_structure\_grandeurAssocie()

```
EnumType2Niveau Tableau_Grandeur_quelconque::Type_structure_grandeurAssocie ( ) const [inline],
[virtual]
```

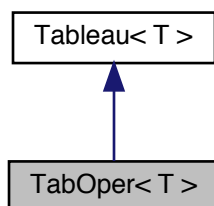
Implémente [TypeQuelconque::Grandeur](#).

La documentation de cette classe a été générée à partir du fichier suivant :

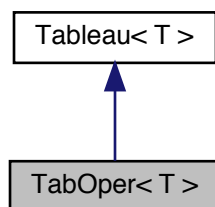
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconqueParticulier\_2.cc

## 6.799 Référence du modèle de la classe TabOper< T >

Graphe d'héritage de TabOper< T >:



Grphe de collaboration de TabOper< T >:



### Fonctions membres publiques

- **TabOper** (int nb)
- **TabOper** (int nb, T val)
- **TabOper** (int nb, T \*tab)
- **TabOper** (const **TabOper**< T > &tab)
- **TabOper**< T > **operator+** (const **TabOper**< T > &vec)
- **TabOper**< T > **operator-** (const **TabOper**< T > &vec)
- **TabOper**< T > **operator-** ()
- void **operator+=** (const **TabOper**< T > &vec)
- void **operator-=** (const **TabOper**< T > &vec)
- void **operator\*=** (double val)
- **TabOper**< T > **operator\*** (double val)
- **TabOper**< T > **operator/** (double val)
- void **operator/=** (double val)
- void **Zero** ()

### Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

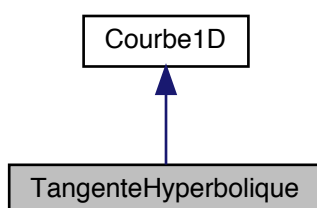
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Tableaux/TabOper\_T.h

## 6.800 Référence de la classe TangenteHyperbolique

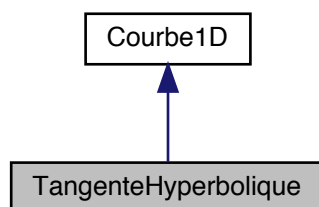
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = a+b*\tanh((x-c)/d)$

```
#include <TangenteHyperbolique.h>
```

Grphe d'héritage de TangenteHyperbolique:



Graphe de collaboration de TangenteHyperbolique:



## Fonctions membres publiques

- **TangenteHyperbolique** (string nom="")
- **TangenteHyperbolique** (const [TangenteHyperbolique](#) &Co)
- **TangenteHyperbolique** (const [Courbe1D](#) &Co)
- void **Affiche** () const  
*affichage de la courbe*
- bool **Complet\_courbe** () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void **LectDonnParticulieres\_courbes** (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void **Info\_commande\_Courbes1D** ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double **Valeur** (double x)  
*ramène la valeur*
- [Courbe1D::ValDer Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double **Derivee** (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2 Valeur\\_Et\\_der12](#) (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double **Der\_sec** (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool Valeur\\_stricte](#) (double x)  
*ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi, ramène le valeur maximale possible de y*
- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double x)  
*ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi, ramène le valeur maximale possible de y et Y' correspondant*
- void **Lecture\_base\_info** (ifstream &ent, const int cas)  
*--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)*
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)  
*cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)*
- void **SchemaXML\_Courbes1D** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)  
*sortie du schemaXML: en fonction de enu*



## Attributs protégés

- double **ax**
- double **bx**
- double **cx**
- double **dx**

## Membres hérités additionnels

### 6.800.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = a+b*\tanh((x-c)/d)$

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x) = a+b*\tanh((x-c)/d)$  ainsi qu'un certain nombre d'information supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

19/01/2001

### 6.800.2 Documentation des fonctions membres

#### 6.800.2.1 Affiche()

```
void TangenteHyperbolique::Affiche ( ) const [virtual]
```

affichage de la courbe  
Implémente [Courbe1D](#).

#### 6.800.2.2 Complet\_courbe()

```
bool TangenteHyperbolique::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon  
Implémente [Courbe1D](#).

#### 6.800.2.3 Der\_sec()

```
double TangenteHyperbolique::Der_sec (
```

double x ) [virtual]

ramène la dérivée seconde  
Implémente [Courbe1D](#).

#### 6.800.2.4 Derivee()

```
double TangenteHyperbolique::Derivee (
```

double x ) [virtual]

ramène la dérivée  
Implémente [Courbe1D](#).

**6.800.2.5 Ecriture\_base\_info()**

```
void TangenteHyperbolique::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.800.2.6 Info\_commande\_Courbes1D()**

```
void TangenteHyperbolique::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

**6.800.2.7 LectDonnParticulieres\_courbes()**

```
void TangenteHyperbolique::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

**6.800.2.8 Lecture\_base\_info()**

```
void TangenteHyperbolique::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

**6.800.2.9 SchemaXML\_Courbes1D()**

```
void TangenteHyperbolique::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

**6.800.2.10 Valeur()**

```
double TangenteHyperbolique::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

**6.800.2.11 Valeur\_Et\_der12()**

```
Courbe1D::ValDer2 TangenteHyperbolique::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

### 6.800.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer TangenteHyperbolique::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

### 6.800.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool TangenteHyperbolique::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

### 6.800.2.14 Valeur\_stricte()

```
Courbe1D::Valbool TangenteHyperbolique::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/TangenteHyperbolique.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/TangenteHyperbolique.cc

## 6.801 Référence de la classe Temps\_CPU\_HZpp

[Temps\\_CPU\\_HZpp](#) une classe dédiée à la gestion des temps d'exécution, il s'agit ici uniquement du temps user.

```
#include <Temps_CPU_HZpp.h>
```

### Fonctions membres publiques

- **Temps\_CPU\_HZpp** (const [Temps\\_CPU\\_HZpp](#) &tps)
- void **Mise\_en\_route\_du\_comptage** ()
- void **Arret\_du\_comptage** ()
- bool **Comptage\_en\_cours** () const
- long long **Temps\_CPU\_User** () const
- long long **Temps\_CPU\_User\_milli** () const
- void **Affiche** (ostream &sort, int niveau=0)
- void **Affiche\_hh\_mn\_s\_ml** (ostream &sort)
- [Temps\\_CPU\\_HZpp](#) & **operator=** (const [Temps\\_CPU\\_HZpp](#) &de)
- [Temps\\_CPU\\_HZpp](#) & **operator+=** (const [Temps\\_CPU\\_HZpp](#) &de)
- [Temps\\_CPU\\_HZpp](#) & **operator-=** (const [Temps\\_CPU\\_HZpp](#) &de)
- istream & **LectXML\_Temps\_CPU\_HZpp** (istream &ent)
- ostream & **EcritXML\_Temps\_CPU\_HZpp** (ostream &sort)
- bool **operator==** (const [Temps\\_CPU\\_HZpp](#) &a) const
- bool **operator!=** (const [Temps\\_CPU\\_HZpp](#) &a) const
- bool **operator>** (const [Temps\\_CPU\\_HZpp](#) &a) const
- bool **operator>=** (const [Temps\\_CPU\\_HZpp](#) &a) const
- bool **operator<** (const [Temps\\_CPU\\_HZpp](#) &a) const
- bool **operator<=** (const [Temps\\_CPU\\_HZpp](#) &a) const
- void **SchemaXML\_Temps\_CPU\_HZpp** (ofstream &sort, const [Enum\\_IO\\_XML](#) enu) const

## Amis

- `istream & operator>>` (`istream &ent`, [Temps\\_CPU\\_HZpp](#) &a)
- `ostream & operator<<` (`ostream &sort`, `const` [Temps\\_CPU\\_HZpp](#) &a)

### 6.801.1 Description détaillée

[Temps\\_CPU\\_HZpp](#) une classe dédiée à la gestion des temps d'exécution, il s'agit ici uniquement du temps user.

Auteur

Gérard Rio

Version

1.0

Date

01/02/2016

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Temps_CPU_HZpp.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Temps_CPU_HZpp.cc`

## 6.802 Référence de la classe Temps\_CPU\_HZpp\_3

[Temps\\_CPU\\_HZpp\\_3](#) une classe dédiée à la gestion des temps d'exécution, ici on cumule 3 compteurs: user, système et réel.

```
#include <Temps_CPU_HZpp_3.h>
```

### Fonctions membres publiques

- `void Mise_en_route_du_comptage ()`
- `void Arret_du_comptage ()`
- `long long Temps_CPU_User () const`
- `long long Temps_CPU_System () const`
- `long long Temps_CPU_Reel () const`
- `void Affiche (ostream &sort, int niveau=0)`
- `void Affiche_hh_mn_s_ml (ostream &sort)`
- `Temps\_CPU\_HZpp\_3 & operator= (const Temps\_CPU\_HZpp\_3 &de)`
- `istream & LectXML_Temps_CPU_HZpp_3 (istream &ent)`
- `ostream & EcritXML_Temps_CPU_HZpp_3 (ostream &sort)`
- `void SchemaXML_Temps_CPU_HZpp_3 (ofstream &sort, const Enum\_IO\_XML enu) const`

## Amis

- `istream & operator>>` (`istream &ent`, [Temps\\_CPU\\_HZpp\\_3](#) &a)
- `ostream & operator<<` (`ostream &sort`, `const` [Temps\\_CPU\\_HZpp\\_3](#) &a)

### 6.802.1 Description détaillée

[Temps\\_CPU\\_HZpp\\_3](#) une classe dédiée à la gestion des temps d'exécution, ici on cumule 3 compteurs: user, système et réel.

Auteur

Gérard Rio

Version

1.0

Date

01/02/2016

La documentation de cette classe a été générée à partir du fichier suivant :

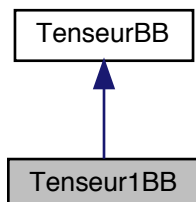
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Temps_CPU_HZpp_3.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Temps_CPU_HZpp_3.cc`

## 6.803 Référence de la classe Tenseur1BB

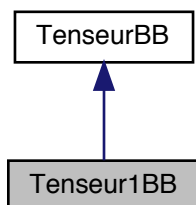
Definition des tenseur derivees de dimension1. cas des composantes deux fois covariantes.

```
#include <Tenseur1.h>
```

Graphe d'héritage de Tenseur1BB:



Graphe de collaboration de Tenseur1BB:



### Fonctions membres publiques

- **Tenseur1BB** (double val)
- **Tenseur1BB** (const [TenseurBB](#) &)
- **Tenseur1BB** (const [Tenseur1BB](#) &B)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurBB](#) & **operator+** (const [TenseurBB](#) &) const  
*operations +*
- void **operator+=** (const [TenseurBB](#) &)  
*operations +=*
- [TenseurBB](#) & **operator-** () const  
*operations -*
- [TenseurBB](#) & **operator-** (const [TenseurBB](#) &) const  
*operations - tens*
- void **operator-=** (const [TenseurBB](#) &)  
*operations -=*
- [TenseurBB](#) & **operator=** (const [TenseurBB](#) &)  
*operations =*
- [TenseurBB](#) & **operator=** (const [Tenseur1BB](#) &B)
- [TenseurBB](#) & **operator\*** (const double &) const

- operations \* double*
- void **operator\***= (const double &)
- operations \*= double*
- **TenseurBB** & **operator/** (const double &) const
- operations /*
- void **operator/=** (const double &)
- operations /=*
- **CoordonneeB** **operator\*** (const **CoordonneeH** &) const
- produit contracte avec un vecteur*
- **TenseurBB** & **operator\*** (const **TenseurHB** &) const
- produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
- **TenseurBH** & **operator\*** (const **TenseurHH** &) const
- idem en BH*
- double **operator&&** (const **TenseurHH** &) const
- produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int **operator==** (const **TenseurBB** &) const
- test*
- int **operator!=** (const **TenseurBB** &) const
- test*
- double **Det** () const
- determinant de la matrice des coordonnees*
- **TenseurBB** & **Transpose** () const
- ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual **TenseurHH** & **Monte2Indices** () const
- manipulation d'indice — -> création de nouveaux tenseurs*
- virtual **TenseurHB** & **MontePremierIndice** () const
- manipulation d'indice — -> création de nouveaux tenseurs*
- virtual **TenseurBH** & **MonteDernierIndice** () const
- manipulation d'indice — -> création de nouveaux tenseurs*
- double **MaxiComposante** () const
- calcul du maximum en valeur absolu des composantes du tenseur*
- void **Affectation\_trans\_dimension** (const **TenseurBB** &B, bool plusZero)
- Affectation\_trans\_dimension.*
- **TenseurHH** & **Inverse** () const
- calcul du tenseur inverse par rapport au produit contracte*
- double & **Coor** (int, int)
- Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double **operator()** (int, int) const
- Retourne la composante i,j du tenseur acces en lecture seulement.*
- istream & **Lecture** (istream &entree)
- lecture et écriture de données*
- ostream & **Ecriture** (ostream &sort) const
- lecture et écriture de données*

## Fonctions membres publiques statiques

- static **TenseurBB** & **Prod\_tensoriel** (const **CoordonneeB** &aB, const **CoordonneeB** &bB)

## Attributs protégés

- listdouble1 lter **ipointe**

## Amis

- istream & **operator**>> (istream &, **Tenseur1BB** &)
- ostream & **operator**<< (ostream &, const **Tenseur1BB** &)

## Membres hérités additionnels

### 6.803.1 Description détaillée

Definition des tenseur derivees de dimension1. cas des composantes deux fois covariantes.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.803.2 Documentation des fonctions membres

#### 6.803.2.1 Affectation\_trans\_dimension()

```
void Tenseur1BB::Affectation_trans_dimension (
    const TenseurBB & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurBB](#).

#### 6.803.2.2 Coor()

```
double & Tenseur1BB::Coor (
    int i,
    int j ) [inline], [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémente [TenseurBB](#).

#### 6.803.2.3 Det()

```
double Tenseur1BB::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees

Implémente [TenseurBB](#).

#### 6.803.2.4 Ecriture()

```
ostream & Tenseur1BB::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurBB](#).

### 6.803.2.5 Inita()

```
void Tenseur1BB::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val  
Implémente [TenseurBB](#).

### 6.803.2.6 Inverse()

```
TenseurHH & Tenseur1BB::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracte  
Implémente [TenseurBB](#).

### 6.803.2.7 Lecture()

```
istream & Tenseur1BB::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données  
Implémente [TenseurBB](#).

### 6.803.2.8 MaxiComposante()

```
double Tenseur1BB::MaxiComposante ( ) const [inline], [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur  
Implémente [TenseurBB](#).

### 6.803.2.9 Monte2Indices()

```
virtual TenseurHH & Tenseur1BB::Monte2Indices ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs  
Implémente [TenseurBB](#).

### 6.803.2.10 MonteDernierIndice()

```
virtual TenseurBH & Tenseur1BB::MonteDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs  
Implémente [TenseurBB](#).

### 6.803.2.11 MontePremierIndice()

```
virtual TenseurHB & Tenseur1BB::MontePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs  
Implémente [TenseurBB](#).

### 6.803.2.12 operator"!=(())

```
int Tenseur1BB::operator!=(
    const TenseurBB & ) const [virtual]
```

test  
Implémente [TenseurBB](#).



**6.803.2.13 operator&&()**

```
double Tenseur1BB::operator&& (
    const TenseurHH & ) const [virtual]
```

produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe  
Implémente [TenseurBB](#).

**6.803.2.14 operator>()()**

```
double Tenseur1BB::operator() (
    int i,
    int j ) const [inline], [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.

Implémente [TenseurBB](#).

**6.803.2.15 operator\*() [1/4]**

```
CoordonneeB Tenseur1BB::operator* (
    const CoordonneeH & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurBB](#).

**6.803.2.16 operator\*() [2/4]**

```
TenseurBB & Tenseur1BB::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurBB](#).

**6.803.2.17 operator\*() [3/4]**

```
TenseurBB & Tenseur1BB::operator* (
    const TenseurHB & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté

Implémente [TenseurBB](#).

**6.803.2.18 operator\*() [4/4]**

```
TenseurBH & Tenseur1BB::operator* (
    const TenseurHH & ) const [virtual]
```

idem en BH

Implémente [TenseurBB](#).

**6.803.2.19 operator\*=( )**

```
void Tenseur1BB::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurBB](#).

**6.803.2.20 operator+()**

```
TenseurBB & Tenseur1BB::operator+ (
    const TenseurBB & ) const [virtual]
```

operations +

Implémente [TenseurBB](#).

**6.803.2.21 operator+=()**

```
void Tenseur1BB::operator+= (
    const TenseurBB & ) [virtual]
```

operations +=

Implémente [TenseurBB](#).

**6.803.2.22 operator-() [1/2]**

```
TenseurBB & Tenseur1BB::operator- ( ) const [virtual]
```

operations -

Implémente [TenseurBB](#).

**6.803.2.23 operator-() [2/2]**

```
TenseurBB & Tenseur1BB::operator- (
    const TenseurBB & ) const [virtual]
```

operations - tens

Implémente [TenseurBB](#).

**6.803.2.24 operator-=()**

```
void Tenseur1BB::operator-= (
    const TenseurBB & ) [virtual]
```

operations -=

Implémente [TenseurBB](#).

**6.803.2.25 operator/()**

```
TenseurBB & Tenseur1BB::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurBB](#).

**6.803.2.26 operator/=()**

```
void Tenseur1BB::operator/= (
    const double & ) [virtual]
```

operations /=

Implémente [TenseurBB](#).

**6.803.2.27 operator=()**

```
TenseurBB & Tenseur1BB::operator= (
    const TenseurBB & ) [virtual]
```

operations =

Implémente [TenseurBB](#).

### 6.803.2.28 operator==()

```
int Tenseur1BB::operator== (
    const TenseurBB & ) const [virtual]
```

test

Implémente [TenseurBB](#).

### 6.803.2.29 Transpose()

```
TenseurBB & Tenseur1BB::Transpose ( ) const [virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libre.

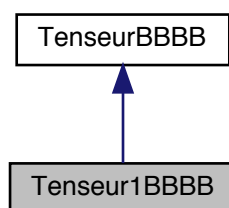
Implémente [TenseurBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

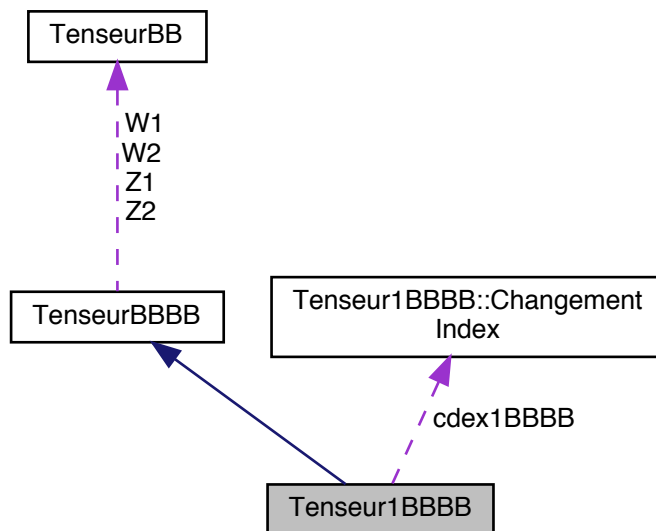
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur1.h

## 6.804 Référence de la classe Tenseur1BBBB

Graphe d'héritage de Tenseur1BBBB:



Graphe de collaboration de Tenseur1BBBB:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur1BBBB** (const double val)
- **Tenseur1BBBB** (bool normal, const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- **Tenseur1BBBB** (const [TenseurBBBB](#) &)
- **Tenseur1BBBB** (const [Tenseur1BBBB](#) &)
- void **Inita** (double val)
- [TenseurBBBB](#) & **operator+** (const [TenseurBBBB](#) &) const
- void **operator+=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator-** () const
- [TenseurBBBB](#) & **operator-** (const [TenseurBBBB](#) &) const
- void **operator-=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator=** (const [Tenseur1HHHH](#) &B)
- [TenseurBBBB](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurBBBB](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBB](#) & **operator&&** (const [TenseurHH](#) &) const
- [TenseurBBBB](#) & **Transpose1et2avec3et4** () const
- void **Affectation\_trans\_dimension** (const [TenseurBBBB](#) &B, bool plusZero)
- void **TransfertDunTenseurGeneral** (const [TenseurBBBB](#) &aBBBB)
- [TenseurHHBB](#) & **Monte2premiersIndices** ()
- [TenseurBBHH](#) & **Monte2derniersIndices** ()
- [TenseurHHHH](#) & **Monte4Indices** ()
- [TenseurBBBB](#) & **Baselocale** ([TenseurBBBB](#) &A, const [BaseB](#) &gi) const
- [TenseurBBBB](#) & **ChangeBase** ([TenseurBBBB](#) &A, const [BaseH](#) &gi) const
- int **operator==** (const [TenseurBBBB](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)

- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const
- `TenseurBB` & `Prod_gauche` (const `TenseurHH` &F) const

### Fonctions membres publiques statiques

- static `TenseurBBBB` & `Prod_tensoriel` (const `TenseurBB` &aBB, const `TenseurBB` &bBB)
- static `TenseurBBBB` & `Prod_tensoriel_barre` (const `TenseurBB` &aBB, const `TenseurBB` &bBB)

### Attributs publics statiques

- static const `ChangementIndex` `cdex1BBBB`  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 1*

### Attributs protégés

- listdouble1 lter `ipointe`

### Amis

- istream & `operator>>` (istream &, `Tenseur1BBBB` &)
- ostream & `operator<<` (ostream &, const `Tenseur1BBBB` &)

### Membres hérités additionnels

#### 6.804.1 Documentation des fonctions membres

##### 6.804.1.1 Affectation\_trans\_dimension()

```
void Tenseur1BBBB::Affectation_trans_dimension (
    const TenseurBBBB & B,
    bool plusZero ) [virtual]
```

Implémente `TenseurBBBB`.

##### 6.804.1.2 Change()

```
void Tenseur1BBBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente `TenseurBBBB`.

##### 6.804.1.3 ChangePlus()

```
void Tenseur1BBBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente `TenseurBBBB`.

#### 6.804.1.4 Ecriture()

```
ostream & Tenseur1BBBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.804.1.5 Inita()

```
void Tenseur1BBBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.804.1.6 Lecture()

```
istream & Tenseur1BBBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.804.1.7 MaxiComposante()

```
double Tenseur1BBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.804.1.8 operator&&()

```
TenseurBB & Tenseur1BBBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.804.1.9 operator>()()

```
double Tenseur1BBBB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.804.1.10 operator\*()

```
TenseurBBBB & Tenseur1BBBB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.804.1.11 operator\*=( )

```
void Tenseur1BBBB::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.12 operator+()**

```
TenseurBBBB & Tenseur1BBBB::operator+ (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.13 operator+=()**

```
void Tenseur1BBBB::operator+= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.14 operator-() [1/2]**

```
TenseurBBBB & Tenseur1BBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.15 operator-() [2/2]**

```
TenseurBBBB & Tenseur1BBBB::operator- (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.16 operator-=()**

```
void Tenseur1BBBB::operator-= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.17 operator/()**

```
TenseurBBBB & Tenseur1BBBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.18 operator/=()**

```
void Tenseur1BBBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.19 operator=()**

```
TenseurBBBB & Tenseur1BBBB::operator= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.804.1.20 operator==(())**

```
int Tenseur1BBBB::operator==(
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.804.1.21 Prod\_gauche()

```
TenseurBB & Tenseur1BBBB::Prod_gauche (
    const TenseurHH & F ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.804.1.22 Transpose1et2avec3et4()

```
TenseurBBBB & Tenseur1BBBB::Transpose1et2avec3et4 ( ) const [virtual]
```

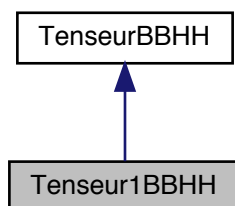
Implémente [TenseurBBBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\\_mai99/Tenseur/TenseurQ-1.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\\_mai99/Tenseur/EnteteTenseur.h](#)

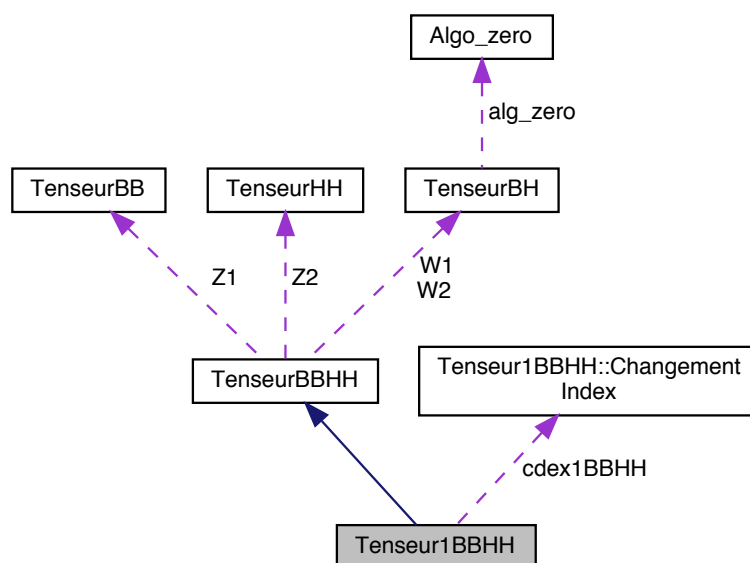
## 6.805 Référence de la classe Tenseur1BBHH

Graphe d'héritage de Tenseur1BBHH:





Graphe de collaboration de Tenseur1BBHH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur1BBHH** (const double val)
- **Tenseur1BBHH** (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)
- **Tenseur1BBHH** (const [TenseurBBHH](#) &)
- **Tenseur1BBHH** (const [Tenseur1BBHH](#) &)
- void **Inita** (double val)
- [TenseurBBHH](#) & **operator+** (const [TenseurBBHH](#) &) const
- void **operator+=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator-** () const
- [TenseurBBHH](#) & **operator-** (const [TenseurBBHH](#) &) const
- void **operator-=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurBBHH](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBB](#) & **operator&&** (const [TenseurBB](#) &) const
- [TenseurHHBB](#) & **Transpose1et2avec3et4** () const
- void **Affectation\_trans\_dimension** (const [TenseurBBHH](#) &B, bool plusZero)
- int **operator==** (const [TenseurBBHH](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort) const
- [TenseurHH](#) & **Prod\_gauche** (const [TenseurHH](#) &F) const

## Fonctions membres publiques statiques

- static [TenseurBBHH](#) & [Prod\\_tensoriel](#) (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)

## Attributs publics statiques

- static const [ChangementIndex](#) [cdex1BBHH](#)  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 1*

## Attributs protégés

- listdouble1Iter [ipointe](#)

## Amis

- istream & [operator](#)>> (istream &, [Tenseur1BBHH](#) &)
- ostream & [operator](#)<< (ostream &, const [Tenseur1BBHH](#) &)

## Membres hérités additionnels

### 6.805.1 Documentation des fonctions membres

#### 6.805.1.1 Affectation\_trans\_dimension()

```
void Tenseur1BBHH::Affectation_trans_dimension (
    const TenseurBBHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.805.1.2 Change()

```
void Tenseur1BBHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.805.1.3 ChangePlus()

```
void Tenseur1BBHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.805.1.4 Ecriture()

```
ostream & Tenseur1BBHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.5 Inita()

```
void Tenseur1BBHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.6 Lecture()

```
istream & Tenseur1BBHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.7 MaxiComposante()

```
double Tenseur1BBHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.8 operator&&()

```
TenseurBB & Tenseur1BBHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.9 operator>()()

```
double Tenseur1BBHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.10 operator\*()

```
TenseurBBHH & Tenseur1BBHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.11 operator\*=( )

```
void Tenseur1BBHH::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.12 operator+()

```
TenseurBBHH & Tenseur1BBHH::operator+ (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.13 operator+=()**

```
void Tenseur1BBHH::operator+= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.14 operator-() [1/2]**

```
TenseurBBHH & Tenseur1BBHH::operator- ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.15 operator-() [2/2]**

```
TenseurBBHH & Tenseur1BBHH::operator- (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.16 operator-=()**

```
void Tenseur1BBHH::operator-= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.17 operator/()**

```
TenseurBBHH & Tenseur1BBHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.18 operator/=()**

```
void Tenseur1BBHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.19 operator=()**

```
TenseurBBHH & Tenseur1BBHH::operator= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.20 operator==( )**

```
int Tenseur1BBHH::operator==(
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.805.1.21 Prod\_gauche()**

```
TenseurHH & Tenseur1BBHH::Prod_gauche (
    const TenseurHH & F ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.805.1.22 Transpose1et2avec3et4()

[TenseurHHBB](#) & [Tenseur1BBHH](#)::Transpose1et2avec3et4 ( ) const [virtual]

Implémente [TenseurBBHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

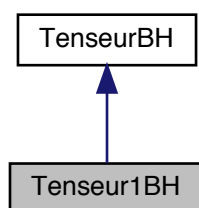
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.806 Référence de la classe Tenseur1BH

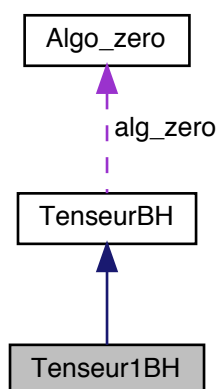
Definition des tenseur derivees de dimension1. cas des composantes mixtes BH.

```
#include <Tenseur1.h>
```

Graphe d'héritage de Tenseur1BH:



Graphe de collaboration de Tenseur1BH:



### Fonctions membres publiques

- [Tenseur1BH](#) (double val)
- [Tenseur1BH](#) (const [TenseurBH](#) &)

- **Tenseur1BH** (const [Tenseur1BH](#) &B)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- **TenseurBH & operator+** (const [TenseurBH](#) &) const  
*operations +*
- void **operator+=** (const [TenseurBH](#) &)  
*operations +=*
- **TenseurBH & operator-** () const  
*operations opposé*
- **TenseurBH & operator-** (const [TenseurBH](#) &) const  
*operations -*
- void **operator-=** (const [TenseurBH](#) &)  
*operations -=*
- **TenseurBH & operator=** (const [TenseurBH](#) &)  
*operations =*
- **TenseurBH & operator=** (const [Tenseur1BH](#) &B)
- **TenseurBH & operator\*** (const double &) const  
*operations \**
- void **operator\*=** (const double &)  
*operations \*=*
- **TenseurBH & operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- **CoordonneeB operator\*** (const [CoordonneeB](#) &) const  
*produit contracte avec un vecteur*
- **TenseurBB & operator\*** (const [TenseurBB](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
- **TenseurBH & operator\*** (const [TenseurBH](#) &) const
- double **operator&&** (const [TenseurBH](#) &) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- double **Trace** () const  
*trace du tenseur ou premier invariant*
- double **II** () const  
*second invariant = trace (A\*A)*
- double **III** () const  
*troisieme invariant = trace ((A\*A)\*A)*
- double **Det** () const  
*determinant de la matrice des coordonnees*
- virtual **Coordonnee ValPropre** (int &cas) const  
*calcul des valeurs propres*
- virtual **Coordonnee ValPropre** (int &cas, [Mat\\_pleine](#) &mat) const  
*calcul des valeurs propres*
- virtual void **VecteursPropres** (const [Coordonnee](#) &Val\_P, int &cas, [Tableau](#)<[Coordonnee](#) > &V\_P) const  
*calcul des vecteurs propres, les valeurs propres étant déjà connues*
- int **operator==** (const [TenseurBH](#) &) const  
*test*
- int **operator!=** (const [TenseurBH](#) &) const
- void **Affectation\_trans\_dimension** (const [TenseurBH](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- **TenseurBH & Inverse** () const  
*calcul du tenseur inverse par rapport au produit contracte*
- **TenseurHB & Transpose** () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- **TenseurHB & Identique** () const
- void **PermuteHautBas** ()  
*permute Bas Haut, mais reste dans le même tenseur*
- double **MaxiComposante** () const

- *calcul du maximum en valeur absolu des composantes du tenseur*
- double & [Coo](#) (int, int)
- *Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double [operator\(\)](#) (int, int) const
- *Retourne la composante i,j du tenseur acces en lecture seulement.*
- istream & [Lecture](#) (istream &entree)
- *lecture et écriture de données*
- ostream & [Ecriture](#) (ostream &sort) const
- *lecture et écriture de données*

## Fonctions membres publiques statiques

- static [TenseurBH](#) & [Prod\\_tensoriel](#) (const [CoordonneeB](#) &aB, const [CoordonneeH](#) &bH)

## Attributs protégés

- listdouble1Iter [ipointe](#)

## Amis

- class [Tenseur3BH](#)
- istream & [operator](#)>> (istream &, [Tenseur1BH](#) &)
- ostream & [operator](#)<< (ostream &, const [Tenseur1BH](#) &)

## Membres hérités additionnels

### 6.806.1 Description détaillée

Definition des tenseur derivees de dimension1. cas des composantes mixtes BH.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

### 6.806.2 Documentation des fonctions membres

#### 6.806.2.1 Affectation\_trans\_dimension()

```
void Tenseur1BH::Affectation_trans_dimension (
    const TenseurBH & B,
    bool plusZero ) [virtual]
```

[Affectation\\_trans\\_dimension](#).

affectation de B dans this, *plusZero* = false: les données manquantes sont inchangées,  
*plusZero* = true: les données manquantes sont mises à 0  
 si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
 des données possibles  
 Implémente [TenseurBH](#).

### 6.806.2.2 Coor()

```
double & Tenseur1BH::Coor (
    int i,
    int j ) [inline], [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture et en ecriture.

Implémente [TenseurBH](#).

### 6.806.2.3 Det()

```
double Tenseur1BH::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees

Implémente [TenseurBH](#).

### 6.806.2.4 Ecriture()

```
ostream & Tenseur1BH::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurBH](#).

### 6.806.2.5 II()

```
double Tenseur1BH::II ( ) const [virtual]
```

second invariant = trace (A\*A)

Implémente [TenseurBH](#).

### 6.806.2.6 III()

```
double Tenseur1BH::III ( ) const [virtual]
```

troisieme invariant = trace ((A\*A)\*A)

Implémente [TenseurBH](#).

### 6.806.2.7 Inita()

```
void Tenseur1BH::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurBH](#).

### 6.806.2.8 Inverse()

```
TenseurBH & Tenseur1BH::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracte

Implémente [TenseurBH](#).

### 6.806.2.9 Lecture()

```
istream & Tenseur1BH::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurBH](#).



**6.806.2.10 MaxiComposante()**

```
double Tenseur1BH::MaxiComposante ( ) const [inline], [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur  
Implémente [TenseurBH](#).

**6.806.2.11 operator"!=( )**

```
int Tenseur1BH::operator!=(
```

```
    const TenseurBH & ) const [virtual]
```

Implémente [TenseurBH](#).

**6.806.2.12 operator&&( )**

```
double Tenseur1BH::operator&& (
```

```
    const TenseurBH & ) const [virtual]
```

produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe  
Implémente [TenseurBH](#).

**6.806.2.13 operator()( )**

```
double Tenseur1BH::operator() (
```

```
    int i,
```

```
    int j ) const [inline], [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.  
Implémente [TenseurBH](#).

**6.806.2.14 operator\*( ) [1/4]**

```
CoordonneeB Tenseur1BH::operator* (
```

```
    const CoordonneeB & ) const [virtual]
```

produit contracte avec un vecteur  
Implémente [TenseurBH](#).

**6.806.2.15 operator\*( ) [2/4]**

```
TenseurBH & Tenseur1BH::operator* (
```

```
    const double & ) const [virtual]
```

operations \*  
Implémente [TenseurBH](#).

**6.806.2.16 operator\*( ) [3/4]**

```
TenseurBB & Tenseur1BH::operator* (
```

```
    const TenseurBB & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté  
Implémente [TenseurBH](#).

**6.806.2.17 operator\*( ) [4/4]**

```
TenseurBH & Tenseur1BH::operator* (
```

```
    const TenseurBH & ) const [virtual]
```

Implémente [TenseurBH](#).

### 6.806.2.18 operator\*=( )

```
void Tenseur1BH::operator*= (
    const double & ) [virtual]
```

operations \*=

Implémente [TenseurBH](#).

### 6.806.2.19 operator+( )

```
TenseurBH & Tenseur1BH::operator+ (
    const TenseurBH & ) const [virtual]
```

operations +

Implémente [TenseurBH](#).

### 6.806.2.20 operator+=( )

```
void Tenseur1BH::operator+= (
    const TenseurBH & ) [virtual]
```

operations +=

Implémente [TenseurBH](#).

### 6.806.2.21 operator-( ) [1/2]

```
TenseurBH & Tenseur1BH::operator- ( ) const [virtual]
```

operations opposé

Implémente [TenseurBH](#).

### 6.806.2.22 operator-( ) [2/2]

```
TenseurBH & Tenseur1BH::operator- (
    const TenseurBH & ) const [virtual]
```

operations -

Implémente [TenseurBH](#).

### 6.806.2.23 operator-=( )

```
void Tenseur1BH::operator-=(
    const TenseurBH & ) [virtual]
```

operations -=

Implémente [TenseurBH](#).

### 6.806.2.24 operator/( )

```
TenseurBH & Tenseur1BH::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurBH](#).

**6.806.2.25 operator/=( )**

```
void Tenseur1BH::operator/= (
    const double & ) [virtual]
```

operations /=

Implémente [TenseurBH](#).

**6.806.2.26 operator=( )**

```
TenseurBH & Tenseur1BH::operator= (
    const TenseurBH & ) [virtual]
```

operations =

Implémente [TenseurBH](#).

**6.806.2.27 operator==( )**

```
int Tenseur1BH::operator== (
    const TenseurBH & ) const [virtual]
```

test

Implémente [TenseurBH](#).

**6.806.2.28 PermuteHautBas( )**

```
void Tenseur1BH::PermuteHautBas ( ) [inline], [virtual]
```

permute Bas Haut, mais reste dans le même tenseur

Implémente [TenseurBH](#).

**6.806.2.29 Trace( )**

```
double Tenseur1BH::Trace ( ) const [virtual]
```

trace du tenseur ou premier invariant

Implémente [TenseurBH](#).

**6.806.2.30 Transpose( )**

```
TenseurHB & Tenseur1BH::Transpose ( ) const [virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

Implémente [TenseurBH](#).

**6.806.2.31 ValPropre( ) [1/2]**

```
virtual Coordonnee Tenseur1BH::ValPropre (
    int & cas ) const [virtual]
```

calcul des valeurs propres

valeurs propre dans le vecteur de retour, classée par ordres "décroissants"

cas indique le cas de valeur propre:

quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire

dans ce cas les valeurs propres de retour sont nulles par défaut

dim = 1, cas=1 pour une valeur propre;

dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques

dim = 3 , cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)

, cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),

, cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du retour)  
 , cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du retour)  
 Implémente [TenseurBH](#).

### 6.806.2.32 ValPropre() [2/2]

```
virtual Coordonnee Tenseur1BH::ValPropre (
    int & cas,
    Mat_pleine & mat ) const [virtual]
```

calcul des valeurs propres

idem met en retour la matrice mat contiend par colonne les vecteurs propre  
 elle doit avoir la dimension du tenseur  
 les vecteurs propre sont exprime dans le repere dual (contrairement au HB) pour les tenseurs dim 3  
 pour dim=2:le premier vecteur propre est exprime dans le repere dual  
 le second vecteur propre est exprimé dans le repère naturel  
 pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
 Implémente [TenseurBH](#).

### 6.806.2.33 VecteursPropres()

```
virtual void Tenseur1BH::VecteursPropres (
    const Coordonnee & Val_P,
    int & cas,
    Tableau< Coordonnee > & V_P ) const [virtual]
```

calcul des vecteurs propres, les valeurs propres  
 étant déjà connues

ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres  
 étant déjà connues  
 en retour VP les vecteurs propre : doivent avoir la dimension du tenseur  
 les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3  
 pour dim=2:le premier vecteur propre est exprime dans le repere naturel  
 le second vecteur propre est exprimé dans le repère dual  
 pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
 en sortie cas = -1 s'il y a eu un problème, dans ce cas, V\_P est quelconque  
 sinon si tout est ok, cas est identique en sortie avec l'entrée  
 Implémente [TenseurBH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

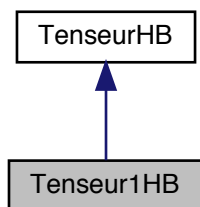
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur1.h

## 6.807 Référence de la classe Tenseur1HB

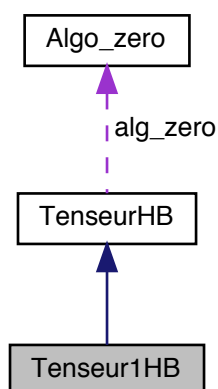
Definition des tenseur derivees de dimension1. cas des composantes mixtes HB.

```
#include <Tenseur1.h>
```

Graphe d'héritage de Tenseur1HB:



Graphe de collaboration de Tenseur1HB:



## Fonctions membres publiques

- **Tenseur1HB** (double val)
- **Tenseur1HB** (const [TenseurHB](#) &)
- **Tenseur1HB** (const [Tenseur1HB](#) &B)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurHB](#) & **operator+** (const [TenseurHB](#) &) const  
*operations +*
- void **operator+=** (const [TenseurHB](#) &)  
*operations +=*
- [TenseurHB](#) & **operator-** () const  
*operations opposé*
- [TenseurHB](#) & **operator-** (const [TenseurHB](#) &) const  
*operations -*
- void **operator-=** (const [TenseurHB](#) &)  
*operations -=*
- [TenseurHB](#) & **operator=** (const [TenseurHB](#) &)  
*operations =*

- [TenseurHB](#) & **operator=** (const [Tenseur1HB](#) &B)
- [TenseurHB](#) & **operator\*** (const double &) const  
*operations \**
- void **operator\*=** (const double &)  
*operations \*=*
- [TenseurHB](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- [CoordonneeH](#) **operator\*** (const [CoordonneeH](#) &) const  
*produit contracte avec un vecteur*
- [TenseurHH](#) & **operator\*** (const [TenseurHH](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
- [TenseurHB](#) & **operator\*** (const [TenseurHB](#) &) const
- double **operator&&** (const [TenseurHB](#) &) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- double **Trace** () const  
*trace du tenseur ou premier invariant*
- double **II** () const  
*second invariant = trace (A\*A)*
- double **III** () const  
*troisieme invariant = trace ((A\*A)\*A)*
- double **Det** () const  
*determinant de la matrice des coordonnees*
- virtual [Coordonnee ValPropre](#) (int &cas) const  
*calcul des valeurs propres*
- virtual [Coordonnee ValPropre](#) (int &cas, [Mat\\_pleine](#) &mat) const  
*calcul des valeurs propres*
- virtual void **VecteursPropres** (const [Coordonnee](#) &Val\_P, int &cas, [Tableau](#)< [Coordonnee](#) > &V\_P) const  
*calcul des vecteurs propres, les valeurs propres étant déjà connues*
- int **operator==** (const [TenseurHB](#) &) const  
*test*
- int **operator!=** (const [TenseurHB](#) &) const
- void **Affectation\_trans\_dimension** (const [TenseurHB](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- [TenseurHB](#) & **Inverse** () const  
*calcul du tenseur inverse par rapport au produit contracte*
- [TenseurBH](#) & **Transpose** () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- [TenseurBH](#) & **Identique** () const
- void **PermuteHautBas** ()  
*permute Bas Haut, mais reste dans le même tenseur*
- double **MaxiComposante** () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- double & **Coor** (int, int)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double **operator()** (int, int) const  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
- istream & **Lecture** (istream &entree)  
*lecture et écriture de données*
- ostream & **Ecriture** (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- static [TenseurHB](#) & **Prod\_tensoriel** (const [CoordonneeH](#) &aH, const [CoordonneeB](#) &bB)

## Attributs protégés

- listdouble1Iter **ipointe**

## Amis

- istream & **operator**>> (istream &, [Tenseur1HB](#) &)
- ostream & **operator**<< (ostream &, const [Tenseur1HB](#) &)

## Membres hérités additionnels

### 6.807.1 Description détaillée

Definition des tenseur derivees de dimension1. cas des composantes mixtes HB.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.807.2 Documentation des fonctions membres

#### 6.807.2.1 Affectation\_trans\_dimension()

```
void Tenseur1HB::Affectation_trans_dimension (
    const TenseurHB & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurHB](#).

#### 6.807.2.2 Coor()

```
double & Tenseur1HB::Coor (
    int i,
    int j ) [inline], [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.  
Implémente [TenseurHB](#).

#### 6.807.2.3 Det()

```
double Tenseur1HB::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees  
Implémente [TenseurHB](#).

#### 6.807.2.4 Ecriture()

```
ostream & Tenseur1HB::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurHB](#).

#### 6.807.2.5 II()

```
double Tenseur1HB::II ( ) const [virtual]
```

second invariant = trace (A\*A)

Implémente [TenseurHB](#).

#### 6.807.2.6 III()

```
double Tenseur1HB::III ( ) const [virtual]
```

troisieme invariant = trace ((A\*A)\*A)

Implémente [TenseurHB](#).

#### 6.807.2.7 Inita()

```
void Tenseur1HB::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurHB](#).

#### 6.807.2.8 Inverse()

```
TenseurHB & Tenseur1HB::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracte

Implémente [TenseurHB](#).

#### 6.807.2.9 Lecture()

```
istream & Tenseur1HB::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurHB](#).

#### 6.807.2.10 MaxiComposante()

```
double Tenseur1HB::MaxiComposante ( ) const [inline], [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurHB](#).

#### 6.807.2.11 operator"!=(())

```
int Tenseur1HB::operator!=(
    const TenseurHB & ) const [virtual]
```

Implémente [TenseurHB](#).



**6.807.2.12 operator&&()**

```
double Tenseur1HB::operator&& (
    const TenseurHB & ) const [virtual]
```

produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe  
Implémente [TenseurHB](#).

**6.807.2.13 operator>()()**

```
double Tenseur1HB::operator() (
    int i,
    int j ) const [inline], [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.

Implémente [TenseurHB](#).

**6.807.2.14 operator\*() [1/4]**

```
CoordonneeH Tenseur1HB::operator* (
    const CoordonneeH & ) const [virtual]
```

produit contracté avec un vecteur

Implémente [TenseurHB](#).

**6.807.2.15 operator\*() [2/4]**

```
TenseurHB & Tenseur1HB::operator* (
    const double & ) const [virtual]
```

operations \*

Implémente [TenseurHB](#).

**6.807.2.16 operator\*() [3/4]**

```
TenseurHB & Tenseur1HB::operator* (
    const TenseurHB & ) const [virtual]
```

Implémente [TenseurHB](#).

**6.807.2.17 operator\*() [4/4]**

```
TenseurHH & Tenseur1HB::operator* (
    const TenseurHH & ) const [virtual]
```

produit contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté

Implémente [TenseurHB](#).

**6.807.2.18 operator\*=( )**

```
void Tenseur1HB::operator*= (
    const double & ) [virtual]
```

operations \*=

Implémente [TenseurHB](#).

**6.807.2.19 operator+()**

```
TenseurHB & Tenseur1HB::operator+ (
    const TenseurHB & ) const [virtual]
```

operations +  
Implémente [TenseurHB](#).

#### 6.807.2.20 operator+=()

```
void Tenseur1HB::operator+= (
    const TenseurHB & ) [virtual]
```

operations +=  
Implémente [TenseurHB](#).

#### 6.807.2.21 operator-() [1/2]

```
TenseurHB & Tenseur1HB::operator- ( ) const [virtual]
```

operations opposé  
Implémente [TenseurHB](#).

#### 6.807.2.22 operator-() [2/2]

```
TenseurHB & Tenseur1HB::operator- (
    const TenseurHB & ) const [virtual]
```

operations -  
Implémente [TenseurHB](#).

#### 6.807.2.23 operator-=()

```
void Tenseur1HB::operator-= (
    const TenseurHB & ) [virtual]
```

operations -=  
Implémente [TenseurHB](#).

#### 6.807.2.24 operator/()

```
TenseurHB & Tenseur1HB::operator/ (
    const double & ) const [virtual]
```

operations /  
Implémente [TenseurHB](#).

#### 6.807.2.25 operator/=()

```
void Tenseur1HB::operator/= (
    const double & ) [virtual]
```

operations /=  
Implémente [TenseurHB](#).

#### 6.807.2.26 operator=()

```
TenseurHB & Tenseur1HB::operator= (
    const TenseurHB & ) [virtual]
```

operations =  
Implémente [TenseurHB](#).

**6.807.2.27 operator==( )**

```
int Tenseur1HB::operator==(
    const TenseurHB & ) const [virtual]
```

test

Implémente [TenseurHB](#).**6.807.2.28 PermuteHautBas()**

```
void Tenseur1HB::PermuteHautBas ( ) [inline], [virtual]
```

permuté Bas Haut, mais reste dans le même tenseur

Implémente [TenseurHB](#).**6.807.2.29 Trace()**

```
double Tenseur1HB::Trace ( ) const [virtual]
```

trace du tenseur ou premier invariant

Implémente [TenseurHB](#).**6.807.2.30 Transpose()**

```
TenseurBH & Tenseur1HB::Transpose ( ) const [virtual]
```

ATTENTION création d'un tenseur transpose qui est supprimé par Libere.

Implémente [TenseurHB](#).**6.807.2.31 ValPropre() [1/2]**

```
virtual Coordonnee Tenseur1HB::ValPropre (
    int & cas ) const [virtual]
```

calcul des valeurs propres

valeurs propre dans le vecteur de retour, classée par ordres "décroissants"

cas indique le cas de valeur propre:

quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire dans ce cas les valeurs propres de retour sont nulles par défaut

dim = 1, cas=1 pour une valeur propre;

dim = 2, cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques

dim = 3, cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)

, cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),

, cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du retour)

, cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du retour)

Implémente [TenseurHB](#).**6.807.2.32 ValPropre() [2/2]**

```
virtual Coordonnee Tenseur1HB::ValPropre (
    int & cas,
    Mat_pleine & mat ) const [virtual]
```

calcul des valeurs propres

idem met en retour la matrice mat contient par colonne les vecteurs propre

elle doit avoir la dimension du tenseur

les vecteurs propre sont exprimés dans le repère naturel (pour les tenseurs dim 3

pour dim=2:le premier vecteur propre est exprimé dans le repère naturel

le second vecteur propre est exprimé dans le repère dual  
 pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
 Implémente [TenseurHB](#).

### 6.807.2.33 VecteursPropres()

```
virtual void Tenseur1HB::VecteursPropres (
    const Coordonnee & Val_P,
    int & cas,
    Tableau< Coordonnee > & V_P ) const [virtual]
```

calcul des vecteurs propres, les valeurs propres  
 étant déjà connues

ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres  
 étant déjà connues

en retour VP les vecteurs propre : doivent avoir la dimension du tenseur  
 les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3  
 pour dim=2:le premier vecteur propre est exprime dans le repere naturel  
 le second vecteur propre est exprimé dans le repère dual  
 pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
 en sortie cas = -1 s'il y a eu un problème, dans ce cas, V\_P est quelconque  
 sinon si tout est ok, cas est identique en sortie avec l'entrée  
 Implémente [TenseurHB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

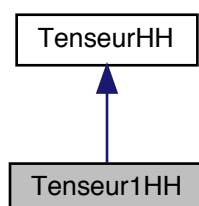
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur1.h

## 6.808 Référence de la classe Tenseur1HH

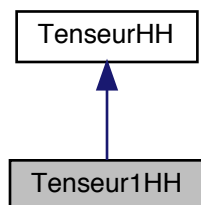
Definition des tenseur derivees de dimension1. cas des composantes deux fois contravariantes.

```
#include <Tenseur1.h>
```

Graphe d'héritage de Tenseur1HH:



Graphe de collaboration de Tenseur1HH:



## Fonctions membres publiques

- **Tenseur1HH** (double val)
- **Tenseur1HH** (const [TenseurHH](#) &)
- **Tenseur1HH** (const [Tenseur1HH](#) &B)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurHH](#) & **operator+** (const [TenseurHH](#) &) const  
*operations +*
- void **operator+=** (const [TenseurHH](#) &)  
*operations +=*
- [TenseurHH](#) & **operator-** () const  
*operations -*
- [TenseurHH](#) & **operator-** (const [TenseurHH](#) &) const  
*operations - tens*
- void **operator-=** (const [TenseurHH](#) &)  
*operations -=*
- [TenseurHH](#) & **operator=** (const [TenseurHH](#) &)  
*operations =*
- [TenseurHH](#) & **operator=** (const [Tenseur1HH](#) &B)
- [TenseurHH](#) & **operator\*** (const double &) const  
*operations \* double*
- void **operator\*=** (const double &)  
*operations \*= double*
- [TenseurHH](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations +=*
- [CoordonneeH](#) **operator\*** (const [CoordonneeB](#) &) const  
*produit contracte avec un vecteur*
- [TenseurHH](#) & **operator\*** (const [TenseurBH](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté avec BH*
- [TenseurHB](#) & **operator\*** (const [TenseurBB](#) &) const  
*idem avec BB*
- double **operator&&** (const [TenseurBB](#) &) const  
*produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int **operator==** (const [TenseurHH](#) &) const  
*test*
- int **operator!=** (const [TenseurHH](#) &) const  
*test*
- double **Det** () const  
*determinant de la matrice des coordonnees*

- [TenseurHH & Transpose](#) () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- double [MaxiComposante](#) () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- virtual [TenseurBB & Baisse2Indices](#) () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual [TenseurBH & BaissePremierIndice](#) () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual [TenseurHB & BaisseDernierIndice](#) () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- void [Affectation\\_trans\\_dimension](#) (const [TenseurHH](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- [TenseurBB & Inverse](#) () const  
*calcul du tenseur inverse par rapport au produit contracte*
- double & [Coor](#) (int, int)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double [operator\(\)](#) (int, int) const  
*Retourne la composante i,j du tenseur acces en lecture seule.*
- istream & [Lecture](#) (istream &entree)  
*lecture et écriture de données*
- ostream & [Ecriture](#) (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- static [TenseurHH](#) & [Prod\\_tensoriel](#) (const [CoordonneeH](#) &aH, const [CoordonneeH](#) &bH)

## Attributs protégés

- listdouble1Iter [ipointe](#)

## Amis

- istream & [operator](#)>> (istream &, [Tenseur1HH](#) &)
- ostream & [operator](#)<< (ostream &, const [Tenseur1HH](#) &)

## Membres hérités additionnels

### 6.808.1 Description détaillée

Definition des tenseur derivees de dimension1. cas des composantes deux fois contravariantes.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

### 6.808.2 Documentation des fonctions membres

### 6.808.2.1 Affectation\_trans\_dimension()

```
void Tenseur1HH::Affectation_trans_dimension (
    const TenseurHH & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles

Implémente [TenseurHH](#).

### 6.808.2.2 Baisse2Indices()

```
virtual TenseurBB & Tenseur1HH::Baisse2Indices ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

### 6.808.2.3 BaisseDernierIndice()

```
virtual TenseurHB & Tenseur1HH::BaisseDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

### 6.808.2.4 BaissePremierIndice()

```
virtual TenseurBH & Tenseur1HH::BaissePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

### 6.808.2.5 Coor()

```
double & Tenseur1HH::Coor (
    int i,
    int j ) [inline], [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémente [TenseurHH](#).

### 6.808.2.6 Det()

```
double Tenseur1HH::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees

Implémente [TenseurHH](#).

### 6.808.2.7 Ecriture()

```
ostream & Tenseur1HH::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurHH](#).

**6.808.2.8 Inita()**

```
void Tenseur1HH::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val  
Implémente [TenseurHH](#).

**6.808.2.9 Inverse()**

```
TenseurBB & Tenseur1HH::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracté  
Implémente [TenseurHH](#).

**6.808.2.10 Lecture()**

```
istream & Tenseur1HH::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données  
Implémente [TenseurHH](#).

**6.808.2.11 MaxiComposante()**

```
double Tenseur1HH::MaxiComposante ( ) const [inline], [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur  
Implémente [TenseurHH](#).

**6.808.2.12 operator"!=(())**

```
int Tenseur1HH::operator!=(
    const TenseurHH & ) const [virtual]
```

test  
Implémente [TenseurHH](#).

**6.808.2.13 operator&&()**

```
double Tenseur1HH::operator&& (
    const TenseurBB & ) const [virtual]
```

produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe  
Implémente [TenseurHH](#).

**6.808.2.14 operator()()**

```
double Tenseur1HH::operator() (
    int i,
    int j ) const [inline], [virtual]
```

Retourne la composante i,j du tenseur acces en lecture seule.  
Implémente [TenseurHH](#).

**6.808.2.15 operator\*() [1/4]**

```
CoordonneeH Tenseur1HH::operator* (
    const CoordonneeB & ) const [virtual]
```

produit contracté avec un vecteur



Implémente [TenseurHH](#).

#### 6.808.2.16 operator\*() [2/4]

```
TenseurHH & Tenseur1HH::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurHH](#).

#### 6.808.2.17 operator\*() [3/4]

```
TenseurHB & Tenseur1HH::operator* (
    const TenseurBB & ) const [virtual]
```

idem avec BB

Implémente [TenseurHH](#).

#### 6.808.2.18 operator\*() [4/4]

```
TenseurHH & Tenseur1HH::operator* (
    const TenseurBH & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B \rightarrow$  donc c'est l'indice du milieu qui est contracté avec BH

Implémente [TenseurHH](#).

#### 6.808.2.19 operator\*=( )

```
void Tenseur1HH::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurHH](#).

#### 6.808.2.20 operator+( )

```
TenseurHH & Tenseur1HH::operator+ (
    const TenseurHH & ) const [virtual]
```

operations +

Implémente [TenseurHH](#).

#### 6.808.2.21 operator+=( )

```
void Tenseur1HH::operator+= (
    const TenseurHH & ) [virtual]
```

operations +=

Implémente [TenseurHH](#).

#### 6.808.2.22 operator-( ) [1/2]

```
TenseurHH & Tenseur1HH::operator- ( ) const [virtual]
```

operations -

Implémente [TenseurHH](#).

**6.808.2.23 operator-() [2/2]**

```
TenseurHH & Tenseur1HH::operator- (
    const TenseurHH & ) const [virtual]
```

operations - tens

Implémente [TenseurHH](#).

**6.808.2.24 operator-=()**

```
void Tenseur1HH::operator-= (
    const TenseurHH & ) [virtual]
```

operations -=

Implémente [TenseurHH](#).

**6.808.2.25 operator/()**

```
TenseurHH & Tenseur1HH::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurHH](#).

**6.808.2.26 operator/=()**

```
void Tenseur1HH::operator/= (
    const double & ) [virtual]
```

operations +=

Implémente [TenseurHH](#).

**6.808.2.27 operator=()**

```
TenseurHH & Tenseur1HH::operator= (
    const TenseurHH & ) [virtual]
```

operations =

Implémente [TenseurHH](#).

**6.808.2.28 operator==(())**

```
int Tenseur1HH::operator==(
    const TenseurHH & ) const [virtual]
```

test

Implémente [TenseurHH](#).

**6.808.2.29 Transpose()**

```
TenseurHH & Tenseur1HH::Transpose ( ) const [virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libre.

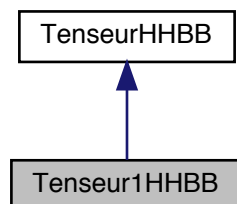
Implémente [TenseurHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

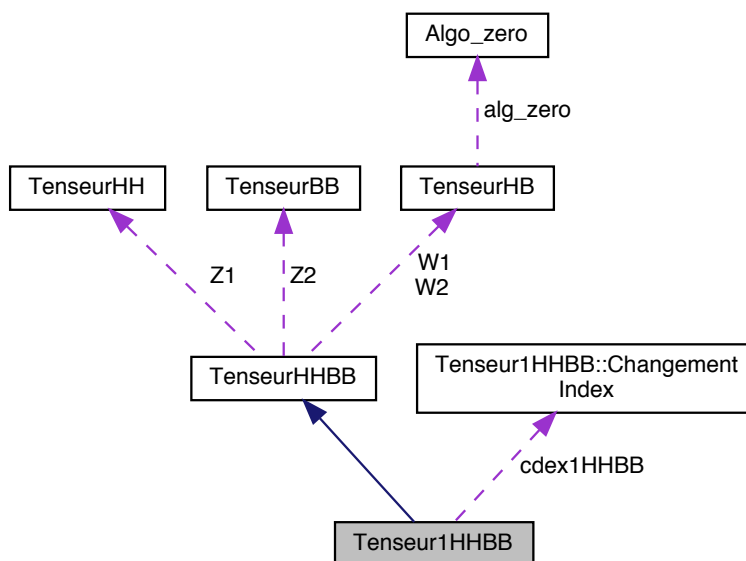
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur1.h

## 6.809 Référence de la classe Tenseur1HHBB

Graphe d'héritage de Tenseur1HHBB:



Graphe de collaboration de Tenseur1HHBB:



### Classes

- class [ChangementIndex](#)

### Fonctions membres publiques

- **Tenseur1HHBB** (const double val)
- **Tenseur1HHBB** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)
- **Tenseur1HHBB** (const [TenseurHHBB](#) &)
- **Tenseur1HHBB** (const [Tenseur1HHBB](#) &)
- void **Inita** (double val)
- [TenseurHHBB](#) & **operator+** (const [TenseurHHBB](#) &) const
- void **operator+=** (const [TenseurHHBB](#) &)

- [TenseurHHBB](#) & [operator-](#) () const
- [TenseurHHBB](#) & [operator-](#) (const [TenseurHHBB](#) &) const
- void [operator-=](#) (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & [operator=](#) (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & [operator\\*](#) (const double &) const
- void [operator\\*=](#) (const double &)
- [TenseurHHBB](#) & [operator/](#) (const double &) const
- void [operator/=](#) (const double &)
- [TenseurHH](#) & [operator&&](#) (const [TenseurHH](#) &) const
- [TenseurBBHH](#) & [Transpose1et2avec3et4](#) () const
- void [Affectation\\_trans\\_dimension](#) (const [TenseurHHBB](#) &B, bool plusZero)
- int [operator==](#) (const [TenseurHHBB](#) &) const
- void [Change](#) (int i, int j, int k, int l, const double &val)
- void [ChangePlus](#) (int i, int j, int k, int l, const double &val)
- double [operator\(\)](#) (int i, int j, int k, int l) const
- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort) const
- [TenseurBB](#) & [Prod\\_gauche](#) (const [TenseurBB](#) &F) const

## Fonctions membres publiques statiques

- static [TenseurHHBB](#) & [Prod\\_tensoriel](#) (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)

## Attributs publics statiques

- static const [ChangementIndex](#) [cdex1HHBB](#)  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 1*

## Attributs protégés

- listdouble1lter [ipointe](#)

## Amis

- istream & [operator>>](#) (istream &, [Tenseur1HHBB](#) &)
- ostream & [operator<<](#) (ostream &, const [Tenseur1HHBB](#) &)

## Membres hérités additionnels

### 6.809.1 Documentation des fonctions membres

#### 6.809.1.1 [Affectation\\_trans\\_dimension\(\)](#)

```
void Tenseur1HHBB::Affectation_trans_dimension (
    const TenseurHHBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.809.1.2 [Change\(\)](#)

```
void Tenseur1HHBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.809.1.3 ChangePlus()

```
void Tenseur1HHBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.809.1.4 Ecriture()

```
ostream & Tenseur1HHBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.809.1.5 Inita()

```
void Tenseur1HHBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.809.1.6 Lecture()

```
istream & Tenseur1HHBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.809.1.7 MaxiComposante()

```
double Tenseur1HHBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.809.1.8 operator&&()

```
TenseurHH & Tenseur1HHBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.809.1.9 operator>()()

```
double Tenseur1HHBB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.809.1.10 operator\*()

```
TenseurHHBB & Tenseur1HHBB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.11 operator\*=( )**

```
void Tenseur1HHBB::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.12 operator+( )**

```
TenseurHHBB & Tenseur1HHBB::operator+ (
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.13 operator+=( )**

```
void Tenseur1HHBB::operator+= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.14 operator-( ) [1/2]**

```
TenseurHHBB & Tenseur1HHBB::operator- ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.15 operator-( ) [2/2]**

```
TenseurHHBB & Tenseur1HHBB::operator- (
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.16 operator-=( )**

```
void Tenseur1HHBB::operator-=(
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.17 operator/( )**

```
TenseurHHBB & Tenseur1HHBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.18 operator/=( )**

```
void Tenseur1HHBB::operator/=(
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.19 operator=()**

```
TenseurHHBB & Tenseur1HHBB::operator= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.20 operator==()**

```
int Tenseur1HHBB::operator== (
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.21 Prod\_gauche()**

```
TenseurBB & Tenseur1HHBB::Prod_gauche (
    const TenseurBB & F ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.809.1.22 Transpose1et2avec3et4()**

```
TenseurBBHH & Tenseur1HHBB::Transpose1et2avec3et4 ( ) const [virtual]
```

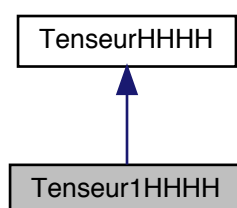
Implémente [TenseurHHBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

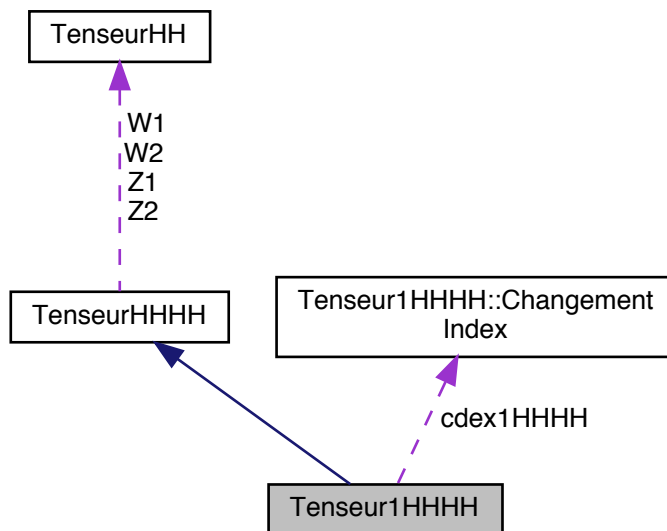
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

**6.810 Référence de la classe Tenseur1HHHH**

Grphe d'héritage de Tenseur1HHHH:



Graphe de collaboration de Tenseur1HHHH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur1HHHH** (const double val)
- **Tenseur1HHHH** (bool normal, const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- **Tenseur1HHHH** (const [TenseurHHHH](#) &)
- **Tenseur1HHHH** (const [Tenseur1HHHH](#) &)
- void **Inita** (double val)
- [TenseurHHHH](#) & **operator+** (const [TenseurHHHH](#) &) const
- void **operator+=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator-** () const
- [TenseurHHHH](#) & **operator-** (const [TenseurHHHH](#) &) const
- void **operator-=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator=** (const [Tenseur1HHHH](#) &B)
- [TenseurHHHH](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurHHHH](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &) const
- [TenseurHHHH](#) & **Transpose1et2avec3et4** () const
- void **Affectation\_trans\_dimension** (const [TenseurHHHH](#) &B, bool plusZero)
- void **TransfertDunTenseurGeneral** (const [TenseurHHHH](#) &aHHHH)
- [TenseurBBHH](#) & **Baisse2premiersIndices** ()
- [TenseurHHBB](#) & **Baisse2derniersIndices** ()
- [TenseurHHHH](#) & **Baselocale** ([TenseurHHHH](#) &A, const [BaseH](#) &gi) const
- [TenseurHHHH](#) & **ChangeBase** ([TenseurHHHH](#) &A, const [BaseB](#) &gi) const
- int **operator==** (const [TenseurHHHH](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const



- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort) const
- [TenseurHH](#) & [Prod\\_gauche](#) (const [TenseurBB](#) &F) const

### Fonctions membres publiques statiques

- static [TenseurHHHH](#) & [Prod\\_tensoriel](#) (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & [Prod\\_tensoriel\\_barre](#) (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)

### Attributs publics statiques

- static const [ChangementIndex](#) [cdex1HHHH](#)  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 1*

### Attributs protégés

- listdouble1Iter [ipointe](#)

### Amis

- istream & [operator>>](#) (istream &, [Tenseur1HHHH](#) &)
- ostream & [operator<<](#) (ostream &, const [Tenseur1HHHH](#) &)

### Membres hérités additionnels

#### 6.810.1 Documentation des fonctions membres

##### 6.810.1.1 Affectation\_trans\_dimension()

```
void Tenseur1HHHH::Affectation_trans_dimension (
    const TenseurHHHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHHH](#).

##### 6.810.1.2 Change()

```
void Tenseur1HHHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

##### 6.810.1.3 ChangePlus()

```
void Tenseur1HHHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.810.1.4 Ecriture()

```
ostream & Tenseur1HHHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.810.1.5 Inita()

```
void Tenseur1HHHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.810.1.6 Lecture()

```
istream & Tenseur1HHHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.810.1.7 MaxiComposante()

```
double Tenseur1HHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.810.1.8 operator&&()

```
TenseurHH & Tenseur1HHHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.810.1.9 operator()()

```
double Tenseur1HHHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.810.1.10 operator\*()

```
TenseurHHHH & Tenseur1HHHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.810.1.11 operator\*=( )

```
void Tenseur1HHHH::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.12 operator+()**

```
TenseurHHHH & Tenseur1HHHH::operator+ (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.13 operator+=()**

```
void Tenseur1HHHH::operator+= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.14 operator-() [1/2]**

```
TenseurHHHH & Tenseur1HHHH::operator- ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.15 operator-() [2/2]**

```
TenseurHHHH & Tenseur1HHHH::operator- (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.16 operator--()**

```
void Tenseur1HHHH::operator--= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.17 operator/()**

```
TenseurHHHH & Tenseur1HHHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.18 operator/=( )**

```
void Tenseur1HHHH::operator/=(
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.19 operator=( )**

```
TenseurHHHH & Tenseur1HHHH::operator= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.810.1.20 operator==( )**

```
int Tenseur1HHHH::operator==(
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.810.1.21 Prod\_gauche()

```
TenseurHH & Tenseur1HHHH::Prod_gauche (
    const TenseurBB & F ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.810.1.22 Transpose1et2avec3et4()

```
TenseurHHHH & Tenseur1HHHH::Transpose1et2avec3et4 ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

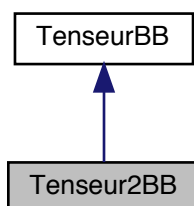
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.811 Référence de la classe Tenseur2BB

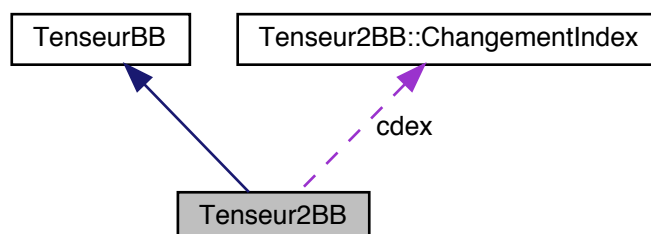
Definition des tenseur derivees de dimension 2. cas des composantes deux fois covariantes symetriques.

```
#include <Tenseur2.h>
```

Graphe d'héritage de Tenseur2BB:



Graphe de collaboration de Tenseur2BB:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur2BB** (const double val)
- **Tenseur2BB** (const double val1, const double val2, const double val3)
- **Tenseur2BB** (const [TenseurBB](#) &)
- **Tenseur2BB** (const [Tenseur2BB](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurBB](#) & **operator+** (const [TenseurBB](#) &) const  
*operations +*
- void **operator+=** (const [TenseurBB](#) &)  
*operations +=*
- [TenseurBB](#) & **operator-** () const  
*operations -*
- [TenseurBB](#) & **operator-** (const [TenseurBB](#) &) const  
*operations - tens*
- void **operator-=** (const [TenseurBB](#) &)  
*operations -=*
- [TenseurBB](#) & **operator=** (const [TenseurBB](#) &)  
*operations =*
- [TenseurBB](#) & **operator=** (const [Tenseur2BB](#) &B)
- [TenseurBB](#) & **operator\*** (const double &) const  
*operations \* double*
- void **operator\*=** (const double &)  
*operations \*= double*
- [TenseurBB](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_3D\_a\_2D** (const [Tenseur3BB](#) &B)
- void **Affectation\_trans\_dimension** (const [TenseurBB](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- [CoordonneeB](#) **operator\*** (const [CoordonneeH](#) &) const  
*produit contracte avec un vecteur*
- [TenseurBB](#) & **operator\*** (const [TenseurHB](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté*
- [TenseurBH](#) & **operator\*** (const [TenseurHH](#) &) const  
*idem en BH*
- double **operator&&** (const [TenseurHH](#) &) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int **operator==** (const [TenseurBB](#) &) const  
*test*
- int **operator!=** (const [TenseurBB](#) &) const  
*test*
- double **Det** () const  
*determinant de la matrice des coordonnees*
- [TenseurBB](#) & **Transpose** () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual [TenseurHH](#) & **Monte2Indices** () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual [TenseurHB](#) & **MontePremierIndice** () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual [TenseurBH](#) & **MonteDernierIndice** () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- double **MaxiComposante** () const  
*calcul du maximum en valeur absolu des composantes du tenseur*

- `TenseurHH & Inverse ()` const  
*calcul du tenseur inverse par rapport au produit contracte*
- `double & Coor (const int i, const int j)`  
*Retourne la composante  $i,j$  du tenseur acces en lecture et en ecriture.*
- `double operator() (const int i, const int j)` const  
*Retourne la composante  $i,j$  du tenseur acces en lecture seulement.*
- `istream & Lecture (istream &entree)`  
*lecture et ecriture de données*
- `ostream & Ecriture (ostream &sort)` const  
*lecture et ecriture de données*

## Fonctions membres publiques statiques

- `static TenseurBB & Prod_tensoriel (const CoordonneeB &aB, const CoordonneeB &bB)`
- `static int OdVect (const int i, const int j)`
- `static int idx_i (const int k)`
- `static int idx_j (const int k)`

## Attributs protégés

- `listdouble3lter ipointe`

## Attributs protégés statiques

- `static const ChangementIndex cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- `class Tenseur3BB`
- `istream & operator>> (istream &, Tenseur2BB &)`
- `ostream & operator<< (ostream &, const Tenseur2BB &)`

## Membres hérités additionnels

### 6.811.1 Description détaillée

Definition des tenseur derivees de dimension 2. cas des composantes deux fois covariantes symetriques.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

### 6.811.2 Documentation des fonctions membres

#### 6.811.2.1 Affectation\_trans\_dimension()

```
void Tenseur2BB::Affectation_trans_dimension (
    const TenseurBB & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,

plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation des données possibles  
Implémente [TenseurBB](#).

### 6.811.2.2 Coor()

```
double & Tenseur2BB::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.  
Implémente [TenseurBB](#).

### 6.811.2.3 Det()

```
double Tenseur2BB::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees  
Implémente [TenseurBB](#).

### 6.811.2.4 Ecriture()

```
ostream & Tenseur2BB::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données  
Implémente [TenseurBB](#).

### 6.811.2.5 Inita()

```
void Tenseur2BB::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val  
Implémente [TenseurBB](#).

### 6.811.2.6 Inverse()

```
TenseurHH & Tenseur2BB::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracte  
Implémente [TenseurBB](#).

### 6.811.2.7 Lecture()

```
istream & Tenseur2BB::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données  
Implémente [TenseurBB](#).

### 6.811.2.8 MaxiComposante()

```
double Tenseur2BB::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur  
Implémente [TenseurBB](#).

**6.811.2.9 Monte2Indices()**

```
virtual TenseurHH & Tenseur2BB::Monte2Indices ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

**6.811.2.10 MonteDernierIndice()**

```
virtual TenseurBH & Tenseur2BB::MonteDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

**6.811.2.11 MontePremierIndice()**

```
virtual TenseurHB & Tenseur2BB::MontePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

**6.811.2.12 operator"!="()**

```
int Tenseur2BB::operator!=(
    const TenseurBB & ) const [virtual]
```

test

Implémente [TenseurBB](#).

**6.811.2.13 operator&&()**

```
double Tenseur2BB::operator&& (
    const TenseurHH & ) const [virtual]
```

produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémente [TenseurBB](#).

**6.811.2.14 operator()()**

```
double Tenseur2BB::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.

Implémente [TenseurBB](#).

**6.811.2.15 operator\*() [1/4]**

```
CoordonneeB Tenseur2BB::operator* (
    const CoordonneeH & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurBB](#).

**6.811.2.16 operator\*() [2/4]**

```
TenseurBB & Tenseur2BB::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurBB](#).



**6.811.2.17 operator\*() [3/4]**

```
TenseurBB & Tenseur2BB::operator* (
    const TenseurHB & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté  
Implémente [TenseurBB](#).

**6.811.2.18 operator\*() [4/4]**

```
TenseurBH & Tenseur2BB::operator* (
    const TenseurHH & ) const [virtual]
```

idem en BH  
Implémente [TenseurBB](#).

**6.811.2.19 operator\*=( )**

```
void Tenseur2BB::operator*= (
    const double & ) [virtual]
```

operations \*= double  
Implémente [TenseurBB](#).

**6.811.2.20 operator+( )**

```
TenseurBB & Tenseur2BB::operator+ (
    const TenseurBB & ) const [virtual]
```

operations +  
Implémente [TenseurBB](#).

**6.811.2.21 operator+=( )**

```
void Tenseur2BB::operator+= (
    const TenseurBB & ) [virtual]
```

operations +=  
Implémente [TenseurBB](#).

**6.811.2.22 operator-( ) [1/2]**

```
TenseurBB & Tenseur2BB::operator- ( ) const [virtual]
```

operations -  
Implémente [TenseurBB](#).

**6.811.2.23 operator-( ) [2/2]**

```
TenseurBB & Tenseur2BB::operator- (
    const TenseurBB & ) const [virtual]
```

operations - tens  
Implémente [TenseurBB](#).

#### 6.811.2.24 operator-=( )

```
void Tenseur2BB::operator-=(  
    const TenseurBB & ) [virtual]
```

operations -=

Implémente [TenseurBB](#).

#### 6.811.2.25 operator/( )

```
TenseurBB & Tenseur2BB::operator/(  
    const double & ) const [virtual]
```

operations /

Implémente [TenseurBB](#).

#### 6.811.2.26 operator/=( )

```
void Tenseur2BB::operator/=(  
    const double & ) [virtual]
```

operations /=

Implémente [TenseurBB](#).

#### 6.811.2.27 operator=( )

```
TenseurBB & Tenseur2BB::operator=(  
    const TenseurBB & ) [virtual]
```

operations =

Implémente [TenseurBB](#).

#### 6.811.2.28 operator==( )

```
int Tenseur2BB::operator==(  
    const TenseurBB & ) const [virtual]
```

test

Implémente [TenseurBB](#).

#### 6.811.2.29 Transpose( )

```
TenseurBB & Tenseur2BB::Transpose ( ) const [virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

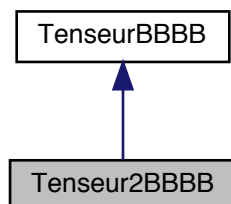
Implémente [TenseurBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

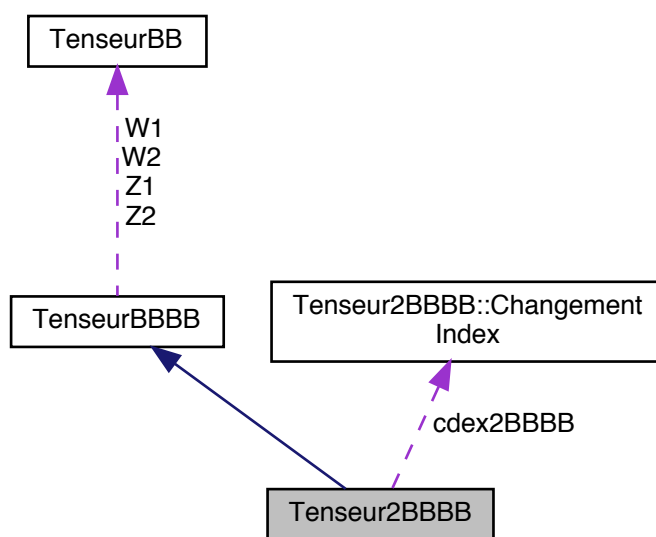
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.812 Référence de la classe Tenseur2BBBB

Graphe d'héritage de Tenseur2BBBB:



Graphe de collaboration de Tenseur2BBBB:



### Classes

- class [ChangementIndex](#)

### Fonctions membres publiques

- **Tenseur2BBBB** (const double val)
- **Tenseur2BBBB** (bool normal, const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- **Tenseur2BBBB** (const [TenseurBBBB](#) &)
- **Tenseur2BBBB** (const [Tenseur2BBBB](#) &)
- void **Inita** (double val)
- [TenseurBBBB](#) & **operator+** (const [TenseurBBBB](#) &) const
- void **operator+=** (const [TenseurBBBB](#) &)

- [TenseurBBBB](#) & [operator-](#) () const
- [TenseurBBBB](#) & [operator-](#) (const [TenseurBBBB](#) &) const
- void [operator-=](#) (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & [operator=](#) (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & [operator=](#) (const [Tenseur2HHHH](#) &B)
- [TenseurBBBB](#) & [operator\\*](#) (const double &) const
- void [operator\\*=](#) (const double &)
- [TenseurBBBB](#) & [operator/](#) (const double &) const
- void [operator/=](#) (const double &)
- [TenseurBB](#) & [operator&&](#) (const [TenseurHH](#) &) const
- [TenseurBBBB](#) & [Transpose1et2avec3et4](#) () const
- virtual void [Affectation\\_trans\\_dimension](#) (const [TenseurBBBB](#) &B, bool plusZero)
- void [TransfertDunTenseurGeneral](#) (const [TenseurBBBB](#) &aBBBB)
- [TenseurHHBB](#) & [Monte2premiersIndices](#) ()
- [TenseurBBHH](#) & [Monte2derniersIndices](#) ()
- [TenseurHHHH](#) & [Monte4Indices](#) ()
- [TenseurBBBB](#) & [Baselocale](#) ([TenseurBBBB](#) &A, const [BaseB](#) &gi) const
- [TenseurBBBB](#) & [ChangeBase](#) ([TenseurBBBB](#) &A, const [BaseH](#) &gi) const
- int [operator==](#) (const [TenseurBBBB](#) &) const
- void [Change](#) (int i, int j, int k, int l, const double &val)
- void [ChangePlus](#) (int i, int j, int k, int l, const double &val)
- double [operator\(\)](#) (int i, int j, int k, int l) const
- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort) const
- void [Affiche\\_bidim](#) (ostream &sort) const
- [TenseurBB](#) & [Prod\\_gauche](#) (const [TenseurHH](#) &F) const

## Fonctions membres publiques statiques

- static [TenseurBBBB](#) & [Prod\\_tensoriel](#) (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- static [TenseurBBBB](#) & [Prod\\_tensoriel\\_barre](#) (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)

## Attributs publics statiques

- static const [ChangementIndex](#) [cdex2BBBB](#)  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 2*

## Attributs protégés

- listdouble9lter [ipointe](#)

## Amis

- istream & [operator>>](#) (istream &, [Tenseur2BBBB](#) &)
- ostream & [operator<<](#) (ostream &, const [Tenseur2BBBB](#) &)

## Membres hérités additionnels

### 6.812.1 Documentation des fonctions membres

#### 6.812.1.1 [Affectation\\_trans\\_dimension\(\)](#)

```
virtual void Tenseur2BBBB::Affectation_trans_dimension (
    const TenseurBBBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.812.1.2 Change()

```
void Tenseur2BBBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.812.1.3 ChangePlus()

```
void Tenseur2BBBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.812.1.4 Ecriture()

```
ostream & Tenseur2BBBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.812.1.5 Inita()

```
void Tenseur2BBBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.812.1.6 Lecture()

```
istream & Tenseur2BBBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.812.1.7 MaxiComposante()

```
double Tenseur2BBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.812.1.8 operator&&()

```
TenseurBB & Tenseur2BBBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.812.1.9 operator>()()

```
double Tenseur2BBBB::operator() (
    int i,
```

```
int j,  
int k,  
int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.812.1.10 operator\*()

```
TenseurBBBB & Tenseur2BBBB::operator* (  
const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.812.1.11 operator\*=( )

```
void Tenseur2BBBB::operator*= (  
const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.812.1.12 operator+( )

```
TenseurBBBB & Tenseur2BBBB::operator+ (  
const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.812.1.13 operator+=( )

```
void Tenseur2BBBB::operator+= (  
const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.812.1.14 operator-( ) [1/2]

```
TenseurBBBB & Tenseur2BBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.812.1.15 operator-( ) [2/2]

```
TenseurBBBB & Tenseur2BBBB::operator- (  
const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.812.1.16 operator-=( )

```
void Tenseur2BBBB::operator-= (  
const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.812.1.17 operator/( )

```
TenseurBBBB & Tenseur2BBBB::operator/ (  
const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.812.1.18 operator/=()**

```
void Tenseur2BBBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.812.1.19 operator=()**

```
TenseurBBBB & Tenseur2BBBB::operator= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.812.1.20 operator==()**

```
int Tenseur2BBBB::operator== (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.812.1.21 Prod\_gauche()**

```
TenseurBB & Tenseur2BBBB::Prod_gauche (
    const TenseurHH & F ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.812.1.22 Transpose1et2avec3et4()**

```
TenseurBBBB & Tenseur2BBBB::Transpose1et2avec3et4 ( ) const [virtual]
```

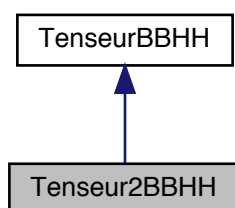
Implémente [TenseurBBBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

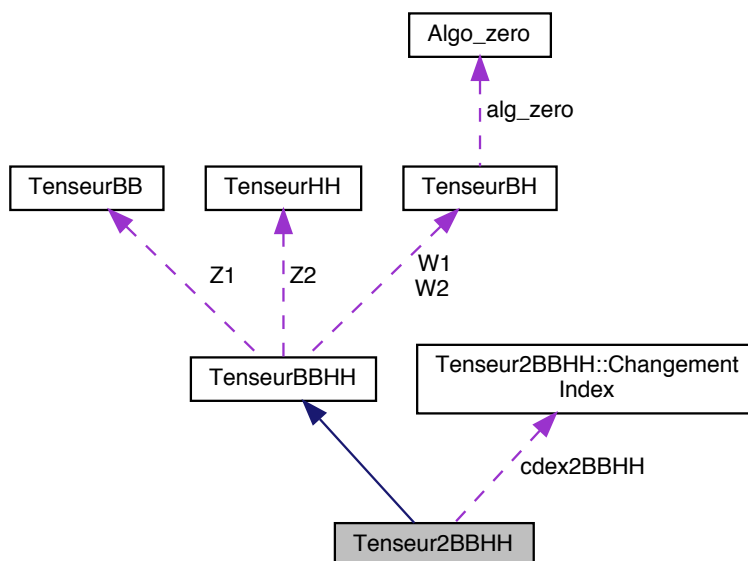
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

**6.813 Référence de la classe Tenseur2BBHH**

Graphe d'héritage de Tenseur2BBHH:



Graphe de collaboration de Tenseur2BBHH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur2BBHH** (const double val)
- **Tenseur2BBHH** (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)
- **Tenseur2BBHH** (const [TenseurBBHH](#) &)
- **Tenseur2BBHH** (const [Tenseur2BBHH](#) &)
- void **Inita** (double val)
- [TenseurBBHH](#) & **operator+** (const [TenseurBBHH](#) &) const
- void **operator+=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator-** () const
- [TenseurBBHH](#) & **operator-** (const [TenseurBBHH](#) &) const
- void **operator-=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator\*** (const double &) const
- void **operator\*= **(const double &)****
- [TenseurBBHH](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBB](#) & **operator&&** (const [TenseurBB](#) &) const
- [TenseurHHBB](#) & **Transpose1et2avec3et4** () const
- virtual void **Affectation\_trans\_dimension** (const [TenseurBBHH](#) &B, bool plusZero)
- int **operator==** (const [TenseurBBHH](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort) const
- void **Affiche\_bidim** (ostream &sort) const
- [TenseurHH](#) & **Prod\_gauche** (const [TenseurHH](#) &F) const



## Fonctions membres publiques statiques

- static [TenseurBBHH](#) & **Prod\_tensoriel** (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)
- static [TenseurBBHH](#) & **Var\_tenseur\_dans\_nouvelle\_base** (const [Mat\\_pleine](#) &beta, [Tenseur2BBHH](#) &var↔  
\_tensBBHH, const [Tableau2](#)< [Tenseur2HH](#) > &var\_beta, const [Tenseur2BB](#) &tensBB)  
— *pour mémoire* —

## Attributs publics statiques

- static const [ChangementIndex](#) **cdex2BBHH**  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 2*

## Attributs protégés

- listdouble9Iter **ipointe**

## Amis

- istream & **operator**>> (istream &, [Tenseur2BBHH](#) &)
- ostream & **operator**<< (ostream &, const [Tenseur2BBHH](#) &)

## Membres hérités additionnels

### 6.813.1 Documentation des fonctions membres

#### 6.813.1.1 Affectation\_trans\_dimension()

```
virtual void Tenseur2BBHH::Affectation_trans_dimension (
    const TenseurBBHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.2 Change()

```
void Tenseur2BBHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.3 ChangePlus()

```
void Tenseur2BBHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.4 Ecriture()

```
ostream & Tenseur2BBHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.5 Inita()

```
void Tenseur2BBHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.6 Lecture()

```
istream & Tenseur2BBHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.7 MaxiComposante()

```
double Tenseur2BBHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.8 operator&&()

```
TenseurBB & Tenseur2BBHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.9 operator()()

```
double Tenseur2BBHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.10 operator\*()

```
TenseurBBHH & Tenseur2BBHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.813.1.11 operator\*=( )

```
void Tenseur2BBHH::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.12 operator+()**

```
TenseurBBHH & Tenseur2BBHH::operator+ (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.13 operator+=()**

```
void Tenseur2BBHH::operator+= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.14 operator-() [1/2]**

```
TenseurBBHH & Tenseur2BBHH::operator- ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.15 operator-() [2/2]**

```
TenseurBBHH & Tenseur2BBHH::operator- (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.16 operator--()**

```
void Tenseur2BBHH::operator--= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.17 operator/()**

```
TenseurBBHH & Tenseur2BBHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.18 operator/=( )**

```
void Tenseur2BBHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.19 operator=( )**

```
TenseurBBHH & Tenseur2BBHH::operator= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.813.1.20 operator==( )**

```
int Tenseur2BBHH::operator==(
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.813.1.21 Prod\_gauche()

```
TenseurHH & Tenseur2BBHH::Prod_gauche (
    const TenseurHH & F ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.813.1.22 Transpose1et2avec3et4()

```
TenseurHHBB & Tenseur2BBHH::Transpose1et2avec3et4 ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

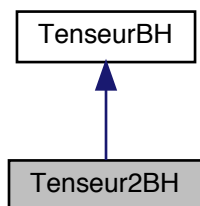
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\\_mai99/Tenseur/TenseurQ-2.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\\_mai99/Tenseur/EnteteTenseur.h](#)

## 6.814 Référence de la classe Tenseur2BH

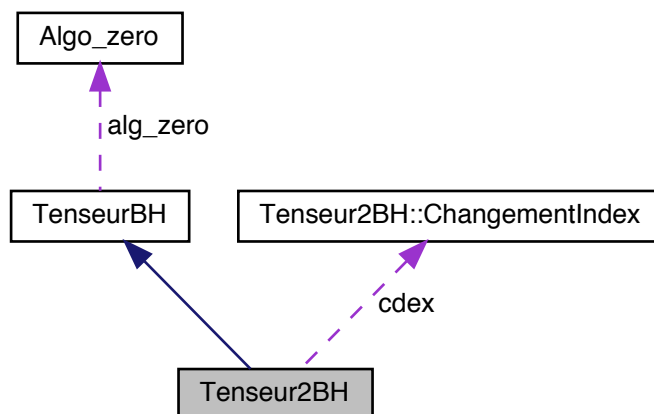
Definition des tenseur derivees de dimension 2. cas des composantes mixtes BH.

```
#include <Tenseur2.h>
```

Graphe d'héritage de Tenseur2BH:



Grphe de collaboration de Tenseur2BH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur2BH** (const double val)
- **Tenseur2BH** (const double val1, const double val2, const double val3, const double val4)
- **Tenseur2BH** (const [TenseurBH](#) &)
- **Tenseur2BH** (const [Tenseur2BH](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurBH](#) & **operator+** (const [TenseurBH](#) &) const  
*operations +*
- void **operator+=** (const [TenseurBH](#) &)  
*operations +=*
- [TenseurBH](#) & **operator-** () const  
*operations opposé*
- [TenseurBH](#) & **operator-** (const [TenseurBH](#) &) const  
*operations -*
- void **operator-=** (const [TenseurBH](#) &)  
*operations -=*
- [TenseurBH](#) & **operator=** (const [TenseurBH](#) &)  
*operations =*
- [TenseurBH](#) & **operator=** (const [Tenseur2BH](#) &B)
- [TenseurBH](#) & **operator\*** (const double &) const  
*operations \**
- void **operator\*= **(const double &)****  
*operations \*=*
- [TenseurBH](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_3D\_a\_2D** (const [Tenseur3BH](#) &B)
- void **Affectation\_trans\_dimension** (const [TenseurBH](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*

- `CoordonneeB operator*` (const `CoordonneeB` &) const  
*produit contracte avec un vecteur*
- `TenseurBB & operator*` (const `TenseurBB` &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
- `TenseurBH & operator*` (const `TenseurBH` &) const
- `double operator&&` (const `TenseurBH` &) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- `int operator==` (const `TenseurBH` &) const  
*test*
- `int operator!=` (const `TenseurBH` &) const
- `TenseurBH & Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
- `double Trace` () const  
*trace du tenseur ou premier invariant*
- `double II` () const  
*second invariant = trace (A\*A)*
- `double III` () const  
*troisieme invariant = trace ((A\*A)\*A)*
- `double Det` () const  
*determinant de la matrice des coordonnees*
- `virtual Coordonnee ValPropre` (int &cas) const  
*calcul des valeurs propres*
- `virtual Coordonnee ValPropre` (int &cas, `Mat_pleine` &mat) const  
*calcul des valeurs propres*
- `virtual void VecteursPropres` (const `Coordonnee` &Val\_P, int &cas, `Tableau`< `Coordonnee` > &V\_P) const  
  
*calcul des vecteurs propres, les valeurs propres étant déjà connues*
- `TenseurHB & Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- `void PermuteHautBas` ()  
*permuté Bas Haut, mais reste dans le même tenseur*
- `double MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- `double & Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- `double operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
- `istream & Lecture` (istream &entree)  
*lecture et écriture de données*
- `ostream & Ecriture` (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- `static TenseurBH & Prod_tensoriel` (const `CoordonneeB` &aB, const `CoordonneeH` &bH)
- `static int OdVect` (const int i, const int j)
- `static int idx_i` (const int k)
- `static int idx_j` (const int k)

## Attributs protégés

- `listdouble4lter ipointe`

## Attributs protégés statiques

- `static const ChangementIndex cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class **Tenseur3BH**
- `istream & operator>>` (`istream &`, [Tenseur2BH &](#))
- `ostream & operator<<` (`ostream &`, `const` [Tenseur2BH &](#))

## Membres hérités additionnels

### 6.814.1 Description détaillée

Definition des tenseur derivees de dimension 2. cas des composantes mixtes BH.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.814.2 Documentation des fonctions membres

#### 6.814.2.1 Affectation\_trans\_dimension()

```
void Tenseur2BH::Affectation_trans_dimension (
    const TenseurBH & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurBH](#).

#### 6.814.2.2 Coor()

```
double & Tenseur2BH::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.  
Implémente [TenseurBH](#).

#### 6.814.2.3 Det()

```
double Tenseur2BH::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees

Implémente [TenseurBH](#).

#### 6.814.2.4 Ecriture()

```
ostream & Tenseur2BH::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurBH](#).

#### 6.814.2.5 II()

```
double Tenseur2BH::II ( ) const [virtual]
```

second invariant = trace (A\*A)

Implémente [TenseurBH](#).

#### 6.814.2.6 III()

```
double Tenseur2BH::III ( ) const [virtual]
```

troisieme invariant = trace ((A\*A)\*A)

Implémente [TenseurBH](#).

#### 6.814.2.7 Inita()

```
void Tenseur2BH::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurBH](#).

#### 6.814.2.8 Inverse()

```
TenseurBH & Tenseur2BH::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémente [TenseurBH](#).

#### 6.814.2.9 Lecture()

```
istream & Tenseur2BH::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurBH](#).

#### 6.814.2.10 MaxiComposante()

```
double Tenseur2BH::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurBH](#).

#### 6.814.2.11 operator"!=(())

```
int Tenseur2BH::operator!=(
    const TenseurBH & ) const [virtual]
```

Implémente [TenseurBH](#).

#### 6.814.2.12 operator&&()

```
double Tenseur2BH::operator&& (
    const TenseurBH & ) const [virtual]
```

produit contracté contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe



Implémente [TenseurBH](#).

#### 6.814.2.13 operator>()

```
double Tenseur2BH::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.

Implémente [TenseurBH](#).

#### 6.814.2.14 operator\*() [1/4]

```
CoordonneeB Tenseur2BH::operator* (
    const CoordonneeB & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurBH](#).

#### 6.814.2.15 operator\*() [2/4]

```
TenseurBH & Tenseur2BH::operator* (
    const double & ) const [virtual]
```

operations \*

Implémente [TenseurBH](#).

#### 6.814.2.16 operator\*() [3/4]

```
TenseurBB & Tenseur2BH::operator* (
    const TenseurBB & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté

Implémente [TenseurBH](#).

#### 6.814.2.17 operator\*() [4/4]

```
TenseurBH & Tenseur2BH::operator* (
    const TenseurBH & ) const [virtual]
```

Implémente [TenseurBH](#).

#### 6.814.2.18 operator\*=( )

```
void Tenseur2BH::operator*= (
    const double & ) [virtual]
```

operations \*=

Implémente [TenseurBH](#).

#### 6.814.2.19 operator+( )

```
TenseurBH & Tenseur2BH::operator+ (
    const TenseurBH & ) const [virtual]
```

operations +

Implémente [TenseurBH](#).

**6.814.2.20 operator+=()**

```
void Tenseur2BH::operator+= (
    const TenseurBH & ) [virtual]
```

operations +=

Implémente [TenseurBH](#).

**6.814.2.21 operator-() [1/2]**

```
TenseurBH & Tenseur2BH::operator- ( ) const [virtual]
```

operations opposé

Implémente [TenseurBH](#).

**6.814.2.22 operator-() [2/2]**

```
TenseurBH & Tenseur2BH::operator- (
    const TenseurBH & ) const [virtual]
```

operations -

Implémente [TenseurBH](#).

**6.814.2.23 operator-=()**

```
void Tenseur2BH::operator-= (
    const TenseurBH & ) [virtual]
```

operations -=

Implémente [TenseurBH](#).

**6.814.2.24 operator/()**

```
TenseurBH & Tenseur2BH::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurBH](#).

**6.814.2.25 operator/=()**

```
void Tenseur2BH::operator/= (
    const double & ) [virtual]
```

operations /=

Implémente [TenseurBH](#).

**6.814.2.26 operator=()**

```
TenseurBH & Tenseur2BH::operator= (
    const TenseurBH & ) [virtual]
```

operations =

Implémente [TenseurBH](#).

**6.814.2.27 operator==(())**

```
int Tenseur2BH::operator==(
    const TenseurBH & ) const [virtual]
```

test

Implémente [TenseurBH](#).

#### 6.814.2.28 PermuteHautBas()

```
void Tenseur2BH::PermuteHautBas ( ) [virtual]
permute Bas Haut, mais reste dans le même tenseur
Implémente TenseurBH.
```

#### 6.814.2.29 Trace()

```
double Tenseur2BH::Trace ( ) const [virtual]
trace du tenseur ou premier invariant
Implémente TenseurBH.
```

#### 6.814.2.30 Transpose()

```
TenseurHB & Tenseur2BH::Transpose ( ) const [virtual]
ATTENTION creation d'un tenseur transpose qui est supprime par Libere.
Implémente TenseurBH.
```

#### 6.814.2.31 ValPropre() [1/2]

```
virtual Coordonnee Tenseur2BH::ValPropre (
    int & cas ) const [virtual]
calcul des valeurs propres
```

valeurs propre dans le vecteur de retour, classée par ordres "décroissants"  
cas indique le cas de valeur propre:  
quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire  
dans ce cas les valeurs propres de retour sont nulles par défaut  
dim = 1, cas=1 pour une valeur propre;  
dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques  
dim = 3 , cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)  
, cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),  
, cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du retour)  
, cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du retour)  
Implémente [TenseurBH](#).

#### 6.814.2.32 ValPropre() [2/2]

```
virtual Coordonnee Tenseur2BH::ValPropre (
    int & cas,
    Mat\_pleine & mat ) const [virtual]
calcul des valeurs propres
```

idem met en retour la matrice mat contient par colonne les vecteurs propre  
elle doit avoir la dimension du tenseur  
les vecteurs propre sont exprime dans le repere dual (contrairement au HB) pour les tenseurs dim 3  
pour dim=2:le premier vecteur propre est exprime dans le repere dual  
le second vecteur propre est exprimé dans le repère naturel  
pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
Implémente [TenseurBH](#).

### 6.814.2.33 VecteursPropres()

```
virtual void Tenseur2BH::VecteursPropres (
    const Coordonnee & Val_P,
    int & cas,
    Tableau<Coordonnee > & V_P ) const [virtual]
```

calcul des vecteurs propres, les valeurs propres  
étant déjà connues

ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres  
étant déjà connues

en retour VP les vecteurs propre : doivent avoir la dimension du tenseur  
les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3  
pour dim=2:le premier vecteur propre est exprime dans le repere naturel  
le second vecteur propre est exprimé dans le repère dual  
pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
en sortie cas = -1 s'il y a eu un problème, dans ce cas, V\_P est quelconque  
sinon si tout est ok, cas est identique en sortie avec l'entrée  
Implémente [TenseurBH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

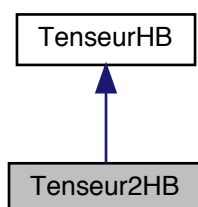
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\\_mai99/Tenseur/Tenseur2.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\\_mai99/Tenseur/EnteteTenseur.h](#)

## 6.815 Référence de la classe Tenseur2HB

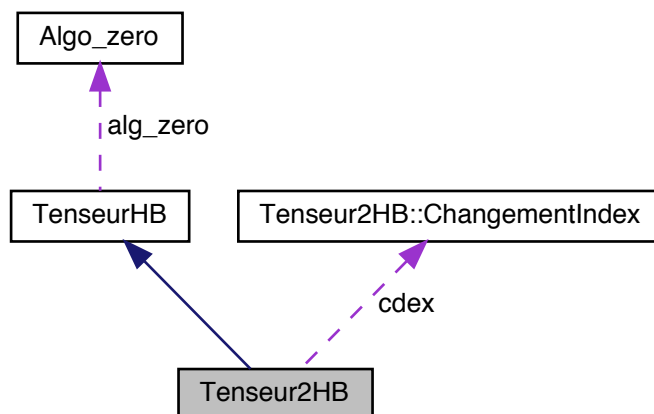
Definition des tenseur derivees de dimension 2. cas des composantes mixtes HB.

```
#include <Tenseur2.h>
```

Graphe d'héritage de Tenseur2HB:



Grphe de collaboration de Tenseur2HB:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur2HB** (const double val)
- **Tenseur2HB** (const double val1, const double val2, const double val3, const double val4)
- **Tenseur2HB** (const [TenseurHB](#) &)
- **Tenseur2HB** (const [Tenseur2HB](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurHB](#) & **operator+** (const [TenseurHB](#) &) const  
*operations +*
- void **operator+=** (const [TenseurHB](#) &)  
*operations +=*
- [TenseurHB](#) & **operator-** () const  
*operations opposé*
- [TenseurHB](#) & **operator-** (const [TenseurHB](#) &) const  
*operations -*
- void **operator-=** (const [TenseurHB](#) &)  
*operations -=*
- [TenseurHB](#) & **operator=** (const [TenseurHB](#) &)  
*operations =*
- [TenseurHB](#) & **operator=** (const [Tenseur2HB](#) &B)
- [TenseurHB](#) & **operator\*** (const double &) const  
*operations \**
- void **operator\*= **(const double &)****  
*operations \*=*
- [TenseurHB](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_3D\_a\_2D** (const [Tenseur3HB](#) &B)
- void **Affectation\_trans\_dimension** (const [TenseurHB](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*

- [CoordonneeH operator\\*](#) (const [CoordonneeH](#) &) const  
*produit contracte avec un vecteur*
- [TenseurHH & operator\\*](#) (const [TenseurHH](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
- [TenseurHB & operator\\*](#) (const [TenseurHB](#) &) const
- [double operator&&](#) (const [TenseurHB](#) &) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- [int operator==](#) (const [TenseurHB](#) &) const  
*test*
- [int operator!=](#) (const [TenseurHB](#) &) const
- [TenseurHB & Inverse](#) () const  
*calcul du tenseur inverse par rapport au produit contracte*
- [double Trace](#) () const  
*trace du tenseur ou premier invariant*
- [double II](#) () const  
*second invariant = trace (A\*A)*
- [double III](#) () const  
*troisieme invariant = trace ((A\*A)\*A)*
- [double Det](#) () const  
*determinant de la matrice des coordonnees*
- [virtual Coordonnee ValPropre](#) (int &cas) const  
*calcul des valeurs propres*
- [virtual Coordonnee ValPropre](#) (int &cas, [Mat\\_pleine](#) &mat) const  
*calcul des valeurs propres*
- [virtual void VecteursPropres](#) (const [Coordonnee](#) &Val\_P, int &cas, [Tableau](#)< [Coordonnee](#) > &V\_P) const  
  
*calcul des vecteurs propres, les valeurs propres étant déjà connues*
- [TenseurBH & Transpose](#) () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- [void PermuteHautBas](#) ()  
*permute Bas Haut, mais reste dans le même tenseur*
- [double MaxiComposante](#) () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- [double & Coor](#) (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- [double operator\(\)](#) (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
- [istream & Lecture](#) (istream &entree)  
*lecture et écriture de données*
- [ostream & Ecriture](#) (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- [static TenseurHB & Prod\\_tensoriel](#) (const [CoordonneeH](#) &aH, const [CoordonneeB](#) &bB)
- [static int OdVect](#) (const int i, const int j)
- [static int idx\\_i](#) (const int k)
- [static int idx\\_j](#) (const int k)

## Attributs protégés

- [listdouble4lter ipointe](#)

## Attributs protégés statiques

- [static const ChangementIndex cdex](#)  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class **Tenseur3HB**
- `istream & operator>>` (`istream &`, [Tenseur2HB &](#))
- `ostream & operator<<` (`ostream &`, `const` [Tenseur2HB &](#))

## Membres hérités additionnels

### 6.815.1 Description détaillée

Definition des tenseur derivees de dimension 2. cas des composantes mixtes HB.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.815.2 Documentation des fonctions membres

#### 6.815.2.1 Affectation\_trans\_dimension()

```
void Tenseur2HB::Affectation_trans_dimension (
    const TenseurHB & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurHB](#).

#### 6.815.2.2 Coor()

```
double & Tenseur2HB::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.  
Implémente [TenseurHB](#).

#### 6.815.2.3 Det()

```
double Tenseur2HB::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees

Implémente [TenseurHB](#).

#### 6.815.2.4 Ecriture()

```
ostream & Tenseur2HB::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurHB](#).

#### 6.815.2.5 II()

```
double Tenseur2HB::II ( ) const [virtual]
```

second invariant = trace (A\*A)

Implémente [TenseurHB](#).

#### 6.815.2.6 III()

```
double Tenseur2HB::III ( ) const [virtual]
```

troisieme invariant = trace ((A\*A)\*A)

Implémente [TenseurHB](#).

#### 6.815.2.7 Inita()

```
void Tenseur2HB::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurHB](#).

#### 6.815.2.8 Inverse()

```
TenseurHB & Tenseur2HB::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémente [TenseurHB](#).

#### 6.815.2.9 Lecture()

```
istream & Tenseur2HB::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurHB](#).

#### 6.815.2.10 MaxiComposante()

```
double Tenseur2HB::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurHB](#).

#### 6.815.2.11 operator"!=(())

```
int Tenseur2HB::operator!=(
    const TenseurHB & ) const [virtual]
```

Implémente [TenseurHB](#).

#### 6.815.2.12 operator&&()

```
double Tenseur2HB::operator&& (
    const TenseurHB & ) const [virtual]
```

produit contracté contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe



Implémente [TenseurHB](#).

### 6.815.2.13 operator>()

```
double Tenseur2HB::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.

Implémente [TenseurHB](#).

### 6.815.2.14 operator\*() [1/4]

```
CoordonneeH Tenseur2HB::operator* (
    const CoordonneeH & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurHB](#).

### 6.815.2.15 operator\*() [2/4]

```
TenseurHB & Tenseur2HB::operator* (
    const double & ) const [virtual]
```

operations \*

Implémente [TenseurHB](#).

### 6.815.2.16 operator\*() [3/4]

```
TenseurHB & Tenseur2HB::operator* (
    const TenseurHB & ) const [virtual]
```

Implémente [TenseurHB](#).

### 6.815.2.17 operator\*() [4/4]

```
TenseurHH & Tenseur2HB::operator* (
    const TenseurHH & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté

Implémente [TenseurHB](#).

### 6.815.2.18 operator\*=( )

```
void Tenseur2HB::operator*= (
    const double & ) [virtual]
```

operations \*=

Implémente [TenseurHB](#).

### 6.815.2.19 operator+( )

```
TenseurHB & Tenseur2HB::operator+ (
    const TenseurHB & ) const [virtual]
```

operations +

Implémente [TenseurHB](#).

**6.815.2.20 operator+=()**

```
void Tenseur2HB::operator+= (
    const TenseurHB & ) [virtual]
```

operations +=

Implémente [TenseurHB](#).

**6.815.2.21 operator-() [1/2]**

```
TenseurHB & Tenseur2HB::operator- ( ) const [virtual]
```

operations opposé

Implémente [TenseurHB](#).

**6.815.2.22 operator-() [2/2]**

```
TenseurHB & Tenseur2HB::operator- (
    const TenseurHB & ) const [virtual]
```

operations -

Implémente [TenseurHB](#).

**6.815.2.23 operator-=()**

```
void Tenseur2HB::operator-= (
    const TenseurHB & ) [virtual]
```

operations -=

Implémente [TenseurHB](#).

**6.815.2.24 operator/()**

```
TenseurHB & Tenseur2HB::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurHB](#).

**6.815.2.25 operator/=()**

```
void Tenseur2HB::operator/= (
    const double & ) [virtual]
```

operations /=

Implémente [TenseurHB](#).

**6.815.2.26 operator=()**

```
TenseurHB & Tenseur2HB::operator= (
    const TenseurHB & ) [virtual]
```

operations =

Implémente [TenseurHB](#).

**6.815.2.27 operator==(())**

```
int Tenseur2HB::operator==(
    const TenseurHB & ) const [virtual]
```

test

Implémente [TenseurHB](#).

### 6.815.2.28 PermuteHautBas()

```
void Tenseur2HB::PermuteHautBas ( ) [virtual]
permute Bas Haut, mais reste dans le même tenseur
Implémente TenseurHB.
```

### 6.815.2.29 Trace()

```
double Tenseur2HB::Trace ( ) const [virtual]
trace du tenseur ou premier invariant
Implémente TenseurHB.
```

### 6.815.2.30 Transpose()

```
TenseurBH & Tenseur2HB::Transpose ( ) const [virtual]
ATTENTION creation d'un tenseur transpose qui est supprime par Libere.
Implémente TenseurHB.
```

### 6.815.2.31 ValPropre() [1/2]

```
virtual Coordonnee Tenseur2HB::ValPropre (
    int & cas ) const [virtual]
calcul des valeurs propres
```

valeurs propre dans le vecteur de retour, classée par ordres "décroissants"  
cas indique le cas de valeur propre:  
quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire  
dans ce cas les valeurs propres de retour sont nulles par défaut  
dim = 1, cas=1 pour une valeur propre;  
dim = 2, cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques  
dim = 3, cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)  
, cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),  
, cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du retour)  
, cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du retour)  
Implémente [TenseurHB](#).

### 6.815.2.32 ValPropre() [2/2]

```
virtual Coordonnee Tenseur2HB::ValPropre (
    int & cas,
    Mat\_pleine & mat ) const [virtual]
calcul des valeurs propres
```

idem met en retour la matrice mat contient par colonne les vecteurs propre  
elle doit avoir la dimension du tenseur  
les vecteurs propre sont exprimé dans le repere naturel (pour les tenseurs dim 3  
pour dim=2:le premier vecteur propre est exprimé dans le repere naturel  
le second vecteur propre est exprimé dans le repère dual  
pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
Implémente [TenseurHB](#).

### 6.815.2.33 VecteursPropres()

```
virtual void Tenseur2HB::VecteursPropres (
    const Coordonnee & Val_P,
    int & cas,
    Tableau<Coordonnee > & V_P ) const [virtual]
```

calcul des vecteurs propres, les valeurs propres  
étant déjà connues

ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres  
étant déjà connues

en retour VP les vecteurs propre : doivent avoir la dimension du tenseur  
les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3  
pour dim=2:le premier vecteur propre est exprime dans le repere naturel  
le second vecteur propre est exprimé dans le repère dual  
pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
en sortie cas = -1 s'il y a eu un problème, dans ce cas, V\_P est quelconque  
sinon si tout est ok, cas est identique en sortie avec l'entrée  
Implémente [TenseurHB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

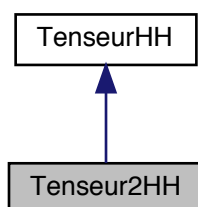
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

## 6.816 Référence de la classe Tenseur2HH

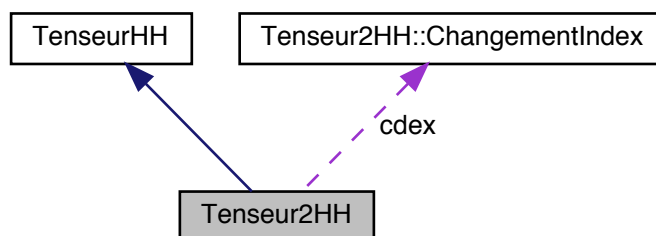
Definition des tenseur derivees de dimension 2. cas des composantes deux fois contravariantes symetriques.

```
#include <Tenseur2.h>
```

Graphe d'héritage de Tenseur2HH:



Grphe de collaboration de Tenseur2HH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur2HH** (const double val)
- **Tenseur2HH** (const double val1, const double val2, const double val3)
- **Tenseur2HH** (const [TenseurHH](#) &)
- **Tenseur2HH** (const [Tenseur2HH](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurHH](#) & **operator+** (const [TenseurHH](#) &) const  
*operations +*
- void **operator+=** (const [TenseurHH](#) &)  
*operations +=*
- [TenseurHH](#) & **operator-** () const  
*operations -*
- [TenseurHH](#) & **operator-** (const [TenseurHH](#) &) const  
*operations - tens*
- void **operator-=** (const [TenseurHH](#) &)  
*operations -=*
- [TenseurHH](#) & **operator=** (const [TenseurHH](#) &)  
*operations =*
- [TenseurHH](#) & **operator=** (const [Tenseur2HH](#) &B)
- [TenseurHH](#) & **operator\*** (const double &) const  
*operations \* double*
- void **operator\*=** (const double &)  
*operations \*= double*
- [TenseurHH](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_3D\_a\_2D** (const [Tenseur3HH](#) &B)
- void **Affectation\_trans\_dimension** (const [TenseurHH](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- [CoordonneeH](#) **operator\*** (const [CoordonneeB](#) &) const  
*produit contracte avec un vecteur*
- [TenseurHH](#) & **operator\*** (const [TenseurBH](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté avec BH*
- [TenseurHB](#) & **operator\*** (const [TenseurBB](#) &) const

- idem avec BB*
- double `operator&&` (const `TenseurBB` &) const  
*produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int `operator==` (const `TenseurHH` &) const  
*test*
- int `operator!=` (const `TenseurHH` &) const  
*test*
- double `Det` () const  
*determinant de la matrice des coordonnees*
- `TenseurHH` & `Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual `TenseurBB` & `Baisse2Indices` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurBH` & `BaissePremierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurHB` & `BaisseDernierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- double `MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- `TenseurBB` & `Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
- double & `Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double `operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seule.*
- istream & `Lecture` (istream &entree)  
*lecture et ecriture de données*
- ostream & `Ecriture` (ostream &sort) const  
*lecture et ecriture de données*

## Fonctions membres publiques statiques

- static `TenseurHH` & `Prod_tensoriel` (const `CoordonneeH` &aH, const `CoordonneeH` &bH)
- static int `OdVect` (const int i, const int j)
- static int `idx_i` (const int k)
- static int `idx_j` (const int k)

## Attributs protégés

- listdouble3lter `ipointe`

## Attributs protégés statiques

- static const `ChangementIndex` `cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class `Tenseur3HH`
- istream & `operator>>` (istream &, `Tenseur2HH` &)
- ostream & `operator<<` (ostream &, const `Tenseur2HH` &)

## Membres hérités additionnels

### 6.816.1 Description détaillée

Definition des tenseur derivees de dimension 2. cas des composantes deux fois contravariantes symetriques.

Auteur

Gérard Rio

## Version

1.0

## Date

23/01/97

## 6.816.2 Documentation des fonctions membres

### 6.816.2.1 Affectation\_trans\_dimension()

```
void Tenseur2HH::Affectation_trans_dimension (
    const TenseurHH & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurHH](#).

### 6.816.2.2 Baisse2Indices()

```
virtual TenseurBB & Tenseur2HH::Baisse2Indices ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

### 6.816.2.3 BaisseDernierIndice()

```
virtual TenseurHB & Tenseur2HH::BaisseDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

### 6.816.2.4 BaissePremierIndice()

```
virtual TenseurBH & Tenseur2HH::BaissePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

### 6.816.2.5 Coor()

```
double & Tenseur2HH::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémente [TenseurHH](#).

### 6.816.2.6 Det()

```
double Tenseur2HH::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees

Implémente [TenseurHH](#).

### 6.816.2.7 Ecriture()

```
ostream & Tenseur2HH::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurHH](#).

### 6.816.2.8 Inita()

```
void Tenseur2HH::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurHH](#).

### 6.816.2.9 Inverse()

```
TenseurBB & Tenseur2HH::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémente [TenseurHH](#).

### 6.816.2.10 Lecture()

```
istream & Tenseur2HH::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurHH](#).

### 6.816.2.11 MaxiComposante()

```
double Tenseur2HH::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurHH](#).

### 6.816.2.12 operator"!="()

```
int Tenseur2HH::operator!= (
    const TenseurHH & ) const [virtual]
```

test

Implémente [TenseurHH](#).

### 6.816.2.13 operator&&()

```
double Tenseur2HH::operator&& (
    const TenseurBB & ) const [virtual]
```

produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémente [TenseurHH](#).

### 6.816.2.14 operator()()

```
double Tenseur2HH::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante i,j du tenseur acces en lecture seule.



Implémente [TenseurHH](#).

#### 6.816.2.15 operator\*() [1/4]

```
CoordonneeH Tenseur2HH::operator* (
    const CoordonneeB & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurHH](#).

#### 6.816.2.16 operator\*() [2/4]

```
TenseurHH & Tenseur2HH::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurHH](#).

#### 6.816.2.17 operator\*() [3/4]

```
TenseurHB & Tenseur2HH::operator* (
    const TenseurBB & ) const [virtual]
```

idem avec BB

Implémente [TenseurHH](#).

#### 6.816.2.18 operator\*() [4/4]

```
TenseurHH & Tenseur2HH::operator* (
    const TenseurBH & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté avec BH

Implémente [TenseurHH](#).

#### 6.816.2.19 operator\*=( )

```
void Tenseur2HH::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurHH](#).

#### 6.816.2.20 operator+( )

```
TenseurHH & Tenseur2HH::operator+ (
    const TenseurHH & ) const [virtual]
```

operations +

Implémente [TenseurHH](#).

#### 6.816.2.21 operator+=( )

```
void Tenseur2HH::operator+= (
    const TenseurHH & ) [virtual]
```

operations +=

Implémente [TenseurHH](#).

**6.816.2.22 operator-() [1/2]**

`TenseurHH & Tenseur2HH::operator- ( ) const [virtual]`  
operations -  
Implémente [TenseurHH](#).

**6.816.2.23 operator-() [2/2]**

`TenseurHH & Tenseur2HH::operator- ( const TenseurHH & ) const [virtual]`  
operations - tens  
Implémente [TenseurHH](#).

**6.816.2.24 operator-=()**

`void Tenseur2HH::operator-= ( const TenseurHH & ) [virtual]`  
operations -=  
Implémente [TenseurHH](#).

**6.816.2.25 operator/()**

`TenseurHH & Tenseur2HH::operator/ ( const double & ) const [virtual]`  
operations /  
Implémente [TenseurHH](#).

**6.816.2.26 operator/=()**

`void Tenseur2HH::operator/= ( const double & ) [virtual]`  
operations +/=  
Implémente [TenseurHH](#).

**6.816.2.27 operator=()**

`TenseurHH & Tenseur2HH::operator= ( const TenseurHH & ) [virtual]`  
operations =  
Implémente [TenseurHH](#).

**6.816.2.28 operator==( )**

`int Tenseur2HH::operator==( const TenseurHH & ) const [virtual]`  
test  
Implémente [TenseurHH](#).

**6.816.2.29 Transpose()**

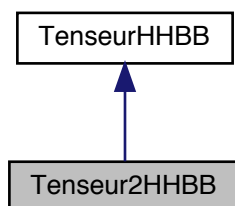
`TenseurHH & Tenseur2HH::Transpose ( ) const [virtual]`  
ATTENTION creation d'un tenseur transpose qui est supprime par Libere.  
Implémente [TenseurHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

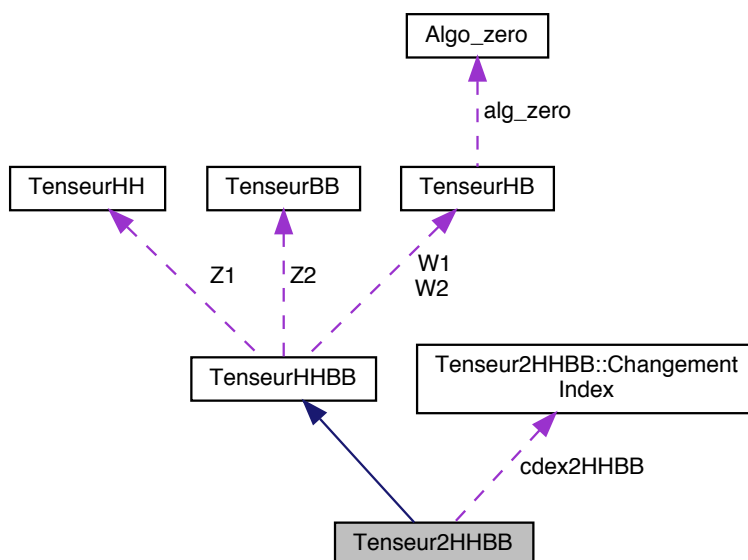
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.817 Référence de la classe Tenseur2HHBB

Grappe d'héritage de Tenseur2HHBB:



Grappe de collaboration de Tenseur2HHBB:



### Classes

- class [ChangementIndex](#)

### Fonctions membres publiques

- `Tenseur2HHBB` (const double val)

- **Tenseur2HHBB** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)
- **Tenseur2HHBB** (const [TenseurHHBB](#) &)
- **Tenseur2HHBB** (const [Tenseur2HHBB](#) &)
- void **Inita** (double val)
- [TenseurHHBB](#) & **operator+** (const [TenseurHHBB](#) &) const
- void **operator+=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator-** () const
- [TenseurHHBB](#) & **operator-** (const [TenseurHHBB](#) &) const
- void **operator-=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurHHBB](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurHH](#) &) const
- [TenseurBBHH](#) & **Transpose1et2avec3et4** () const
- virtual void **Affectation\_trans\_dimension** (const [TenseurHHBB](#) &B, bool plusZero)
- int **operator==** (const [TenseurHHBB](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort) const
- void **Affiche\_bidim** (ostream &sort) const
- [TenseurBB](#) & **Prod\_gauche** (const [TenseurBB](#) &F) const

## Fonctions membres publiques statiques

- static [TenseurHHBB](#) & **Prod\_tensoriel** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)

## Attributs publics statiques

- static const [ChangementIndex](#) **cdex2HHBB**  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 2*

## Attributs protégés

- listdouble9Iter **ipointe**

## Amis

- istream & **operator>>** (istream &, [Tenseur2HHBB](#) &)
- ostream & **operator<<** (ostream &, const [Tenseur2HHBB](#) &)

## Membres hérités additionnels

### 6.817.1 Documentation des fonctions membres

#### 6.817.1.1 Affectation\_trans\_dimension()

```
virtual void Tenseur2HHBB::Affectation_trans_dimension (
    const TenseurHHBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.817.1.2 Change()

```
void Tenseur2HHBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.817.1.3 ChangePlus()

```
void Tenseur2HHBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.817.1.4 Ecriture()

```
ostream & Tenseur2HHBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.817.1.5 Inita()

```
void Tenseur2HHBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.817.1.6 Lecture()

```
istream & Tenseur2HHBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.817.1.7 MaxiComposante()

```
double Tenseur2HHBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.817.1.8 operator&&()

```
TenseurHH & Tenseur2HHBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.817.1.9 operator>()()

```
double Tenseur2HHBB::operator() (
    int i,
```

```
int j,  
int k,  
int l ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.817.1.10 operator\*()

```
TenseurHHBB & Tenseur2HHBB::operator* (  
const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.817.1.11 operator\*=( )

```
void Tenseur2HHBB::operator*= (  
const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.817.1.12 operator+( )

```
TenseurHHBB & Tenseur2HHBB::operator+ (  
const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.817.1.13 operator+=( )

```
void Tenseur2HHBB::operator+= (  
const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.817.1.14 operator-( ) [1/2]

```
TenseurHHBB & Tenseur2HHBB::operator- ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.817.1.15 operator-( ) [2/2]

```
TenseurHHBB & Tenseur2HHBB::operator- (  
const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.817.1.16 operator-=( )

```
void Tenseur2HHBB::operator-=  
const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.817.1.17 operator/( )

```
TenseurHHBB & Tenseur2HHBB::operator/ (  
const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.817.1.18 operator/=()**

```
void Tenseur2HHBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.817.1.19 operator=()**

```
TenseurHHBB & Tenseur2HHBB::operator= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.817.1.20 operator==()**

```
int Tenseur2HHBB::operator== (
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.817.1.21 Prod\_gauche()**

```
TenseurBB & Tenseur2HHBB::Prod_gauche (
    const TenseurBB & F ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.817.1.22 Transpose1et2avec3et4()**

```
TenseurBBHH & Tenseur2HHBB::Transpose1et2avec3et4 ( ) const [virtual]
```

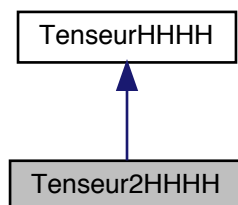
Implémente [TenseurHHBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

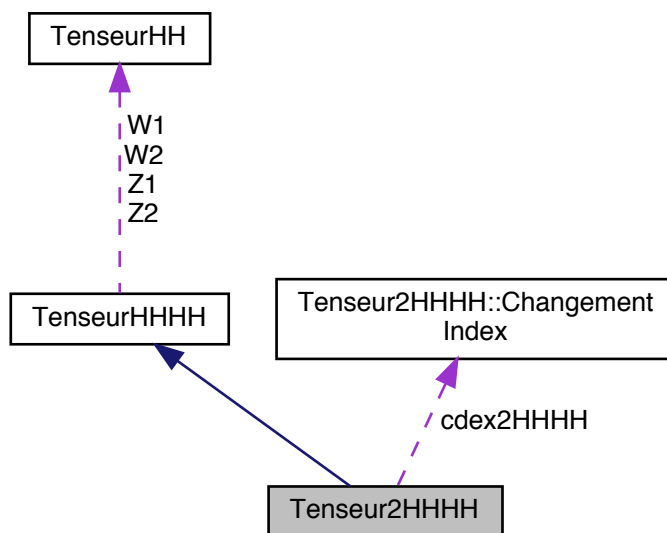
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[EnteteTenseur.h](#)

**6.818 Référence de la classe Tenseur2HHHH**

Graphe d'héritage de Tenseur2HHHH:



Graphe de collaboration de Tenseur2HHHH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur2HHHH** (const double val)
- **Tenseur2HHHH** (bool normal, const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- **Tenseur2HHHH** (const [TenseurHHHH](#) &)
- **Tenseur2HHHH** (const [Tenseur2HHHH](#) &)
- void **Inita** (double val)
- [TenseurHHHH](#) & **operator+** (const [TenseurHHHH](#) &) const
- void **operator+=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator-** () const
- [TenseurHHHH](#) & **operator-** (const [TenseurHHHH](#) &) const
- void **operator-=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator=** (const [Tenseur2HHHH](#) &B)
- [TenseurHHHH](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurHHHH](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &) const
- [TenseurHHHH](#) & **Transpose1et2avec3et4** () const
- virtual void **Affectation\_trans\_dimension** (const [TenseurHHHH](#) &B, bool plusZero)
- void **TransfertDunTenseurGeneral** (const [TenseurHHHH](#) &aHHHH)
- [TenseurBBHH](#) & **Baisse2premiersIndices** ()
- [TenseurHHBB](#) & **Baisse2derniersIndices** ()
- [TenseurHHHH](#) & **Baselocale** ([TenseurHHHH](#) &A, const [BaseH](#) &gi) const
- [TenseurHHHH](#) & **ChangeBase** ([TenseurHHHH](#) &A, const [BaseB](#) &gi) const
- int **operator==** (const [TenseurHHHH](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const



- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort) const
- void [Affiche\\_bidim](#) (ostream &sort) const
- [TenseurHH](#) & [Prod\\_gauche](#) (const [TenseurBB](#) &F) const

## Fonctions membres publiques statiques

- static [TenseurHHHH](#) & [Prod\\_tensoriel](#) (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & [Prod\\_tensoriel\\_barre](#) (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & [Var\\_tenseur\\_dans\\_nouvelle\\_base](#) (const [Mat\\_pleine](#) &gamma, [Tenseur2HHHH](#) &var\_tensHHHH, const [Tableau2](#)< [Tenseur2HH](#) > &var\_gamma, const [Tenseur2HH](#) &tensHH)  
— *pour mémoire* —

## Attributs publics statiques

- static const [ChangementIndex](#) [cdex2HHHH](#)  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 2*

## Attributs protégés

- listdouble9Iter [ipointe](#)

## Amis

- istream & [operator](#)>> (istream &, [Tenseur2HHHH](#) &)
- ostream & [operator](#)<< (ostream &, const [Tenseur2HHHH](#) &)

## Membres hérités additionnels

### 6.818.1 Documentation des fonctions membres

#### 6.818.1.1 Affectation\_trans\_dimension()

```
virtual void Tenseur2HHHH::Affectation_trans_dimension (
    const TenseurHHHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.2 Change()

```
void Tenseur2HHHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.3 ChangePlus()

```
void Tenseur2HHHH::ChangePlus (
    int i,
    int j,
    int k,
```

```
int l,  
const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.4 Ecriture()

```
ostream & Tenseur2HHHH::Ecriture (  
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.5 Inita()

```
void Tenseur2HHHH::Inita (  
    double val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.6 Lecture()

```
istream & Tenseur2HHHH::Lecture (  
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.7 MaxiComposante()

```
double Tenseur2HHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.8 operator&&()

```
TenseurHH & Tenseur2HHHH::operator&& (  
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.9 operator>()()

```
double Tenseur2HHHH::operator() (  
    int i,  
    int j,  
    int k,  
    int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.818.1.10 operator\*()

```
TenseurHHHH & Tenseur2HHHH::operator* (  
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.11 operator\*=( )**

```
void Tenseur2HHHH::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.12 operator+( )**

```
TenseurHHHH & Tenseur2HHHH::operator+ (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.13 operator+=( )**

```
void Tenseur2HHHH::operator+= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.14 operator-( ) [1/2]**

```
TenseurHHHH & Tenseur2HHHH::operator- ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.15 operator-( ) [2/2]**

```
TenseurHHHH & Tenseur2HHHH::operator- (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.16 operator--( )**

```
void Tenseur2HHHH::operator--= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.17 operator/( )**

```
TenseurHHHH & Tenseur2HHHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.18 operator/=( )**

```
void Tenseur2HHHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.818.1.19 operator=( )**

```
TenseurHHHH & Tenseur2HHHH::operator= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.818.1.20 operator==( )

```
int Tenseur2HHHH::operator==(
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.818.1.21 Prod\_gauche( )

```
TenseurHH & Tenseur2HHHH::Prod_gauche (
    const TenseurBB & F ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.818.1.22 Transpose1et2avec3et4( )

```
TenseurHHHH & Tenseur2HHHH::Transpose1et2avec3et4 ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

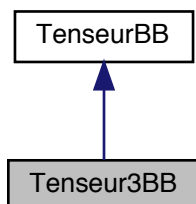
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.819 Référence de la classe Tenseur3BB

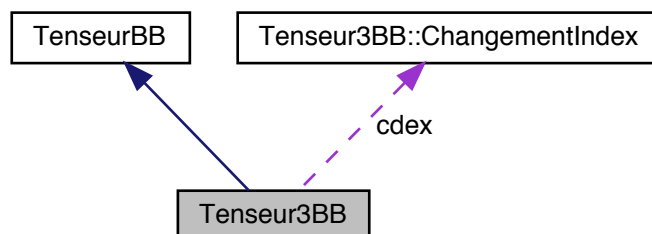
Definition des tenseur derivees de dimension 3. cas des composantes deux fois covariantes.

```
#include <Tenseur3.h>
```

Graphe d'héritage de Tenseur3BB:



Graphe de collaboration de Tenseur3BB:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur3BB** (const double val)
- **Tenseur3BB** (const double val1, const double val2, const double val3, const double val4, const double val5, const double val6)
- **Tenseur3BB** (const [TenseurBB](#) &)
- **Tenseur3BB** (const [Tenseur3BB](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurBB](#) & **operator+** (const [TenseurBB](#) &) const  
*operations +*
- void **operator+=** (const [TenseurBB](#) &)  
*operations +=*
- [TenseurBB](#) & **operator-** () const  
*operations -*
- [TenseurBB](#) & **operator-** (const [TenseurBB](#) &) const  
*operations - tens*
- void **operator-=** (const [TenseurBB](#) &)  
*operations -=*
- [TenseurBB](#) & **operator=** (const [TenseurBB](#) &)  
*operations =*
- [TenseurBB](#) & **operator=** (const [Tenseur3BB](#) &B)
- [TenseurBB](#) & **operator=** (const [Tenseur\\_ns3BB](#) &B)
- [TenseurBB](#) & **operator\*** (const double &) const  
*operations \* double*
- void **operator\*=** (const double &)  
*operations \*= double*
- [TenseurBB](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_2D\_a\_3D** (const [Tenseur2BB](#) &B, bool plusZero)
- void **Affectation\_trans\_dimension** (const [TenseurBB](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- [CoordonneeB](#) **operator\*** (const [CoordonneeH](#) &) const  
*produit contracte avec un vecteur*
- [TenseurBB](#) & **operator\*** (const [TenseurHB](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté*

- `TenseurBH & operator*` (const `TenseurHH &`) const  
*idem en BH*
- double `operator&&` (const `TenseurHH &`) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int `operator==` (const `TenseurBB &`) const  
*test*
- int `operator!=` (const `TenseurBB &`) const  
*test*
- double `Det` () const  
*determinant de la matrice des coordonnees*
- `TenseurHH & Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
- `TenseurBB & Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual `TenseurHH & Monte2Indices` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurHB & MontePremierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurBH & MonteDernierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- double `MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- double & `Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double `operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
- istream & `Lecture` (istream &entree)  
*lecture et écriture de données*
- ostream & `Ecriture` (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- static `TenseurBB & Prod_tensoriel` (const `CoordonneeB &aB`, const `CoordonneeB &bB`)
- static int `OdVect` (const int i, const int j)
- static int `idx_i` (const int k)
- static int `idx_j` (const int k)

## Attributs protégés

- listdouble6lter `ipointe`

## Attributs protégés statiques

- static const `ChangementIndex cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class `Tenseur2BB`
- class `Tenseur3HH`
- istream & `operator>>` (istream &, `Tenseur3BB &`)
- ostream & `operator<<` (ostream &, const `Tenseur3BB &`)

## Membres hérités additionnels

### 6.819.1 Description détaillée

Definition des tenseur derivees de dimension 3. cas des composantes deux fois covariantes.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

**6.819.2 Documentation des fonctions membres****6.819.2.1 Affectation\_trans\_dimension()**

```
void Tenseur3BB::Affectation_trans_dimension (
    const TenseurBB & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurBB](#).

**6.819.2.2 Coor()**

```
double & Tenseur3BB::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.  
Implémente [TenseurBB](#).

**6.819.2.3 Det()**

```
double Tenseur3BB::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees  
Implémente [TenseurBB](#).

**6.819.2.4 Ecriture()**

```
ostream & Tenseur3BB::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données  
Implémente [TenseurBB](#).

**6.819.2.5 Inita()**

```
void Tenseur3BB::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val  
Implémente [TenseurBB](#).

**6.819.2.6 Inverse()**

`TenseurHH & Tenseur3BB::Inverse ( ) const [virtual]`  
 calcul du tenseur inverse par rapport au produit contracté  
 Implémente [TenseurBB](#).

**6.819.2.7 Lecture()**

`istream & Tenseur3BB::Lecture (`  
     `istream & entree ) [virtual]`  
 lecture et écriture de données  
 Implémente [TenseurBB](#).

**6.819.2.8 MaxiComposante()**

`double Tenseur3BB::MaxiComposante ( ) const [virtual]`  
 calcul du maximum en valeur absolu des composantes du tenseur  
 Implémente [TenseurBB](#).

**6.819.2.9 Monte2Indices()**

`virtual TenseurHH & Tenseur3BB::Monte2Indices ( ) const [virtual]`  
 — manipulation d'indice — -> création de nouveaux tenseurs  
 Implémente [TenseurBB](#).

**6.819.2.10 MonteDernierIndice()**

`virtual TenseurBH & Tenseur3BB::MonteDernierIndice ( ) const [virtual]`  
 — manipulation d'indice — -> création de nouveaux tenseurs  
 Implémente [TenseurBB](#).

**6.819.2.11 MontePremierIndice()**

`virtual TenseurHB & Tenseur3BB::MontePremierIndice ( ) const [virtual]`  
 — manipulation d'indice — -> création de nouveaux tenseurs  
 Implémente [TenseurBB](#).

**6.819.2.12 operator"!="()**

`int Tenseur3BB::operator!= (`  
     `const TenseurBB & ) const [virtual]`  
 test  
 Implémente [TenseurBB](#).

**6.819.2.13 operator&&()**

`double Tenseur3BB::operator&& (`  
     `const TenseurHH & ) const [virtual]`  
 produit contracté contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe  
 Implémente [TenseurBB](#).



**6.819.2.14 operator>()**

```
double Tenseur3BB::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante i,j du tenseur acces en lecture seulement.

Implémente [TenseurBB](#).

**6.819.2.15 operator\*() [1/4]**

```
CoordonneeB Tenseur3BB::operator* (
    const CoordonneeH & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurBB](#).

**6.819.2.16 operator\*() [2/4]**

```
TenseurBB & Tenseur3BB::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurBB](#).

**6.819.2.17 operator\*() [3/4]**

```
TenseurBB & Tenseur3BB::operator* (
    const TenseurHB & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B \rightarrow$  donc c'est l'indice du milieu qui est contracté

Implémente [TenseurBB](#).

**6.819.2.18 operator\*() [4/4]**

```
TenseurBH & Tenseur3BB::operator* (
    const TenseurHH & ) const [virtual]
```

idem en BH

Implémente [TenseurBB](#).

**6.819.2.19 operator\*=( )**

```
void Tenseur3BB::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurBB](#).

**6.819.2.20 operator+()**

```
TenseurBB & Tenseur3BB::operator+ (
    const TenseurBB & ) const [virtual]
```

operations +

Implémente [TenseurBB](#).

**6.819.2.21 operator+=()**

```
void Tenseur3BB::operator+= (
    const TenseurBB & ) [virtual]
```

operations +=

Implémente [TenseurBB](#).

**6.819.2.22 operator-() [1/2]**

```
TenseurBB & Tenseur3BB::operator- ( ) const [virtual]
```

operations -

Implémente [TenseurBB](#).

**6.819.2.23 operator-() [2/2]**

```
TenseurBB & Tenseur3BB::operator- (
    const TenseurBB & ) const [virtual]
```

operations - tens

Implémente [TenseurBB](#).

**6.819.2.24 operator-=()**

```
void Tenseur3BB::operator-= (
    const TenseurBB & ) [virtual]
```

operations -=

Implémente [TenseurBB](#).

**6.819.2.25 operator/()**

```
TenseurBB & Tenseur3BB::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurBB](#).

**6.819.2.26 operator/=()**

```
void Tenseur3BB::operator/= (
    const double & ) [virtual]
```

operations /=

Implémente [TenseurBB](#).

**6.819.2.27 operator=()**

```
TenseurBB & Tenseur3BB::operator= (
    const TenseurBB & ) [virtual]
```

operations =

Implémente [TenseurBB](#).

**6.819.2.28 operator==(())**

```
int Tenseur3BB::operator==(
    const TenseurBB & ) const [virtual]
```

test

Implémente [TenseurBB](#).

### 6.819.2.29 Transpose()

`TenseurBB & Tenseur3BB::Transpose ( ) const [virtual]`

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

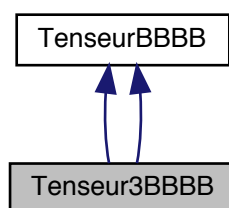
Implémente [TenseurBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

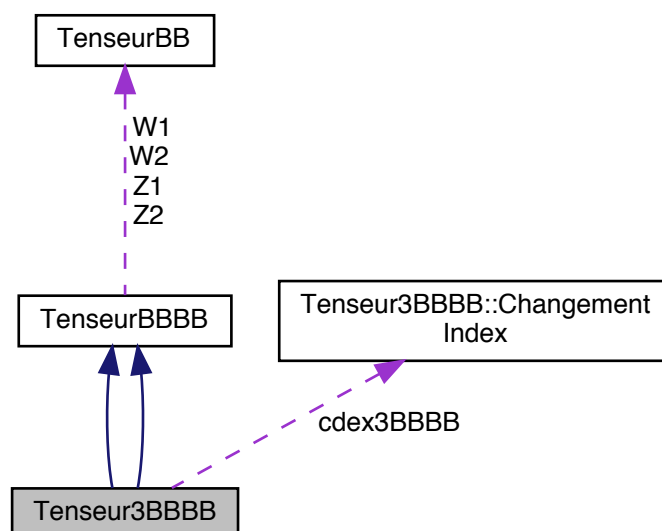
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.820 Référence de la classe Tenseur3BBBB

Graphe d'héritage de Tenseur3BBBB:



Graphe de collaboration de Tenseur3BBBB:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur3BBBB** (const double val)
- **Tenseur3BBBB** (bool normal, const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- **Tenseur3BBBB** (const [TenseurBBBB](#) &)
- **Tenseur3BBBB** (const [Tenseur3BBBB](#) &)
- void **Inita** (double val)
- [TenseurBBBB](#) & **operator+** (const [TenseurBBBB](#) &) const
- void **operator+=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator-** () const
- [TenseurBBBB](#) & **operator-** (const [TenseurBBBB](#) &) const
- void **operator-=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator=** (const [Tenseur3HHHH](#) &B)
- [TenseurBBBB](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurBBBB](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBB](#) & **operator&&** (const [TenseurHH](#) &) const
- [TenseurBB](#) & **ContractionVerticale** (const [TenseurHH](#) &) const
- [TenseurBBBB](#) & **operator&&** (const [TenseurHHBB](#) &) const
- [TenseurBBHH](#) & **operator&&** (const [TenseurHHHH](#) &) const
- [TenseurBBBB](#) & **Transpose1et2avec3et4** () const
- virtual void **Affectation\_trans\_dimension** (const [TenseurBBBB](#) &B, bool plusZero)
- void **TransfertDunTenseurGeneral** (const [TenseurBBBB](#) &aBBBB)
- [TenseurHHBB](#) & **Monte2premiersIndices** ()
- [TenseurBBHH](#) & **Monte2derniersIndices** ()
- [TenseurHHHH](#) & **Monte4Indices** ()
- [TenseurBBBB](#) & **Baselocale** ([TenseurBBBB](#) &A, const [BaseB](#) &gi) const
- [TenseurBBBB](#) & **ChangeBase** ([TenseurBBBB](#) &A, const [BaseH](#) &gi) const
- int **operator==** (const [TenseurBBBB](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort) const
- void **Affiche\_bidim** (ostream &sort) const
- [TenseurBB](#) & **Prod\_gauche** (const [TenseurHH](#) &F) const
- **Tenseur3BBBB** (const double val)
- **Tenseur3BBBB** (bool normal, const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- **Tenseur3BBBB** (const [TenseurBBBB](#) &)
- **Tenseur3BBBB** (const [Tenseur3BBBB](#) &)
- [TenseurBBBB](#) & **operator+** (const [TenseurBBBB](#) &) const
- void **operator+=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator-** () const
- [TenseurBBBB](#) & **operator-** (const [TenseurBBBB](#) &) const
- void **operator-=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator=** (const [Tenseur3HHHH](#) &B)
- [TenseurBBBB](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurBBBB](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBB](#) & **operator&&** (const [TenseurHH](#) &) const
- [TenseurBBBB](#) & **Transpose1et2avec3et4** () const
- int **operator==** (const [TenseurBBBB](#) &) const
- void **change** (int i, int j, int k, int l, double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort)

## Fonctions membres publiques statiques

- static [TenseurBBBB](#) & [Prod\\_tensoriel](#) (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- static [TenseurBBBB](#) & [Prod\\_tensoriel\\_barre](#) (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- static [TenseurBBBB](#) & [Prod\\_tensoriel](#) (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- static [TenseurBBBB](#) & [Prod\\_tensoriel\\_barre](#) (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)

## Attributs publics statiques

- static const [ChangementIndex](#) [cdex3BBBB](#)  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3*

## Attributs protégés

- listdouble36lter [ipointe](#)

## Attributs protégés statiques

- static [ChangementIndex](#) [cdex3BBBB](#)

## Amis

- istream & [operator](#)>> (istream &, [Tenseur3BBBB](#) &)
- ostream & [operator](#)<< (ostream &, const [Tenseur3BBBB](#) &)
- istream & [operator](#)>> (istream &, [Tenseur3BBBB](#) &)
- ostream & [operator](#)<< (ostream &, const [Tenseur3BBBB](#) &)

## Membres hérités additionnels

### 6.820.1 Documentation des fonctions membres

#### 6.820.1.1 Affectation\_trans\_dimension()

```
virtual void Tenseur3BBBB::Affectation_trans_dimension (
    const TenseurBBBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.2 Change()

```
void Tenseur3BBBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.3 ChangePlus()

```
void Tenseur3BBBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.4 Ecriture()

```
ostream & Tenseur3BBBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.5 Initia()

```
void Tenseur3BBBB::Initia (
    double val ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.6 Lecture() [1/2]

```
istream & Tenseur3BBBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.7 Lecture() [2/2]

```
istream & Tenseur3BBBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.8 MaxiComposante() [1/2]

```
double Tenseur3BBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.9 MaxiComposante() [2/2]

```
double Tenseur3BBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.10 operator&&() [1/2]

```
TenseurBB & Tenseur3BBBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.11 operator&&() [2/2]

```
TenseurBB & Tenseur3BBBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.12 operator>() [1/2]**

```
double Tenseur3BBBB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.13 operator>() [2/2]**

```
double Tenseur3BBBB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.14 operator\*() [1/2]**

```
TenseurBBBB & Tenseur3BBBB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.15 operator\*() [2/2]**

```
TenseurBBBB & Tenseur3BBBB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.16 operator\*=( ) [1/2]**

```
void Tenseur3BBBB::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.17 operator\*=( ) [2/2]**

```
void Tenseur3BBBB::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.18 operator+() [1/2]**

```
TenseurBBBB & Tenseur3BBBB::operator+ (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.19 operator+() [2/2]**

```
TenseurBBBB & Tenseur3BBBB::operator+ (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.20 operator+=()** [1/2]

```
void Tenseur3BBBB::operator+= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.21 operator+=()** [2/2]

```
void Tenseur3BBBB::operator+= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.22 operator-()** [1/4]

```
TenseurBBBB & Tenseur3BBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.23 operator-()** [2/4]

```
TenseurBBBB & Tenseur3BBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.24 operator-()** [3/4]

```
TenseurBBBB & Tenseur3BBBB::operator- (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.25 operator-()** [4/4]

```
TenseurBBBB & Tenseur3BBBB::operator- (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.26 operator-=()** [1/2]

```
void Tenseur3BBBB::operator-= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.27 operator-=()** [2/2]

```
void Tenseur3BBBB::operator-= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).



**6.820.1.28 operator()** [1/2]

```
TenseurBBBB & Tenseur3BBBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.29 operator()** [2/2]

```
TenseurBBBB & Tenseur3BBBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.30 operator/=( )** [1/2]

```
void Tenseur3BBBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.31 operator/=( )** [2/2]

```
void Tenseur3BBBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.32 operator=( )** [1/2]

```
TenseurBBBB & Tenseur3BBBB::operator= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.33 operator=( )** [2/2]

```
TenseurBBBB & Tenseur3BBBB::operator= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.34 operator==( )** [1/2]

```
int Tenseur3BBBB::operator==(
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.35 operator==( )** [2/2]

```
int Tenseur3BBBB::operator==(
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.820.1.36 Prod\_gauche()**

```
TenseurBB & Tenseur3BBBB::Prod_gauche (
    const TenseurHH & F ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.820.1.37 `Transpose1et2avec3et4()` [1/2]

[TenseurBBBB](#) & `Tenseur3BBBB::Transpose1et2avec3et4 ( ) const [virtual]`

Implémente [TenseurBBBB](#).

#### 6.820.1.38 `Transpose1et2avec3et4()` [2/2]

[TenseurBBBB](#) & `Tenseur3BBBB::Transpose1et2avec3et4 ( ) const [virtual]`

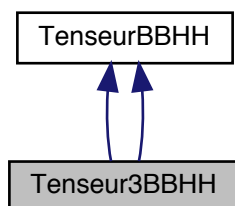
Implémente [TenseurBBBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

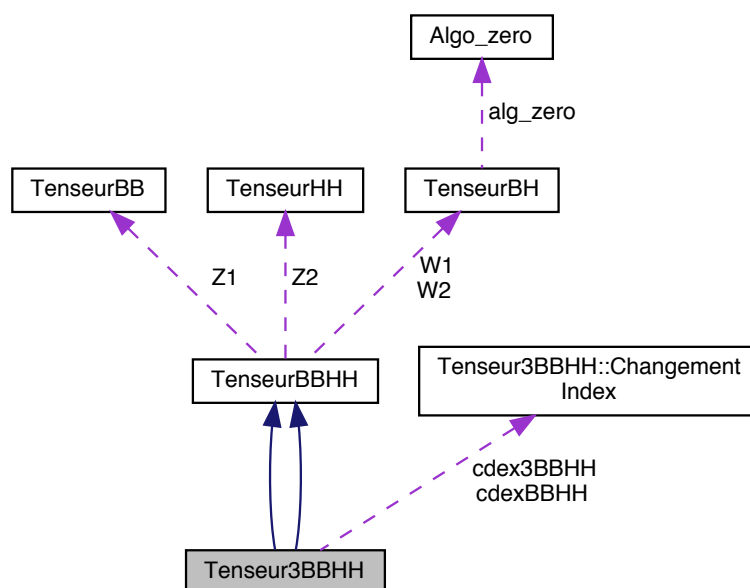
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ-3.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/TenseurQ3.h.old.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Tenseur/EnteteTenseur.h`

## 6.821 Référence de la classe `Tenseur3BBHH`

Grphe d'héritage de `Tenseur3BBHH`:



Graphe de collaboration de Tenseur3BBHH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur3BBHH** (const double val)
- **Tenseur3BBHH** (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)
- **Tenseur3BBHH** (const [TenseurBBHH](#) &)
- **Tenseur3BBHH** (const [Tenseur3BBHH](#) &)
- void **Inita** (double val)
- [TenseurBBHH](#) & **operator+** (const [TenseurBBHH](#) &) const
- void **operator+=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator-** () const
- [TenseurBBHH](#) & **operator-** (const [TenseurBBHH](#) &) const
- void **operator-=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurBBHH](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBB](#) & **operator&&** (const [TenseurBB](#) &) const
- [TenseurBB](#) & **ContractionVerticale** (const [TenseurBB](#) &) const
- [TenseurBBBB](#) & **operator&&** (const [TenseurBBBB](#) &) const
- [TenseurBBHH](#) & **operator&&** (const [TenseurBBHH](#) &) const
- [TenseurHHBB](#) & **Transpose1et2avec3et4** () const
- virtual void **Affectation\_trans\_dimension** (const [TenseurBBHH](#) &B, bool plusZero)
- int **operator==** (const [TenseurBBHH](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)

- ostream & [Ecriture](#) (ostream &sort) const
- void [Affiche\\_bidim](#) (ostream &sort) const
- [TenseurHH](#) & [Prod\\_gauche](#) (const [TenseurHH](#) &F) const
- [Tenseur3BBHH](#) (const double val)
- [Tenseur3BBHH](#) (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)
- [Tenseur3BBHH](#) (const [TenseurBBHH](#) &)
- [Tenseur3BBHH](#) (const [Tenseur3BBHH](#) &)
- [TenseurBBHH](#) & [operator+](#) (const [TenseurBBHH](#) &) const
- void [operator+=](#) (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & [operator-](#) () const
- [TenseurBBHH](#) & [operator-](#) (const [TenseurBBHH](#) &) const
- void [operator-=](#) (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & [operator=](#) (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & [operator\\*](#) (const double &) const
- void [operator\\*=](#) (const double &)
- [TenseurBBHH](#) & [operator/](#) (const double &) const
- void [operator/=](#) (const double &)
- [TenseurBB](#) & [operator&&](#) (const [TenseurBB](#) &) const
- [TenseurHHBB](#) & [Transpose1et2avec3et4](#) () const
- int [operator==](#) (const [TenseurBBHH](#) &) const
- void [change](#) (int i, int j, int k, int l, double &val)
- double [operator\(\)](#) (int i, int j, int k, int l) const
- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort)

## Fonctions membres publiques statiques

- static [TenseurBBHH](#) & [Prod\\_tensoriel](#) (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)
- static [TenseurBBHH](#) & [Prod\\_tensoriel\\_barre](#) (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)
- static [TenseurBBHH](#) & [Var\\_tenseur\\_dans\\_nouvelle\\_base](#) (const [Mat\\_pleine](#) &beta, [Tenseur3BBHH](#) &var↔  
\_tensBBHH, const [Tableau2](#)< [Tenseur3HH](#) > &var\_beta, const [Tenseur3BB](#) &tensBB)  
— *pour mémoire* —
- static [TenseurBBHH](#) & [Prod\\_tensoriel](#) (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)

## Attributs publics statiques

- static const [ChangementIndex](#) [cdex3BBHH](#)  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3*

## Attributs protégés

- listdouble36lter [ipointe](#)

## Attributs protégés statiques

- static [ChangementIndex](#) [cdexBBHH](#)

## Amis

- istream & [operator](#)>> (istream &, [Tenseur3BBHH](#) &)
- ostream & [operator](#)<< (ostream &, const [Tenseur3BBHH](#) &)
- istream & [operator](#)>> (istream &, [Tenseur3BBHH](#) &)
- ostream & [operator](#)<< (ostream &, const [Tenseur3BBHH](#) &)

## Membres hérités additionnels

### 6.821.1 Documentation des fonctions membres

### 6.821.1.1 Affectation\_trans\_dimension()

```
virtual void Tenseur3BBHH::Affectation_trans_dimension (
    const TenseurBBHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.821.1.2 Change()

```
void Tenseur3BBHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.821.1.3 ChangePlus()

```
void Tenseur3BBHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.821.1.4 Ecriture()

```
ostream & Tenseur3BBHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.821.1.5 Inita()

```
void Tenseur3BBHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.821.1.6 Lecture() [1/2]

```
istream & Tenseur3BBHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.821.1.7 Lecture() [2/2]

```
istream & Tenseur3BBHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.8 MaxiComposante()** [1/2]

```
double Tenseur3BBHH::MaxiComposante ( ) const [virtual]
Implémente TenseurBBHH.
```

**6.821.1.9 MaxiComposante()** [2/2]

```
double Tenseur3BBHH::MaxiComposante ( ) const [virtual]
Implémente TenseurBBHH.
```

**6.821.1.10 operator&&()** [1/2]

```
TenseurBB & Tenseur3BBHH::operator&& (
    const TenseurBB & ) const [virtual]
Implémente TenseurBBHH.
```

**6.821.1.11 operator&&()** [2/2]

```
TenseurBB & Tenseur3BBHH::operator&& (
    const TenseurBB & ) const [virtual]
Implémente TenseurBBHH.
```

**6.821.1.12 operator>()()** [1/2]

```
double Tenseur3BBHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
Implémente TenseurBBHH.
```

**6.821.1.13 operator>()()** [2/2]

```
double Tenseur3BBHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
Implémente TenseurBBHH.
```

**6.821.1.14 operator\*()** [1/2]

```
TenseurBBHH & Tenseur3BBHH::operator* (
    const double & ) const [virtual]
Implémente TenseurBBHH.
```

**6.821.1.15 operator\*()** [2/2]

```
TenseurBBHH & Tenseur3BBHH::operator* (
    const double & ) const [virtual]
Implémente TenseurBBHH.
```

**6.821.1.16 operator\*=( ) [1/2]**

```
void Tenseur3BBHH::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.17 operator\*=( ) [2/2]**

```
void Tenseur3BBHH::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.18 operator+( ) [1/2]**

```
TenseurBBHH & Tenseur3BBHH::operator+ (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.19 operator+( ) [2/2]**

```
TenseurBBHH & Tenseur3BBHH::operator+ (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.20 operator+=( ) [1/2]**

```
void Tenseur3BBHH::operator+= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.21 operator+=( ) [2/2]**

```
void Tenseur3BBHH::operator+= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.22 operator-( ) [1/4]**

```
TenseurBBHH & Tenseur3BBHH::operator- ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.23 operator-( ) [2/4]**

```
TenseurBBHH & Tenseur3BBHH::operator- ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.24 operator-( ) [3/4]**

```
TenseurBBHH & Tenseur3BBHH::operator- (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.25 operator-() [4/4]**

```
TenseurBBHH & Tenseur3BBHH::operator- (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.26 operator-=() [1/2]**

```
void Tenseur3BBHH::operator-= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.27 operator-=() [2/2]**

```
void Tenseur3BBHH::operator-= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.28 operator/() [1/2]**

```
TenseurBBHH & Tenseur3BBHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.29 operator/() [2/2]**

```
TenseurBBHH & Tenseur3BBHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.30 operator/=() [1/2]**

```
void Tenseur3BBHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.31 operator/=() [2/2]**

```
void Tenseur3BBHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.32 operator=() [1/2]**

```
TenseurBBHH & Tenseur3BBHH::operator= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.821.1.33 operator=() [2/2]**

```
TenseurBBHH & Tenseur3BBHH::operator= (
    const TenseurBBHH & ) [virtual]
```



Implémente [TenseurBBHH](#).

#### 6.821.1.34 operator==( ) [1/2]

```
int Tenseur3BBHH::operator==(
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.821.1.35 operator==( ) [2/2]

```
int Tenseur3BBHH::operator==(
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.821.1.36 Prod\_gauche()

```
TenseurHH & Tenseur3BBHH::Prod_gauche (
    const TenseurHH & F ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.821.1.37 Transpose1et2avec3et4() [1/2]

```
TenseurHHBB & Tenseur3BBHH::Transpose1et2avec3et4 ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.821.1.38 Transpose1et2avec3et4() [2/2]

```
TenseurHHBB & Tenseur3BBHH::Transpose1et2avec3et4 ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

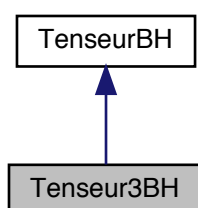
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3.h.old.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.822 Référence de la classe Tenseur3BH

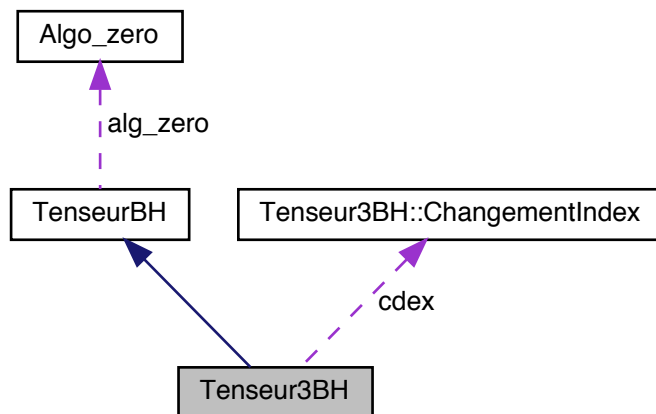
Definition des tenseur derivees de dimension 3. cas des composantes mixtes BH.

```
#include <Tenseur3.h>
```

Grphe d'héritage de Tenseur3BH:



Graphe de collaboration de Tenseur3BH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur3BH** (const double val)
- **Tenseur3BH** (const double val1, const double val2, const double val3, const double val4, const double val5, const double val6, const double val7, const double val8, const double val9)
- **Tenseur3BH** (const [TenseurBH](#) &)
- **Tenseur3BH** (const [Tenseur3BH](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurBH](#) & **operator+** (const [TenseurBH](#) &) const  
*operations +*
- void **operator+=** (const [TenseurBH](#) &)  
*operations +=*
- [TenseurBH](#) & **operator-** () const  
*operations opposé*
- [TenseurBH](#) & **operator-** (const [TenseurBH](#) &) const  
*operations -*
- void **operator-=** (const [TenseurBH](#) &)  
*operations -=*
- [TenseurBH](#) & **operator=** (const [TenseurBH](#) &)  
*operations =*
- [TenseurBH](#) & **operator=** (const [Tenseur3BH](#) &B)
- [TenseurBH](#) & **operator\*** (const double &) const  
*operations \**
- void **operator\*=** (const double &)  
*operations \*=*
- [TenseurBH](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_2D\_a\_3D** (const [Tenseur2BH](#) &B, bool plusZero)
- void **Affectation\_trans\_dimension** (const [TenseurBH](#) &B, bool plusZero)

- Affectation\_trans\_dimension.*
- `CoordonneeB operator*` (const `CoordonneeB` &) const  
*produit contracte avec un vecteur*
  - `TenseurBB & operator*` (const `TenseurBB` &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
  - `TenseurBH & operator*` (const `TenseurBH` &) const
  - `double operator&&` (const `TenseurBH` &) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
  - `int operator==` (const `TenseurBH` &) const  
*test*
  - `int operator!=` (const `TenseurBH` &) const
  - `TenseurBH & Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
  - `double Trace` () const  
*trace du tenseur ou premier invariant*
  - `double II` () const  
*second invariant = trace (A\*A)*
  - `double III` () const  
*troisieme invariant = trace ((A\*A)\*A)*
  - `double Det` () const  
*determinant de la matrice des coordonnees*
  - `virtual Coordonnee ValPropre` (int &cas) const  
*calcul des valeurs propres*
  - `virtual Coordonnee ValPropre` (int &cas, `Mat_pleine` &mat) const  
*calcul des valeurs propres*
  - `virtual void VecteursPropres` (const `Coordonnee` &Val\_P, int &cas, `Tableau`< `Coordonnee` > &V\_P) const  
*calcul des vecteurs propres, les valeurs propres étant déjà connues*
  - `TenseurHB & Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
  - `void PermuteHautBas` ()  
*permute Bas Haut, mais reste dans le même tenseur*
  - `double MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
  - `double & Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
  - `double operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
  - `istream & Lecture` (istream &entree)  
*lecture et écriture de données*
  - `ostream & Ecriture` (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- `static TenseurBH & Prod_tensoriel` (const `CoordonneeB` &aB, const `CoordonneeH` &bH)
- `static int OdVect` (const int i, const int j)
- `static int idx_i` (const int k)
- `static int idx_j` (const int k)

## Attributs protégés

- `listdouble9lter ipointe`

## Attributs protégés statiques

- `static const ChangementIndex cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class **Tenseur2BH**
- istream & **operator>>** (istream &, [Tenseur3BH](#) &)
- ostream & **operator<<** (ostream &, const [Tenseur3BH](#) &)

## Membres hérités additionnels

### 6.822.1 Description détaillée

Definition des tenseur derivees de dimension 3. cas des composantes mixtes BH.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.822.2 Documentation des fonctions membres

#### 6.822.2.1 Affectation\_trans\_dimension()

```
void Tenseur3BH::Affectation_trans_dimension (
    const TenseurBH & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
 plusZero = true: les données manquantes sont mises à 0  
 si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
 des données possibles

Implémente [TenseurBH](#).

#### 6.822.2.2 Coor()

```
double & Tenseur3BH::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémente [TenseurBH](#).

#### 6.822.2.3 Det()

```
double Tenseur3BH::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees

Implémente [TenseurBH](#).

#### 6.822.2.4 Ecriture()

```
ostream & Tenseur3BH::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurBH](#).

#### 6.822.2.5 II()

```
double Tenseur3BH::II ( ) const [virtual]
```

second invariant = trace (A\*A)

Implémente [TenseurBH](#).

#### 6.822.2.6 III()

```
double Tenseur3BH::III ( ) const [virtual]
```

troisieme invariant = trace ((A\*A)\*A)

Implémente [TenseurBH](#).

#### 6.822.2.7 Inita()

```
void Tenseur3BH::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurBH](#).

#### 6.822.2.8 Inverse()

```
TenseurBH & Tenseur3BH::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracte

Implémente [TenseurBH](#).

#### 6.822.2.9 Lecture()

```
istream & Tenseur3BH::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurBH](#).

#### 6.822.2.10 MaxiComposante()

```
double Tenseur3BH::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurBH](#).

#### 6.822.2.11 operator"!=(())

```
int Tenseur3BH::operator!=(
    const TenseurBH & ) const [virtual]
```

Implémente [TenseurBH](#).

#### 6.822.2.12 operator&&()

```
double Tenseur3BH::operator&& (
    const TenseurBH & ) const [virtual]
```

produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémente [TenseurBH](#).

#### 6.822.2.13 operator>()

```
double Tenseur3BH::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante i,j du tenseur acces en lecture seulement.

Implémente [TenseurBH](#).

#### 6.822.2.14 operator\*() [1/4]

```
CoordonneeB Tenseur3BH::operator* (
    const CoordonneeB & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurBH](#).

#### 6.822.2.15 operator\*() [2/4]

```
TenseurBH & Tenseur3BH::operator* (
    const double & ) const [virtual]
```

operations \*

Implémente [TenseurBH](#).

#### 6.822.2.16 operator\*() [3/4]

```
TenseurBB & Tenseur3BH::operator* (
    const TenseurBB & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté

Implémente [TenseurBH](#).

#### 6.822.2.17 operator\*() [4/4]

```
TenseurBH & Tenseur3BH::operator* (
    const TenseurBH & ) const [virtual]
```

Implémente [TenseurBH](#).

#### 6.822.2.18 operator\*=( )

```
void Tenseur3BH::operator*= (
    const double & ) [virtual]
```

operations \*=

Implémente [TenseurBH](#).

#### 6.822.2.19 operator+( )

```
TenseurBH & Tenseur3BH::operator+ (
    const TenseurBH & ) const [virtual]
```

operations +

Implémente [TenseurBH](#).

**6.822.2.20 operator+=()**

```
void Tenseur3BH::operator+= (
    const TenseurBH & ) [virtual]
```

operations +=  
Implémente [TenseurBH](#).

**6.822.2.21 operator-() [1/2]**

```
TenseurBH & Tenseur3BH::operator- ( ) const [virtual]
```

operations opposé  
Implémente [TenseurBH](#).

**6.822.2.22 operator-() [2/2]**

```
TenseurBH & Tenseur3BH::operator- (
    const TenseurBH & ) const [virtual]
```

operations -  
Implémente [TenseurBH](#).

**6.822.2.23 operator-=()**

```
void Tenseur3BH::operator-= (
    const TenseurBH & ) [virtual]
```

operations -=  
Implémente [TenseurBH](#).

**6.822.2.24 operator/()**

```
TenseurBH & Tenseur3BH::operator/ (
    const double & ) const [virtual]
```

operations /  
Implémente [TenseurBH](#).

**6.822.2.25 operator/=()**

```
void Tenseur3BH::operator/= (
    const double & ) [virtual]
```

operations /=  
Implémente [TenseurBH](#).

**6.822.2.26 operator=()**

```
TenseurBH & Tenseur3BH::operator= (
    const TenseurBH & ) [virtual]
```

operations =  
Implémente [TenseurBH](#).

**6.822.2.27 operator==(())**

```
int Tenseur3BH::operator==(
    const TenseurBH & ) const [virtual]
```

test

Implémente [TenseurBH](#).

### 6.822.2.28 PermuteHautBas()

```
void Tenseur3BH::PermuteHautBas ( ) [virtual]
```

permuté Bas Haut, mais reste dans le même tenseur  
Implémente [TenseurBH](#).

### 6.822.2.29 Trace()

```
double Tenseur3BH::Trace ( ) const [virtual]
```

trace du tenseur ou premier invariant  
Implémente [TenseurBH](#).

### 6.822.2.30 Transpose()

```
TenseurHB & Tenseur3BH::Transpose ( ) const [virtual]
```

ATTENTION création d'un tenseur transpose qui est supprimé par Libere.  
Implémente [TenseurBH](#).

### 6.822.2.31 ValPropre() [1/2]

```
virtual Coordonnee Tenseur3BH::ValPropre (
    int & cas ) const [virtual]
```

calcul des valeurs propres

valeurs propre dans le vecteur de retour, classée par ordres "décroissants"  
cas indique le cas de valeur propre:  
quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire  
dans ce cas les valeurs propres de retour sont nulles par défaut  
dim = 1, cas=1 pour une valeur propre;  
dim = 2, cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques  
dim = 3, cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)  
, cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),  
, cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du retour)  
, cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du retour)  
Implémente [TenseurBH](#).

### 6.822.2.32 ValPropre() [2/2]

```
virtual Coordonnee Tenseur3BH::ValPropre (
    int & cas,
    Mat_pleine & mat ) const [virtual]
```

calcul des valeurs propres

idem met en retour la matrice mat contient par colonne les vecteurs propre  
elle doit avoir la dimension du tenseur  
les vecteurs propre sont exprimés dans le repère dual (contrairement au HB) pour les tenseurs dim 3  
pour dim=2: le premier vecteur propre est exprimé dans le repère dual  
le second vecteur propre est exprimé dans le repère naturel  
pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
Implémente [TenseurBH](#).



### 6.822.2.33 VecteursPropres()

```
virtual void Tenseur3BH::VecteursPropres (
    const Coordonnee & Val_P,
    int & cas,
    Tableau< Coordonnee > & V_P ) const [virtual]
```

calcul des vecteurs propres, les valeurs propres étant déjà connues

ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres étant déjà connues

en retour VP les vecteurs propre : doivent avoir la dimension du tenseur  
les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3 pour dim=2:le premier vecteur propre est exprime dans le repere naturel le second vecteur propre est exprimé dans le repère dual pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
en sortie cas = -1 s'il y a eu un problème, dans ce cas, V\_P est quelconque sinon si tout est ok, cas est identique en sortie avec l'entrée  
Implémente [TenseurBH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

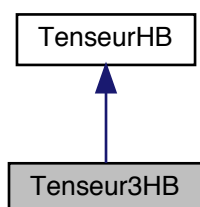
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.823 Référence de la classe Tenseur3HB

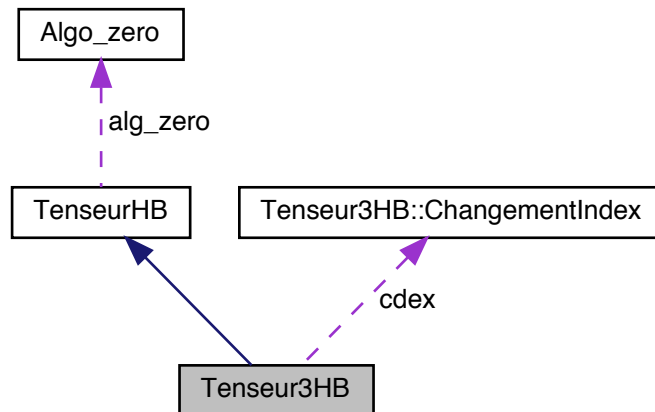
Definition des tenseur derivees de dimension 3. cas des composantes mixtes HB.

```
#include <Tenseur3.h>
```

Graphe d'héritage de Tenseur3HB:



Graphe de collaboration de Tenseur3HB:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur3HB** (const double val)
- **Tenseur3HB** (const double val1, const double val2, const double val3, const double val4, const double val5, const double val6, const double val7, const double val8, const double val9)
- **Tenseur3HB** (const [TenseurHB](#) &)
- **Tenseur3HB** (const [Tenseur3HB](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurHB](#) & **operator+** (const [TenseurHB](#) &) const  
*operations +*
- void **operator+=** (const [TenseurHB](#) &)  
*operations +=*
- [TenseurHB](#) & **operator-** () const  
*operations opposé*
- [TenseurHB](#) & **operator-** (const [TenseurHB](#) &) const  
*operations -*
- void **operator-=** (const [TenseurHB](#) &)  
*operations -=*
- [TenseurHB](#) & **operator=** (const [TenseurHB](#) &)  
*operations =*
- [TenseurHB](#) & **operator=** (const [Tenseur3HB](#) &B)
- [TenseurHB](#) & **operator\*** (const double &) const  
*operations \**
- void **operator\*=** (const double &)  
*operations \*=*
- [TenseurHB](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_2D\_a\_3D** (const [Tenseur2HB](#) &B, bool plusZero)
- void **Affectation\_trans\_dimension** (const [TenseurHB](#) &B, bool plusZero)

- Affectation\_trans\_dimension.*
- `CoordonneeH operator*` (const `CoordonneeH` &) const  
*produit contracte avec un vecteur*
  - `TenseurHH & operator*` (const `TenseurHH` &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
  - `TenseurHB & operator*` (const `TenseurHB` &) const
  - `double operator&&` (const `TenseurHB` &) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
  - `int operator==` (const `TenseurHB` &) const  
*test*
  - `int operator!=` (const `TenseurHB` &) const
  - `TenseurHB & Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
  - `double Trace` () const  
*trace du tenseur ou premier invariant*
  - `double II` () const  
*second invariant = trace (A\*A)*
  - `double III` () const  
*troisieme invariant = trace ((A\*A)\*A)*
  - `double Det` () const  
*determinant de la matrice des coordonnees*
  - `virtual Coordonnee ValPropre` (int &cas) const  
*calcul des valeurs propres*
  - `virtual Coordonnee ValPropre` (int &cas, `Mat_pleine` &mat) const  
*calcul des valeurs propres*
  - `virtual void VecteursPropres` (const `Coordonnee` &Val\_P, int &cas, `Tableau`< `Coordonnee` > &V\_P) const  
*calcul des vecteurs propres, les valeurs propres étant déjà connues*
  - `TenseurBH & Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
  - `void PermuteHautBas` ()  
*permute Bas Haut, mais reste dans le même tenseur*
  - `double MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
  - `double & Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
  - `double operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
  - `istream & Lecture` (istream &entree)  
*lecture et écriture de données*
  - `ostream & Ecriture` (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- `static TenseurHB & Prod_tensoriel` (const `CoordonneeH` &aH, const `CoordonneeB` &bB)
- `static int OdVect` (const int i, const int j)
- `static int idx_i` (const int k)
- `static int idx_j` (const int k)

## Attributs protégés

- `listdouble9lter ipointe`

## Attributs protégés statiques

- `static const ChangementIndex cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class **Tenseur2HB**
- istream & **operator>>** (istream &, [Tenseur3HB](#) &)
- ostream & **operator<<** (ostream &, const [Tenseur3HB](#) &)

## Membres hérités additionnels

### 6.823.1 Description détaillée

Definition des tenseur derivees de dimension 3. cas des composantes mixtes HB.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.823.2 Documentation des fonctions membres

#### 6.823.2.1 Affectation\_trans\_dimension()

```
void Tenseur3HB::Affectation_trans_dimension (
    const TenseurHB & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
 plusZero = true: les données manquantes sont mises à 0  
 si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
 des données possibles  
 Implémente [TenseurHB](#).

#### 6.823.2.2 Coor()

```
double & Tenseur3HB::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.  
 Implémente [TenseurHB](#).

#### 6.823.2.3 Det()

```
double Tenseur3HB::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees  
 Implémente [TenseurHB](#).

#### 6.823.2.4 Ecriture()

```
ostream & Tenseur3HB::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurHB](#).

#### 6.823.2.5 II()

```
double Tenseur3HB::II ( ) const [virtual]
```

second invariant = trace (A\*A)

Implémente [TenseurHB](#).

#### 6.823.2.6 III()

```
double Tenseur3HB::III ( ) const [virtual]
```

troisieme invariant = trace ((A\*A)\*A)

Implémente [TenseurHB](#).

#### 6.823.2.7 Inita()

```
void Tenseur3HB::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurHB](#).

#### 6.823.2.8 Inverse()

```
TenseurHB & Tenseur3HB::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracte

Implémente [TenseurHB](#).

#### 6.823.2.9 Lecture()

```
istream & Tenseur3HB::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurHB](#).

#### 6.823.2.10 MaxiComposante()

```
double Tenseur3HB::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurHB](#).

#### 6.823.2.11 operator"!=(())

```
int Tenseur3HB::operator!=(
    const TenseurHB & ) const [virtual]
```

Implémente [TenseurHB](#).

#### 6.823.2.12 operator&&()

```
double Tenseur3HB::operator&& (
    const TenseurHB & ) const [virtual]
```

produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémente [TenseurHB](#).

#### 6.823.2.13 operator>()

```
double Tenseur3HB::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.

Implémente [TenseurHB](#).

#### 6.823.2.14 operator\*() [1/4]

```
CoordonneeH Tenseur3HB::operator* (
    const CoordonneeH & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurHB](#).

#### 6.823.2.15 operator\*() [2/4]

```
TenseurHB & Tenseur3HB::operator* (
    const double & ) const [virtual]
```

operations \*

Implémente [TenseurHB](#).

#### 6.823.2.16 operator\*() [3/4]

```
TenseurHB & Tenseur3HB::operator* (
    const TenseurHB & ) const [virtual]
```

Implémente [TenseurHB](#).

#### 6.823.2.17 operator\*() [4/4]

```
TenseurHH & Tenseur3HB::operator* (
    const TenseurHH & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté

Implémente [TenseurHB](#).

#### 6.823.2.18 operator\*=( )

```
void Tenseur3HB::operator*= (
    const double & ) [virtual]
```

operations \*=

Implémente [TenseurHB](#).

#### 6.823.2.19 operator+( )

```
TenseurHB & Tenseur3HB::operator+ (
    const TenseurHB & ) const [virtual]
```

operations +

Implémente [TenseurHB](#).

**6.823.2.20 operator+=()**

```
void Tenseur3HB::operator+= (
    const TenseurHB & ) [virtual]
```

operations +=

Implémente [TenseurHB](#).

**6.823.2.21 operator-() [1/2]**

```
TenseurHB & Tenseur3HB::operator- ( ) const [virtual]
```

operations opposé

Implémente [TenseurHB](#).

**6.823.2.22 operator-() [2/2]**

```
TenseurHB & Tenseur3HB::operator- (
    const TenseurHB & ) const [virtual]
```

operations -

Implémente [TenseurHB](#).

**6.823.2.23 operator-=()**

```
void Tenseur3HB::operator-= (
    const TenseurHB & ) [virtual]
```

operations -=

Implémente [TenseurHB](#).

**6.823.2.24 operator/()**

```
TenseurHB & Tenseur3HB::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurHB](#).

**6.823.2.25 operator/=()**

```
void Tenseur3HB::operator/= (
    const double & ) [virtual]
```

operations /=

Implémente [TenseurHB](#).

**6.823.2.26 operator=()**

```
TenseurHB & Tenseur3HB::operator= (
    const TenseurHB & ) [virtual]
```

operations =

Implémente [TenseurHB](#).

**6.823.2.27 operator==(())**

```
int Tenseur3HB::operator==(
    const TenseurHB & ) const [virtual]
```

test

Implémente [TenseurHB](#).

### 6.823.2.28 PermuteHautBas()

```
void Tenseur3HB::PermuteHautBas ( ) [virtual]
```

permuté Bas Haut, mais reste dans le même tenseur  
Implémente [TenseurHB](#).

### 6.823.2.29 Trace()

```
double Tenseur3HB::Trace ( ) const [virtual]
```

trace du tenseur ou premier invariant  
Implémente [TenseurHB](#).

### 6.823.2.30 Transpose()

```
TenseurBH & Tenseur3HB::Transpose ( ) const [virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprimé par Libere.  
Implémente [TenseurHB](#).

### 6.823.2.31 ValPropre() [1/2]

```
virtual Coordonnee Tenseur3HB::ValPropre (
    int & cas ) const [virtual]
```

calcul des valeurs propres

valeurs propre dans le vecteur de retour, classée par ordres "décroissants"  
cas indique le cas de valeur propre:  
quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire  
dans ce cas les valeurs propres de retour sont nulles par défaut  
dim = 1, cas=1 pour une valeur propre;  
dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques  
dim = 3 , cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)  
, cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),  
, cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du retour)  
, cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du retour)  
Implémente [TenseurHB](#).

### 6.823.2.32 ValPropre() [2/2]

```
virtual Coordonnee Tenseur3HB::ValPropre (
    int & cas,
    Mat_pleine & mat ) const [virtual]
```

calcul des valeurs propres

idem met en retour la matrice mat contient par colonne les vecteurs propre  
elle doit avoir la dimension du tenseur  
les vecteurs propre sont exprimés dans le repère naturel (pour les tenseurs dim 3  
pour dim=2: le premier vecteur propre est exprimé dans le repère naturel  
le second vecteur propre est exprimé dans le repère dual  
pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
Implémente [TenseurHB](#).



### 6.823.2.33 VecteursPropres()

```
virtual void Tenseur3HB::VecteursPropres (
    const Coordonnee & Val_P,
    int & cas,
    Tableau< Coordonnee > & V_P ) const [virtual]
```

calcul des vecteurs propres, les valeurs propres étant déjà connues

ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres étant déjà connues

en retour VP les vecteurs propre : doivent avoir la dimension du tenseur  
les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3 pour dim=2:le premier vecteur propre est exprime dans le repere naturel le second vecteur propre est exprimé dans le repère dual pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
en sortie cas = -1 s'il y a eu un problème, dans ce cas, V\_P est quelconque sinon si tout est ok, cas est identique en sortie avec l'entrée  
Implémente [TenseurHB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

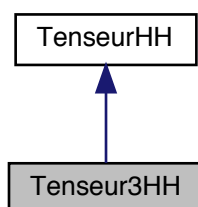
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.824 Référence de la classe Tenseur3HH

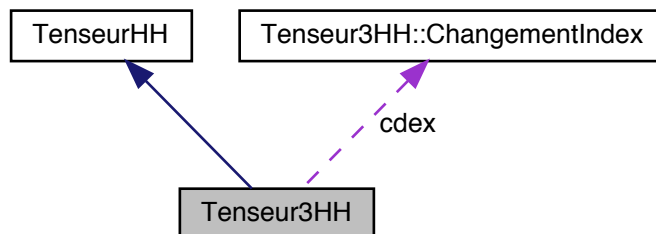
Definition des tenseur derivees de dimension 3. cas des composantes deux fois contravariantes.

```
#include <Tenseur3.h>
```

Graphe d'héritage de Tenseur3HH:



Graphe de collaboration de Tenseur3HH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur3HH** (const double val)
- **Tenseur3HH** (const double val1, const double val2, const double val3, const double val4, const double val5, const double val6)
- **Tenseur3HH** (const [TenseurHH](#) &)
- **Tenseur3HH** (const [Tenseur3HH](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurHH](#) & **operator+** (const [TenseurHH](#) &) const  
*operations +*
- void **operator+=** (const [TenseurHH](#) &)  
*operations +=*
- [TenseurHH](#) & **operator-** () const  
*operations -*
- [TenseurHH](#) & **operator-** (const [TenseurHH](#) &) const  
*operations - tens*
- void **operator-=** (const [TenseurHH](#) &)  
*operations -=*
- [TenseurHH](#) & **operator=** (const [TenseurHH](#) &)  
*operations =*
- [TenseurHH](#) & **operator=** (const [Tenseur3HH](#) &B)
- [TenseurHH](#) & **operator=** (const [Tenseur\\_ns3HH](#) &B)
- [TenseurHH](#) & **operator\*** (const double &) const  
*operations \* double*
- void **operator\*=** (const double &)  
*operations \*= double*
- [TenseurHH](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_2D\_a\_3D** (const [Tenseur2HH](#) &B, bool plusZero)
- void **Affectation\_trans\_dimension** (const [TenseurHH](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- [CoordonneeH](#) **operator\*** (const [CoordonneeB](#) &) const  
*produit contracte avec un vecteur*
- [TenseurHH](#) & **operator\*** (const [TenseurBH](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté avec BH*

- `TenseurHB & operator*` (const `TenseurBB` &) const  
*idem avec BB*
- double `operator&&` (const `TenseurBB` &) const  
*produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int `operator==` (const `TenseurHH` &) const  
*test*
- int `operator!=` (const `TenseurHH` &) const  
*test*
- double `Det` () const  
*determinant de la matrice des coordonnees*
- `TenseurBB & Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
- `TenseurHH & Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual `TenseurBB & Baisse2Indices` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurBH & BaissePremierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurHB & BaisseDernierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- double `MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- double & `Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double `operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seule.*
- istream & `Lecture` (istream &entree)  
*lecture et écriture de données*
- ostream & `Ecriture` (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- static `TenseurHH & Prod_tensoriel` (const `CoordonneeH` &aH, const `CoordonneeH` &bH)
- static int `OdVect` (const int i, const int j)
- static int `idx_i` (const int k)
- static int `idx_j` (const int k)

## Attributs protégés

- listdouble6lter `ipointe`

## Attributs protégés statiques

- static const `ChangementIndex` `cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class `Tenseur2HH`
- class `Tenseur3BB`
- istream & `operator>>` (istream &, `Tenseur3HH` &)
- ostream & `operator<<` (ostream &, const `Tenseur3HH` &)

## Membres hérités additionnels

### 6.824.1 Description détaillée

Definition des tenseur derivees de dimension 3. cas des composantes deux fois contravariantes.

**Auteur**

G rard Rio

**Version**

1.0

**Date**

23/01/97

**6.824.2 Documentation des fonctions membres****6.824.2.1 Affectation\_trans\_dimension()**

```
void Tenseur3HH::Affectation_trans_dimension (
    const TenseurHH & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les donn es manquantes sont inchang es,  
plusZero = true: les donn es manquantes sont mises   0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des donn es possibles  
Impl mente [TenseurHH](#).

**6.824.2.2 Baisse2Indices()**

```
virtual TenseurBB & Tenseur3HH::Baisse2Indices ( ) const [virtual]
```

— manipulation d'indice — -&gt; cr ation de nouveaux tenseurs

Impl mente [TenseurHH](#).**6.824.2.3 BaisseDernierIndice()**

```
virtual TenseurHB & Tenseur3HH::BaisseDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -&gt; cr ation de nouveaux tenseurs

Impl mente [TenseurHH](#).**6.824.2.4 BaissePremierIndice()**

```
virtual TenseurBH & Tenseur3HH::BaissePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -&gt; cr ation de nouveaux tenseurs

Impl mente [TenseurHH](#).**6.824.2.5 Coor()**

```
double & Tenseur3HH::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Impl mente [TenseurHH](#).

### 6.824.2.6 Det()

```
double Tenseur3HH::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees  
Implémente [TenseurHH](#).

### 6.824.2.7 Ecriture()

```
ostream & Tenseur3HH::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données  
Implémente [TenseurHH](#).

### 6.824.2.8 Inita()

```
void Tenseur3HH::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val  
Implémente [TenseurHH](#).

### 6.824.2.9 Inverse()

```
TenseurBB & Tenseur3HH::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracte  
Implémente [TenseurHH](#).

### 6.824.2.10 Lecture()

```
istream & Tenseur3HH::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données  
Implémente [TenseurHH](#).

### 6.824.2.11 MaxiComposante()

```
double Tenseur3HH::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur  
Implémente [TenseurHH](#).

### 6.824.2.12 operator"!="()

```
int Tenseur3HH::operator!= (
    const TenseurHH & ) const [virtual]
```

test  
Implémente [TenseurHH](#).

### 6.824.2.13 operator&&()

```
double Tenseur3HH::operator&& (
    const TenseurBB & ) const [virtual]
```

produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe  
Implémente [TenseurHH](#).

**6.824.2.14 operator>()**

```
double Tenseur3HH::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante i,j du tenseur acces en lecture seule.

Implémente [TenseurHH](#).

**6.824.2.15 operator\*() [1/4]**

```
CoordonneeH Tenseur3HH::operator* (
    const CoordonneeB & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurHH](#).

**6.824.2.16 operator\*() [2/4]**

```
TenseurHH & Tenseur3HH::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurHH](#).

**6.824.2.17 operator\*() [3/4]**

```
TenseurHB & Tenseur3HH::operator* (
    const TenseurBB & ) const [virtual]
```

idem avec BB

Implémente [TenseurHH](#).

**6.824.2.18 operator\*() [4/4]**

```
TenseurHH & Tenseur3HH::operator* (
    const TenseurBH & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté avec BH

Implémente [TenseurHH](#).

**6.824.2.19 operator\*=( )**

```
void Tenseur3HH::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurHH](#).

**6.824.2.20 operator+()**

```
TenseurHH & Tenseur3HH::operator+ (
    const TenseurHH & ) const [virtual]
```

operations +

Implémente [TenseurHH](#).

**6.824.2.21 operator+=()**

```
void Tenseur3HH::operator+= (
    const TenseurHH & ) [virtual]
```

operations +=

Implémente [TenseurHH](#).

**6.824.2.22 operator-() [1/2]**

```
TenseurHH & Tenseur3HH::operator- ( ) const [virtual]
```

operations -

Implémente [TenseurHH](#).

**6.824.2.23 operator-() [2/2]**

```
TenseurHH & Tenseur3HH::operator- (
    const TenseurHH & ) const [virtual]
```

operations - tens

Implémente [TenseurHH](#).

**6.824.2.24 operator-=()**

```
void Tenseur3HH::operator-= (
    const TenseurHH & ) [virtual]
```

operations -=

Implémente [TenseurHH](#).

**6.824.2.25 operator/()**

```
TenseurHH & Tenseur3HH::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurHH](#).

**6.824.2.26 operator/=()**

```
void Tenseur3HH::operator/= (
    const double & ) [virtual]
```

operations +=

Implémente [TenseurHH](#).

**6.824.2.27 operator=()**

```
TenseurHH & Tenseur3HH::operator= (
    const TenseurHH & ) [virtual]
```

operations =

Implémente [TenseurHH](#).

**6.824.2.28 operator==(())**

```
int Tenseur3HH::operator==(
    const TenseurHH & ) const [virtual]
```

test

Implémente [TenseurHH](#).

### 6.824.2.29 Transpose()

`TenseurHH & Tenseur3HH::Transpose ( ) const [virtual]`

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

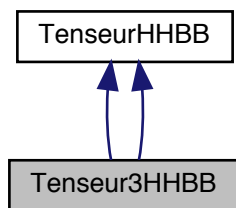
Implémente [TenseurHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

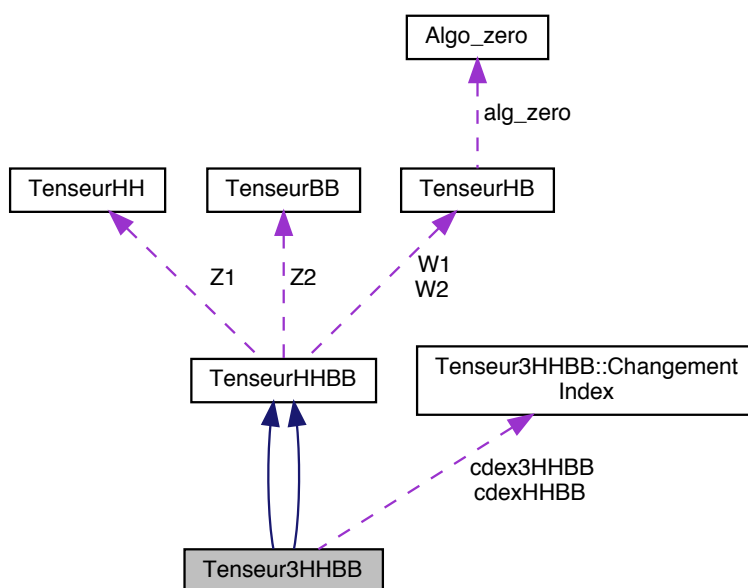
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.825 Référence de la classe Tenseur3HHBB

Graphe d'héritage de Tenseur3HHBB:



Graphe de collaboration de Tenseur3HHBB:





## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur3HHBB** (const double val)
- **Tenseur3HHBB** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)
- **Tenseur3HHBB** (const [TenseurHHBB](#) &)
- **Tenseur3HHBB** (const [Tenseur3HHBB](#) &)
- void **Inita** (double val)
- [TenseurHHBB](#) & **operator+** (const [TenseurHHBB](#) &) const
- void **operator+=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator-** () const
- [TenseurHHBB](#) & **operator-** (const [TenseurHHBB](#) &) const
- void **operator-=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurHHBB](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurHH](#) &) const
- [TenseurHH](#) & **ContractionVerticale** (const [TenseurHH](#) &) const
- [TenseurHHHH](#) & **operator&&** (const [TenseurHHHH](#) &) const
- [TenseurHHBB](#) & **operator&&** (const [TenseurHHBB](#) &) const
- [TenseurBBHH](#) & **Transpose1et2avec3et4** () const
- virtual void **Affectation\_trans\_dimension** (const [TenseurHHBB](#) &B, bool plusZero)
- int **operator==** (const [TenseurHHBB](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort) const
- void **Affiche\_bidim** (ostream &sort) const
- [TenseurBB](#) & **Prod\_gauche** (const [TenseurBB](#) &F) const
- **Tenseur3HHBB** (const double val)
- **Tenseur3HHBB** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)
- **Tenseur3HHBB** (const [TenseurHHBB](#) &)
- **Tenseur3HHBB** (const [Tenseur3HHBB](#) &)
- [TenseurHHBB](#) & **operator+** (const [TenseurHHBB](#) &) const
- void **operator+=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator-** () const
- [TenseurHHBB](#) & **operator-** (const [TenseurHHBB](#) &) const
- void **operator-=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurHHBB](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurHH](#) &) const
- [TenseurBBHH](#) & **Transpose1et2avec3et4** () const
- int **operator==** (const [TenseurHHBB](#) &) const
- void **change** (int i, int j, int k, int l, double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort)

## Fonctions membres publiques statiques

- static [TenseurHHBB](#) & **Prod\_tensoriel** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)
- static [TenseurHHBB](#) & **Prod\_tensoriel\_barre** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)
- static [TenseurHHBB](#) & **Prod\_tensoriel** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)

## Attributs publics statiques

- static const [ChangementIndex](#) `cdex3HHBB`  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3*

## Attributs protégés

- `listdouble36lter ipointe`

## Attributs protégés statiques

- static [ChangementIndex](#) `cdexHHBB`

## Amis

- `istream & operator>>` (`istream &`, [Tenseur3HHBB](#) &)
- `ostream & operator<<` (`ostream &`, const [Tenseur3HHBB](#) &)
- `istream & operator>>` (`istream &`, [Tenseur3HHBB](#) &)
- `ostream & operator<<` (`ostream &`, const [Tenseur3HHBB](#) &)

## Membres hérités additionnels

### 6.825.1 Documentation des fonctions membres

#### 6.825.1.1 Affectation\_trans\_dimension()

```
virtual void Tenseur3HHBB::Affectation_trans_dimension (
    const TenseurHHBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.2 Change()

```
void Tenseur3HHBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.3 ChangePlus()

```
void Tenseur3HHBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.4 Ecriture()

```
ostream & Tenseur3HHBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.5 Inita()

```
void Tenseur3HHBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.6 Lecture() [1/2]

```
istream & Tenseur3HHBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.7 Lecture() [2/2]

```
istream & Tenseur3HHBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.8 MaxiComposante() [1/2]

```
double Tenseur3HHBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.9 MaxiComposante() [2/2]

```
double Tenseur3HHBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.10 operator&&() [1/2]

```
TenseurHH & Tenseur3HHBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.11 operator&&() [2/2]

```
TenseurHH & Tenseur3HHBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.12 operator>() [1/2]

```
double Tenseur3HHBB::operator() (
    int i,
    int j,
```

```
        int k,  
        int l ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.13 operator>() [2/2]

```
double Tenseur3HHBB::operator() (  
    int i,  
    int j,  
    int k,  
    int l ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.14 operator\*() [1/2]

```
TenseurHHBB & Tenseur3HHBB::operator* (  
    const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.15 operator\*() [2/2]

```
TenseurHHBB & Tenseur3HHBB::operator* (  
    const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.16 operator\*=() [1/2]

```
void Tenseur3HHBB::operator*=(  
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.17 operator\*=() [2/2]

```
void Tenseur3HHBB::operator*=(  
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.18 operator+() [1/2]

```
TenseurHHBB & Tenseur3HHBB::operator+ (  
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.825.1.19 operator+() [2/2]

```
TenseurHHBB & Tenseur3HHBB::operator+ (  
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.20 operator+=( ) [1/2]**

```
void Tenseur3HHBB::operator+=(  
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.21 operator+=( ) [2/2]**

```
void Tenseur3HHBB::operator+=(  
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.22 operator-( ) [1/4]**

```
TenseurHHBB & Tenseur3HHBB::operator-( ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.23 operator-( ) [2/4]**

```
TenseurHHBB & Tenseur3HHBB::operator-( ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.24 operator-( ) [3/4]**

```
TenseurHHBB & Tenseur3HHBB::operator-(  
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.25 operator-( ) [4/4]**

```
TenseurHHBB & Tenseur3HHBB::operator-(  
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.26 operator-=( ) [1/2]**

```
void Tenseur3HHBB::operator-=(  
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.27 operator-=( ) [2/2]**

```
void Tenseur3HHBB::operator-=(  
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.28 operator/( ) [1/2]**

```
TenseurHHBB & Tenseur3HHBB::operator/(  
    const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.29 operator/()** [2/2]

```
TenseurHHBB & Tenseur3HHBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.30 operator/=( )** [1/2]

```
void Tenseur3HHBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.31 operator/=( )** [2/2]

```
void Tenseur3HHBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.32 operator=( )** [1/2]

```
TenseurHHBB & Tenseur3HHBB::operator= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.33 operator=( )** [2/2]

```
TenseurHHBB & Tenseur3HHBB::operator= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.34 operator==( )** [1/2]

```
int Tenseur3HHBB::operator==(
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.35 operator==( )** [2/2]

```
int Tenseur3HHBB::operator==(
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.36 Prod\_gauche()**

```
TenseurBB & Tenseur3HHBB::Prod_gauche (
    const TenseurBB & F ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.825.1.37 Transpose1et2avec3et4()** [1/2]

```
TenseurBBHH & Tenseur3HHBB::Transpose1et2avec3et4 ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

## 6.825.1.38 Transpose1et2avec3et4() [2/2]

`TenseurBBHH` & `Tenseur3HHBB::Transpose1et2avec3et4 ( ) const` [virtual]

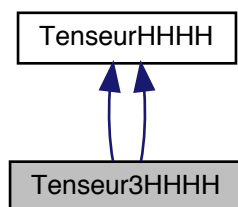
Implémente `TenseurHHBB`.

La documentation de cette classe a été générée à partir du fichier suivant :

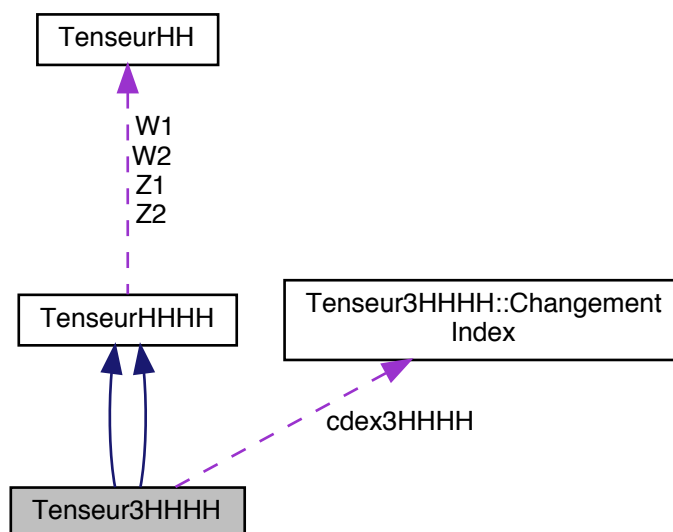
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3.h.old.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.826 Référence de la classe Tenseur3HHHH

Graphe d'héritage de Tenseur3HHHH:



Graphe de collaboration de Tenseur3HHHH:



## Classes

— class [ChangementIndex](#)

## Fonctions membres publiques

— **Tenseur3HHHH** (const double val)  
 — **Tenseur3HHHH** (bool normal, const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)  
 — **Tenseur3HHHH** (const [TenseurHHHH](#) &)  
 — **Tenseur3HHHH** (const [Tenseur3HHHH](#) &)  
 — void **Inita** (double val)  
 — [TenseurHHHH](#) & **operator+** (const [TenseurHHHH](#) &) const  
 — void **operator+=** (const [TenseurHHHH](#) &)  
 — [TenseurHHHH](#) & **operator-** () const  
 — [TenseurHHHH](#) & **operator-** (const [TenseurHHHH](#) &) const  
 — void **operator-=** (const [TenseurHHHH](#) &)  
 — [TenseurHHHH](#) & **operator=** (const [TenseurHHHH](#) &)  
 — [TenseurHHHH](#) & **operator=** (const [Tenseur3HHHH](#) &B)  
 — [TenseurHHHH](#) & **operator\*** (const double &) const  
 — void **operator\*=** (const double &)  
 — [TenseurHHHH](#) & **operator/** (const double &) const  
 — void **operator/=** (const double &)  
 — [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &) const  
 — [TenseurHH](#) & **ContractionVerticale** (const [TenseurBB](#) &) const  
 — [TenseurHHHH](#) & **operator&&** (const [TenseurBBHH](#) &) const  
 — [TenseurHHBB](#) & **operator&&** (const [TenseurBBBB](#) &) const  
 — [TenseurHHHH](#) & **Transpose1et2avec3et4** () const  
 — virtual void **Affectation\_trans\_dimension** (const [TenseurHHHH](#) &B, bool plusZero)  
 — void **TransfertDunTenseurGeneral** (const [TenseurHHHH](#) &aHHHH)  
 — [TenseurBBHH](#) & **Baisse2premiersIndices** ()  
 — [TenseurHHBB](#) & **Baisse2derniersIndices** ()  
 — [TenseurHHHH](#) & **Baselocale** ([TenseurHHHH](#) &A, const [BaseH](#) &gi) const  
 — [TenseurHHHH](#) & **ChangeBase** ([TenseurHHHH](#) &A, const [BaseB](#) &gi) const  
 — int **operator==** (const [TenseurHHHH](#) &) const  
 — void **Change** (int i, int j, int k, int l, const double &val)  
 — void **ChangePlus** (int i, int j, int k, int l, const double &val)  
 — double **operator()** (int i, int j, int k, int l) const  
 — double **MaxiComposante** () const  
 — istream & **Lecture** (istream &entree)  
 — ostream & **Ecriture** (ostream &sort) const  
 — void **Affiche\_bidim** (ostream &sort) const  
 — [TenseurHH](#) & **Prod\_gauche** (const [TenseurBB](#) &F) const  
 — **Tenseur3HHHH** (const double val)  
 — **Tenseur3HHHH** (bool normal, const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)  
 — **Tenseur3HHHH** (const [TenseurHHHH](#) &)  
 — **Tenseur3HHHH** (const [Tenseur3HHHH](#) &)  
 — [TenseurHHHH](#) & **operator+** (const [TenseurHHHH](#) &) const  
 — void **operator+=** (const [TenseurHHHH](#) &)  
 — [TenseurHHHH](#) & **operator-** () const  
 — [TenseurHHHH](#) & **operator-** (const [TenseurHHHH](#) &) const  
 — void **operator-=** (const [TenseurHHHH](#) &)  
 — [TenseurHHHH](#) & **operator=** (const [TenseurHHHH](#) &)  
 — [TenseurHHHH](#) & **operator\*** (const double &) const  
 — void **operator\*=** (const double &)  
 — [TenseurHHHH](#) & **operator/** (const double &) const  
 — void **operator/=** (const double &)  
 — [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &) const  
 — [TenseurHHHH](#) & **Transpose1et2avec3et4** () const  
 — int **operator==** (const [TenseurHHHH](#) &) const  
 — void **change** (int i, int j, int k, int l, double &val)  
 — double **operator()** (int i, int j, int k, int l) const  
 — double **MaxiComposante** () const  
 — istream & **Lecture** (istream &entree)  
 — ostream & **Ecriture** (ostream &sort)



## Fonctions membres publiques statiques

- static [TenseurHHHH](#) & **Prod\_tensoriel** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & **Prod\_tensoriel\_croise** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & **Prod\_tensoriel\_croise\_croise** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & **Prod\_tensoriel\_barre** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & **Var\_tenseur\_dans\_nouvelle\_base** (const [Mat\\_pleine](#) &gamma, [Tenseur3HHHH](#) &var\_tensHHHH, const [Tableau2](#)< [Tenseur3HH](#) > &var\_gamma, const [Tenseur3HH](#) &tensHH)  
— *pour mémoire* —
- static [TenseurHHHH](#) & **Prod\_tensoriel** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & **Prod\_tensoriel\_barre** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)

## Attributs publics statiques

- static const [ChangementIndex](#) **cdex3HHHH**  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3*

## Attributs protégés

- listdouble36lter **ipointe**

## Attributs protégés statiques

- static [ChangementIndex](#) **cdex3HHHH**

## Amis

- istream & **operator**>> (istream &, [Tenseur3HHHH](#) &)
- ostream & **operator**<< (ostream &, const [Tenseur3HHHH](#) &)
- istream & **operator**>> (istream &, [Tenseur3HHHH](#) &)
- ostream & **operator**<< (ostream &, const [Tenseur3HHHH](#) &)

## Membres hérités additionnels

### 6.826.1 Documentation des fonctions membres

#### 6.826.1.1 Affectation\_trans\_dimension()

```
virtual void Tenseur3HHHH::Affectation_trans_dimension (
    const TenseurHHHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.826.1.2 Change()

```
void Tenseur3HHHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.3 ChangePlus()

```
void Tenseur3HHHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.4 Ecriture()

```
ostream & Tenseur3HHHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.5 Inita()

```
void Tenseur3HHHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.6 Lecture() [1/2]

```
istream & Tenseur3HHHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.7 Lecture() [2/2]

```
istream & Tenseur3HHHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.8 MaxiComposante() [1/2]

```
double Tenseur3HHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.9 MaxiComposante() [2/2]

```
double Tenseur3HHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.10 operator&&() [1/2]

```
TenseurHH & Tenseur3HHHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.11 operator&&() [2/2]**

```
TenseurHH & Tenseur3HHHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.12 operator>() [1/2]**

```
double Tenseur3HHHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.13 operator>() [2/2]**

```
double Tenseur3HHHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.14 operator\*() [1/2]**

```
TenseurHHHH & Tenseur3HHHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.15 operator\*() [2/2]**

```
TenseurHHHH & Tenseur3HHHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.16 operator\*=() [1/2]**

```
void Tenseur3HHHH::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.17 operator\*=() [2/2]**

```
void Tenseur3HHHH::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.18 operator+() [1/2]**

```
TenseurHHHH & Tenseur3HHHH::operator+ (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.19 operator+()** [2/2]

```
TenseurHHHH & Tenseur3HHHH::operator+ (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.20 operator+=()** [1/2]

```
void Tenseur3HHHH::operator+= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.21 operator+=()** [2/2]

```
void Tenseur3HHHH::operator+= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.22 operator-()** [1/4]

```
TenseurHHHH & Tenseur3HHHH::operator- ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.23 operator-()** [2/4]

```
TenseurHHHH & Tenseur3HHHH::operator- ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.24 operator-()** [3/4]

```
TenseurHHHH & Tenseur3HHHH::operator- (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.25 operator-()** [4/4]

```
TenseurHHHH & Tenseur3HHHH::operator- (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.26 operator-=()** [1/2]

```
void Tenseur3HHHH::operator-= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.27 operator-=()** [2/2]

```
void Tenseur3HHHH::operator-= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.28 operator/()** [1/2]

```
TenseurHHHH & Tenseur3HHHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.29 operator/()** [2/2]

```
TenseurHHHH & Tenseur3HHHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.30 operator/=()** [1/2]

```
void Tenseur3HHHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.31 operator/=()** [2/2]

```
void Tenseur3HHHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.32 operator=()** [1/2]

```
TenseurHHHH & Tenseur3HHHH::operator= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.33 operator=()** [2/2]

```
TenseurHHHH & Tenseur3HHHH::operator= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.34 operator==(())** [1/2]

```
int Tenseur3HHHH::operator==(
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.826.1.35 operator==(())** [2/2]

```
int Tenseur3HHHH::operator==(
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.36 Prod\_gauche()

```
TenseurHH & Tenseur3HHHH::Prod_gauche (
    const TenseurBB & F ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.37 Transpose1et2avec3et4() [1/2]

```
TenseurHHHH & Tenseur3HHHH::Transpose1et2avec3et4 ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.826.1.38 Transpose1et2avec3et4() [2/2]

```
TenseurHHHH & Tenseur3HHHH::Transpose1et2avec3et4 ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

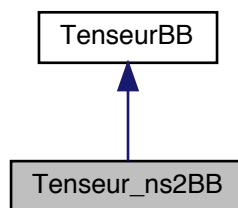
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ-3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3.h.old.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.827 Référence de la classe Tenseur\_ns2BB

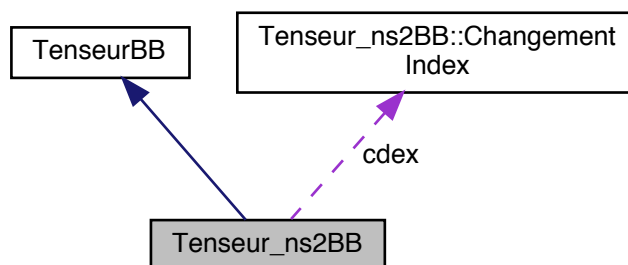
Definition des tenseur derivees de dimension 2. cas des composantes deux fois covariantes non symetriques pour les differencier la dimension = -2.

```
#include <Tenseur2.h>
```

Graphe d'héritage de Tenseur\_ns2BB:



Graphe de collaboration de Tenseur\_ns2BB:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur\_ns2BB** (const double val)
- **Tenseur\_ns2BB** (const double val1, const double val2, const double val3, const double val4)
- **Tenseur\_ns2BB** (const [TenseurBB](#) &)
- **Tenseur\_ns2BB** (const [Tenseur\\_ns2BB](#) &)
- void **lnita** (double val)  
*initialise toutes les composantes à val*
- [TenseurBB](#) & **operator+** (const [TenseurBB](#) &) const  
*operations +*
- void **operator+=** (const [TenseurBB](#) &)  
*operations +=*
- [TenseurBB](#) & **operator-** () const  
*operations -*
- [TenseurBB](#) & **operator-** (const [TenseurBB](#) &) const  
*operations - tens*
- void **operator-=** (const [TenseurBB](#) &)  
*operations -=*
- [TenseurBB](#) & **operator=** (const [TenseurBB](#) &)  
*operations =*
- [TenseurBB](#) & **operator=** (const [Tenseur\\_ns2BB](#) &B)
- [TenseurBB](#) & **operator\*** (const double &) const  
*operations \* double*
- void **operator\*= **(const double &)****  
*operations \*= double*
- [TenseurBB](#) & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_3D\_a\_2D** (const [Tenseur\\_ns3BB](#) &B)
- void **Affectation\_trans\_dimension** (const [TenseurBB](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- [CoordonneeB](#) **operator\*** (const [CoordonneeH](#) &) const  
*produit contracte avec un vecteur*
- [TenseurBB](#) & **operator\*** (const [TenseurHB](#) &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté*
- [TenseurBH](#) & **operator\*** (const [TenseurHH](#) &) const

- idem en BH*
- double `operator&&` (const `TenseurHH` &) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int `operator==` (const `TenseurBB` &) const
- test*
- int `operator!=` (const `TenseurBB` &) const
- test*
- double `Det` () const  
*determinant de la matrice des coordonnees*
- `TenseurBB` & `Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual `TenseurHH` & `Monte2Indices` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurHB` & `MontePremierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurBH` & `MonteDernierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- double `MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- `TenseurHH` & `Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
- double & `Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double `operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
- istream & `Lecture` (istream &entree)  
*lecture et écriture de données*
- ostream & `Ecriture` (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- static int `OdVect` (const int i, const int j)
- static int `idx_i` (const int k)
- static int `idx_j` (const int k)

## Attributs protégés

- listdouble4lter `ipointe`

## Attributs protégés statiques

- static const `ChangementIndex` `cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class `Tenseur_ns3HH`
- class `Tenseur_ns2HH`
- istream & `operator>>` (istream &, `Tenseur_ns2BB` &)
- ostream & `operator<<` (ostream &, const `Tenseur_ns2BB` &)

## Membres hérités additionnels

### 6.827.1 Description détaillée

Definition des tenseur derivees de dimension 2. cas des composantes deux fois covariantes non symetriques pour les differencier la dimension = -2.



**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

**6.827.2 Documentation des fonctions membres****6.827.2.1 Affectation\_trans\_dimension()**

```
void Tenseur_ns2BB::Affectation_trans_dimension (
    const TenseurBB & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurBB](#).

**6.827.2.2 Coor()**

```
double & Tenseur_ns2BB::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.  
Implémente [TenseurBB](#).

**6.827.2.3 Det()**

```
double Tenseur_ns2BB::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees  
Implémente [TenseurBB](#).

**6.827.2.4 Ecriture()**

```
ostream & Tenseur_ns2BB::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données  
Implémente [TenseurBB](#).

**6.827.2.5 Inita()**

```
void Tenseur_ns2BB::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val  
Implémente [TenseurBB](#).

### 6.827.2.6 Inverse()

```
TenseurHH & Tenseur_ns2BB::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémente [TenseurBB](#).

### 6.827.2.7 Lecture()

```
istream & Tenseur_ns2BB::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurBB](#).

### 6.827.2.8 MaxiComposante()

```
double Tenseur_ns2BB::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurBB](#).

### 6.827.2.9 Monte2Indices()

```
virtual TenseurHH & Tenseur_ns2BB::Monte2Indices ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

### 6.827.2.10 MonteDernierIndice()

```
virtual TenseurBH & Tenseur_ns2BB::MonteDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

### 6.827.2.11 MontePremierIndice()

```
virtual TenseurHB & Tenseur_ns2BB::MontePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

### 6.827.2.12 operator"!="()

```
int Tenseur_ns2BB::operator!= (
    const TenseurBB & ) const [virtual]
```

test

Implémente [TenseurBB](#).

### 6.827.2.13 operator&&()

```
double Tenseur_ns2BB::operator&& (
    const TenseurHH & ) const [virtual]
```

produit contracté contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémente [TenseurBB](#).

**6.827.2.14 operator>()**

```
double Tenseur_ns2BB::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante i,j du tenseur acces en lecture seulement.

Implémente [TenseurBB](#).

**6.827.2.15 operator\*() [1/4]**

```
CoordonneeB Tenseur_ns2BB::operator* (
    const CoordonneeH & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurBB](#).

**6.827.2.16 operator\*() [2/4]**

```
TenseurBB & Tenseur_ns2BB::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurBB](#).

**6.827.2.17 operator\*() [3/4]**

```
TenseurBB & Tenseur_ns2BB::operator* (
    const TenseurHB & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté

Implémente [TenseurBB](#).

**6.827.2.18 operator\*() [4/4]**

```
TenseurBH & Tenseur_ns2BB::operator* (
    const TenseurHH & ) const [virtual]
```

idem en BH

Implémente [TenseurBB](#).

**6.827.2.19 operator\*=( )**

```
void Tenseur_ns2BB::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurBB](#).

**6.827.2.20 operator+()**

```
TenseurBB & Tenseur_ns2BB::operator+ (
    const TenseurBB & ) const [virtual]
```

operations +

Implémente [TenseurBB](#).

**6.827.2.21 operator+=()**

```
void Tenseur_ns2BB::operator+= (
    const TenseurBB & ) [virtual]
```

operations +=

Implémente [TenseurBB](#).

**6.827.2.22 operator-() [1/2]**

```
TenseurBB & Tenseur_ns2BB::operator- ( ) const [virtual]
```

operations -

Implémente [TenseurBB](#).

**6.827.2.23 operator-() [2/2]**

```
TenseurBB & Tenseur_ns2BB::operator- (
    const TenseurBB & ) const [virtual]
```

operations - tens

Implémente [TenseurBB](#).

**6.827.2.24 operator-=()**

```
void Tenseur_ns2BB::operator-= (
    const TenseurBB & ) [virtual]
```

operations -=

Implémente [TenseurBB](#).

**6.827.2.25 operator/()**

```
TenseurBB & Tenseur_ns2BB::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurBB](#).

**6.827.2.26 operator/=()**

```
void Tenseur_ns2BB::operator/= (
    const double & ) [virtual]
```

operations /=

Implémente [TenseurBB](#).

**6.827.2.27 operator=()**

```
TenseurBB & Tenseur_ns2BB::operator= (
    const TenseurBB & ) [virtual]
```

operations =

Implémente [TenseurBB](#).

**6.827.2.28 operator==(())**

```
int Tenseur_ns2BB::operator==(
    const TenseurBB & ) const [virtual]
```

test

Implémente [TenseurBB](#).

### 6.827.2.29 Transpose()

`TenseurBB & Tenseur_ns2BB::Transpose ( ) const [virtual]`

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

Implémente [TenseurBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

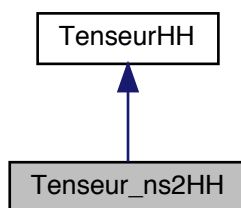
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.828 Référence de la classe Tenseur\_ns2HH

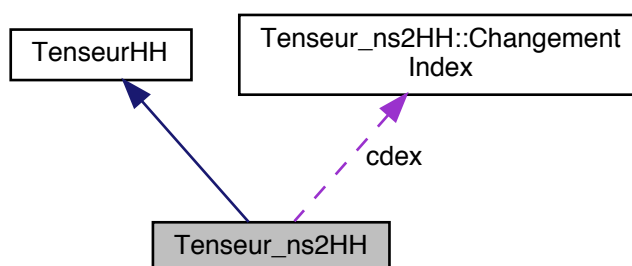
Definition des tenseur derivees de dimension 2. cas des composantes deux fois contravariantes non symetriques pour les differencier la dimension = -2.

```
#include <Tenseur2.h>
```

Graphe d'héritage de Tenseur\_ns2HH:



Graphe de collaboration de Tenseur\_ns2HH:



### Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur\_ns2HH** (const double val)
- **Tenseur\_ns2HH** (const double val1, const double val2, const double val3, const double val4)
- **Tenseur\_ns2HH** (const **TenseurHH** &)
- **Tenseur\_ns2HH** (const **Tenseur\_ns2HH** &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- **TenseurHH** & **operator+** (const **TenseurHH** &) const  
*operations +*
- void **operator+=** (const **TenseurHH** &)  
*operations +=*
- **TenseurHH** & **operator-** () const  
*operations -*
- **TenseurHH** & **operator-** (const **TenseurHH** &) const  
*operations - tens*
- void **operator-=** (const **TenseurHH** &)  
*operations -=*
- **TenseurHH** & **operator=** (const **TenseurHH** &)  
*operations =*
- **TenseurHH** & **operator=** (const **Tenseur\_ns2HH** &B)
- **TenseurHH** & **operator\*** (const double &) const  
*operations \* double*
- void **operator\*=** (const double &)  
*operations \*= double*
- **TenseurHH** & **operator/** (const double &) const  
*operations /*
- void **operator/=** (const double &)  
*operations /=*
- void **Affectation\_3D\_a\_2D** (const **Tenseur\_ns3HH** &B)
- void **Affectation\_trans\_dimension** (const **TenseurHH** &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- **CoordonneeH** **operator\*** (const **CoordonneeB** &) const  
*produit contracte avec un vecteur*
- **TenseurHH** & **operator\*** (const **TenseurBH** &) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté avec BH*
- **TenseurHB** & **operator\*** (const **TenseurBB** &) const  
*idem avec BB*
- double **operator&&** (const **TenseurBB** &) const  
*produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int **operator==** (const **TenseurHH** &) const  
*test*
- int **operator!=** (const **TenseurHH** &) const  
*test*
- double **Det** () const  
*determinant de la matrice des coordonnees*
- **TenseurHH** & **Transpose** () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual **TenseurBB** & **Baisse2Indices** () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual **TenseurBH** & **BaissePremierIndice** () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual **TenseurHB** & **BaisseDernierIndice** () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- double **MaxiComposante** () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- **TenseurBB** & **Inverse** () const  
*calcul du tenseur inverse par rapport au produit contracte*
- double & **Coor** (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*

- `double operator()` (`const int i, const int j`) `const`  
*Retourne la composante  $i,j$  du tenseur acces en lecture seule.*
- `istream & Lecture` (`istream &entree`)  
*lecture et écriture de données*
- `ostream & Ecriture` (`ostream &sort`) `const`  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- `static int OdVect` (`const int i, const int j`)
- `static int idx_i` (`const int k`)
- `static int idx_j` (`const int k`)

## Attributs protégés

- `listdouble4Iter ipointe`

## Attributs protégés statiques

- `static const ChangementIndex cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- `class Tenseur_ns3HH`
- `class Tenseur_ns2BB`
- `istream & operator>>` (`istream &, Tenseur_ns2HH &`)
- `ostream & operator<<` (`ostream &, const Tenseur_ns2HH &`)

## Membres hérités additionnels

### 6.828.1 Description détaillée

Definition des tenseur derivees de dimension 2. cas des composantes deux fois contravariantes non symetriques pour les differencier la dimension = -2.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.828.2 Documentation des fonctions membres

#### 6.828.2.1 Affectation\_trans\_dimension()

```
void Tenseur_ns2HH::Affectation_trans_dimension (
    const TenseurHH & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation des données possibles

Implémente [TenseurHH](#).

#### 6.828.2.2 Baisse2Indices()

```
virtual TenseurBB & Tenseur_ns2HH::Baisse2Indices ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

#### 6.828.2.3 BaisseDernierIndice()

```
virtual TenseurHB & Tenseur_ns2HH::BaisseDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

#### 6.828.2.4 BaissePremierIndice()

```
virtual TenseurBH & Tenseur_ns2HH::BaissePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurHH](#).

#### 6.828.2.5 Coor()

```
double & Tenseur_ns2HH::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémente [TenseurHH](#).

#### 6.828.2.6 Det()

```
double Tenseur_ns2HH::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees

Implémente [TenseurHH](#).

#### 6.828.2.7 Ecriture()

```
ostream & Tenseur_ns2HH::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurHH](#).

#### 6.828.2.8 Inita()

```
void Tenseur_ns2HH::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurHH](#).



**6.828.2.9 Inverse()**

```
TenseurBB & Tenseur_ns2HH::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémente [TenseurHH](#).

**6.828.2.10 Lecture()**

```
istream & Tenseur_ns2HH::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurHH](#).

**6.828.2.11 MaxiComposante()**

```
double Tenseur_ns2HH::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurHH](#).

**6.828.2.12 operator"!="()**

```
int Tenseur_ns2HH::operator!= (
    const TenseurHH & ) const [virtual]
```

test

Implémente [TenseurHH](#).

**6.828.2.13 operator&&()**

```
double Tenseur_ns2HH::operator&& (
    const TenseurBB & ) const [virtual]
```

produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémente [TenseurHH](#).

**6.828.2.14 operator>()()**

```
double Tenseur_ns2HH::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seule.

Implémente [TenseurHH](#).

**6.828.2.15 operator\*() [1/4]**

```
CoordonneeH Tenseur_ns2HH::operator* (
    const CoordonneeB & ) const [virtual]
```

produit contracté avec un vecteur

Implémente [TenseurHH](#).

**6.828.2.16 operator\*() [2/4]**

```
TenseurHH & Tenseur_ns2HH::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurHH](#).

#### 6.828.2.17 operator\*() [3/4]

```
TenseurHB & Tenseur_ns2HH::operator* (
    const TenseurBB & ) const [virtual]
```

idem avec BB

Implémente [TenseurHH](#).

#### 6.828.2.18 operator\*() [4/4]

```
TenseurHH & Tenseur_ns2HH::operator* (
    const TenseurBH & ) const [virtual]
```

produit contracté contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté avec BH

Implémente [TenseurHH](#).

#### 6.828.2.19 operator\*=( )

```
void Tenseur_ns2HH::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurHH](#).

#### 6.828.2.20 operator+( )

```
TenseurHH & Tenseur_ns2HH::operator+ (
    const TenseurHH & ) const [virtual]
```

operations +

Implémente [TenseurHH](#).

#### 6.828.2.21 operator+=( )

```
void Tenseur_ns2HH::operator+= (
    const TenseurHH & ) [virtual]
```

operations +=

Implémente [TenseurHH](#).

#### 6.828.2.22 operator-( ) [1/2]

```
TenseurHH & Tenseur_ns2HH::operator- ( ) const [virtual]
```

operations -

Implémente [TenseurHH](#).

#### 6.828.2.23 operator-( ) [2/2]

```
TenseurHH & Tenseur_ns2HH::operator- (
    const TenseurHH & ) const [virtual]
```

operations - tens

Implémente [TenseurHH](#).

#### 6.828.2.24 operator-=( )

```
void Tenseur_ns2HH::operator-=(  
    const TenseurHH & ) [virtual]
```

operations -=  
Implémente [TenseurHH](#).

#### 6.828.2.25 operator/( )

```
TenseurHH & Tenseur_ns2HH::operator/(  
    const double & ) const [virtual]
```

operations /  
Implémente [TenseurHH](#).

#### 6.828.2.26 operator+=( )

```
void Tenseur_ns2HH::operator+=(  
    const double & ) [virtual]
```

operations +=  
Implémente [TenseurHH](#).

#### 6.828.2.27 operator=( )

```
TenseurHH & Tenseur_ns2HH::operator=(  
    const TenseurHH & ) [virtual]
```

operations =  
Implémente [TenseurHH](#).

#### 6.828.2.28 operator==( )

```
int Tenseur_ns2HH::operator==(  
    const TenseurHH & ) const [virtual]
```

test  
Implémente [TenseurHH](#).

#### 6.828.2.29 Transpose( )

```
TenseurHH & Tenseur_ns2HH::Transpose ( ) const [virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

Implémente [TenseurHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

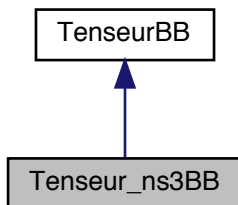
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\\_mai99/Tenseur/Tenseur2.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\\_mai99/Tenseur/EnteteTenseur.h](#)

## 6.829 Référence de la classe Tenseur\_ns3BB

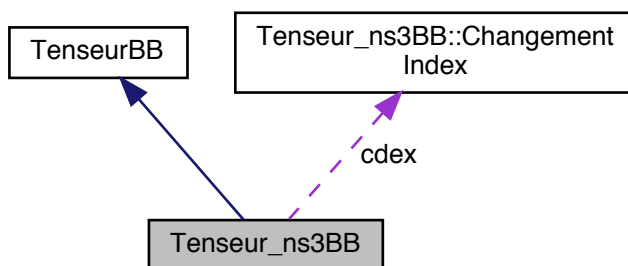
Definition des tenseur derivees de dimension 3. cas des composantes deux fois covariantes non symetriques pour les differencier la dimension = -3.

```
#include <Tenseur3.h>
```

Grappe d'héritage de Tenseur\_ns3BB:



Grappe de collaboration de Tenseur\_ns3BB:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- **Tenseur\_ns3BB** (const double val)
- **Tenseur\_ns3BB** (const double val1, const double val2, const double val3, const double val4, const double val5, const double val6, const double val7, const double val8, const double val9)
- **Tenseur\_ns3BB** (const [TenseurBB](#) &)
- **Tenseur\_ns3BB** (const [Tenseur\\_ns3BB](#) &)
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [TenseurBB](#) & **operator+** (const [TenseurBB](#) &) const  
*operations +*
- void **operator+=** (const [TenseurBB](#) &)  
*operations +=*
- [TenseurBB](#) & **operator-** () const  
*operations -*
- [TenseurBB](#) & **operator-** (const [TenseurBB](#) &) const  
*operations - tens*
- void **operator-=** (const [TenseurBB](#) &)  
*operations -=*

- `TenseurBB & operator=` (const `TenseurBB &`)  
*operations =*
- `TenseurBB & operator=` (const `Tenseur3BB &B`)
- `TenseurBB & operator=` (const `Tenseur_ns3BB &B`)
- `TenseurBB & operator*` (const double &) const  
*operations \* double*
- void `operator*=` (const double &)  
*operations \*= double*
- `TenseurBB & operator/` (const double &) const  
*operations /*
- void `operator/=` (const double &)  
*operations /=*
- void `Affectation_2D_a_3D` (const `Tenseur_ns2BB &B`, bool plusZero)
- void `Affectation_trans_dimension` (const `TenseurBB &B`, bool plusZero)  
*Affectation\_trans\_dimension.*
- `CoordonneeB operator*` (const `CoordonneeH &`) const  
*produit contracte avec un vecteur*
- `TenseurBB & operator*` (const `TenseurHB &`) const  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
- `TenseurBH & operator*` (const `TenseurHH &`) const  
*idem en BH*
- double `operator&&` (const `TenseurHH &`) const  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- int `operator==` (const `TenseurBB &`) const  
*test*
- int `operator!=` (const `TenseurBB &`) const  
*test*
- double `Det` () const  
*determinant de la matrice des coordonnees*
- `TenseurHH & Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
- `TenseurBB & Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual `TenseurHH & Monte2Indices` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurHB & MontePremierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurBH & MonteDernierIndice` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- double `MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- double & `Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- double `operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
- istream & `Lecture` (istream &entree)  
*lecture et écriture de données*
- ostream & `Ecriture` (ostream &sort) const  
*lecture et écriture de données*

## Fonctions membres publiques statiques

- static int `OdVect` (const int i, const int j)
- static int `idx_i` (const int k)
- static int `idx_j` (const int k)

## Attributs protégés

- listdouble9lter `ipointe`

## Attributs protégés statiques

- static const [ChangementIndex](#) `cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- class [Tenseur\\_ns2BB](#)
- class [Tenseur\\_ns3HH](#)
- `istream & operator>>` (`istream &`, [Tenseur\\_ns3BB](#) &)
- `ostream & operator<<` (`ostream &`, const [Tenseur\\_ns3BB](#) &)

## Membres hérités additionnels

### 6.829.1 Description détaillée

Definition des tenseur derivees de dimension 3. cas des composantes deux fois covariantes non symetriques pour les differencier la dimension = -3.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

### 6.829.2 Documentation des fonctions membres

#### 6.829.2.1 Affectation\_trans\_dimension()

```
void Tenseur_ns3BB::Affectation_trans_dimension (
    const TenseurBB & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurBB](#).

#### 6.829.2.2 Coor()

```
double & Tenseur_ns3BB::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.  
Implémente [TenseurBB](#).

#### 6.829.2.3 Det()

```
double Tenseur_ns3BB::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees  
Implémente [TenseurBB](#).

#### 6.829.2.4 Ecriture()

```
ostream & Tenseur_ns3BB::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données

Implémente [TenseurBB](#).

#### 6.829.2.5 Inita()

```
void Tenseur_ns3BB::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val

Implémente [TenseurBB](#).

#### 6.829.2.6 Inverse()

```
TenseurHH & Tenseur_ns3BB::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémente [TenseurBB](#).

#### 6.829.2.7 Lecture()

```
istream & Tenseur_ns3BB::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données

Implémente [TenseurBB](#).

#### 6.829.2.8 MaxiComposante()

```
double Tenseur_ns3BB::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémente [TenseurBB](#).

#### 6.829.2.9 Monte2Indices()

```
virtual TenseurHH & Tenseur_ns3BB::Monte2Indices ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

#### 6.829.2.10 MonteDernierIndice()

```
virtual TenseurBH & Tenseur_ns3BB::MonteDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

#### 6.829.2.11 MontePremierIndice()

```
virtual TenseurHB & Tenseur_ns3BB::MontePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémente [TenseurBB](#).

**6.829.2.12 operator"!=()**

```
int Tenseur_ns3BB::operator!= (
    const TenseurBB & ) const [virtual]
```

test

Implémente [TenseurBB](#).

**6.829.2.13 operator&&()**

```
double Tenseur_ns3BB::operator&& (
    const TenseurHH & ) const [virtual]
```

produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémente [TenseurBB](#).

**6.829.2.14 operator>()()**

```
double Tenseur_ns3BB::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.

Implémente [TenseurBB](#).

**6.829.2.15 operator\*() [1/4]**

```
CoordonneeB Tenseur_ns3BB::operator* (
    const CoordonneeH & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurBB](#).

**6.829.2.16 operator\*() [2/4]**

```
TenseurBB & Tenseur_ns3BB::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurBB](#).

**6.829.2.17 operator\*() [3/4]**

```
TenseurBB & Tenseur_ns3BB::operator* (
    const TenseurHB & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté

Implémente [TenseurBB](#).

**6.829.2.18 operator\*() [4/4]**

```
TenseurBH & Tenseur_ns3BB::operator* (
    const TenseurHH & ) const [virtual]
```

idem en BH

Implémente [TenseurBB](#).



**6.829.2.19 operator\*=( )**

```
void Tenseur_ns3BB::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurBB](#).

**6.829.2.20 operator+( )**

```
TenseurBB & Tenseur_ns3BB::operator+ (
    const TenseurBB & ) const [virtual]
```

operations +

Implémente [TenseurBB](#).

**6.829.2.21 operator+=( )**

```
void Tenseur_ns3BB::operator+= (
    const TenseurBB & ) [virtual]
```

operations +=

Implémente [TenseurBB](#).

**6.829.2.22 operator-( ) [1/2]**

```
TenseurBB & Tenseur_ns3BB::operator- ( ) const [virtual]
```

operations -

Implémente [TenseurBB](#).

**6.829.2.23 operator-( ) [2/2]**

```
TenseurBB & Tenseur_ns3BB::operator- (
    const TenseurBB & ) const [virtual]
```

operations - tens

Implémente [TenseurBB](#).

**6.829.2.24 operator==( )**

```
void Tenseur_ns3BB::operator== (
    const TenseurBB & ) [virtual]
```

operations ==

Implémente [TenseurBB](#).

**6.829.2.25 operator/( )**

```
TenseurBB & Tenseur_ns3BB::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurBB](#).

**6.829.2.26 operator/=( )**

```
void Tenseur_ns3BB::operator/= (
    const double & ) [virtual]
```

operations /=

Implémente [TenseurBB](#).

### 6.829.2.27 operator=()

```
TenseurBB & Tenseur_ns3BB::operator= (
    const TenseurBB & ) [virtual]
```

operations =

Implémente [TenseurBB](#).

### 6.829.2.28 operator==()

```
int Tenseur_ns3BB::operator== (
    const TenseurBB & ) const [virtual]
```

test

Implémente [TenseurBB](#).

### 6.829.2.29 Transpose()

```
TenseurBB & Tenseur_ns3BB::Transpose ( ) const [virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libre.

Implémente [TenseurBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

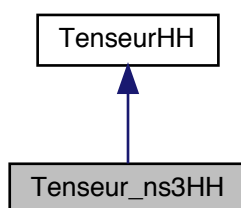
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.830 Référence de la classe Tenseur\_ns3HH

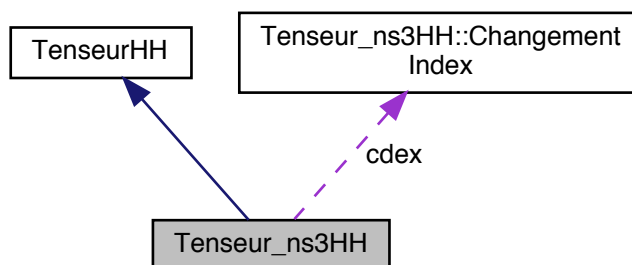
Definition des tenseur derivees de dimension 3. cas des composantes deux fois contravariantes non symetriques pour les differencier la dimension = -3.

```
#include <Tenseur3.h>
```

Grappe d'héritage de Tenseur\_ns3HH:



Graphe de collaboration de Tenseur\_ns3HH:



## Classes

- class [ChangementIndex](#)

## Fonctions membres publiques

- `Tenseur_ns3HH` (double val)
- `Tenseur_ns3HH` (const double val1, const double val2, const double val3, const double val4, const double val5, const double val6, const double val7, const double val8, const double val9)
- `Tenseur_ns3HH` (const [TenseurHH](#) &)
- `Tenseur_ns3HH` (const [Tenseur\\_ns3HH](#) &)
- void `Inita` (double val)  
*initialise toutes les composantes à val*
- [TenseurHH](#) & `operator+` (const [TenseurHH](#) &) const  
*operations +*
- void `operator+=` (const [TenseurHH](#) &)  
*operations +=*
- [TenseurHH](#) & `operator-` () const  
*operations -*
- [TenseurHH](#) & `operator-` (const [TenseurHH](#) &) const  
*operations - tens*
- void `operator-=` (const [TenseurHH](#) &)  
*operations -=*
- [TenseurHH](#) & `operator=` (const [TenseurHH](#) &)  
*operations =*
- [TenseurHH](#) & `operator=` (const [Tenseur3HH](#) &B)
- [TenseurHH](#) & `operator=` (const [Tenseur\\_ns3HH](#) &B)
- [TenseurHH](#) & `operator*` (const double &) const  
*operations \* double*
- void `operator*=` (const double &)  
*operations \*= double*
- [TenseurHH](#) & `operator/` (const double &) const  
*operations /*
- void `operator/=` (const double &)  
*operations +=*
- void `Affectation_2D_a_3D` (const [Tenseur\\_ns2HH](#) &B, bool plusZero)
- void `Affectation_trans_dimension` (const [TenseurHH](#) &B, bool plusZero)  
*Affectation\_trans\_dimension.*
- `CoordonneeH` `operator*` (const `CoordonneeB` &) const  
*produit contracte avec un vecteur*
- [TenseurHH](#) & `operator*` (const [TenseurBH](#) &) const

- *produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté avec BH*
- `TenseurHB & operator*` (const `TenseurBB` &) const  
*idem avec BB*
- `double operator&&` (const `TenseurBB` &) const  
*produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- `int operator==` (const `TenseurHH` &) const  
*test*
- `int operator!=` (const `TenseurHH` &) const  
*test*
- `double Det` () const  
*determinant de la matrice des coordonnees*
- `TenseurBB & Inverse` () const  
*calcul du tenseur inverse par rapport au produit contracte*
- `TenseurHH & Transpose` () const  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libre.*
- `virtual TenseurBB & Baisse2Indices` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- `virtual TenseurBH & BaissePremierIndex` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- `virtual TenseurHB & BaisseDernierIndex` () const  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- `double MaxiComposante` () const  
*calcul du maximum en valeur absolu des composantes du tenseur*
- `double & Coor` (const int i, const int j)  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- `double operator()` (const int i, const int j) const  
*Retourne la composante i,j du tenseur acces en lecture seule.*
- `istream & Lecture` (istream &entree)  
*lecture et ecriture de données*
- `ostream & Ecriture` (ostream &sort) const  
*lecture et ecriture de données*

## Fonctions membres publiques statiques

- `static int OdVect` (const int i, const int j)
- `static int idx_i` (const int k)
- `static int idx_j` (const int k)

## Attributs protégés

- `listdouble9lter ipointe`

## Attributs protégés statiques

- `static const ChangementIndex cdex`  
*initialisation des tableaux d'index pour tenseur du second ordre*

## Amis

- `class Tenseur_ns2HH`
- `class Tenseur_ns3BB`
- `istream & operator>>` (istream &, `Tenseur_ns3HH` &)
- `ostream & operator<<` (ostream &, const `Tenseur_ns3HH` &)

## Membres hérités additionnels

### 6.830.1 Description détaillée

Definition des tenseur derivees de dimension 3. cas des composantes deux fois contravariantes non symetriques pour les differencier la dimension = -3.

**Auteur**

Gérard Rio

**Version**

1.0

**Date**

23/01/97

**6.830.2 Documentation des fonctions membres****6.830.2.1 Affectation\_trans\_dimension()**

```
void Tenseur_ns3HH::Affectation_trans_dimension (
    const TenseurHH & B,
    bool plusZero ) [virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles  
Implémente [TenseurHH](#).

**6.830.2.2 Baisse2Indices()**

```
virtual TenseurBB & Tenseur_ns3HH::Baisse2Indices ( ) const [virtual]
```

— manipulation d'indice — -&gt; création de nouveaux tenseurs

Implémente [TenseurHH](#).**6.830.2.3 BaisseDernierIndice()**

```
virtual TenseurHB & Tenseur_ns3HH::BaisseDernierIndice ( ) const [virtual]
```

— manipulation d'indice — -&gt; création de nouveaux tenseurs

Implémente [TenseurHH](#).**6.830.2.4 BaissePremierIndice()**

```
virtual TenseurBH & Tenseur_ns3HH::BaissePremierIndice ( ) const [virtual]
```

— manipulation d'indice — -&gt; création de nouveaux tenseurs

Implémente [TenseurHH](#).**6.830.2.5 Coor()**

```
double & Tenseur_ns3HH::Coor (
    const int i,
    const int j ) [virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémente [TenseurHH](#).

### 6.830.2.6 Det()

```
double Tenseur_ns3HH::Det ( ) const [virtual]
```

determinant de la matrice des coordonnees  
Implémente [TenseurHH](#).

### 6.830.2.7 Ecriture()

```
ostream & Tenseur_ns3HH::Ecriture (
    ostream & sort ) const [virtual]
```

lecture et écriture de données  
Implémente [TenseurHH](#).

### 6.830.2.8 Inita()

```
void Tenseur_ns3HH::Inita (
    double val ) [virtual]
```

initialise toutes les composantes à val  
Implémente [TenseurHH](#).

### 6.830.2.9 Inverse()

```
TenseurBB & Tenseur_ns3HH::Inverse ( ) const [virtual]
```

calcul du tenseur inverse par rapport au produit contracte  
Implémente [TenseurHH](#).

### 6.830.2.10 Lecture()

```
istream & Tenseur_ns3HH::Lecture (
    istream & entree ) [virtual]
```

lecture et écriture de données  
Implémente [TenseurHH](#).

### 6.830.2.11 MaxiComposante()

```
double Tenseur_ns3HH::MaxiComposante ( ) const [virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur  
Implémente [TenseurHH](#).

### 6.830.2.12 operator"!="()

```
int Tenseur_ns3HH::operator!= (
    const TenseurHH & ) const [virtual]
```

test  
Implémente [TenseurHH](#).

### 6.830.2.13 operator&&()

```
double Tenseur_ns3HH::operator&& (
    const TenseurBB & ) const [virtual]
```

produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe  
Implémente [TenseurHH](#).

**6.830.2.14 operator>()**

```
double Tenseur_ns3HH::operator() (
    const int i,
    const int j ) const [virtual]
```

Retourne la composante i,j du tenseur acces en lecture seule.

Implémente [TenseurHH](#).

**6.830.2.15 operator\*() [1/4]**

```
CoordonneeH Tenseur_ns3HH::operator* (
    const CoordonneeB & ) const [virtual]
```

produit contracte avec un vecteur

Implémente [TenseurHH](#).

**6.830.2.16 operator\*() [2/4]**

```
TenseurHH & Tenseur_ns3HH::operator* (
    const double & ) const [virtual]
```

operations \* double

Implémente [TenseurHH](#).

**6.830.2.17 operator\*() [3/4]**

```
TenseurHB & Tenseur_ns3HH::operator* (
    const TenseurBB & ) const [virtual]
```

idem avec BB

Implémente [TenseurHH](#).

**6.830.2.18 operator\*() [4/4]**

```
TenseurHH & Tenseur_ns3HH::operator* (
    const TenseurBH & ) const [virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté avec BH

Implémente [TenseurHH](#).

**6.830.2.19 operator\*=( )**

```
void Tenseur_ns3HH::operator*= (
    const double & ) [virtual]
```

operations \*= double

Implémente [TenseurHH](#).

**6.830.2.20 operator+( )**

```
TenseurHH & Tenseur_ns3HH::operator+ (
    const TenseurHH & ) const [virtual]
```

operations +

Implémente [TenseurHH](#).

**6.830.2.21 operator+=()**

```
void Tenseur_ns3HH::operator+= (
    const TenseurHH & ) [virtual]
```

operations +=

Implémente [TenseurHH](#).

**6.830.2.22 operator-() [1/2]**

```
TenseurHH & Tenseur_ns3HH::operator- ( ) const [virtual]
```

operations -

Implémente [TenseurHH](#).

**6.830.2.23 operator-() [2/2]**

```
TenseurHH & Tenseur_ns3HH::operator- (
    const TenseurHH & ) const [virtual]
```

operations - tens

Implémente [TenseurHH](#).

**6.830.2.24 operator-=()**

```
void Tenseur_ns3HH::operator-= (
    const TenseurHH & ) [virtual]
```

operations -=

Implémente [TenseurHH](#).

**6.830.2.25 operator/()**

```
TenseurHH & Tenseur_ns3HH::operator/ (
    const double & ) const [virtual]
```

operations /

Implémente [TenseurHH](#).

**6.830.2.26 operator/=()**

```
void Tenseur_ns3HH::operator/= (
    const double & ) [virtual]
```

operations +=

Implémente [TenseurHH](#).

**6.830.2.27 operator=()**

```
TenseurHH & Tenseur_ns3HH::operator= (
    const TenseurHH & ) [virtual]
```

operations =

Implémente [TenseurHH](#).

**6.830.2.28 operator==(())**

```
int Tenseur_ns3HH::operator==(
    const TenseurHH & ) const [virtual]
```

test



Implémente [TenseurHH](#).

### 6.830.2.29 Transpose()

```
TenseurHH & Tenseur_ns3HH::Transpose ( ) const [virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

Implémente [TenseurHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

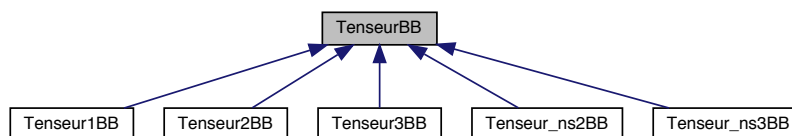
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.831 Référence de la classe TenseurBB

[TenseurBB](#): cas des composantes deux fois covariantes.

```
#include <Tenseur.h>
```

Graphe d'héritage de TenseurBB:



### Fonctions membres publiques

- virtual `~TenseurBB ()`  
*DESTRUCTEUR :*
- int `Dimension () const`  
*retourne la dimension du tenseur*
- virtual void `Inita (double val)=0`  
*initialise toutes les composantes à val*
- virtual `TenseurBB & operator+ (const TenseurBB &) const =0`  
*operations +*
- virtual void `operator+= (const TenseurBB &)=0`  
*operations +=*
- virtual `TenseurBB & operator- () const =0`  
*operations -*
- virtual `TenseurBB & operator- (const TenseurBB &) const =0`  
*operations - tens*
- virtual void `operator-= (const TenseurBB &)=0`  
*operations -=*
- virtual `TenseurBB & operator= (const TenseurBB &)=0`  
*operations =*
- virtual `TenseurBB & operator* (const double &) const =0`  
*operations \* double*
- virtual void `operator*=(const double &)=0`  
*operations \*= double*
- virtual `TenseurBB & operator/ (const double &) const =0`  
*operations /*
- virtual void `operator/=(const double &)=0`  
*operations /=*
- virtual `CoordonneeB operator* (const CoordonneeH &) const =0`  
*produit contracte avec un vecteur*

- virtual **TenseurBB** & **operator\*** (const **TenseurHB** &) const =0  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté*
- virtual **TenseurBH** & **operator\*** (const **TenseurHH** &) const =0  
*idem en BH*
- virtual double **operator&&** (const **TenseurHH** &) const =0  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- virtual int **operator==** (const **TenseurBB** &) const =0  
*test*
- virtual int **operator!=** (const **TenseurBB** &) const =0  
*test*
- virtual double **Det** () const =0  
*determinant de la matrice des coordonnees*
- void **ChBase** (const **Mat\_pleine** &beta)
- void **Var\_tenseur\_dans\_nouvelle\_base** (const **Mat\_pleine** &beta, **TenseurBB** &var\_tensBB, const **Mat\_pleine** &var\_beta)  
*il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base*
- virtual void **Affectation\_trans\_dimension** (const **TenseurBB** &B, bool plusZero)=0  
*Affectation\_trans\_dimension.*
- **TenseurBB** & **BaseAbsolue** (**TenseurBB** &A, const **BaseH** &gi) const  
*calcul des composantes du tenseur dans la base absolue: cas BB*
- **TenseurHH** & **BaseAbsolue** (**TenseurHH** &A, const **BaseH** &gi) const  
*idem pour HH*
- **TenseurBH** & **BaseAbsolue** (**TenseurBH** &A, const **BaseH** &gi) const  
*idem pour BH*
- **TenseurHB** & **BaseAbsolue** (**TenseurHB** &A, const **BaseH** &gi) const  
*idem pour HB*
- **TenseurBB** & **Baselocale** (**TenseurBB** &A, const **BaseB** &gi) const  
*calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue*
- virtual **TenseurBB** & **Transpose** () const =0  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual **TenseurHH** & **Monte2Indices** () const =0  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual **TenseurHB** & **MontePremierIndice** () const =0  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual **TenseurBH** & **MonteDernierIndice** () const =0  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual double **MaxiComposante** () const =0  
*calcul du maximum en valeur absolu des composantes du tenseur*
- virtual **TenseurHH** & **Inverse** () const =0  
*calcul du tenseur inverse par rapport au produit contracte*
- virtual double & **Coor** (int i, int j)=0  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- **Mat\_pleine** & **Matrice\_composante** (**Mat\_pleine** &a) const  
*retourne la matrice contenant les composantes du tenseur*
- void **Affectation** (const **Mat\_pleine** &a)  
*opération inverse: affectation en fonction d'une matrice donc sans vérif de variance!*
- virtual double **operator()** (int i, int j) const =0  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
- virtual istream & **Lecture** (istream &entree)=0  
*lecture et écriture de données*
- virtual ostream & **Ecriture** (ostream &sort) const =0  
*lecture et écriture de données*

## Attributs publics

- int **dimension**
- double \* **t**

## Fonctions membres protégées

- void **Message** (int dim, string mes) const  
*sortie d'un message standard dim = dimension du tenseur argument*

## Amis

- [TenseurBB](#) & **operator\*** (double r, const [TenseurBB](#) &t)
- [CoordonneeB](#) **operator\*** (const [CoordonneeH](#) &v, const [TenseurBB](#) &t)  
*produit contracte avec un vecteur*

### 6.831.1 Description détaillée

[TenseurBB](#): cas des composantes deux fois covariantes.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.831.2 Documentation des fonctions membres

#### 6.831.2.1 Affectation\_trans\_dimension()

```
virtual void TenseurBB::Affectation_trans_dimension (
    const TenseurBB & B,
    bool plusZero ) [pure virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

#### 6.831.2.2 BaseAbsolue()

```
TenseurBB & TenseurBB::BaseAbsolue (
    TenseurBB & A,
    const BaseH & gi ) const
```

calcul des composantes du tenseur dans la base absolue: cas BB

calcul des composantes du tenseur dans la base absolue  
la variance du résultat peut-être quelconque d'où quatre possibilités  
en fonction de l'argument A. Dans tous les cas les composantes sont identiques  
car la sortie est en absolue  
en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension  
différente du tenseur courant suivant que la dimension absolue et la dimension locale  
sont égales ou différentes, retour d'une référence sur A

### 6.831.2.3 Baselocale()

```
TenseurBB & TenseurBB::Baselocale (
    TenseurBB & A,
    const BaseB & gi ) const
```

calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue

calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension différente du tenseur courant suivant que la dimension absolue et la dimension locale sont égales ou différentes , retour d'une référence sur A

### 6.831.2.4 ChBase()

```
void TenseurBB::ChBase (
    const Mat_pleine & beta )
```

changement de base (cf. théorie)

changement de base (cf. théorie) : la matrice beta est telle que:

$gpB(i) = beta(i,j) * gB(j) \iff gp_i = beta_i^j * g_j$

et la matrice gamma telle que:

gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH

$gpH(i) = gamma(i,j) * gH(j)$ , i indice de ligne, j indice de colonne

c-a-d=  $gp^i = gamma^i_j * g^j$

rappel des différentes relations entre beta et gamma

$[beta]^{-1} = [gamma]^T$ ;  $[beta]^{-1T} = [gamma]$

$[beta] = [gamma]^{-1T}$ ;  $[beta]^T = [gamma]^{-1}$

changement de base pour de deux fois covariants:

$[Ap_{kl}] = [beta] * [A_{ij}] * [beta]^T$

### 6.831.2.5 Coor()

```
virtual double & TenseurBB::Coor (
    int i,
    int j ) [pure virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémenté dans [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), [Tenseur\\_ns3BB](#), et [Tenseur1BB](#).

### 6.831.2.6 Det()

```
virtual double TenseurBB::Det ( ) const [pure virtual]
```

determinant de la matrice des coordonnees

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.7 Ecriture()

```
virtual ostream & TenseurBB::Ecriture (
    ostream & sort ) const [pure virtual]
```

lecture et écriture de données

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.8 Inita()

```
virtual void TenseurBB::Inita (
    double val ) [pure virtual]
```

initialise toutes les composantes à val

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.9 Inverse()

```
virtual TenseurHH & TenseurBB::Inverse ( ) const [pure virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.10 Lecture()

```
virtual istream & TenseurBB::Lecture (
    istream & entree ) [pure virtual]
```

lecture et écriture de données

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.11 MaxiComposante()

```
virtual double TenseurBB::MaxiComposante ( ) const [pure virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.12 Monte2Indices()

```
virtual TenseurHH & TenseurBB::Monte2Indices ( ) const [pure virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.13 MonteDernierIndice()

```
virtual TenseurBH & TenseurBB::MonteDernierIndice ( ) const [pure virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.14 MontePremierIndice()

```
virtual TenseurHB & TenseurBB::MontePremierIndice ( ) const [pure virtual]
```

— manipulation d'indice — -> création de nouveaux tenseurs

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.15 operator"!=()

```
virtual int TenseurBB::operator!= (
    const TenseurBB & ) const [pure virtual]
```

test

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.16 operator&&()

```
virtual double TenseurBB::operator&& (
    const TenseurHH & ) const [pure virtual]
```

produit contracté contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.17 operator()()**

```
virtual double TenseurBB::operator() (
    int i,
    int j ) const [pure virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seulement.

Implémenté dans [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), [Tenseur\\_ns3BB](#), et [Tenseur1BB](#).

**6.831.2.18 operator\*() [1/4]**

```
virtual CoordonneeB TenseurBB::operator* (
    const CoordonneeH & ) const [pure virtual]
```

produit contracte avec un vecteur

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.19 operator\*() [2/4]**

```
virtual TenseurBB & TenseurBB::operator* (
    const double & ) const [pure virtual]
```

operations \* double

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.20 operator\*() [3/4]**

```
virtual TenseurBB & TenseurBB::operator* (
    const TenseurHB & ) const [pure virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B \rightarrow$  donc c'est l'indice du milieu qui est contracté

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.21 operator\*() [4/4]**

```
virtual TenseurBH & TenseurBB::operator* (
    const TenseurHH & ) const [pure virtual]
```

idem en BH

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.22 operator\*=( )**

```
virtual void TenseurBB::operator*= (
    const double & ) [pure virtual]
```

operations \*= double

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.23 operator+( )**

```
virtual TenseurBB & TenseurBB::operator+ (
    const TenseurBB & ) const [pure virtual]
```

operations +

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.24 operator+=()**

```
virtual void TenseurBB::operator+= (
    const TenseurBB & ) [pure virtual]
```

operations +=

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.25 operator-() [1/2]**

```
virtual TenseurBB & TenseurBB::operator- ( ) const [pure virtual]
```

operations -

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.26 operator-() [2/2]**

```
virtual TenseurBB & TenseurBB::operator- (
    const TenseurBB & ) const [pure virtual]
```

operations - tens

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.27 operator-=()**

```
virtual void TenseurBB::operator-= (
    const TenseurBB & ) [pure virtual]
```

operations -=

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.28 operator/()**

```
virtual TenseurBB & TenseurBB::operator/ (
    const double & ) const [pure virtual]
```

operations /

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.29 operator/=()**

```
virtual void TenseurBB::operator/= (
    const double & ) [pure virtual]
```

operations /=

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.30 operator=()**

```
virtual TenseurBB & TenseurBB::operator= (
    const TenseurBB & ) [pure virtual]
```

operations =

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

**6.831.2.31 operator==(())**

```
virtual int TenseurBB::operator==(
    const TenseurBB & ) const [pure virtual]
```

test

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.32 Transpose()

```
virtual TenseurBB & TenseurBB::Transpose ( ) const [pure virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

Implémenté dans [Tenseur1BB](#), [Tenseur2BB](#), [Tenseur\\_ns2BB](#), [Tenseur3BB](#), et [Tenseur\\_ns3BB](#).

### 6.831.2.33 Var\_tenseur\_dans\_nouvelle\_base()

```
void TenseurBB::Var_tenseur_dans_nouvelle_base (
    const Mat_pleine & beta,
    TenseurBB & var_tensBB,
    const Mat_pleine & var_beta )
```

il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base

il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base  
connaissant sa variation dans la base actuelle

this : le tenseur

var\_tensBB : en entrée: la variation du tenseur dans la base initiale qu'on appelle  $g^i$

var\_tensBB : en sortie: la variation du tenseur dans la base finale qu'on appelle  $gp^i$

beta : en entrée  $gpB(i) = beta(i,j) * gB(j)$

var\_beta : en entrée : la variation de beta

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur.h

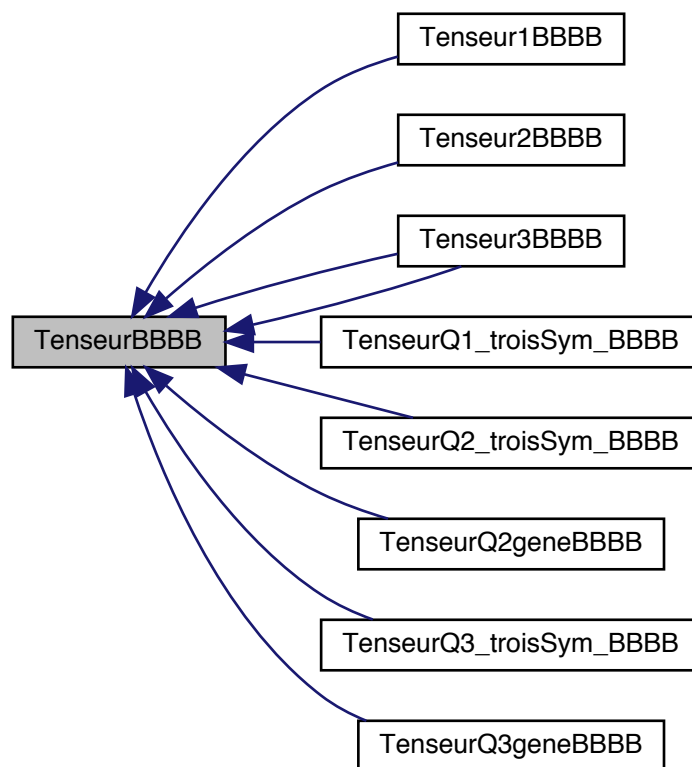
## 6.832 Référence de la classe TenseurBBBB

cas des composantes 4 fois covariantes

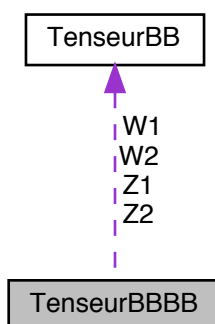
```
#include <TenseurQ.h>
```



Grappe d'héritage de TenseurBBBB:



Grappe de collaboration de TenseurBBBB:



## Fonctions membres publiques

- **TenseurBBBB** (const [TenseurBB](#) \*Zi1, const [TenseurBB](#) \*Zi2, const [TenseurBB](#) \*Wi1, const [TenseurBB](#) \*Wi2)
- **TenseurBBBB** (const [TenseurBBBB](#) &)
- void **Libere** (int cas)
- int **Dimension** () const
- **TenseurBBBB operator+** (const [TenseurBBBB](#) &) const
- void **operator+=** (const [TenseurBBBB](#) &)
- **TenseurBBBB operator-** () const
- **TenseurBBBB operator-** (const [TenseurBBBB](#) &) const
- void **operator-=** (const [TenseurBBBB](#) &)
- **TenseurBBBB & operator=** (const [TenseurBBBB](#) &)
- **TenseurBBBB operator\*** (const double &) const
- void **operator\*=** (const double &)
- **TenseurBBBB operator/** (const double &) const
- void **operator/=** (const double &)
- **TenseurBB & operator&&** (const [TenseurHH](#) &) const
- bool **operator==** (const [TenseurBBBB](#) &) const
- bool **operator!=** (const [TenseurBBBB](#) &) const
- **operator TenseurBBBB &** (void)
- void **change** (int i, int j, int k, int l, double &val)
- double **operator()** (int i, int j, int k, int l) const
- const [TenseurBB](#) & **Z\_1** () const
- const [TenseurBB](#) & **Z\_2** () const
- const [TenseurBB](#) & **W\_1** () const
- const [TenseurBB](#) & **W\_2** () const
- const [TenseurBB](#) \* **Z\_1P** () const
- const [TenseurBB](#) \* **Z\_2P** () const
- const [TenseurBB](#) \* **W\_1P** () const
- const [TenseurBB](#) \* **W\_2P** () const
- int **Dimension** () const
- virtual void **Inita** (double val)=0
- virtual [TenseurBBBB](#) & **operator+** (const [TenseurBBBB](#) &) const =0
- virtual void **operator+=** (const [TenseurBBBB](#) &)=0
- virtual [TenseurBBBB](#) & **operator-** () const =0
- virtual [TenseurBBBB](#) & **operator-** (const [TenseurBBBB](#) &) const =0
- virtual void **operator-=** (const [TenseurBBBB](#) &)=0
- virtual [TenseurBBBB](#) & **operator=** (const [TenseurBBBB](#) &)=0
- virtual [TenseurBBBB](#) & **operator\*** (const double &) const =0
- virtual void **operator\*=** (const double &)=0
- virtual [TenseurBBBB](#) & **operator/** (const double &) const =0
- virtual void **operator/=** (const double &)=0
- virtual [TenseurBB](#) & **operator&&** (const [TenseurHH](#) &) const =0
- virtual [TenseurBBBB](#) & **Transpose1et2avec3et4** () const =0
- virtual void **Affectation\_trans\_dimension** (const [TenseurBBBB](#) &B, bool plusZero)=0
- virtual int **operator==** (const [TenseurBBBB](#) &) const =0
- int **operator!=** (const [TenseurBBBB](#) &a) const
- virtual void **Change** (int i, int j, int k, int l, const double &val)=0
- virtual void **ChangePlus** (int i, int j, int k, int l, const double &val)=0
- virtual double **operator()** (int i, int j, int k, int l) const =0
- virtual double **MaxiComposante** () const =0
- virtual istream & **Lecture** (istream &entree)=0
- virtual ostream & **Ecriture** (ostream &sort) const =0

## Fonctions membres publiques statiques

- static [TenseurBBBB](#) **Prod\_tensoriel** (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- static [TenseurBBBB](#) **Prod\_tensoriel\_barre** (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)

## Attributs publics

- int **dimension**
- double \* **t**

## Fonctions membres protégées

- void **Message** (int dim, string mes) const
- void **Message** (int dim, string mes) const
- virtual **TenseurBB** & **Prod\_gauche** (const **TenseurHH** &F) const =0

## Attributs protégés

- **TenseurBB** \* **Z1**
- **TenseurBB** \* **Z2**
- **TenseurBB** \* **W1**
- **TenseurBB** \* **W2**

## Amis

- **TenseurBBBB** & **operator\*** (double r, const **TenseurBBBB** &t)
- **TenseurBB** & **operator&&** (const **TenseurHH** &F, const **TenseurBBBB** &T)
- istream & **operator>>** (istream &, **TenseurBBBB** &)
- ostream & **operator<<** (ostream &, const **TenseurBBBB** &)
- **TenseurBBBB** & **operator\*** (double r, const **TenseurBBBB** &t)
- **TenseurBB** & **operator&&** (const **TenseurHH** &F, const **TenseurBBBB** &T)

### 6.832.1 Description détaillée

cas des composantes 4 fois covariantes

Auteur

Gérard Rio

Version

1.0

Date

2/5/2002

La documentation de cette classe a été générée à partir du fichier suivant :

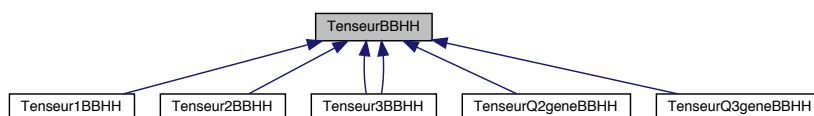
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurO4.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)

## 6.833 Référence de la classe TenseurBBHH

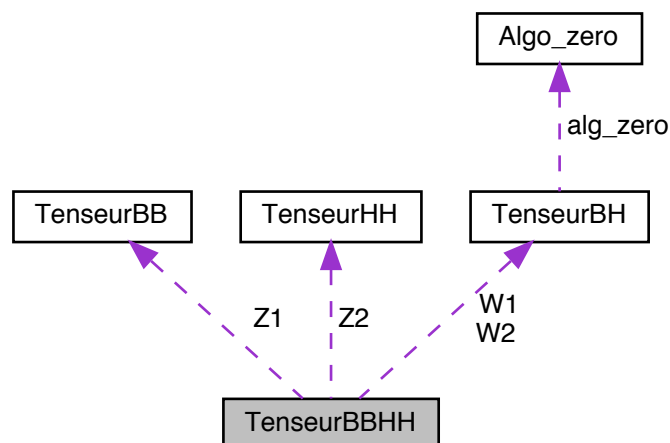
cas des composantes mixte BBHH

```
#include <TenseurQ.h>
```

Graphe d'héritage de TenseurBBHH:



Graphe de collaboration de TenseurBBHH:



## Fonctions membres publiques

- **TenseurBBHH** (const [TenseurBB](#) \*Z1, const [TenseurHH](#) \*Z2, const [TenseurBH](#) \*W1, const [TenseurBH](#) \*W2)
- **TenseurBBHH** (const [TenseurBBHH](#) &)
- void **Libere** (int cas)
- int **Dimension** () const
- [TenseurBBHH](#) **operator+** (const [TenseurBBHH](#) &) const
- void **operator+=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) **operator-** () const
- [TenseurBBHH](#) **operator-** (const [TenseurBBHH](#) &) const
- void **operator-=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) & **operator=** (const [TenseurBBHH](#) &)
- [TenseurBBHH](#) **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurBBHH](#) **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBB](#) & **operator&&** (const [TenseurBB](#) &) const
- bool **operator==** (const [TenseurBBHH](#) &) const
- bool **operator!=** (const [TenseurBBHH](#) &) const
- **operator** [TenseurBBHH](#) & (void)
- void **change** (int i, int j, int k, int l, double &val)
- double **operator()** (int i, int j, int k, int l) const
- const [TenseurBB](#) & **Z\_1** () const
- const [TenseurHH](#) & **Z\_2** () const
- const [TenseurBH](#) & **W\_1** () const
- const [TenseurBH](#) & **W\_2** () const
- const [TenseurBB](#) \* **Z\_1P** () const
- const [TenseurHH](#) \* **Z\_2P** () const
- const [TenseurBH](#) \* **W\_1P** () const
- const [TenseurBH](#) \* **W\_2P** () const
- int **Dimension** () const
- virtual void **Inita** (double val)=0
- virtual [TenseurBBHH](#) & **operator+** (const [TenseurBBHH](#) &) const =0
- virtual void **operator+=** (const [TenseurBBHH](#) &)=0
- virtual [TenseurBBHH](#) & **operator-** () const =0
- virtual [TenseurBBHH](#) & **operator-** (const [TenseurBBHH](#) &) const =0

- virtual void **operator=** (const TenseurBBHH &)=0
- virtual TenseurBBHH & **operator=** (const TenseurBBHH &)=0
- virtual TenseurBBHH & **operator\*** (const double &) const =0
- virtual void **operator\*=** (const double &)=0
- virtual TenseurBBHH & **operator/** (const double &) const =0
- virtual void **operator/=** (const double &)=0
- virtual TenseurBB & **operator&&** (const TenseurBB &) const =0
- virtual TenseurHHBB & **Transpose1et2avec3et4** () const =0
- virtual void **Affectation\_trans\_dimension** (const TenseurBBHH &B, bool plusZero)=0
- virtual int **operator==** (const TenseurBBHH &) const =0
- int **operator!=** (const TenseurBBHH &a) const
- virtual void **Change** (int i, int j, int k, int l, const double &val)=0
- virtual void **ChangePlus** (int i, int j, int k, int l, const double &val)=0
- virtual double **operator()** (int i, int j, int k, int l) const =0
- virtual double **MaxiComposante** () const =0
- virtual istream & **Lecture** (istream &entree)=0
- virtual ostream & **Ecriture** (ostream &sort) const =0

### Fonctions membres publiques statiques

- static TenseurBBHH **Prod\_tensoriel** (const TenseurBB &aBB, const TenseurHH &bHH)
- static TenseurBBHH **Prod\_tensoriel\_barre** (const TenseurBH &aBH, const TenseurBH &bBH)

### Attributs publics

- int **dimension**
- double \* **t**

### Fonctions membres protégées

- void **Message** (int dim, string mes) const
- void **Message** (int dim, string mes) const
- virtual TenseurHH & **Prod\_gauche** (const TenseurHH &F) const =0

### Attributs protégés

- TenseurBB \* **Z1**
- TenseurHH \* **Z2**
- TenseurBH \* **W1**
- TenseurBH \* **W2**

### Amis

- TenseurBBHH & **operator\*** (double r, const TenseurBBHH &t)
- TenseurHH & **operator&&** (const TenseurHH &F, const TenseurBBHH &T)
- istream & **operator>>** (istream &, TenseurBBHH &)
- ostream & **operator<<** (ostream &, const TenseurBBHH &)
- TenseurBBHH & **operator\*** (double r, const TenseurBBHH &t)
- TenseurHH & **operator&&** (const TenseurHH &F, const TenseurBBHH &T)

## 6.833.1 Description détaillée

cas des composantes mixte BBHH

Auteur

Gérard Rio

Version

1.0

## Date

2/5/2002

La documentation de cette classe a été générée à partir du fichier suivant :

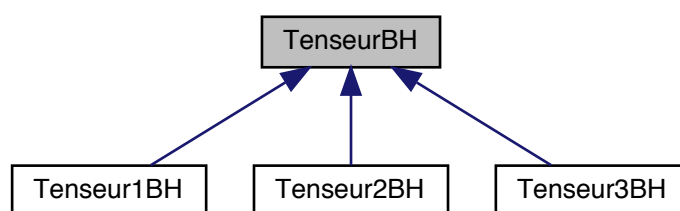
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurO4.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ.h

## 6.834 Référence de la classe TenseurBH

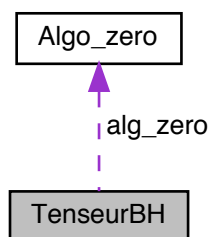
**TenseurBH**: cas des composantes mixtes BH.

```
#include <Tenseur.h>
```

Graphe d'héritage de TenseurBH:



Graphe de collaboration de TenseurBH:



### Fonctions membres publiques

- virtual `~TenseurBH` ()  
*DESTRUCTEUR :*
- int `Dimension` () const  
*retourne la dimension du tenseur*
- virtual void `Inita` (double val)=0  
*initialise toutes les composantes à val*
- virtual `TenseurBH & operator+` (const `TenseurBH` &) const =0  
*operations +*
- virtual void `operator+=` (const `TenseurBH` &)=0  
*operations +=*

- virtual `TenseurBH & operator-` (const `TenseurBH &`) const =0  
*operations -*
- virtual `TenseurBH & operator-` () const =0  
*operations opposé*
- virtual void `operator-=` (const `TenseurBH &`)=0  
*operations -=*
- virtual `TenseurBH & operator=` (const `TenseurBH &`)=0  
*operations =*
- virtual `TenseurBH & operator*` (const double &) const =0  
*operations \**
- virtual void `operator*=` (const double &)=0  
*operations \*=*
- virtual `TenseurBH & operator/` (const double &) const =0  
*operations /*
- virtual void `operator/=` (const double &)=0  
*operations /=*
- virtual `CoordonneeB operator*` (const `CoordonneeB &`) const =0  
*produit contracte avec un vecteur*
- virtual `TenseurBB & operator*` (const `TenseurBB &`) const =0  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté*
- virtual `TenseurBH & operator*` (const `TenseurBH &`) const =0
- virtual double `operator&&` (const `TenseurBH &`) const =0  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A.B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- virtual int `operator==` (const `TenseurBH &`) const =0  
*test*
- virtual int `operator!=` (const `TenseurBH &`) const =0
- virtual `TenseurBH & Inverse` () const =0  
*calcul du tenseur inverse par rapport au produit contracte*
- virtual double `Trace` () const =0  
*trace du tenseur ou premier invariant*
- virtual double `II` () const =0  
*second invariant = trace (A\*A)*
- virtual double `III` () const =0  
*troisieme invariant = trace ((A\*A)\*A)*
- virtual double `Det` () const =0  
*determinant de la matrice des coordonnees*
- virtual `Coordonnee ValPropre` (int &cas) const =0  
*calcul des valeurs propres*
- virtual `Coordonnee ValPropre` (int &cas, `Mat_pleine &mat`) const =0  
*calcul des valeurs propres*
- virtual void `VecteursPropres` (const `Coordonnee &Val_P`, int &cas, `Tableau< Coordonnee > &V_P`) const =0  
*calcul des vecteurs propres, les valeurs propres étant déjà connues*
- void `ChBase` (const `Mat_pleine &beta`, const `Mat_pleine &gamma`)  
*changement de base (cf. théorie)*
- void `Var_tenseur_dans_nouvelle_base` (const `Mat_pleine &beta`, `TenseurBH &var_tensBH`, const `Mat_pleine &var_beta`, const `Mat_pleine &gamma`, const `Mat_pleine &var_gamma`)  
*il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base*
- virtual void `Affectation_trans_dimension` (const `TenseurBH &B`, bool plusZero)=0  
*Affectation\_trans\_dimension.*
- `TenseurBH & BaseAbsolue` (`TenseurBH &A`, const `BaseB &giB`, const `BaseH &giH`) const  
*calcul des composantes du tenseur dans la base absolue: cas HH*
- `TenseurHB & BaseAbsolue` (`TenseurHB &A`, const `BaseB &giB`, const `BaseH &giH`) const  
*idem cas HB*
- `TenseurBH & BaseLocale` (`TenseurBH &A`, const `BaseH &gih`, const `BaseB &gib`) const  
*calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue*
- virtual `TenseurHB & Transpose` () const =0  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual void `PermuteHautBas` ()=0

- *permuté Bas Haut, mais reste dans le même tenseur*
- virtual double **MaxiComposante** () const =0
- *calcul du maximum en valeur absolu des composantes du tenseur*
- virtual double & **Coor** (int i, int j)=0
- *Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- **Mat\_pleine** & **Matrice\_composante** (**Mat\_pleine** &a) const
- *retourne la matrice contenant les composantes du tenseur*
- void **Affectation** (const **Mat\_pleine** &a)
- *opération inverse: affectation en fonction d'une matrice donc sans vérif de variance!*
- virtual double **operator()** (int i, int j) const =0
- *Retourne la composante i,j du tenseur acces en lecture seulement.*
- virtual istream & **Lecture** (istream &entree)=0
- *lecture et écriture de données*
- virtual ostream & **Ecriture** (ostream &sort) const =0
- *lecture et écriture de données*

### Attributs publics

- int **dimension**
- double \* **t**

### Fonctions membres protégées

- void **Message** (int dim, string mes) const
- *sortie d'un message standard dim = dimension du tenseur argument*
- **Coordonnee ValPropre\_int** (int &cas, **Mat\_pleine** &mat) const
- *méthode interne pour le calcul de vecteurs propres*

### Attributs protégés statiques

- static **Algo\_zero alg\_zero**
- *initialisation d'une instance d'outil permettant la recherche de zero*

### Amis

- **TenseurBH** & **operator\*** (double r, const **TenseurBH** &t)
- **CoordonneeH operator\*** (const **CoordonneeH** &v, const **TenseurBH** &t)
- *produit contracté avec un vecteur*

#### 6.834.1 Description détaillée

**TenseurBH**: cas des composantes mixtes BH.

Auteur

Gérard Rio

Version

1.0

Date

23/01/97

#### 6.834.2 Documentation des fonctions membres



**6.834.2.1 Affectation\_trans\_dimension()**

```
virtual void TenseurBH::Affectation_trans_dimension (
    const TenseurBH & B,
    bool plusZero ) [pure virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les données manquantes sont inchangées,  
plusZero = true: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
des données possibles

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.2 BaseAbsolue()**

```
TenseurBH & TenseurBH::BaseAbsolue (
    TenseurBH & A,
    const BaseB & giB,
    const BaseH & giH ) const
```

calcul des composantes du tenseur dans la base absolue: cas HH

calcul des composantes du tenseur dans la base absolue  
la variance du résultat peut-être quelconque d'où quatre possibilités  
mais on n'en conserve que les non symétriques qui sont le cas général  
se qui évite de symétriser accidentellement un tenseur non symétrique  
en fonction de l'argument A. Dans tous les cas les composantes sont identiques  
car la sortie est en absolue.

en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension  
différente du tenseur courant, retour d'une référence sur A

**6.834.2.3 Baselocale()**

```
TenseurBH & TenseurBH::Baselocale (
    TenseurBH & A,
    const BaseH & gih,
    const BaseB & gib ) const
```

calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue

calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue  
en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension  
différente du tenseur courant suivant que la dimension absolue et la dimension locale  
sont égales ou différentes , retour d'une référence sur A

**6.834.2.4 ChBase()**

```
void TenseurBH::ChBase (
    const Mat_pleine & beta,
    const Mat_pleine & gamma )
```

changement de base (cf. théorie)

changement de base (cf. théorie) : la matrice beta est telle que:

$$gpB(i) = beta(i,j) * gB(j) \iff gp\_i = beta\_i^j * g\_j$$

et la matrice gamma telle que:

gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH

$$gpH(i) = gamma(i,j) * gH(j), \text{ i indice de ligne, j indice de colonne}$$

$$c-a-d= gp^i = gamma^i_j * g^j$$

rappel des différentes relations entre beta et gamma

$$[beta]^{-1} = [gamma]^T; [beta]^{-1T} = [gamma]$$

$[\text{beta}] = [\text{gamma}]^{-1T}$ ;  $[\text{beta}]^T = [\text{gamma}]^{-1}$

formule de changement de base

$[\text{Ap}_k^I] = [\text{beta}] * [\text{A}_j^I] * [\text{gamma}]^T$

$\text{beta}(i,j)$  représente les coordonnées de la nouvelle base naturelle  $\text{gpB}$  dans l'ancienne  $\text{gB}$

$\text{gpB}(i) = \text{beta}(i,j) * \text{gB}(j)$ ,  $i$  indice de ligne,  $j$  indice de colonne

#### 6.834.2.5 Coor()

```
virtual double & TenseurBH::Coor (
    int i,
    int j ) [pure virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture et en ecriture.

Implémenté dans [Tenseur2BH](#), [Tenseur3BH](#), et [Tenseur1BH](#).

#### 6.834.2.6 Det()

```
virtual double TenseurBH::Det ( ) const [pure virtual]
```

determinant de la matrice des coordonnees

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.7 Ecriture()

```
virtual ostream & TenseurBH::Ecriture (
    ostream & sort ) const [pure virtual]
```

lecture et écriture de données

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.8 II()

```
virtual double TenseurBH::II ( ) const [pure virtual]
```

second invariant = trace ( $A*A$ )

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.9 III()

```
virtual double TenseurBH::III ( ) const [pure virtual]
```

troisieme invariant = trace ( $(A*A)*A$ )

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.10 Inita()

```
virtual void TenseurBH::Inita (
    double val ) [pure virtual]
```

initialise toutes les composantes à val

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.11 Inverse()

```
virtual TenseurBH & TenseurBH::Inverse ( ) const [pure virtual]
```

calcul du tenseur inverse par rapport au produit contracte

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.12 Lecture()**

```
virtual istream & TenseurBH::Lecture (
    istream & entree ) [pure virtual]
```

lecture et écriture de données

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.13 MaxiComposante()**

```
virtual double TenseurBH::MaxiComposante ( ) const [pure virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.14 operator&&()**

```
virtual double TenseurBH::operator&& (
    const TenseurBH & ) const [pure virtual]
```

produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.15 operator()()**

```
virtual double TenseurBH::operator() (
    int i,
    int j ) const [pure virtual]
```

Retourne la composante i,j du tenseur acces en lecture seulement.

Implémenté dans [Tenseur2BH](#), [Tenseur3BH](#), et [Tenseur1BH](#).

**6.834.2.16 operator\*() [1/3]**

```
virtual CoordonneeB TenseurBH::operator* (
    const CoordonneeB & ) const [pure virtual]
```

produit contracte avec un vecteur

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.17 operator\*() [2/3]**

```
virtual TenseurBH & TenseurBH::operator* (
    const double & ) const [pure virtual]
```

operations \*

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.18 operator\*() [3/3]**

```
virtual TenseurBB & TenseurBH::operator* (
    const TenseurBB & ) const [pure virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A..B$  -> donc c'est l'indice du milieu qui est contracté

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.19 operator\*=( )**

```
virtual void TenseurBH::operator*= (
    const double & ) [pure virtual]
```

operations \*=

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.20 operator+()

```
virtual TenseurBH & TenseurBH::operator+ (
    const TenseurBH & ) const [pure virtual]
```

operations +

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.21 operator+=()

```
virtual void TenseurBH::operator+= (
    const TenseurBH & ) [pure virtual]
```

operations +=

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.22 operator-() [1/2]

```
virtual TenseurBH & TenseurBH::operator- ( ) const [pure virtual]
```

operations opposé

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.23 operator-() [2/2]

```
virtual TenseurBH & TenseurBH::operator- (
    const TenseurBH & ) const [pure virtual]
```

operations -

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.24 operator-=()

```
virtual void TenseurBH::operator-= (
    const TenseurBH & ) [pure virtual]
```

operations -=

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.25 operator/()

```
virtual TenseurBH & TenseurBH::operator/ (
    const double & ) const [pure virtual]
```

operations /

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.26 operator/=()

```
virtual void TenseurBH::operator/= (
    const double & ) [pure virtual]
```

operations /=

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.27 operator=()**

```
virtual TenseurBH & TenseurBH::operator= (
    const TenseurBH & ) [pure virtual]
```

operations =

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.28 operator==( )**

```
virtual int TenseurBH::operator==(
    const TenseurBH & ) const [pure virtual]
```

test

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.29 PermuteHautBas()**

```
virtual void TenseurBH::PermuteHautBas ( ) [pure virtual]
```

permuté Bas Haut, mais reste dans le même tenseur

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.30 Trace()**

```
virtual double TenseurBH::Trace ( ) const [pure virtual]
```

trace du tenseur ou premier invariant

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.31 Transpose()**

```
virtual TenseurBH & TenseurBH::Transpose ( ) const [pure virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprimé par Libere.

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.32 ValPropre() [1/2]**

```
virtual Coordonnee TenseurBH::ValPropre (
    int & cas ) const [pure virtual]
```

calcul des valeurs propres

valeurs propre dans le vecteur de retour, classée par ordres "décroissants"

cas indique le cas de valeur propre:

quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire dans ce cas les valeurs propres de retour sont nulles par défaut

dim = 1, cas=1 pour une valeur propre;

dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques

dim = 3 , cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)

, cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),

, cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du retour)

, cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du retour)

Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

**6.834.2.33 ValPropre() [2/2]**

```
virtual Coordonnee TenseurBH::ValPropre (
    int & cas,
```

```

    Mat_pleine & mat ) const [pure virtual]
calcul des valeurs propres

```

idem met en retour la matrice mat contient par colonne les vecteurs propre  
elle doit avoir la dimension du tenseur

les vecteurs propre sont exprimés dans le repère dual (contrairement au HB) pour les tenseurs dim 3  
pour dim=2: le premier vecteur propre est exprimé dans le repère dual  
le second vecteur propre est exprimé dans le repère naturel  
pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

#### 6.834.2.34 Var\_tenseur\_dans\_nouvelle\_base()

```

void TenseurBH::Var_tenseur_dans_nouvelle_base (
    const Mat_pleine & beta,
    TenseurBH & var_tensBH,
    const Mat_pleine & var_beta,
    const Mat_pleine & gamma,
    const Mat_pleine & var_gamma )

```

il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base

il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base  
connaissant sa variation dans la base actuelle

this : le tenseur

var\_tensBB : en entrée: la variation du tenseur dans les bases initiales qu'on appelle  $g_i$  et  $g^i$   
var\_tensBH : en sortie: la variation du tenseur dans les bases finales qu'on appelle  $gp_i$  et  $gp^j$   
beta : en entrée  $gpB(i) = beta(i,j) * gB(j)$   
var\_beta : en entrée : la variation de beta  
gamma : en entrée  $gpH(i) = gamma(i,j) * gH(j)$   
var\_gamma : en entrée : la variation de gamma  
 $[Ap_k^i] = [beta] * [A_i^j] * [gamma]^T$

#### 6.834.2.35 VecteursPropres()

```

virtual void TenseurBH::VecteursPropres (
    const Coordonnee & Val_P,
    int & cas,
    Tableau< Coordonnee > & V_P ) const [pure virtual]

```

calcul des vecteurs propres, les valeurs propres  
étant déjà connues

ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres  
étant déjà connues

en retour VP les vecteurs propre : doivent avoir la dimension du tenseur  
les vecteurs propre sont exprimés dans le repère naturel (pour les tenseurs dim 3  
pour dim=2: le premier vecteur propre est exprimé dans le repère naturel  
le second vecteur propre est exprimé dans le repère dual  
pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
en sortie cas = -1 s'il y a eu un problème, dans ce cas, V\_P est quelconque  
sinon si tout est ok, cas est identique en sortie avec l'entrée  
Implémenté dans [Tenseur1BH](#), [Tenseur2BH](#), et [Tenseur3BH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

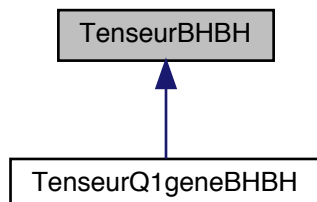
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.835 Référence de la classe TenseurBHBH

cas des composantes 2 fois mixtes BHBH

```
#include <TenseurQ.h>
```

Graphe d'héritage de TenseurBHBH:



### Fonctions membres publiques

- int **Dimension** () const
- virtual void **Inita** (double val)=0
- virtual [TenseurBHBH](#) & **operator+** (const [TenseurBHBH](#) &) const =0
- virtual void **operator+=** (const [TenseurBHBH](#) &)=0
- virtual [TenseurBHBH](#) & **operator-** () const =0
- virtual [TenseurBHBH](#) & **operator-** (const [TenseurBHBH](#) &) const =0
- virtual void **operator-=** (const [TenseurBHBH](#) &)=0
- virtual [TenseurBHBH](#) & **operator=** (const [TenseurBHBH](#) &)=0
- virtual [TenseurBHBH](#) & **operator\*** (const double &) const =0
- virtual void **operator\*-=** (const double &)=0
- virtual [TenseurBHBH](#) & **operator/** (const double &) const =0
- virtual void **operator/=** (const double &)=0
- virtual [TenseurBH](#) & **operator&&** (const [TenseurBH](#) &) const =0
- virtual [TenseurHBHB](#) & **Transpose1et2avec3et4** () const =0
- virtual void **Affectation\_trans\_dimension** (const [TenseurBHBH](#) &B, bool plusZero)=0
- virtual int **operator==** (const [TenseurBHBH](#) &) const =0
- int **operator!=** (const [TenseurBHBH](#) &a) const
- virtual void **Change** (int i, int j, int k, int l, const double &val)=0
- virtual void **ChangePlus** (int i, int j, int k, int l, const double &val)=0
- virtual double **operator()** (int i, int j, int k, int l) const =0
- virtual double **MaxiComposante** () const =0
- virtual istream & **Lecture** (istream &entree)=0
- virtual ostream & **Ecriture** (ostream &sort) const =0

### Attributs publics

- int **dimension**
- double \* **t**

### Fonctions membres protégées

- void **Message** (int dim, string mes) const
- virtual [TenseurBH](#) & **Prod\_gauche** (const [TenseurBH](#) &F) const =0

### Amis

- [TenseurBHBH](#) & **operator\*** (double r, const [TenseurBHBH](#) &t)
- [TenseurBH](#) & **operator&&** (const [TenseurBH](#) &F, const [TenseurBHBH](#) &T)

### 6.835.1 Description détaillée

cas des composantes 2 fois mixtes BHBH

Auteur

Gérard Rio

Version

1.0

Date

2/5/2002

La documentation de cette classe a été générée à partir du fichier suivant :

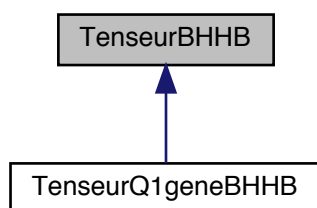
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ.h

### 6.836 Référence de la classe TenseurBHHB

cas des composantes 2 fois mixtes BHHB

```
#include <TenseurQ.h>
```

Graphe d'héritage de TenseurBHHB:



### Fonctions membres publiques

- int **Dimension** () const
- virtual void **Inita** (double val)=0
- virtual **TenseurBHHB** & **operator+** (const **TenseurBHHB** &) const =0
- virtual void **operator+=** (const **TenseurBHHB** &)=0
- virtual **TenseurBHHB** & **operator-** () const =0
- virtual **TenseurBHHB** & **operator-** (const **TenseurBHHB** &) const =0
- virtual void **operator-=** (const **TenseurBHHB** &)=0
- virtual **TenseurBHHB** & **operator=** (const **TenseurBHHB** &)=0
- virtual **TenseurBHHB** & **operator\*** (const double &) const =0
- virtual void **operator\*-=** (const double &)=0
- virtual **TenseurBHHB** & **operator/** (const double &) const =0
- virtual void **operator/=** (const double &)=0
- virtual **TenseurBH** & **operator&&** (const **TenseurBH** &) const =0
- virtual **TenseurHBBH** & **Transpose1et2avec3et4** () const =0
- virtual void **Affectation\_trans\_dimension** (const **TenseurBHHB** &B, bool plusZero)=0
- virtual int **operator==** (const **TenseurBHHB** &) const =0
- int **operator!=** (const **TenseurBHHB** &a) const
- virtual void **Change** (int i, int j, int k, int l, const double &val)=0
- virtual void **ChangePlus** (int i, int j, int k, int l, const double &val)=0
- virtual double **operator()** (int i, int j, int k, int l) const =0
- virtual double **MaxiComposante** () const =0
- virtual istream & **Lecture** (istream &entree)=0
- virtual ostream & **Ecriture** (ostream &sort) const =0



## Attributs publics

- int **dimension**
- double \* **t**

## Fonctions membres protégées

- void **Message** (int dim, string mes) const
- virtual **TenseurHB & Prod\_gauche** (const **TenseurBH** &F) const =0

## Amis

- **TenseurBHHB** & **operator\*** (double r, const **TenseurBHHB** &t)
- **TenseurHB** & **operator&&** (const **TenseurBH** &F, const **TenseurBHHB** &T)

### 6.836.1 Description détaillée

cas des composantes 2 fois mixtes BHHB

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

2/5/2002

La documentation de cette classe a été générée à partir du fichier suivant :

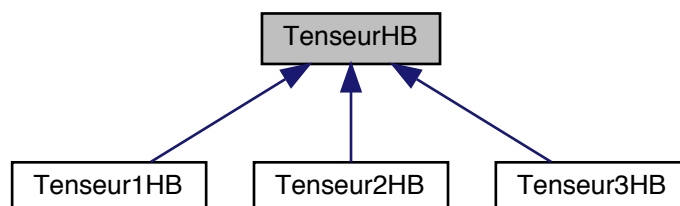
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)

## 6.837 Référence de la classe TenseurHB

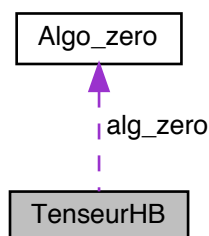
**TenseurHB**: cas des composantes mixtes HB.

```
#include <Tenseur.h>
```

Graphe d'héritage de TenseurHB:



Graphe de collaboration de TenseurHB:



## Fonctions membres publiques

- virtual `~TenseurHB ()`  
*DESTRUCTEUR :*
- int `Dimension ()` const  
*retourne la dimension du tenseur*
- virtual void `Inita (double val)=0`  
*initialise toutes les composantes à val*
- virtual `TenseurHB & operator+ (const TenseurHB &) const =0`  
*operations +*
- virtual void `operator+= (const TenseurHB &)=0`  
*operations +=*
- virtual `TenseurHB & operator- ()` const =0  
*operations opposé*
- virtual `TenseurHB & operator- (const TenseurHB &) const =0`  
*operations -*
- virtual void `operator-= (const TenseurHB &)=0`  
*operations -=*
- virtual `TenseurHB & operator= (const TenseurHB &)=0`  
*operations =*
- virtual `TenseurHB & operator* (const double &) const =0`  
*operations \**
- virtual void `operator*-= (const double &)=0`  
*operations \*=*
- virtual `TenseurHB & operator/ (const double &) const =0`  
*operations /*
- virtual void `operator/= (const double &)=0`  
*operations /=*
- virtual `CoordonneeH operator* (const CoordonneeH &) const =0`  
*produit contracte avec un vecteur*
- virtual `TenseurHH & operator* (const TenseurHH &) const =0`  
*produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté*
- virtual `TenseurHB & operator* (const TenseurHB &) const =0`
- virtual double `operator&& (const TenseurHB &) const =0`  
*produit contracte contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- virtual int `operator== (const TenseurHB &) const =0`  
*test*
- virtual int `operator!= (const TenseurHB &) const =0`
- virtual `TenseurHB & Inverse ()` const =0  
*calcul du tenseur inverse par rapport au produit contracte*
- virtual double `Trace ()` const =0

- *trace du tenseur ou premier invariant*  
virtual double `II ()` const =0
- *second invariant = trace (A\*A)*  
virtual double `III ()` const =0
- *troisieme invariant = trace ((A\*A)\*A)*  
virtual double `Det ()` const =0
- *determinant de la matrice des coordonnees*  
virtual `Coordonnee ValPropre (int &cas)` const =0
- *calcul des valeurs propres*  
virtual `Coordonnee ValPropre (int &cas, Mat_pleine &mat)` const =0
- *calcul des valeurs propres*  
virtual void `VecteursPropres (const Coordonnee &Val_P, int &cas, Tableau< Coordonnee > &V_P)` const =0
  
- *calcul des vecteurs propres, les valeurs propres étant déjà connues*  
void `ChBase (const Mat_pleine &beta, const Mat_pleine &gamma)`  
*changement de base (cf. théorie)*
- void `Var_tenseur_dans_nouvelle_base (const Mat_pleine &beta, TenseurBH &var_tensHB, const Mat_pleine &var_beta, const Mat_pleine &gamma, const Mat_pleine &var_gamma)`  
*il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base*
- virtual void `Affectation_trans_dimension (const TenseurHB &B, bool plusZero)=0`  
*Affectation\_trans\_dimension.*
- `TenseurHB & BaseAbsolue (TenseurHB &A, const BaseH &giH, const BaseB &giB)` const  
*calcul des composantes du tenseur dans la base absolue: cas HH*
- `TenseurBH & BaseAbsolue (TenseurBH &A, const BaseH &giH, const BaseB &giB)` const
- `TenseurHB & BaseLocale (TenseurHB &A, const BaseB &gib, const BaseH &gih)` const
  
- *calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension différente du tenseur courant suivant que la dimension absolue et la dimension locale sont égales ou différentes , retour d'une référence sur A*  
virtual `TenseurBH & Transpose ()` const =0  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual void `PermuteHautBas ()=0`  
*permuté Bas Haut, mais reste dans le même tenseur*
- virtual double `MaxiComposante ()` const =0  
*calcul du maximum en valeur absolu des composantes du tenseur*
- virtual double `& Coor (int i, int j)=0`  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- `Mat_pleine & Matrice_composante (Mat_pleine &a)` const  
*retourne la matrice contenant les composantes du tenseur*
- void `Affectation (const Mat_pleine &a)`  
*opération inverse: affectation en fonction d'une matrice donc sans vérif de variance!*
- virtual double `operator() (int i, int j)` const =0  
*Retourne la composante i,j du tenseur acces en lecture seulement.*
- virtual istream `& Lecture (istream &entree)=0`  
*lecture et écriture de données*
- virtual ostream `& Ecriture (ostream &sort)` const =0  
*lecture et écriture de données*

## Attributs publics

- int `dimension`
- double `* t`

## Fonctions membres protégées

- void `Message (int dim, string mes)` const  
*sortie d'un message standard dim = dimension du tenseur argument*
- `Coordonnee ValPropre_int (int &cas, Mat_pleine &mat)` const  
*méthode interne pour le calcul de vecteurs propres*

## Attributs protégés statiques

- static [Algo\\_zero](#) `alg_zero`  
*initialisation d'une instance d'outil permettant la recherche de zero*

## Amis

- [TenseurHB](#) & `operator*` (double r, const [TenseurHB](#) &t)
- [CoordonneeB](#) `operator*` (const [CoordonneeB](#) &v, const [TenseurHB](#) &t)  
*produit contracte avec un vecteur*

### 6.837.1 Description détaillée

[TenseurHB](#): cas des composantes mixtes HB.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

### 6.837.2 Documentation des fonctions membres

#### 6.837.2.1 `Affectation_trans_dimension()`

```
virtual void TenseurHB::Affectation_trans_dimension (
    const TenseurHB & B,
    bool plusZero ) [pure virtual]
```

`Affectation_trans_dimension`.

affectation de B dans this, `plusZero = false`: les données manquantes sont inchangées,  
`plusZero = true`: les données manquantes sont mises à 0  
si au contraire la dimension de B est plus grande que `*this`, il y a uniquement affectation  
des données possibles

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

#### 6.837.2.2 `BaseAbsolue()`

```
TenseurHB & TenseurHB::BaseAbsolue (
    TenseurHB & A,
    const BaseH & giH,
    const BaseB & giB ) const
```

calcul des composantes du tenseur dans la base absolue: cas HH

calcul des composantes du tenseur dans la base absolue  
la variance du résultat peut-être quelconque d'où quatre possibilités  
mais on n'en conserve que les non symétriques qui sont le cas général  
se qui évite de symétriser accidentellement un tenseur non symétrique  
en fonction de l'argument A. Dans tous les cas les composantes sont identiques  
car la sortie est en absolue.

en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension  
différente du tenseur courant, retour d'une référence sur A

**6.837.2.3 ChBase()**

```
void TenseurHB::ChBase (
    const Mat_pleine & beta,
    const Mat_pleine & gamma )
```

changement de base (cf. théorie)

changement de base (cf. théorie) : la matrice beta est telle que:

$$gpB(i) = beta(i,j) * gB(j) \iff gp\_i = beta\_i^j * g\_j$$

et la matrice gamma telle que:

gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH

$$gpH(i) = gamma(i,j) * gH(j), \text{ i indice de ligne, j indice de colonne}$$

$$c-a-d= gp^i = gamma^i_j * g^j$$

rappel des différentes relations entre beta et gamma

$$[beta]^{-1} = [gamma]^T; [beta]^{-1T} = [gamma]$$

$$[beta] = [gamma]^{-1T}; [beta]^T = [gamma]^{-1}$$

formule de changement de base

$$[Ap^k_l] = [gamma] * [A^i_j] * [beta]^T$$

beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB

$$gpB(i) = beta(i,j) * gB(j), \text{ i indice de ligne, j indice de colonne}$$

**6.837.2.4 Coor()**

```
virtual double & TenseurHB::Coor (
    int i,
    int j ) [pure virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémenté dans [Tenseur2HB](#), [Tenseur3HB](#), et [Tenseur1HB](#).

**6.837.2.5 Det()**

```
virtual double TenseurHB::Det ( ) const [pure virtual]
```

determinant de la matrice des coordonnees

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.6 Ecriture()**

```
virtual ostream & TenseurHB::Ecriture (
    ostream & sort ) const [pure virtual]
```

lecture et écriture de données

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.7 II()**

```
virtual double TenseurHB::II ( ) const [pure virtual]
```

second invariant = trace (A\*A)

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.8 III()**

```
virtual double TenseurHB::III ( ) const [pure virtual]
```

troisieme invariant = trace ((A\*A)\*A)

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.9 Inita()**

```
virtual void TenseurHB::Inita (
    double val ) [pure virtual]
```

initialise toutes les composantes à val

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.10 Inverse()**

```
virtual TenseurHB & TenseurHB::Inverse ( ) const [pure virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.11 Lecture()**

```
virtual istream & TenseurHB::Lecture (
    istream & entree ) [pure virtual]
```

lecture et écriture de données

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.12 MaxiComposante()**

```
virtual double TenseurHB::MaxiComposante ( ) const [pure virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.13 operator&&()**

```
virtual double TenseurHB::operator&& (
    const TenseurHB & ) const [pure virtual]
```

produit contracté contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.14 operator>()()**

```
virtual double TenseurHB::operator() (
    int i,
    int j ) const [pure virtual]
```

Retourne la composante i,j du tenseur acces en lecture seulement.

Implémenté dans [Tenseur2HB](#), [Tenseur3HB](#), et [Tenseur1HB](#).

**6.837.2.15 operator\*() [1/3]**

```
virtual CoordonneeH TenseurHB::operator* (
    const CoordonneeH & ) const [pure virtual]
```

produit contracté avec un vecteur

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.16 operator\*() [2/3]**

```
virtual TenseurHB & TenseurHB::operator* (
    const double & ) const [pure virtual]
```

operations \*

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

#### 6.837.2.17 operator\*() [3/3]

```
virtual TenseurHH & TenseurHB::operator* (
    const TenseurHH & ) const [pure virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté  
Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

#### 6.837.2.18 operator\*=( )

```
virtual void TenseurHB::operator*= (
    const double & ) [pure virtual]
```

operations \*=  
Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

#### 6.837.2.19 operator+( )

```
virtual TenseurHB & TenseurHB::operator+ (
    const TenseurHB & ) const [pure virtual]
```

operations +  
Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

#### 6.837.2.20 operator+=( )

```
virtual void TenseurHB::operator+= (
    const TenseurHB & ) [pure virtual]
```

operations +=  
Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

#### 6.837.2.21 operator-( ) [1/2]

```
virtual TenseurHB & TenseurHB::operator- ( ) const [pure virtual]
```

operations opposé  
Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

#### 6.837.2.22 operator-( ) [2/2]

```
virtual TenseurHB & TenseurHB::operator- (
    const TenseurHB & ) const [pure virtual]
```

operations -  
Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

#### 6.837.2.23 operator==( )

```
virtual void TenseurHB::operator==(
    const TenseurHB & ) [pure virtual]
```

operations ==  
Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.24 operator/()**

```
virtual TenseurHB & TenseurHB::operator/ (
    const double & ) const [pure virtual]
```

operations /

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.25 operator/=()**

```
virtual void TenseurHB::operator/= (
    const double & ) [pure virtual]
```

operations /=

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.26 operator=()**

```
virtual TenseurHB & TenseurHB::operator= (
    const TenseurHB & ) [pure virtual]
```

operations =

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.27 operator==(())**

```
virtual int TenseurHB::operator==(
    const TenseurHB & ) const [pure virtual]
```

test

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.28 PermuteHautBas()**

```
virtual void TenseurHB::PermuteHautBas ( ) [pure virtual]
```

permuté Bas Haut, mais reste dans le même tenseur

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.29 Trace()**

```
virtual double TenseurHB::Trace ( ) const [pure virtual]
```

trace du tenseur ou premier invariant

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.30 Transpose()**

```
virtual TenseurBH & TenseurHB::Transpose ( ) const [pure virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprimé par Libere.

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

**6.837.2.31 ValPropre() [1/2]**

```
virtual Coordonnee TenseurHB::ValPropre (
    int & cas ) const [pure virtual]
```

calcul des valeurs propres

valeurs propre dans le vecteur de retour, classée par ordres "décroissants"



cas indique le cas de valeur propre:

quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire dans ce cas les valeurs propres de retour sont nulles par défaut

dim = 1, cas=1 pour une valeur propre;

dim = 2, cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques

dim = 3, cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)

, cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),

, cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du retour)

, cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du retour)

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

### 6.837.2.32 ValPropre() [2/2]

```
virtual Coordonnee TenseurHB::ValPropre (
    int & cas,
    Mat_pleine & mat ) const [pure virtual]
```

calcul des valeurs propres

idem met en retour la matrice mat contient par colonne les vecteurs propre

elle doit avoir la dimension du tenseur

les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3

pour dim=2:le premier vecteur propre est exprime dans le repere naturel

le second vecteur propre est exprimé dans le repère dual

pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)

Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

### 6.837.2.33 Var\_tenseur\_dans\_nouvelle\_base()

```
void TenseurHB::Var_tenseur_dans_nouvelle_base (
    const Mat_pleine & beta,
    TenseurBH & var_tensHB,
    const Mat_pleine & var_beta,
    const Mat_pleine & gamma,
    const Mat_pleine & var_gamma )
```

il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base

il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base

connaissant sa variation dans la base actuelle

this : le tenseur

var\_tensBB : en entrée: la variation du tenseur dans les bases initiales qu'on appelle  $g_i$  et  $g^i$

var\_tensHB : en sortie: la variation du tenseur dans les bases finales qu'on appelle  $gp_i$  et  $gp^j$

beta : en entrée  $gpB(i) = beta(i,j) * gB(j)$

var\_beta : en entrée : la variation de beta

gamma : en entrée  $gpH(i) = gamma(i,j) * gH(j)$

var\_gamma : en entrée : la variation de gamma

$[Ap^k_l] = [gamma] * [A^{i^_j}] * [beta]^T$

### 6.837.2.34 VecteursPropres()

```
virtual void TenseurHB::VecteursPropres (
    const Coordonnee & Val_P,
    int & cas,
    Tableau< Coordonnee > & V_P ) const [pure virtual]
```

calcul des vecteurs propres, les valeurs propres  
étant déjà connues

ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres étant déjà connues  
 en retour VP les vecteurs propre : doivent avoir la dimension du tenseur  
 les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3 pour dim=2:le premier vecteur propre est exprime dans le repere naturel le second vecteur propre est exprimé dans le repère dual pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)  
 en sortie cas = -1 s'il y a eu un problème, dans ce cas, V\_P est quelconque sinon si tout est ok, cas est identique en sortie avec l'entrée  
 Implémenté dans [Tenseur1HB](#), [Tenseur2HB](#), et [Tenseur3HB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

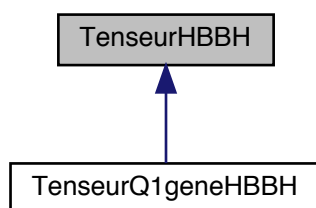
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.838 Référence de la classe TenseurHBBH

cas des composantes 2 fois mixtes HBBH

```
#include <TenseurQ.h>
```

Graphe d'héritage de TenseurHBBH:



### Fonctions membres publiques

- int **Dimension** () const
- virtual void **Inita** (double val)=0
- virtual [TenseurHBBH](#) & **operator+** (const [TenseurHBBH](#) &) const =0
- virtual void **operator+=** (const [TenseurHBBH](#) &)=0
- virtual [TenseurHBBH](#) & **operator-** () const =0
- virtual [TenseurHBBH](#) & **operator-** (const [TenseurHBBH](#) &) const =0
- virtual void **operator-=** (const [TenseurHBBH](#) &)=0
- virtual [TenseurHBBH](#) & **operator=** (const [TenseurHBBH](#) &)=0
- virtual [TenseurHBBH](#) & **operator\*** (const double &) const =0
- virtual void **operator\*=** (const double &)=0
- virtual [TenseurHBBH](#) & **operator/** (const double &) const =0
- virtual void **operator/=** (const double &)=0
- virtual [TenseurHB](#) & **operator&&** (const [TenseurBH](#) &) const =0
- virtual [TenseurBHHB](#) & **Transpose1et2avec3et4** () const =0
- virtual void **Affectation\_trans\_dimension** (const [TenseurHBBH](#) &B, bool plusZero)=0
- virtual int **operator==** (const [TenseurHBBH](#) &) const =0
- int **operator!=** (const [TenseurHBBH](#) &a) const
- virtual void **Change** (int i, int j, int k, int l, const double &val)=0
- virtual void **ChangePlus** (int i, int j, int k, int l, const double &val)=0
- virtual double **operator()** (int i, int j, int k, int l) const =0
- virtual double **MaxiComposante** () const =0
- virtual istream & **Lecture** (istream &entree)=0
- virtual ostream & **Ecriture** (ostream &sort) const =0

## Attributs publics

- int **dimension**
- double \* **t**

## Fonctions membres protégées

- void **Message** (int dim, string mes) const
- virtual **TenseurBH & Prod\_gauche** (const **TenseurHB &F**) const =0

## Amis

- **TenseurHBBH & operator\*** (double r, const **TenseurHBBH &t**)
- **TenseurBH & operator&&** (const **TenseurHB &F**, const **TenseurHBBH &T**)

### 6.838.1 Description détaillée

cas des composantes 2 fois mixtes HBBH

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

2/5/2002

La documentation de cette classe a été générée à partir du fichier suivant :

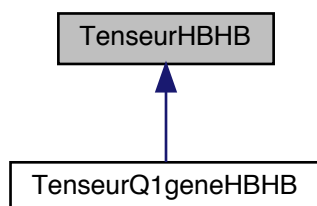
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)

## 6.839 Référence de la classe TenseurHBHB

cas des composantes 2 fois mixtes HBHB

```
#include <TenseurQ.h>
```

Graphe d'héritage de TenseurHBHB:



## Fonctions membres publiques

- int **Dimension** () const
- virtual void **Inita** (double val)=0
- virtual **TenseurHBHB & operator+** (const **TenseurHBHB &**) const =0
- virtual void **operator+=** (const **TenseurHBHB &**)=0
- virtual **TenseurHBHB & operator-** () const =0

- virtual [TenseurHBHB](#) & **operator-** (const [TenseurHBHB](#) &) const =0
- virtual void **operator=** (const [TenseurHBHB](#) &)=0
- virtual [TenseurHBHB](#) & **operator=** (const [TenseurHBHB](#) &)=0
- virtual [TenseurHBHB](#) & **operator\*** (const double &) const =0
- virtual void **operator\*=** (const double &)=0
- virtual [TenseurHBHB](#) & **operator/** (const double &) const =0
- virtual void **operator/=** (const double &)=0
- virtual [TenseurHB](#) & **operator&&** (const [TenseurHB](#) &) const =0
- virtual [TenseurBHBH](#) & **Transpose1et2avec3et4** () const =0
- virtual void **Affectation\_trans\_dimension** (const [TenseurHBHB](#) &B, bool plusZero)=0
- virtual int **operator==** (const [TenseurHBHB](#) &) const =0
- int **operator!=** (const [TenseurHBHB](#) &a) const
- virtual void **Change** (int i, int j, int k, int l, const double &val)=0
- virtual void **ChangePlus** (int i, int j, int k, int l, const double &val)=0
- virtual double **operator()** (int i, int j, int k, int l) const =0
- virtual double **MaxiComposante** () const =0
- virtual istream & **Lecture** (istream &entree)=0
- virtual ostream & **Ecriture** (ostream &sort) const =0

### Attributs publics

- int **dimension**
- double \* **t**

### Fonctions membres protégées

- void **Message** (int dim, string mes) const
- virtual [TenseurHB](#) & **Prod\_gauche** (const [TenseurHB](#) &F) const =0

### Amis

- [TenseurHBHB](#) & **operator\*** (double r, const [TenseurHBHB](#) &t)
- [TenseurHB](#) & **operator&&** (const [TenseurHB](#) &F, const [TenseurHBHB](#) &T)

### 6.839.1 Description détaillée

cas des composantes 2 fois mixtes HBHB

Auteur

Gérard Rio

Version

1.0

Date

2/5/2002

La documentation de cette classe a été générée à partir du fichier suivant :

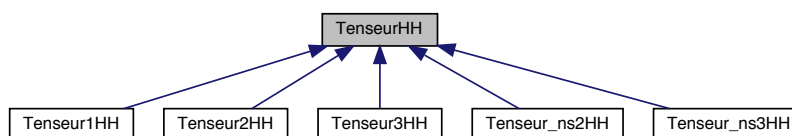
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)

### 6.840 Référence de la classe TenseurHH

[TenseurHH](#): cas des composantes deux fois contravariantes.

```
#include <Tenseur.h>
```

Graphe d'héritage de TenseurHH:



## Fonctions membres publiques

- virtual `~TenseurHH ()`  
*DESTRUCTEUR :*
- int `Dimension ()` const  
*retourne la dimension du tenseur*
- virtual void `Inita (double val)=0`  
*initialise toutes les composantes à val*
- virtual `TenseurHH & operator+ (const TenseurHH &) const =0`  
*operations +*
- virtual void `operator+= (const TenseurHH &)=0`  
*operations +=*
- virtual `TenseurHH & operator- () const =0`  
*operations -*
- virtual `TenseurHH & operator- (const TenseurHH &) const =0`  
*operations - tens*
- virtual void `operator-= (const TenseurHH &)=0`  
*operations -=*
- virtual `TenseurHH & operator= (const TenseurHH &)=0`  
*operations =*
- virtual `TenseurHH & operator* (const double &) const =0`  
*operations \* double*
- virtual void `operator*= (const double &)=0`  
*operations \*= double*
- virtual `TenseurHH & operator/ (const double &) const =0`  
*operations /*
- virtual void `operator/= (const double &)=0`  
*operations +/-*
- virtual `CoordonneeH operator* (const CoordonneeB &) const =0`  
*produit contracté avec un vecteur*
- virtual `TenseurHH & operator* (const TenseurBH &) const =0`  
*produit contracté contracté une fois  $A(i,j)*B(j,k)=A.B$  -> donc c'est l'indice du milieu qui est contracté avec BH*
- virtual `TenseurHB & operator* (const TenseurBB &) const =0`  
*idem avec BB*
- virtual double `operator&& (const TenseurBB &) const =0`  
*produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe*
- virtual int `operator== (const TenseurHH &) const =0`  
*test*
- virtual int `operator!= (const TenseurHH &) const =0`  
*test*
- virtual double `Det ()` const =0  
*determinant de la matrice des coordonnées*
- virtual `TenseurBB & Inverse ()` const =0  
*calcul du tenseur inverse par rapport au produit contracté*
- void `ChBase (const Mat_pleine &gamma, bool inverse=false)`  
*changement de base (cf. théorie)*

- void `Var_tenseur_dans_nouvelle_base` (const `Mat_pleine` &gamma, `TenseurHH` &var\_tensHH, const `Mat_pleine` &var\_gamma)  
*il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base*
- virtual void `Affectation_trans_dimension` (const `TenseurHH` &B, bool plusZero)=0  
*Affectation\_trans\_dimension.*
- `TenseurHH` & `BaseAbsolue` (`TenseurHH` &A, const `BaseB` &gi) const  
*calcul des composantes du tenseur dans la base absolue: cas HH*
- `TenseurBB` & `BaseAbsolue` (`TenseurBB` &A, const `BaseB` &gi) const  
*idem: cas BB*
- `TenseurBH` & `BaseAbsolue` (`TenseurBH` &A, const `BaseB` &gi) const  
*idem: cas BH*
- `TenseurHB` & `BaseAbsolue` (`TenseurHB` &A, const `BaseB` &gi) const  
*idem: cas HB*
- `TenseurHH` & `Baselocale` (`TenseurHH` &A, const `BaseH` &gi) const  
*calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue*
- virtual `TenseurHH` & `Transpose` () const =0  
*ATTENTION creation d'un tenseur transpose qui est supprime par Libere.*
- virtual double `MaxiComposante` () const =0  
*calcul du maximum en valeur absolu des composantes du tenseur*
- virtual `TenseurBB` & `Baisse2Indices` () const =0  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurBH` & `BaissePremierIndice` () const =0  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual `TenseurHB` & `BaisseDernierIndice` () const =0  
*— manipulation d'indice — -> création de nouveaux tenseurs*
- virtual double & `Coor` (int i, int j)=0  
*Retourne la composante i,j du tenseur acces en lecture et en ecriture.*
- `Mat_pleine` & `Matrice_composante` (`Mat_pleine` &a) const  
*retourne la matrice contenant les composantes du tenseur*
- void `Affectation` (const `Mat_pleine` &a)  
*opération inverse: affectation en fonction d'une matrice donc sans vérif de variance!*
- virtual double `operator()` (int i, int j) const =0  
*Retourne la composante i,j du tenseur acces en lecture seule.*
- virtual istream & `Lecture` (istream &entree)=0  
*lecture et écriture de données*
- virtual ostream & `Ecriture` (ostream &sort) const =0  
*lecture et écriture de données*

## Attributs publics

- int `dimension`
- double \* `t`

## Fonctions membres protégées

- void `Message` (int dim, string mes) const  
*sortie d'un message standard dim = dimension du tenseur argument*

## Amis

- `TenseurHH` & `operator*` (double r, const `TenseurHH` &t)
- `CoordonneeH` `operator*` (const `CoordonneeB` &v, const `TenseurHH` &t)  
*produit contracte avec un vecteur*

### 6.840.1 Description détaillée

`TenseurHH`: cas des composantes deux fois contravariantes.

**Auteur**

G rard Rio

**Version**

1.0

**Date**

23/01/97

**6.840.2 Documentation des fonctions membres****6.840.2.1 Affectation\_trans\_dimension()**

```
virtual void TenseurHH::Affectation_trans_dimension (
    const TenseurHH & B,
    bool plusZero ) [pure virtual]
```

Affectation\_trans\_dimension.

affectation de B dans this, plusZero = false: les donn es manquantes sont inchang es,  
 plusZero = true: les donn es manquantes sont mises   0  
 si au contraire la dimension de B est plus grande que \*this, il y a uniquement affectation  
 des donn es possibles

Impl ment  dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.2 Baisse2Indices()**

```
virtual TenseurBB & TenseurHH::Baisse2Indices ( ) const [pure virtual]
```

— manipulation d'indice — -> cr ation de nouveaux tenseurs

Impl ment  dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.3 BaisseDernierIndice()**

```
virtual TenseurHB & TenseurHH::BaisseDernierIndice ( ) const [pure virtual]
```

— manipulation d'indice — -> cr ation de nouveaux tenseurs

Impl ment  dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.4 BaissePremierIndice()**

```
virtual TenseurBH & TenseurHH::BaissePremierIndice ( ) const [pure virtual]
```

— manipulation d'indice — -> cr ation de nouveaux tenseurs

Impl ment  dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.5 BaseAbsolue()**

```
TenseurHH & TenseurHH::BaseAbsolue (
    TenseurHH & A,
    const BaseB & gi ) const
```

calcul des composantes du tenseur dans la base absolue: cas HH

calcul des composantes du tenseur dans la base absolue  
 la variance du r sultat peut  tre quelconque d'o  quatre possibilit s  
 en fonction de l'argument A. Dans tous les cas les composantes sont identiques  
 car la sortie est en absolue

en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension différente du tenseur courant suivant que la dimension absolue et la dimension locale sont égales ou différentes, retour d'une référence sur A

### 6.840.2.6 Baselocale()

```
TenseurHH & TenseurHH::Baselocale (
    TenseurHH & A,
    const BaseH & gi ) const
```

calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue

calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension différente du tenseur courant suivant que la dimension absolue et la dimension locale sont égales ou différentes , retour d'une référence sur A

### 6.840.2.7 ChBase()

```
void TenseurHH::ChBase (
    const Mat_pleine & gamma,
    bool inverse = false )
```

changement de base (cf. théorie)

changement de base (cf. théorie) : la matrice beta est telle que:

par défaut inverse = false : (c'est l'utilisation historique)

beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB

$gpB(i) = beta(i,j) * gB(j)$ , i indice de ligne, j indice de colonne

$gpB(i) = beta(i,j) * gB(j) \iff gp_i = beta_i^j * g_j$

et on a la matrice gamma telle que:

gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH

$gpH(i) = gamma(i,j) * gH(j)$ , i indice de ligne, j indice de colonne

$c-a-d= gp^i = gamma^i_j * g^j$

rappel des différentes relations entre beta et gamma

$[beta]^{-1} = [gamma]^T$ ;  $[beta]^{-1T} = [gamma]$

$[beta] = [gamma]^{-1T}$ ;  $[beta]^T = [gamma]^{-1}$

changement de base pour de deux fois contravariants:

$[Ap^{kl}] = [gamma] * [A^{ij}] * [gamma]^T$

donc le changement de base s'effectue directement à l'aide de gamma

si inverse = true:

beta(i,j) représente les coordonnées de l'ancienne base gB dans la nouvelle gpB

$gB(i) = beta(i,j) * gpB(j)$ , i indice de ligne, j indice de colonne

la formule de changement de base est alors:

$[Ap^{kl}] = [beta]^T * [A^{ij}] * [beta]$

nécessite d'inverser la matrice gamma pour calculer beta

### 6.840.2.8 Coor()

```
virtual double & TenseurHH::Coor (
    int i,
    int j ) [pure virtual]
```

Retourne la composante i,j du tenseur acces en lecture et en ecriture.

Implémenté dans [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), [Tenseur\\_ns3HH](#), et [Tenseur1HH](#).

### 6.840.2.9 Det()

```
virtual double TenseurHH::Det ( ) const [pure virtual]
```

determinant de la matrice des coordonnees

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).



### 6.840.2.10 Ecriture()

```
virtual ostream & TenseurHH::Ecriture (
    ostream & sort ) const [pure virtual]
```

lecture et écriture de données

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

### 6.840.2.11 Inita()

```
virtual void TenseurHH::Inita (
    double val ) [pure virtual]
```

initialise toutes les composantes à val

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

### 6.840.2.12 Inverse()

```
virtual TenseurBB & TenseurHH::Inverse ( ) const [pure virtual]
```

calcul du tenseur inverse par rapport au produit contracté

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

### 6.840.2.13 Lecture()

```
virtual istream & TenseurHH::Lecture (
    istream & entree ) [pure virtual]
```

lecture et écriture de données

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

### 6.840.2.14 MaxiComposante()

```
virtual double TenseurHH::MaxiComposante ( ) const [pure virtual]
```

calcul du maximum en valeur absolu des composantes du tenseur

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

### 6.840.2.15 operator"!=(())

```
virtual int TenseurHH::operator!=(
    const TenseurHH & ) const [pure virtual]
```

test

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

### 6.840.2.16 operator&&()

```
virtual double TenseurHH::operator&& (
    const TenseurBB & ) const [pure virtual]
```

produit contracté deux fois  $A(i,j)*B(j,i)=A..B$  -> on contracte d'abord l'indice du milieu puis l'indice externe

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.17 operator>()**

```
virtual double TenseurHH::operator() (
    int i,
    int j ) const [pure virtual]
```

Retourne la composante  $i,j$  du tenseur acces en lecture seule.

Implémenté dans [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), [Tenseur\\_ns3HH](#), et [Tenseur1HH](#).

**6.840.2.18 operator\*() [1/4]**

```
virtual CoordonneeH TenseurHH::operator* (
    const CoordonneeB & ) const [pure virtual]
```

produit contracte avec un vecteur

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.19 operator\*() [2/4]**

```
virtual TenseurHH & TenseurHH::operator* (
    const double & ) const [pure virtual]
```

operations \* double

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.20 operator\*() [3/4]**

```
virtual TenseurHB & TenseurHH::operator* (
    const TenseurBB & ) const [pure virtual]
```

idem avec BB

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.21 operator\*() [4/4]**

```
virtual TenseurHH & TenseurHH::operator* (
    const TenseurBH & ) const [pure virtual]
```

produit contracte contracté une fois  $A(i,j)*B(j,k)=A.B \rightarrow$  donc c'est l'indice du milieu qui est contracté avec BH

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.22 operator\*=( )**

```
virtual void TenseurHH::operator*= (
    const double & ) [pure virtual]
```

operations \*= double

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.23 operator+()**

```
virtual TenseurHH & TenseurHH::operator+ (
    const TenseurHH & ) const [pure virtual]
```

operations +

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.24 operator+=()**

```
virtual void TenseurHH::operator+= (
    const TenseurHH & ) [pure virtual]
```

operations +=

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.25 operator-() [1/2]**

```
virtual TenseurHH & TenseurHH::operator- ( ) const [pure virtual]
```

operations -

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.26 operator-() [2/2]**

```
virtual TenseurHH & TenseurHH::operator- (
    const TenseurHH & ) const [pure virtual]
```

operations - tens

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.27 operator-=()**

```
virtual void TenseurHH::operator-= (
    const TenseurHH & ) [pure virtual]
```

operations -=

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.28 operator/()**

```
virtual TenseurHH & TenseurHH::operator/ (
    const double & ) const [pure virtual]
```

operations /

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.29 operator/=()**

```
virtual void TenseurHH::operator/= (
    const double & ) [pure virtual]
```

operations /=

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.30 operator=()**

```
virtual TenseurHH & TenseurHH::operator= (
    const TenseurHH & ) [pure virtual]
```

operations =

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

**6.840.2.31 operator==(())**

```
virtual int TenseurHH::operator==(
    const TenseurHH & ) const [pure virtual]
```

test

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

### 6.840.2.32 Transpose()

```
virtual TenseurHH & TenseurHH::Transpose ( ) const [pure virtual]
```

ATTENTION creation d'un tenseur transpose qui est supprime par Libere.

Implémenté dans [Tenseur1HH](#), [Tenseur2HH](#), [Tenseur\\_ns2HH](#), [Tenseur3HH](#), et [Tenseur\\_ns3HH](#).

### 6.840.2.33 Var\_tenseur\_dans\_nouvelle\_base()

```
void TenseurHH::Var_tenseur_dans_nouvelle_base (
    const Mat_pleine & gamma,
    TenseurHH & var_tensHH,
    const Mat_pleine & var_gamma )
```

il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base

il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base  
connaissant sa variation dans la base actuelle

this : le tenseur

var\_tensBB : en entrée: la variation du tenseur dans la base initiale qu'on appelle g\_i

var\_tensHH : en sortie: la variation du tenseur dans la base finale qu'on appelle gp\_i

gamma : en entrée gpH(i) = gamma(i,j) \* gH(j)

var\_gamma : en entrée : la variation de gamma

$[A_p^{kl}] = [\gamma] * [A_{ij}] * [\gamma]^T$

La documentation de cette classe a été générée à partir du fichier suivant :

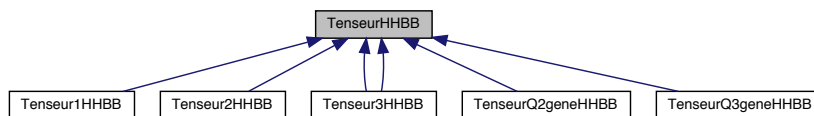
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur.h

## 6.841 Référence de la classe TenseurHHBB

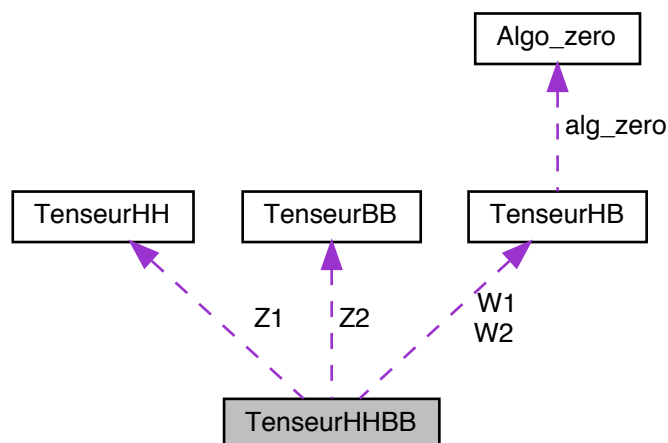
cas des composantes mixte HHBB

```
#include <TenseurQ.h>
```

Grphe d'héritage de TenseurHHBB:



Graphe de collaboration de TenseurHHBB:



## Fonctions membres publiques

- **TenseurHHBB** (const [TenseurHH](#) \*Zi1, const [TenseurBB](#) \*Zi2, const [TenseurHB](#) \*Wi1, const [TenseurHB](#) \*Wi2)
- **TenseurHHBB** (const [TenseurHHBB](#) &)
- void **Libere** (int cas)
- int **Dimension** () const
- [TenseurHHBB](#) **operator+** (const [TenseurHHBB](#) &) const
- void **operator+=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) **operator-** () const
- [TenseurHHBB](#) **operator-** (const [TenseurHHBB](#) &) const
- void **operator-=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurHHBB](#) **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurHH](#) &) const
- bool **operator==** (const [TenseurHHBB](#) &) const
- bool **operator!=** (const [TenseurHHBB](#) &) const
- **operator** [TenseurHHBB](#) & (void)
- void **change** (int i, int j, int k, int l, double &val)
- double **operator()** (int i, int j, int k, int l) const
- const [TenseurHH](#) & **Z\_1** () const
- const [TenseurBB](#) & **Z\_2** () const
- const [TenseurHB](#) & **W\_1** () const
- const [TenseurHB](#) & **W\_2** () const
- const [TenseurHH](#) \* **Z\_1P** () const
- const [TenseurBB](#) \* **Z\_2P** () const
- const [TenseurHB](#) \* **W\_1P** () const
- const [TenseurHB](#) \* **W\_2P** () const
- int **Dimension** () const
- virtual void **Inita** (double val)=0
- virtual [TenseurHHBB](#) & **operator+** (const [TenseurHHBB](#) &) const =0
- virtual void **operator+=** (const [TenseurHHBB](#) &)=0
- virtual [TenseurHHBB](#) & **operator-** () const =0
- virtual [TenseurHHBB](#) & **operator-** (const [TenseurHHBB](#) &) const =0

- virtual void **operator=** (const [TenseurHHBB](#) &)=0
- virtual [TenseurHHBB](#) & **operator=** (const [TenseurHHBB](#) &)=0
- virtual [TenseurHHBB](#) & **operator\*** (const double &) const =0
- virtual void **operator\*=** (const double &)=0
- virtual [TenseurHHBB](#) & **operator/** (const double &) const =0
- virtual void **operator/=** (const double &)=0
- virtual [TenseurHH](#) & **operator&&** (const [TenseurHH](#) &) const =0
- virtual [TenseurBBHH](#) & **Transpose1et2avec3et4** () const =0
- virtual void **Affectation\_trans\_dimension** (const [TenseurHHBB](#) &B, bool plusZero)=0
- virtual int **operator==** (const [TenseurHHBB](#) &) const =0
- int **operator!=** (const [TenseurHHBB](#) &a) const
- virtual void **Change** (int i, int j, int k, int l, const double &val)=0
- virtual void **ChangePlus** (int i, int j, int k, int l, const double &val)=0
- virtual double **operator()** (int i, int j, int k, int l) const =0
- virtual double **MaxiComposante** () const =0
- virtual istream & **Lecture** (istream &entree)=0
- virtual ostream & **Ecriture** (ostream &sort) const =0

### Fonctions membres publiques statiques

- static [TenseurHHBB](#) **Prod\_tensoriel** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)
- static [TenseurHHBB](#) **Prod\_tensoriel\_barre** (const [TenseurHB](#) &aHB, const [TenseurHB](#) &bHB)

### Attributs publics

- int **dimension**
- double \* **t**

### Fonctions membres protégées

- void **Message** (int dim, string mes) const
- void **Message** (int dim, string mes) const
- virtual [TenseurBB](#) & **Prod\_gauche** (const [TenseurBB](#) &F) const =0

### Attributs protégés

- [TenseurHH](#) \* **Z1**
- [TenseurBB](#) \* **Z2**
- [TenseurHB](#) \* **W1**
- [TenseurHB](#) \* **W2**

### Amis

- [TenseurHHBB](#) & **operator\*** (double r, const [TenseurHHBB](#) &t)
- [TenseurBB](#) & **operator&&** (const [TenseurBB](#) &F, const [TenseurHHBB](#) &T)
- istream & **operator>>** (istream &, [TenseurHHBB](#) &)
- ostream & **operator<<** (ostream &, const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator\*** (double r, const [TenseurHHBB](#) &t)
- [TenseurBB](#) & **operator&&** (const [TenseurBB](#) &F, const [TenseurHHBB](#) &T)

## 6.841.1 Description détaillée

cas des composantes mixte HHBB

Auteur

Gérard Rio

Version

1.0

Date

2/5/2002

La documentation de cette classe a été générée à partir du fichier suivant :

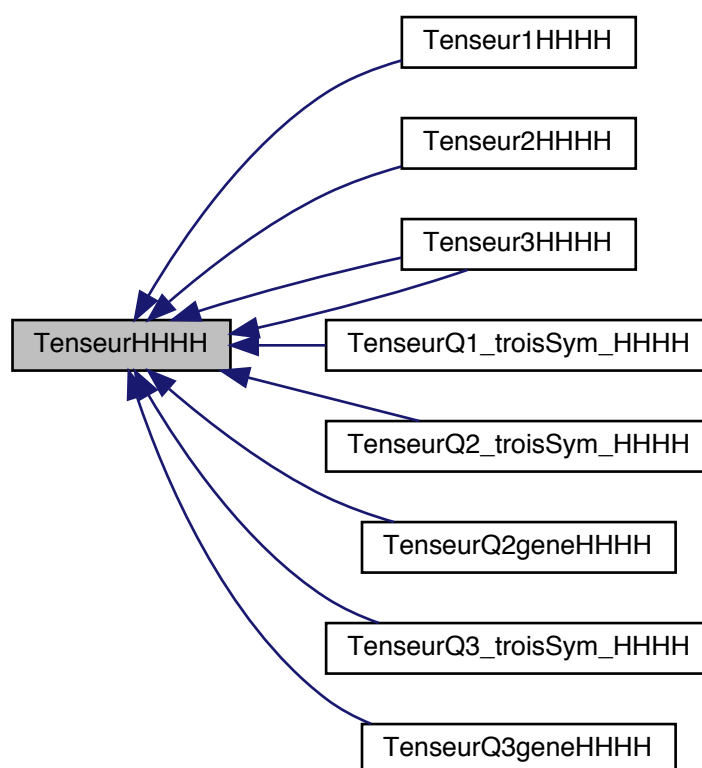
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurO4.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ.h

## 6.842 Référence de la classe TenseurHHHH

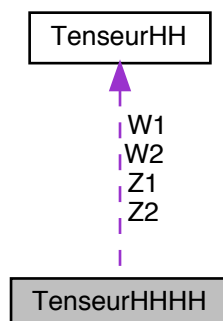
cas des composantes 4 fois contravariantes

```
#include <TenseurQ.h>
```

Graphe d'héritage de TenseurHHHH:



Grphe de collaboration de TenseurHHHH:



## Fonctions membres publiques

- **TenseurHHHH** (const [TenseurHH](#) \*Zi1, const [TenseurHH](#) \*Zi2, const [TenseurHH](#) \*Wi1, const [TenseurHH](#) \*Wi2)
- **TenseurHHHH** (const [TenseurHHHH](#) &)
- void **Libere** (int cas)
- int **Dimension** () const
- [TenseurHHHH](#) **operator+** (const [TenseurHHHH](#) &) const
- void **operator+=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) **operator-** () const
- [TenseurHHHH](#) **operator-** (const [TenseurHHHH](#) &) const
- void **operator-=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) **operator\*** (const double &) const
- void **operator\*:=** (const double &)
- [TenseurHHHH](#) **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &) const
- bool **operator==** (const [TenseurHHHH](#) &) const
- bool **operator!=** (const [TenseurHHHH](#) &) const
- **operator** [TenseurHHHH](#) & (void)
- void **change** (int i, int j, int k, int l, double &val)
- double **operator()** (int i, int j, int k, int l) const
- const [TenseurHH](#) & **Z\_1** () const
- const [TenseurHH](#) & **Z\_2** () const
- const [TenseurHH](#) & **W\_1** () const
- const [TenseurHH](#) & **W\_2** () const
- const [TenseurHH](#) \* **Z\_1P** () const
- const [TenseurHH](#) \* **Z\_2P** () const
- const [TenseurHH](#) \* **W\_1P** () const
- const [TenseurHH](#) \* **W\_2P** () const
- int **Dimension** () const
- virtual void **Inita** (double val)=0
- virtual [TenseurHHHH](#) & **operator+** (const [TenseurHHHH](#) &) const =0
- virtual void **operator+=** (const [TenseurHHHH](#) &)=0
- virtual [TenseurHHHH](#) & **operator-** () const =0
- virtual [TenseurHHHH](#) & **operator-** (const [TenseurHHHH](#) &) const =0
- virtual void **operator-=** (const [TenseurHHHH](#) &)=0
- virtual [TenseurHHHH](#) & **operator=** (const [TenseurHHHH](#) &)=0
- virtual [TenseurHHHH](#) & **operator\*** (const double &) const =0
- virtual void **operator\*:=** (const double &)=0
- virtual [TenseurHHHH](#) & **operator/** (const double &) const =0



- virtual void **operator/=** (const double &)=0
- virtual [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &) const =0
- virtual [TenseurHHHH](#) & **Transpose1et2avec3et4** () const =0
- virtual void **Affectation\_trans\_dimension** (const [TenseurHHHH](#) &B, bool plusZero)=0
- virtual int **operator==** (const [TenseurHHHH](#) &) const =0
- int **operator!=** (const [TenseurHHHH](#) &a) const
- virtual void **Change** (int i, int j, int k, int l, const double &val)=0
- virtual void **ChangePlus** (int i, int j, int k, int l, const double &val)=0
- virtual double **operator()** (int i, int j, int k, int l) const =0
- virtual double **MaxiComposante** () const =0
- virtual istream & **Lecture** (istream &entree)=0
- virtual ostream & **Ecriture** (ostream &sort) const =0

### Fonctions membres publiques statiques

- static [TenseurHHHH](#) **Prod\_tensoriel** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) **Prod\_tensoriel\_barre** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)

### Attributs publics

- int **dimension**
- double \* **t**

### Fonctions membres protégées

- void **Message** (int dim, string mes) const
- void **Message** (int dim, string mes) const
- virtual [TenseurHH](#) & **Prod\_gauche** (const [TenseurBB](#) &F) const =0

### Attributs protégés

- [TenseurHH](#) \* **Z1**
- [TenseurHH](#) \* **Z2**
- [TenseurHH](#) \* **W1**
- [TenseurHH](#) \* **W2**

### Amis

- [TenseurHHHH](#) & **operator\*** (double r, const [TenseurHHHH](#) &t)
- [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &F, const [TenseurHHHH](#) &T)
- istream & **operator>>** (istream &, [TenseurHHHH](#) &)
- ostream & **operator<<** (ostream &, const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator\*** (double r, const [TenseurHHHH](#) &t)
- [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &F, const [TenseurHHHH](#) &T)

## 6.842.1 Description détaillée

cas des composantes 4 fois contravariantes

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

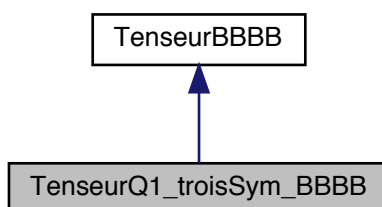
2/5/2002

La documentation de cette classe a été générée à partir du fichier suivant :

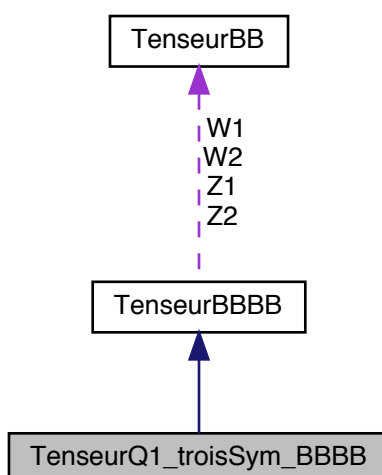
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurO4.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/[TenseurQ.h](#)

## 6.843 Référence de la classe TenseurQ1\_troisSym\_BBBB

Graphe d'héritage de TenseurQ1\_troisSym\_BBBB:



Graphe de collaboration de TenseurQ1\_troisSym\_BBBB:



### Fonctions membres publiques

- `TenseurQ1_troisSym_BBBB` (const double &x1111)
- `TenseurQ1_troisSym_BBBB` (const `TenseurBBBB` &)
- `TenseurQ1_troisSym_BBBB` (const `TenseurQ1_troisSym_BBBB` &)
- void `Inita` (double val)
- `TenseurBBBB` & `operator+` (const `TenseurBBBB` &) const
- void `operator+=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator-` () const
- `TenseurBBBB` & `operator-` (const `TenseurBBBB` &) const
- void `operator-=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator*` (const double &) const
- void `operator*+=` (const double &)
- `TenseurBBBB` & `operator/` (const double &) const

- void `operator/=` (const double &)
- `TenseurBB` & `operator&&` (const `TenseurHH` &) const
- `TenseurBBHH` & `operator&&` (const `TenseurHHHH` &) const
- `TenseurBBBB` & `operator&&` (const `TenseurHHBB` &) const
- `TenseurBBBB` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurBBBB` &B, bool plusZero)
- int `operator==` (const `TenseurBBBB` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres protégées

- `TenseurBB` & `Prod_gauche` (const `TenseurHH` &F) const
- `TenseurHHBB` & `Prod_gauche` (const `TenseurHHHH` &) const
- `TenseurBBBB` & `Prod_gauche` (const `TenseurBBHH` &) const

## Attributs protégés

- listdouble1Iter `ipointe`

## Amis

- istream & `operator>>` (istream &, `TenseurQ1_troisSym_BBBB` &)
- ostream & `operator<<` (ostream &, const `TenseurQ1_troisSym_BBBB` &)

## Membres hérités additionnels

### 6.843.1 Documentation des fonctions membres

#### 6.843.1.1 `Affectation_trans_dimension()`

```
void TenseurQ1_troisSym_BBBB::Affectation_trans_dimension (
    const TenseurBBBB & B,
    bool plusZero ) [virtual]
```

Implémente `TenseurBBBB`.

#### 6.843.1.2 `Change()`

```
void TenseurQ1_troisSym_BBBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente `TenseurBBBB`.

#### 6.843.1.3 `ChangePlus()`

```
void TenseurQ1_troisSym_BBBB::ChangePlus (
    int i,
    int j,
    int k,
```

```
int l,  
const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.843.1.4 Ecriture()

```
ostream & TenseurQ1_troisSym_BBBB::Ecriture (  
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.843.1.5 Inita()

```
void TenseurQ1_troisSym_BBBB::Inita (  
    double val ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.843.1.6 Lecture()

```
istream & TenseurQ1_troisSym_BBBB::Lecture (  
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.843.1.7 MaxiComposante()

```
double TenseurQ1_troisSym_BBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.843.1.8 operator&&()

```
TenseurBB & TenseurQ1_troisSym_BBBB::operator&& (  
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.843.1.9 operator>()()

```
double TenseurQ1_troisSym_BBBB::operator() (  
    int i,  
    int j,  
    int k,  
    int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.843.1.10 operator\*()

```
TenseurBBBB & TenseurQ1_troisSym_BBBB::operator* (  
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.11 operator\*=( )**

```
void TenseurQ1_troisSym_BBBB::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.12 operator+( )**

```
TenseurBBBB & TenseurQ1_troisSym_BBBB::operator+ (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.13 operator+=( )**

```
void TenseurQ1_troisSym_BBBB::operator+= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.14 operator-( ) [1/2]**

```
TenseurBBBB & TenseurQ1_troisSym_BBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.15 operator-( ) [2/2]**

```
TenseurBBBB & TenseurQ1_troisSym_BBBB::operator- (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.16 operator==( )**

```
void TenseurQ1_troisSym_BBBB::operator==(
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.17 operator/( )**

```
TenseurBBBB & TenseurQ1_troisSym_BBBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.18 operator/=( )**

```
void TenseurQ1_troisSym_BBBB::operator/=(
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.843.1.19 operator=( )**

```
TenseurBBBB & TenseurQ1_troisSym_BBBB::operator= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.843.1.20 operator==()

```
int TenseurQ1_troisSym_BBBB::operator== (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.843.1.21 Prod\_gauche()

```
TenseurBB & TenseurQ1_troisSym_BBBB::Prod_gauche (
    const TenseurHH & F ) const [inline], [protected], [virtual]
```

Implémente [TenseurBBBB](#).

### 6.843.1.22 Transpose1et2avec3et4()

```
TenseurBBBB & TenseurQ1_troisSym_BBBB::Transpose1et2avec3et4 ( ) const [virtual]
```

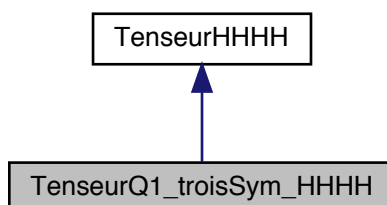
Implémente [TenseurBBBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

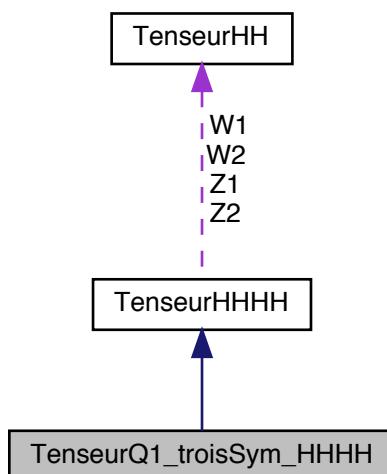
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur1\_TroisSym.h

## 6.844 Référence de la classe TenseurQ1\_troisSym\_HHHH

Grappe d'héritage de TenseurQ1\_troisSym\_HHHH:



Graph de collaboration de TenseurQ1\_troisSym\_HHHH:



## Fonctions membres publiques

- [TenseurQ1\\_troisSym\\_HHHH](#) (const double &x1111)
- [TenseurQ1\\_troisSym\\_HHHH](#) (const [TenseurHHHH](#) &)
- [TenseurQ1\\_troisSym\\_HHHH](#) (const [TenseurQ1\\_troisSym\\_HHHH](#) &)
- void [Inita](#) (double val)
- [TenseurHHHH](#) & [operator+](#) (const [TenseurHHHH](#) &) const
- void [operator+=](#) (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & [operator-](#) () const
- [TenseurHHHH](#) & [operator-](#) (const [TenseurHHHH](#) &) const
- void [operator-=](#) (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & [operator=](#) (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & [operator\\*](#) (const double &) const
- void [operator\\*=](#) (const double &)
- [TenseurHHHH](#) & [operator/](#) (const double &) const
- void [operator/=](#) (const double &)
- [TenseurHH](#) & [operator&&](#) (const [TenseurBB](#) &) const
- [TenseurHHHH](#) & [operator&&](#) (const [TenseurBBHH](#) &) const
- [TenseurHHBB](#) & [operator&&](#) (const [TenseurBBBB](#) &) const
- [TenseurHHHH](#) & [Transpose1et2avec3et4](#) () const
- void [Affectation\\_trans\\_dimension](#) (const [TenseurHHHH](#) &B, bool plusZero)
- int [operator==](#) (const [TenseurHHHH](#) &) const
- void [Change](#) (int i, int j, int k, int l, const double &val)
- void [ChangePlus](#) (int i, int j, int k, int l, const double &val)
- double [operator\(\)](#) (int i, int j, int k, int l) const
- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort) const

## Fonctions membres protégées

- [TenseurHH](#) & [Prod\\_gauche](#) (const [TenseurBB](#) &F) const
- [TenseurBBHH](#) & [Prod\\_gauche](#) (const [TenseurBBBB](#) &) const
- [TenseurHHHH](#) & [Prod\\_gauche](#) (const [TenseurHHBB](#) &) const

## Attributs protégés

- listdoubleIter **ipointe**

## Amis

- istream & **operator**>> (istream &, [TenseurQ1\\_troisSym\\_HHHH](#) &)
- ostream & **operator**<< (ostream &, const [TenseurQ1\\_troisSym\\_HHHH](#) &)

## Membres hérités additionnels

### 6.844.1 Documentation des fonctions membres

#### 6.844.1.1 Affectation\_trans\_dimension()

```
void TenseurQ1_troisSym_HHHH::Affectation_trans_dimension (
    const TenseurHHHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.844.1.2 Change()

```
void TenseurQ1_troisSym_HHHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.844.1.3 ChangePlus()

```
void TenseurQ1_troisSym_HHHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.844.1.4 Ecriture()

```
ostream & TenseurQ1_troisSym_HHHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.844.1.5 Inita()

```
void TenseurQ1_troisSym_HHHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHHH](#).



### 6.844.1.6 Lecture()

```
istream & TenseurQ1_troisSym_HHHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.844.1.7 MaxiComposante()

```
double TenseurQ1_troisSym_HHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.844.1.8 operator&&()

```
TenseurHH & TenseurQ1_troisSym_HHHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.844.1.9 operator()()

```
double TenseurQ1_troisSym_HHHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.844.1.10 operator\*()

```
TenseurHHHH & TenseurQ1_troisSym_HHHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.844.1.11 operator\*=( )

```
void TenseurQ1_troisSym_HHHH::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.844.1.12 operator+()

```
TenseurHHHH & TenseurQ1_troisSym_HHHH::operator+ (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.844.1.13 operator+=( )

```
void TenseurQ1_troisSym_HHHH::operator+= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.844.1.14 operator-() [1/2]**

`TenseurHHHH & TenseurQ1_troisSym_HHHH::operator- ( ) const [virtual]`  
Implémente [TenseurHHHH](#).

**6.844.1.15 operator-() [2/2]**

`TenseurHHHH & TenseurQ1_troisSym_HHHH::operator- ( const TenseurHHHH & ) const [virtual]`  
Implémente [TenseurHHHH](#).

**6.844.1.16 operator==( )**

`void TenseurQ1_troisSym_HHHH::operator==( const TenseurHHHH & ) [virtual]`  
Implémente [TenseurHHHH](#).

**6.844.1.17 operator/()**

`TenseurHHHH & TenseurQ1_troisSym_HHHH::operator/ ( const double & ) const [virtual]`  
Implémente [TenseurHHHH](#).

**6.844.1.18 operator/=()**

`void TenseurQ1_troisSym_HHHH::operator/= ( const double & ) [virtual]`  
Implémente [TenseurHHHH](#).

**6.844.1.19 operator=( )**

`TenseurHHHH & TenseurQ1_troisSym_HHHH::operator= ( const TenseurHHHH & ) [virtual]`  
Implémente [TenseurHHHH](#).

**6.844.1.20 operator==( )**

`int TenseurQ1_troisSym_HHHH::operator==( const TenseurHHHH & ) const [virtual]`  
Implémente [TenseurHHHH](#).

**6.844.1.21 Prod\_gauche()**

`TenseurHH & TenseurQ1_troisSym_HHHH::Prod_gauche ( const TenseurBB & F ) const [inline], [protected], [virtual]`  
Implémente [TenseurHHHH](#).

**6.844.1.22 Transpose1et2avec3et4()**

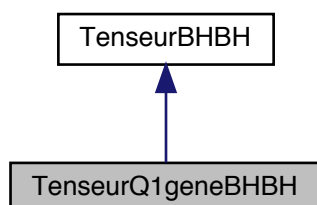
`TenseurHHHH & TenseurQ1_troisSym_HHHH::Transpose1et2avec3et4 ( ) const [virtual]`  
Implémente [TenseurHHHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

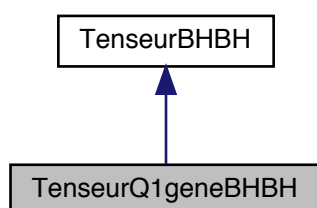
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur1\_TroisSym.h

## 6.845 Référence de la classe TenseurQ1geneBHBH

Graphe d'héritage de TenseurQ1geneBHBH:



Graphe de collaboration de TenseurQ1geneBHBH:



### Fonctions membres publiques

- **TenseurQ1geneBHBH** (const double &x1111)
- **TenseurQ1geneBHBH** (const [TenseurBHBH](#) &)
- **TenseurQ1geneBHBH** (const [TenseurQ1geneBHBH](#) &)
- void **Inita** (double val)
- [TenseurBHBH](#) & **operator+** (const [TenseurBHBH](#) &) const
- void **operator+=** (const [TenseurBHBH](#) &)
- [TenseurBHBH](#) & **operator-** () const
- [TenseurBHBH](#) & **operator-** (const [TenseurBHBH](#) &) const
- void **operator-=** (const [TenseurBHBH](#) &)
- [TenseurBHBH](#) & **operator=** (const [TenseurBHBH](#) &)
- [TenseurBHBH](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurBHBH](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBH](#) & **operator&&** (const [TenseurBH](#) &) const
- [TenseurHBHB](#) & **Transpose1et2avec3et4** () const
- virtual void **Affectation\_trans\_dimension** (const [TenseurBHBH](#) &B, bool plusZero)
- int **operator==** (const [TenseurBHBH](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)

- void [ChangePlus](#) (int i, int j, int k, int l, const double &val)
- double [operator\(\)](#) (int i, int j, int k, int l) const
- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort) const

### Fonctions membres protégées

- [TenseurBH](#) & [Prod\\_gauche](#) (const [TenseurBH](#) &F) const

### Attributs protégés

- listdouble1Iter [ipointe](#)

### Amis

- istream & [operator](#)>> (istream &, [TenseurQ1geneBHBH](#) &)
- ostream & [operator](#)<< (ostream &, const [TenseurQ1geneBHBH](#) &)

### Membres hérités additionnels

#### 6.845.1 Documentation des fonctions membres

##### 6.845.1.1 Affectation\_trans\_dimension()

```
virtual void TenseurQ1geneBHBH::Affectation_trans_dimension (
    const TenseurBHBH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBHBH](#).

##### 6.845.1.2 Change()

```
void TenseurQ1geneBHBH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBHBH](#).

##### 6.845.1.3 ChangePlus()

```
void TenseurQ1geneBHBH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBHBH](#).

##### 6.845.1.4 Ecriture()

```
ostream & TenseurQ1geneBHBH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.5 Inita()

```
void TenseurQ1geneBHBH::Inita (
    double val ) [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.6 Lecture()

```
istream & TenseurQ1geneBHBH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.7 MaxiComposante()

```
double TenseurQ1geneBHBH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.8 operator&&()

```
TenseurBH & TenseurQ1geneBHBH::operator&& (
    const TenseurBH & ) const [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.9 operator()()

```
double TenseurQ1geneBHBH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.10 operator\*()

```
TenseurBHBH & TenseurQ1geneBHBH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.11 operator\*=( )

```
void TenseurQ1geneBHBH::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.12 operator+()

```
TenseurBHBH & TenseurQ1geneBHBH::operator+ (
    const TenseurBHBH & ) const [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.13 operator+=()**

```
void TenseurQlgeneBHBH::operator+= (
    const TenseurBHBH & ) [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.14 operator-() [1/2]**

```
TenseurBHBH & TenseurQlgeneBHBH::operator- ( ) const [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.15 operator-() [2/2]**

```
TenseurBHBH & TenseurQlgeneBHBH::operator- (
    const TenseurBHBH & ) const [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.16 operator-=()**

```
void TenseurQlgeneBHBH::operator-= (
    const TenseurBHBH & ) [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.17 operator/()**

```
TenseurBHBH & TenseurQlgeneBHBH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.18 operator/=()**

```
void TenseurQlgeneBHBH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.19 operator=()**

```
TenseurBHBH & TenseurQlgeneBHBH::operator= (
    const TenseurBHBH & ) [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.20 operator==()**

```
int TenseurQlgeneBHBH::operator== (
    const TenseurBHBH & ) const [virtual]
```

Implémente [TenseurBHBH](#).

**6.845.1.21 Prod\_gauche()**

```
TenseurBH & TenseurQlgeneBHBH::Prod_gauche (
    const TenseurBH & F ) const [protected], [virtual]
```

Implémente [TenseurBHBH](#).

### 6.845.1.22 Transpose1et2avec3et4()

`TenseurBHHB` & `TenseurQ1geneBHHB::Transpose1et2avec3et4 ( ) const` [virtual]

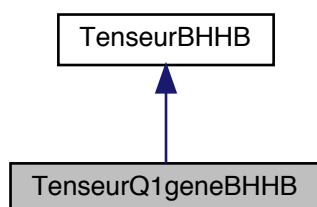
Implémente `TenseurBHHB`.

La documentation de cette classe a été générée à partir du fichier suivant :

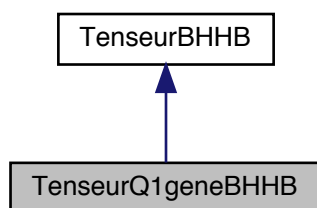
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ1gene.h

## 6.846 Référence de la classe TenseurQ1geneBHHB

Grappe d'héritage de `TenseurQ1geneBHHB`:



Grappe de collaboration de `TenseurQ1geneBHHB`:



### Fonctions membres publiques

- `TenseurQ1geneBHHB` (const double &x1111)
- `TenseurQ1geneBHHB` (const `TenseurBHHB` &)
- `TenseurQ1geneBHHB` (const `TenseurQ1geneBHHB` &)
- void `Inita` (double val)
- `TenseurBHHB` & `operator+` (const `TenseurBHHB` &) const
- void `operator+=` (const `TenseurBHHB` &)
- `TenseurBHHB` & `operator-` () const
- `TenseurBHHB` & `operator-` (const `TenseurBHHB` &) const
- void `operator-=` (const `TenseurBHHB` &)
- `TenseurBHHB` & `operator=` (const `TenseurBHHB` &)
- `TenseurBHHB` & `operator*` (const double &) const
- void `operator*=` (const double &)

- [TenseurBHHB](#) & [operator/](#) (const double &) const
- void [operator/=](#) (const double &)
- [TenseurBH](#) & [operator&&](#) (const [TenseurHB](#) &) const
- [TenseurHBBH](#) & [Transpose1et2avec3et4](#) () const
- virtual void [Affectation\\_trans\\_dimension](#) (const [TenseurBHHB](#) &B, bool plusZero)
- int [operator==](#) (const [TenseurBHHB](#) &) const
- void [Change](#) (int i, int j, int k, int l, const double &val)
- void [ChangePlus](#) (int i, int j, int k, int l, const double &val)
- double [operator\(\)](#) (int i, int j, int k, int l) const
- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort) const

## Fonctions membres protégées

- [TenseurHB](#) & [Prod\\_gauche](#) (const [TenseurBH](#) &F) const

## Attributs protégés

- listdouble1Iter [ipointe](#)

## Amis

- istream & [operator>>](#) (istream &, [TenseurQ1geneBHHB](#) &)
- ostream & [operator<<](#) (ostream &, const [TenseurQ1geneBHHB](#) &)

## Membres hérités additionnels

### 6.846.1 Documentation des fonctions membres

#### 6.846.1.1 [Affectation\\_trans\\_dimension\(\)](#)

```
virtual void TenseurQ1geneBHHB::Affectation_trans_dimension (
    const TenseurBHHB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBHHB](#).

#### 6.846.1.2 [Change\(\)](#)

```
void TenseurQ1geneBHHB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBHHB](#).

#### 6.846.1.3 [ChangePlus\(\)](#)

```
void TenseurQ1geneBHHB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBHHB](#).



**6.846.1.4 Ecriture()**

```
ostream & TenseurQ1geneBHHB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.5 Inita()**

```
void TenseurQ1geneBHHB::Inita (
    double val ) [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.6 Lecture()**

```
istream & TenseurQ1geneBHHB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.7 MaxiComposante()**

```
double TenseurQ1geneBHHB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.8 operator&&()**

```
TenseurBH & TenseurQ1geneBHHB::operator&& (
    const TenseurHB & ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.9 operator()()**

```
double TenseurQ1geneBHHB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.10 operator\*()**

```
TenseurBHHB & TenseurQ1geneBHHB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.11 operator\*=( )**

```
void TenseurQ1geneBHHB::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.12 operator+()**

```
TenseurBHHB & TenseurQlgeneBHHB::operator+ (
    const TenseurBHHB & ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.13 operator+=()**

```
void TenseurQlgeneBHHB::operator+= (
    const TenseurBHHB & ) [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.14 operator-() [1/2]**

```
TenseurBHHB & TenseurQlgeneBHHB::operator- ( ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.15 operator-() [2/2]**

```
TenseurBHHB & TenseurQlgeneBHHB::operator- (
    const TenseurBHHB & ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.16 operator-=()**

```
void TenseurQlgeneBHHB::operator-= (
    const TenseurBHHB & ) [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.17 operator/()**

```
TenseurBHHB & TenseurQlgeneBHHB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.18 operator/=()**

```
void TenseurQlgeneBHHB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.19 operator=()**

```
TenseurBHHB & TenseurQlgeneBHHB::operator= (
    const TenseurBHHB & ) [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.20 operator==(())**

```
int TenseurQlgeneBHHB::operator==(
    const TenseurBHHB & ) const [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.21 Prod\_gauche()**

```
TenseurHB & TenseurQ1geneBHHB::Prod_gauche (
    const TenseurBH & F ) const [protected], [virtual]
```

Implémente [TenseurBHHB](#).

**6.846.1.22 Transpose1et2avec3et4()**

```
TenseurHBBH & TenseurQ1geneBHHB::Transpose1et2avec3et4 ( ) const [virtual]
```

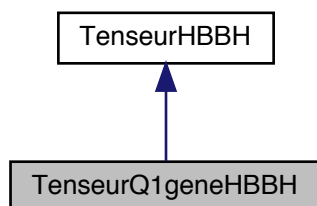
Implémente [TenseurBHHB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

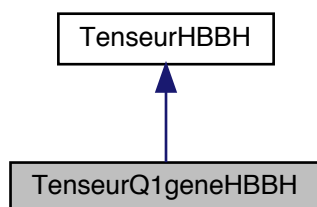
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ1gene.h

**6.847 Référence de la classe TenseurQ1geneHBBH**

Graphe d'héritage de TenseurQ1geneHBBH:



Graphe de collaboration de TenseurQ1geneHBBH:

**Fonctions membres publiques**

- [TenseurQ1geneHBBH](#) (const double &x1111)
- [TenseurQ1geneHBBH](#) (const [TenseurHBBH](#) &)
- [TenseurQ1geneHBBH](#) (const [TenseurQ1geneHBBH](#) &)
- void [Inita](#) (double val)

- [TenseurHBBH](#) & [operator+](#) (const [TenseurHBBH](#) &) const
- void [operator+=](#) (const [TenseurHBBH](#) &)
- [TenseurHBBH](#) & [operator-](#) () const
- [TenseurHBBH](#) & [operator-](#) (const [TenseurHBBH](#) &) const
- void [operator-=](#) (const [TenseurHBBH](#) &)
- [TenseurHBBH](#) & [operator=](#) (const [TenseurHBBH](#) &)
- [TenseurHBBH](#) & [operator\\*](#) (const double &) const
- void [operator\\*=](#) (const double &)
- [TenseurHBBH](#) & [operator/](#) (const double &) const
- void [operator/=](#) (const double &)
- [TenseurHB](#) & [operator&&](#) (const [TenseurBH](#) &) const
- [TenseurBH](#) & [Transpose1et2avec3et4](#) () const
- virtual void [Affectation\\_trans\\_dimension](#) (const [TenseurHBBH](#) &B, bool plusZero)
- int [operator==](#) (const [TenseurHBBH](#) &) const
- void [Change](#) (int i, int j, int k, int l, const double &val)
- void [ChangePlus](#) (int i, int j, int k, int l, const double &val)
- double [operator\(\)](#) (int i, int j, int k, int l) const
- double [MaxiComposante](#) () const
- istream & [Lecture](#) (istream &entree)
- ostream & [Ecriture](#) (ostream &sort) const

### Fonctions membres protégées

- [TenseurBH](#) & [Prod\\_gauche](#) (const [TenseurHB](#) &F) const

### Attributs protégés

- listdouble1Iter [ipointe](#)

### Amis

- istream & [operator>>](#) (istream &, [TenseurQ1geneHBBH](#) &)
- ostream & [operator<<](#) (ostream &, const [TenseurQ1geneHBBH](#) &)

### Membres hérités additionnels

#### 6.847.1 Documentation des fonctions membres

##### 6.847.1.1 [Affectation\\_trans\\_dimension\(\)](#)

```
virtual void TenseurQ1geneHBBH::Affectation_trans_dimension (
    const TenseurHBBH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHBBH](#).

##### 6.847.1.2 [Change\(\)](#)

```
void TenseurQ1geneHBBH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHBBH](#).

### 6.847.1.3 ChangePlus()

```
void TenseurQ1geneHBBH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHBBH](#).

### 6.847.1.4 Ecriture()

```
ostream & TenseurQ1geneHBBH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHBBH](#).

### 6.847.1.5 Inita()

```
void TenseurQ1geneHBBH::Inita (
    double val ) [virtual]
```

Implémente [TenseurHBBH](#).

### 6.847.1.6 Lecture()

```
istream & TenseurQ1geneHBBH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHBBH](#).

### 6.847.1.7 MaxiComposante()

```
double TenseurQ1geneHBBH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHBBH](#).

### 6.847.1.8 operator&&()

```
TenseurHB & TenseurQ1geneHBBH::operator&& (
    const TenseurBH & ) const [virtual]
```

Implémente [TenseurHBBH](#).

### 6.847.1.9 operator()()

```
double TenseurQ1geneHBBH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHBBH](#).

### 6.847.1.10 operator\*()

```
TenseurHBBH & TenseurQ1geneHBBH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.11 operator\*=( )**

```
void TenseurQlgeneHBBH::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.12 operator+( )**

```
TenseurHBBH & TenseurQlgeneHBBH::operator+ (
    const TenseurHBBH & ) const [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.13 operator+=( )**

```
void TenseurQlgeneHBBH::operator+= (
    const TenseurHBBH & ) [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.14 operator-( ) [1/2]**

```
TenseurHBBH & TenseurQlgeneHBBH::operator- ( ) const [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.15 operator-( ) [2/2]**

```
TenseurHBBH & TenseurQlgeneHBBH::operator- (
    const TenseurHBBH & ) const [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.16 operator-=( )**

```
void TenseurQlgeneHBBH::operator-=(
    const TenseurHBBH & ) [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.17 operator/( )**

```
TenseurHBBH & TenseurQlgeneHBBH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.18 operator/=( )**

```
void TenseurQlgeneHBBH::operator/=(
    const double & ) [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.19 operator=()**

```
TenseurHBBH & TenseurQ1geneHBBH::operator= (
    const TenseurHBBH & ) [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.20 operator==()**

```
int TenseurQ1geneHBBH::operator== (
    const TenseurHBBH & ) const [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.21 Prod\_gauche()**

```
TenseurBH & TenseurQ1geneHBBH::Prod_gauche (
    const TenseurHB & F ) const [protected], [virtual]
```

Implémente [TenseurHBBH](#).

**6.847.1.22 Transpose1et2avec3et4()**

```
TenseurBHHB & TenseurQ1geneHBBH::Transpose1et2avec3et4 ( ) const [virtual]
```

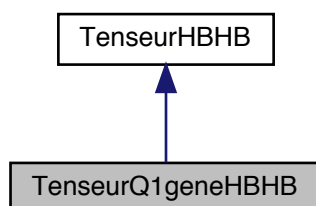
Implémente [TenseurHBBH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

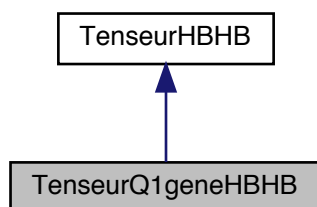
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ1gene.h

**6.848 Référence de la classe TenseurQ1geneHBHB**

Grappe d'héritage de TenseurQ1geneHBHB:



Graphe de collaboration de TenseurQ1geneHBHB:



## Fonctions membres publiques

- `TenseurQ1geneHBHB` (const double &x1111)
- `TenseurQ1geneHBHB` (const `TenseurHBHB` &)
- `TenseurQ1geneHBHB` (const `TenseurQ1geneHBHB` &)
- void `Inita` (double val)
- `TenseurHBHB` & `operator+` (const `TenseurHBHB` &) const
- void `operator+=` (const `TenseurHBHB` &)
- `TenseurHBHB` & `operator-` () const
- `TenseurHBHB` & `operator-` (const `TenseurHBHB` &) const
- void `operator-=` (const `TenseurHBHB` &)
- `TenseurHBHB` & `operator=` (const `TenseurHBHB` &)
- `TenseurHBHB` & `operator*` (const double &) const
- void `operator*=` (const double &)
- `TenseurHBHB` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurHB` & `operator&&` (const `TenseurHB` &) const
- `TenseurBHBH` & `Transpose1et2avec3et4` () const
- virtual void `Affectation_trans_dimension` (const `TenseurHBHB` &B, bool plusZero)
- int `operator==` (const `TenseurHBHB` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres protégées

- `TenseurHB` & `Prod_gauche` (const `TenseurHB` &F) const

## Attributs protégés

- listdouble1 lter `ipointe`

## Amis

- istream & `operator>>` (istream &, `TenseurQ1geneHBHB` &)
- ostream & `operator<<` (ostream &, const `TenseurQ1geneHBHB` &)

## Membres hérités additionnels

### 6.848.1 Documentation des fonctions membres



### 6.848.1.1 Affectation\_trans\_dimension()

```
virtual void TenseurQ1geneHBHB::Affectation_trans_dimension (
    const TenseurHBHB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.2 Change()

```
void TenseurQ1geneHBHB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.3 ChangePlus()

```
void TenseurQ1geneHBHB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.4 Ecriture()

```
ostream & TenseurQ1geneHBHB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.5 Inita()

```
void TenseurQ1geneHBHB::Inita (
    double val ) [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.6 Lecture()

```
istream & TenseurQ1geneHBHB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.7 MaxiComposante()

```
double TenseurQ1geneHBHB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.8 operator&&()

```
TenseurHB & TenseurQlgeneHBHB::operator&& (
    const TenseurHB & ) const [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.9 operator>()()

```
double TenseurQlgeneHBHB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.10 operator\*()

```
TenseurHBHB & TenseurQlgeneHBHB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.11 operator\*=( )

```
void TenseurQlgeneHBHB::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.12 operator+()

```
TenseurHBHB & TenseurQlgeneHBHB::operator+ (
    const TenseurHBHB & ) const [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.13 operator+=( )

```
void TenseurQlgeneHBHB::operator+= (
    const TenseurHBHB & ) [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.14 operator-() [1/2]

```
TenseurHBHB & TenseurQlgeneHBHB::operator- ( ) const [virtual]
```

Implémente [TenseurHBHB](#).

### 6.848.1.15 operator-() [2/2]

```
TenseurHBHB & TenseurQlgeneHBHB::operator- (
    const TenseurHBHB & ) const [virtual]
```

Implémente [TenseurHBHB](#).

**6.848.1.16 operator-=()**

```
void TenseurQ1geneHBHB::operator-= (
    const TenseurHBHB & ) [virtual]
```

Implémente [TenseurHBHB](#).

**6.848.1.17 operator/()**

```
TenseurHBHB & TenseurQ1geneHBHB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHBHB](#).

**6.848.1.18 operator/=()**

```
void TenseurQ1geneHBHB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHBHB](#).

**6.848.1.19 operator=()**

```
TenseurHBHB & TenseurQ1geneHBHB::operator= (
    const TenseurHBHB & ) [virtual]
```

Implémente [TenseurHBHB](#).

**6.848.1.20 operator==()**

```
int TenseurQ1geneHBHB::operator== (
    const TenseurHBHB & ) const [virtual]
```

Implémente [TenseurHBHB](#).

**6.848.1.21 Prod\_gauche()**

```
TenseurHB & TenseurQ1geneHBHB::Prod_gauche (
    const TenseurHB & F ) const [protected], [virtual]
```

Implémente [TenseurHBHB](#).

**6.848.1.22 Transpose1et2avec3et4()**

```
TenseurBHBH & TenseurQ1geneHBHB::Transpose1et2avec3et4 ( ) const [virtual]
```

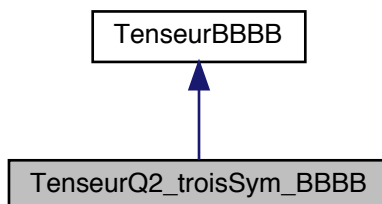
Implémente [TenseurHBHB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

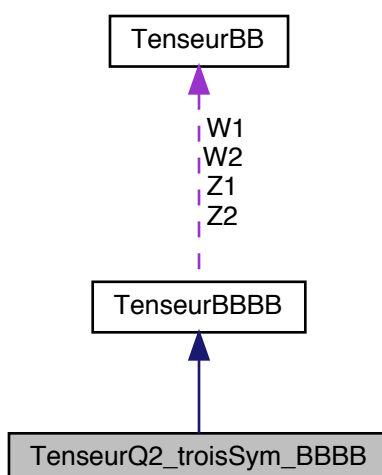
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ1gene.h

## 6.849 Référence de la classe TenseurQ2\_troisSym\_BBBB

Graphe d'héritage de TenseurQ2\_troisSym\_BBBB:



Graphe de collaboration de TenseurQ2\_troisSym\_BBBB:



### Fonctions membres publiques

- `TenseurQ2_troisSym_BBBB` (const double &val)
- `TenseurQ2_troisSym_BBBB` (const double &x1111, const double &x2222, const double &x1122, const double &x1212, const double &x1211, const double &x1222)
- `TenseurQ2_troisSym_BBBB` (const `TenseurBBBB` &)
- `TenseurQ2_troisSym_BBBB` (const `TenseurQ2_troisSym_BBBB` &)
- void `Inita` (double val)
- `TenseurBBBB` & `operator+` (const `TenseurBBBB` &) const
- void `operator+=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator-` () const
- `TenseurBBBB` & `operator-` (const `TenseurBBBB` &) const
- void `operator-=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator*` (const double &) const

- void `operator*=(const double &)`
- `TenseurBBBB & operator/(const double &) const`
- void `operator/=(const double &)`
- `TenseurBB & operator&&(const TenseurHH &) const`
- `TenseurBBHH & operator&&(const TenseurHHHH &) const`
- `TenseurBBBB & operator&&(const TenseurHHBB &) const`
- `TenseurBBBB & Transpose1et2avec3et4()` const
- void `Affectation_trans_dimension(const TenseurBBBB &B, bool plusZero)`
- int `operator==(const TenseurBBBB &) const`
- void `Change(int i, int j, int k, int l, const double &val)`
- void `ChangePlus(int i, int j, int k, int l, const double &val)`
- double `operator()(int i, int j, int k, int l) const`
- double `MaxiComposante()` const
- istream & `Lecture(istream &entree)`
- ostream & `Ecriture(ostream &sort) const`

### Fonctions membres protégées

- `TenseurBB & Prod_gauche(const TenseurHH &F) const`
- `TenseurHHBB & Prod_gauche(const TenseurHHHH &) const`
- `TenseurBBBB & Prod_gauche(const TenseurBBHH &) const`

### Attributs protégés

- listdouble6lter `ipointe`

### Amis

- istream & `operator>>(istream &, TenseurQ2_troisSym_BBBB &)`
- ostream & `operator<<(ostream &, const TenseurQ2_troisSym_BBBB &)`

### Membres hérités additionnels

#### 6.849.1 Documentation des fonctions membres

##### 6.849.1.1 Affectation\_trans\_dimension()

```
void TenseurQ2_troisSym_BBBB::Affectation_trans_dimension (
    const TenseurBBBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBBB](#).

##### 6.849.1.2 Change()

```
void TenseurQ2_troisSym_BBBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.3 ChangePlus()

```
void TenseurQ2_troisSym_BBBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.4 Ecriture()

```
ostream & TenseurQ2_troisSym_BBBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.5 Inita()

```
void TenseurQ2_troisSym_BBBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.6 Lecture()

```
istream & TenseurQ2_troisSym_BBBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.7 MaxiComposante()

```
double TenseurQ2_troisSym_BBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.8 operator&&()

```
TenseurBB & TenseurQ2_troisSym_BBBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.9 operator>()()

```
double TenseurQ2_troisSym_BBBB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.10 operator\*()

```
TenseurBBBB & TenseurQ2_troisSym_BBBB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.849.1.11 operator\*=( )**

```
void TenseurQ2_troisSym_BBBB::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.849.1.12 operator+( )**

```
TenseurBBBB & TenseurQ2_troisSym_BBBB::operator+ (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.849.1.13 operator+=( )**

```
void TenseurQ2_troisSym_BBBB::operator+= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.849.1.14 operator-( ) [1/2]**

```
TenseurBBBB & TenseurQ2_troisSym_BBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.849.1.15 operator-( ) [2/2]**

```
TenseurBBBB & TenseurQ2_troisSym_BBBB::operator- (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.849.1.16 operator--( )**

```
void TenseurQ2_troisSym_BBBB::operator--= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.849.1.17 operator/( )**

```
TenseurBBBB & TenseurQ2_troisSym_BBBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.849.1.18 operator/=( )**

```
void TenseurQ2_troisSym_BBBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.19 operator=()

```
TenseurBBBB & TenseurQ2_troisSym_BBBB::operator= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.20 operator==()

```
int TenseurQ2_troisSym_BBBB::operator== (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.21 Prod\_gauche()

```
TenseurBB & TenseurQ2_troisSym_BBBB::Prod_gauche (
    const TenseurHH & F ) const [inline], [protected], [virtual]
```

Implémente [TenseurBBBB](#).

### 6.849.1.22 Transpose1et2avec3et4()

```
TenseurBBBB & TenseurQ2_troisSym_BBBB::Transpose1et2avec3et4 ( ) const [virtual]
```

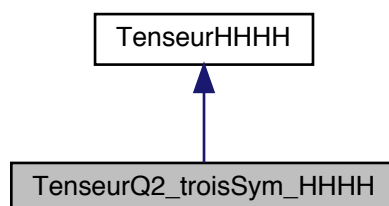
Implémente [TenseurBBBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2\_TroisSym.h

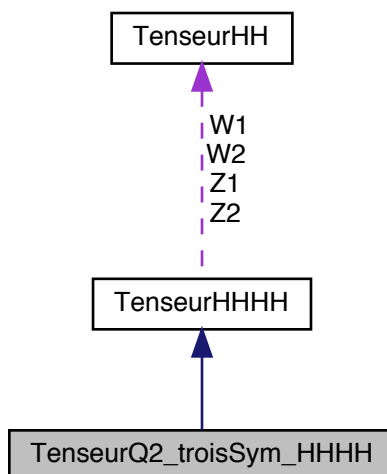
## 6.850 Référence de la classe TenseurQ2\_troisSym\_HHHH

Graphes d'héritage de TenseurQ2\_troisSym\_HHHH:





Graphe de collaboration de TenseurQ2\_troisSym\_HHHH:



## Fonctions membres publiques

- `TenseurQ2_troisSym_HHHH` (const double &val)
- `TenseurQ2_troisSym_HHHH` (const double &x1111, const double &x2222, const double &x1122, const double &x1212, const double &x1211, const double &x1222)
- `TenseurQ2_troisSym_HHHH` (const `TenseurHHHH` &)
- `TenseurQ2_troisSym_HHHH` (const `TenseurQ2_troisSym_HHHH` &)
- void `Inita` (double val)
- `TenseurHHHH` & `operator+` (const `TenseurHHHH` &) const
- void `operator+=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator-` () const
- `TenseurHHHH` & `operator-` (const `TenseurHHHH` &) const
- void `operator-=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator*` (const double &) const
- void `operator*=` (const double &)
- `TenseurHHHH` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurHH` & `operator&&` (const `TenseurBB` &) const
- `TenseurHHHH` & `operator&&` (const `TenseurBBHH` &) const
- `TenseurHHBB` & `operator&&` (const `TenseurBBBB` &) const
- `TenseurHHHH` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurHHHH` &B, bool plusZero)
- int `operator==` (const `TenseurHHHH` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres protégées

- `TenseurHH` & `Prod_gauche` (const `TenseurBB` &F) const
- `TenseurBBHH` & `Prod_gauche` (const `TenseurBBBB` &) const
- `TenseurHHHH` & `Prod_gauche` (const `TenseurHHBB` &) const

## Attributs protégés

- listdouble6lter **ipointe**

## Amis

- istream & **operator**>> (istream &, [TenseurQ2\\_troisSym\\_HHHH](#) &)
- ostream & **operator**<< (ostream &, const [TenseurQ2\\_troisSym\\_HHHH](#) &)

## Membres hérités additionnels

### 6.850.1 Documentation des fonctions membres

#### 6.850.1.1 Affectation\_trans\_dimension()

```
void TenseurQ2_troisSym_HHHH::Affectation_trans_dimension (
    const TenseurHHHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.850.1.2 Change()

```
void TenseurQ2_troisSym_HHHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.850.1.3 ChangePlus()

```
void TenseurQ2_troisSym_HHHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.850.1.4 Ecriture()

```
ostream & TenseurQ2_troisSym_HHHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.850.1.5 Inita()

```
void TenseurQ2_troisSym_HHHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.850.1.6 Lecture()

```
istream & TenseurQ2_troisSym_HHHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.850.1.7 MaxiComposante()

```
double TenseurQ2_troisSym_HHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.850.1.8 operator&&()

```
TenseurHH & TenseurQ2_troisSym_HHHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.850.1.9 operator()()

```
double TenseurQ2_troisSym_HHHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.850.1.10 operator\*()

```
TenseurHHHH & TenseurQ2_troisSym_HHHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.850.1.11 operator\*=( )

```
void TenseurQ2_troisSym_HHHH::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.850.1.12 operator+()

```
TenseurHHHH & TenseurQ2_troisSym_HHHH::operator+ (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.850.1.13 operator+=( )

```
void TenseurQ2_troisSym_HHHH::operator+= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.850.1.14 operator-() [1/2]**

`TenseurHHHH & TenseurQ2_troisSym_HHHH::operator- ( ) const [virtual]`  
Implémente [TenseurHHHH](#).

**6.850.1.15 operator-() [2/2]**

`TenseurHHHH & TenseurQ2_troisSym_HHHH::operator- ( const TenseurHHHH & ) const [virtual]`  
Implémente [TenseurHHHH](#).

**6.850.1.16 operator==( )**

`void TenseurQ2_troisSym_HHHH::operator==( const TenseurHHHH & ) [virtual]`  
Implémente [TenseurHHHH](#).

**6.850.1.17 operator/()**

`TenseurHHHH & TenseurQ2_troisSym_HHHH::operator/ ( const double & ) const [virtual]`  
Implémente [TenseurHHHH](#).

**6.850.1.18 operator/=( )**

`void TenseurQ2_troisSym_HHHH::operator/=( const double & ) [virtual]`  
Implémente [TenseurHHHH](#).

**6.850.1.19 operator=( )**

`TenseurHHHH & TenseurQ2_troisSym_HHHH::operator=( const TenseurHHHH & ) [virtual]`  
Implémente [TenseurHHHH](#).

**6.850.1.20 operator==( )**

`int TenseurQ2_troisSym_HHHH::operator==( const TenseurHHHH & ) const [virtual]`  
Implémente [TenseurHHHH](#).

**6.850.1.21 Prod\_gauche()**

`TenseurHH & TenseurQ2_troisSym_HHHH::Prod_gauche ( const TenseurBB & F ) const [inline], [protected], [virtual]`  
Implémente [TenseurHHHH](#).

**6.850.1.22 Transpose1et2avec3et4()**

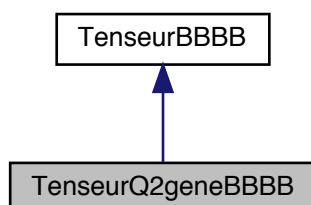
`TenseurHHHH & TenseurQ2_troisSym_HHHH::Transpose1et2avec3et4 ( ) const [virtual]`  
Implémente [TenseurHHHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

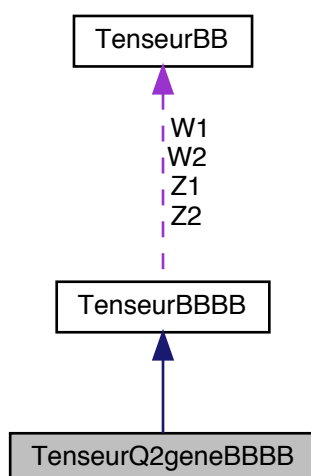
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur2\_TroisSym.h

## 6.851 Référence de la classe TenseurQ2geneBBBB

Graphe d'héritage de TenseurQ2geneBBBB:



Graphe de collaboration de TenseurQ2geneBBBB:



### Fonctions membres publiques

- **TenseurQ2geneBBBB** (const double val)
- **TenseurQ2geneBBBB** (int cas, const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- **TenseurQ2geneBBBB** (int cas, const [Tenseur2BB](#) &aBB, const [Tenseur2BB](#) &bBB)
- **TenseurQ2geneBBBB** (const [TenseurBBBB](#) &)
- **TenseurQ2geneBBBB** (const [TenseurQ2geneBBBB](#) &)
- void **Inita** (double val)
- [TenseurBBBB](#) & **operator+** (const [TenseurBBBB](#) &) const
- void **operator+=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator-** () const
- [TenseurBBBB](#) & **operator-** (const [TenseurBBBB](#) &) const

- void `operator-=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator=` (const `TenseurQ2geneBBBB` &B)
- `TenseurBBBB` & `operator*` (const double &) const
- void `operator*=` (const double &)
- `TenseurBBBB` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurBB` & `operator&&` (const `TenseurHH` &) const
- `TenseurBBBB` & `operator&&` (const `TenseurHHBB` &) const
- `TenseurBBHH` & `operator&&` (const `TenseurHHHH` &) const
- `TenseurBBBB` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurBBBB` &B, bool `plusZero`)
- `TenseurBBBB` & `Symetrise1et2_3et4` () const
- int `operator==` (const `TenseurBBBB` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

### Fonctions membres publiques statiques

- static `TenseurBBBB` & `Prod_tensoriel` (const `TenseurBB` &aBB, const `TenseurBB` &bBB)
- static `TenseurBBBB` & `Prod_tensoriel_barre` (const `TenseurBB` &aBB, const `TenseurBB` &bBB)
- static `TenseurBBBB` & `Prod_tensoriel_under_barre` (const `TenseurBB` &aBB, const `TenseurBB` &bBB)

### Fonctions membres protégées

- `TenseurBB` & `Prod_gauche` (const `TenseurHH` &F) const
- `TenseurHHBB` & `Prod_gauche` (const `TenseurHHHH` &F) const
- `TenseurBBBB` & `Prod_gauche` (const `TenseurBBHH` &F) const

### Attributs protégés

- listdouble16Iter `ipointe`

### Amis

- istream & `operator>>` (istream &, `TenseurQ2geneBBBB` &)
- ostream & `operator<<` (ostream &, const `TenseurQ2geneBBBB` &)

### Membres hérités additionnels

#### 6.851.1 Documentation des fonctions membres

##### 6.851.1.1 `Affectation_trans_dimension()`

```
void TenseurQ2geneBBBB::Affectation_trans_dimension (
    const TenseurBBBB & B,
    bool plusZero ) [virtual]
```

Implémente `TenseurBBBB`.

### 6.851.1.2 Change()

```
void TenseurQ2geneBBBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.851.1.3 ChangePlus()

```
void TenseurQ2geneBBBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.851.1.4 Ecriture()

```
ostream & TenseurQ2geneBBBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.851.1.5 Inita()

```
void TenseurQ2geneBBBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.851.1.6 Lecture()

```
istream & TenseurQ2geneBBBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.851.1.7 MaxiComposante()

```
double TenseurQ2geneBBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.851.1.8 operator&&()

```
TenseurBB & TenseurQ2geneBBBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.851.1.9 operator>()()

```
double TenseurQ2geneBBBB::operator() (
    int i,
```

```
int j,  
int k,  
int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.851.1.10 operator\*()

```
TenseurBBBB & TenseurQ2geneBBBB::operator* (  
const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.851.1.11 operator\*=( )

```
void TenseurQ2geneBBBB::operator*= (  
const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.851.1.12 operator+( )

```
TenseurBBBB & TenseurQ2geneBBBB::operator+ (  
const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.851.1.13 operator+=( )

```
void TenseurQ2geneBBBB::operator+= (  
const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.851.1.14 operator-( ) [1/2]

```
TenseurBBBB & TenseurQ2geneBBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.851.1.15 operator-( ) [2/2]

```
TenseurBBBB & TenseurQ2geneBBBB::operator- (  
const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.851.1.16 operator-=( )

```
void TenseurQ2geneBBBB::operator-=  
const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.851.1.17 operator/( )

```
TenseurBBBB & TenseurQ2geneBBBB::operator/  
const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).



**6.851.1.18 operator/=()**

```
void TenseurQ2geneBBBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.851.1.19 operator=()**

```
TenseurBBBB & TenseurQ2geneBBBB::operator= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.851.1.20 operator==()**

```
int TenseurQ2geneBBBB::operator== (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.851.1.21 Prod\_gauche()**

```
TenseurBB & TenseurQ2geneBBBB::Prod_gauche (
    const TenseurHH & F ) const [protected], [virtual]
```

Implémente [TenseurBBBB](#).

**6.851.1.22 Transpose1et2avec3et4()**

```
TenseurBBBB & TenseurQ2geneBBBB::Transpose1et2avec3et4 ( ) const [virtual]
```

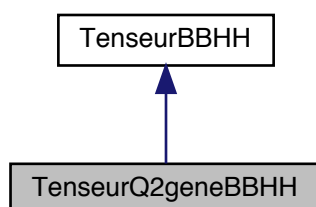
Implémente [TenseurBBBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

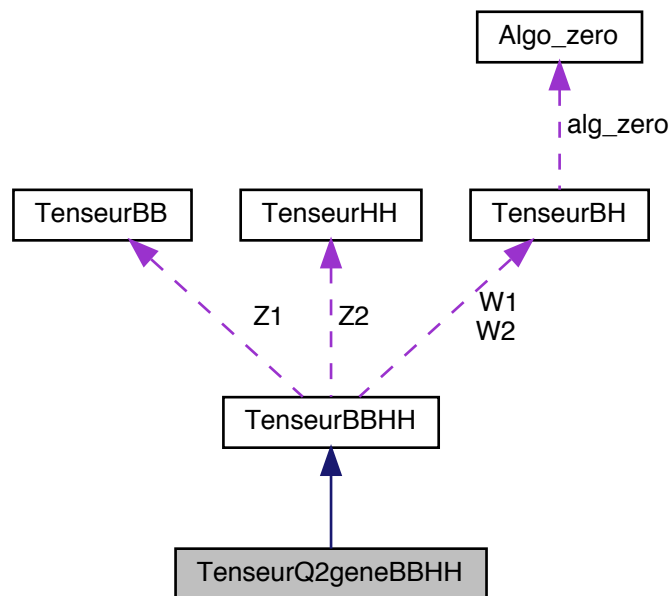
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ2gene.h

## 6.852 Référence de la classe TenseurQ2geneBBHH

Graphe d'héritage de TenseurQ2geneBBHH:



Grphe de collaboration de TenseurQ2geneBBHH:



## Fonctions membres publiques

- `TenseurQ2geneBBHH` (const double val)
- `TenseurQ2geneBBHH` (const `TenseurBB` &aBB, const `TenseurHH` &bHH)
- `TenseurQ2geneBBHH` (const `Tenseur2BB` &aBB, const `Tenseur2HH` &bHH)
- `TenseurQ2geneBBHH` (const `TenseurBBHH` &)
- `TenseurQ2geneBBHH` (const `TenseurQ2geneBBHH` &)
- void `Inita` (double val)
- `TenseurBBHH` & `operator+` (const `TenseurBBHH` &) const
- void `operator+=` (const `TenseurBBHH` &)
- `TenseurBBHH` & `operator-` () const
- `TenseurBBHH` & `operator-` (const `TenseurBBHH` &) const
- void `operator-=` (const `TenseurBBHH` &)
- `TenseurBBHH` & `operator=` (const `TenseurBBHH` &)
- `TenseurBBHH` & `operator*` (const double &) const
- void `operator*+=` (const double &)
- `TenseurBBHH` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurBB` & `operator&&` (const `TenseurBB` &) const
- `TenseurBBBB` & `operator&&` (const `TenseurBBBB` &) const
- `TenseurBBHH` & `operator&&` (const `TenseurBBHH` &) const
- `TenseurHHBB` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurBBHH` &B, bool plusZero)
- int `operator==` (const `TenseurBBHH` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres publiques statiques

- static [TenseurBBHH](#) & **Prod\_tensoriel** (const [TenseurBB](#) &aBB, const [TenseurHH](#) &bHH)

## Fonctions membres protégées

- [TenseurHH](#) & **Prod\_gauche** (const [TenseurHH](#) &F) const
- [TenseurBBHH](#) & **Prod\_gauche** (const [TenseurBBHH](#) &F) const
- [TenseurHHHH](#) & **Prod\_gauche** (const [TenseurHHHH](#) &F) const

## Attributs protégés

- listdouble16lter **ipointe**

## Amis

- istream & **operator>>** (istream &, [TenseurQ2geneBBHH](#) &)
- ostream & **operator<<** (ostream &, const [TenseurQ2geneBBHH](#) &)

## Membres hérités additionnels

### 6.852.1 Documentation des fonctions membres

#### 6.852.1.1 Affectation\_trans\_dimension()

```
void TenseurQ2geneBBHH::Affectation_trans_dimension (
    const TenseurBBHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.852.1.2 Change()

```
void TenseurQ2geneBBHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.852.1.3 ChangePlus()

```
void TenseurQ2geneBBHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.852.1.4 Ecriture()

```
ostream & TenseurQ2geneBBHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.5 Inita()

```
void TenseurQ2geneBBHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.6 Lecture()

```
istream & TenseurQ2geneBBHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.7 MaxiComposante()

```
double TenseurQ2geneBBHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.8 operator&&()

```
TenseurBB & TenseurQ2geneBBHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.9 operator()()

```
double TenseurQ2geneBBHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.10 operator\*()

```
TenseurBBHH & TenseurQ2geneBBHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.11 operator\*=( )

```
void TenseurQ2geneBBHH::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.12 operator+()

```
TenseurBBHH & TenseurQ2geneBBHH::operator+ (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.13 operator+=()**

```
void TenseurQ2geneBBHH::operator+= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.14 operator-() [1/2]**

```
TenseurBBHH & TenseurQ2geneBBHH::operator- ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.15 operator-() [2/2]**

```
TenseurBBHH & TenseurQ2geneBBHH::operator- (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.16 operator-=()**

```
void TenseurQ2geneBBHH::operator-= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.17 operator/()**

```
TenseurBBHH & TenseurQ2geneBBHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.18 operator/=()**

```
void TenseurQ2geneBBHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.19 operator=()**

```
TenseurBBHH & TenseurQ2geneBBHH::operator= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.20 operator==(())**

```
int TenseurQ2geneBBHH::operator==(
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.852.1.21 Prod\_gauche()**

```
TenseurHH & TenseurQ2geneBBHH::Prod_gauche (
    const TenseurHH & F ) const [protected], [virtual]
```

Implémente [TenseurBBHH](#).

### 6.852.1.22 Transpose1et2avec3et4()

`TenseurHHBB` & `TenseurQ2geneBBHH::Transpose1et2avec3et4 ( ) const [virtual]`

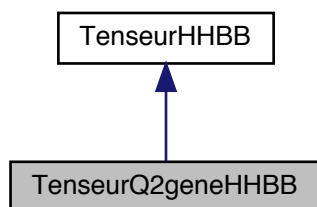
Implémente `TenseurBBHH`.

La documentation de cette classe a été générée à partir du fichier suivant :

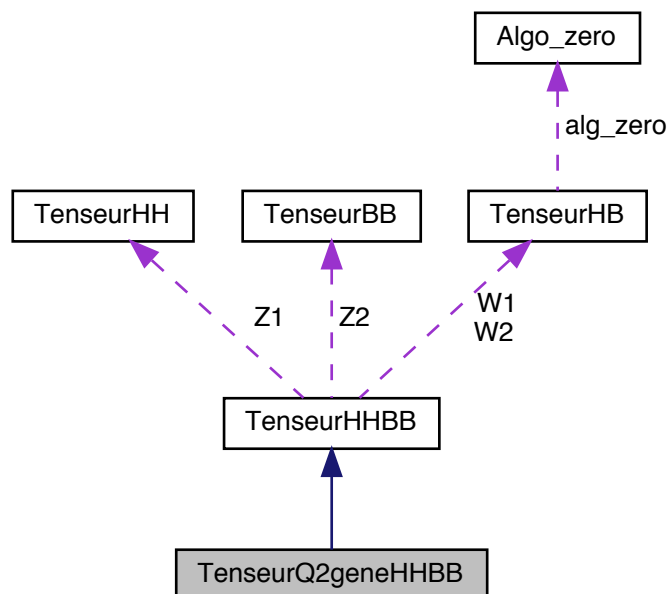
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ2gene.h

## 6.853 Référence de la classe TenseurQ2geneHHBB

Graphe d'héritage de TenseurQ2geneHHBB:



Graphe de collaboration de TenseurQ2geneHHBB:



## Fonctions membres publiques

- `TenseurQ2geneHHBB` (const double val)
- `TenseurQ2geneHHBB` (const `TenseurHH` &aHH, const `TenseurBB` &bBB)
- `TenseurQ2geneHHBB` (const `Tenseur2HH` &aHH, const `Tenseur2BB` &bBB)
- `TenseurQ2geneHHBB` (const `TenseurHHBB` &)
- `TenseurQ2geneHHBB` (const `TenseurQ2geneHHBB` &)
- void `Inita` (double val)
- `TenseurHHBB` & `operator+` (const `TenseurHHBB` &) const
- void `operator+=` (const `TenseurHHBB` &)
- `TenseurHHBB` & `operator-` () const
- `TenseurHHBB` & `operator-` (const `TenseurHHBB` &) const
- void `operator-=` (const `TenseurHHBB` &)
- `TenseurHHBB` & `operator=` (const `TenseurHHBB` &)
- `TenseurHHBB` & `operator*` (const double &) const
- void `operator*=` (const double &)
- `TenseurHHBB` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurHH` & `operator&&` (const `TenseurHH` &) const
- `TenseurHHHH` & `operator&&` (const `TenseurHHHH` &) const
- `TenseurHHBB` & `operator&&` (const `TenseurHHBB` &) const
- `TenseurBBHH` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurHHBB` &B, bool plusZero)
- int `operator==` (const `TenseurHHBB` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres publiques statiques

- static `TenseurHHBB` & `Prod_tensoriel` (const `TenseurHH` &aHH, const `TenseurBB` &bBB)

## Fonctions membres protégées

- `TenseurBB` & `Prod_gauche` (const `TenseurBB` &F) const
- `TenseurHHBB` & `Prod_gauche` (const `TenseurHHBB` &F) const
- `TenseurBBBB` & `Prod_gauche` (const `TenseurBBBB` &F) const

## Attributs protégés

- listdouble16Iter `ipointe`

## Amis

- istream & `operator>>` (istream &, `TenseurQ2geneHHBB` &)
- ostream & `operator<<` (ostream &, const `TenseurQ2geneHHBB` &)

## Membres hérités additionnels

### 6.853.1 Documentation des fonctions membres

#### 6.853.1.1 `Affectation_trans_dimension()`

```
void TenseurQ2geneHHBB::Affectation_trans_dimension (
    const TenseurHHBB & B,
    bool plusZero ) [virtual]
```

Implémente `TenseurHHBB`.

### 6.853.1.2 Change()

```
void TenseurQ2geneHHBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.3 ChangePlus()

```
void TenseurQ2geneHHBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.4 Ecriture()

```
ostream & TenseurQ2geneHHBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.5 Inita()

```
void TenseurQ2geneHHBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.6 Lecture()

```
istream & TenseurQ2geneHHBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.7 MaxiComposante()

```
double TenseurQ2geneHHBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.8 operator&&()

```
TenseurHH & TenseurQ2geneHHBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.9 operator>()()

```
double TenseurQ2geneHHBB::operator() (
    int i,
```



```
int j,  
int k,  
int l ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.853.1.10 operator\*()

```
TenseurHHBB & TenseurQ2geneHHBB::operator* (  
const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.853.1.11 operator\*=( )

```
void TenseurQ2geneHHBB::operator*= (  
const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.853.1.12 operator+( )

```
TenseurHHBB & TenseurQ2geneHHBB::operator+ (  
const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.853.1.13 operator+=( )

```
void TenseurQ2geneHHBB::operator+= (  
const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.853.1.14 operator-( ) [1/2]

```
TenseurHHBB & TenseurQ2geneHHBB::operator- ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.853.1.15 operator-( ) [2/2]

```
TenseurHHBB & TenseurQ2geneHHBB::operator- (  
const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.853.1.16 operator-=( )

```
void TenseurQ2geneHHBB::operator-=  
const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.853.1.17 operator/( )

```
TenseurHHBB & TenseurQ2geneHHBB::operator/  
const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.18 operator/=()

```
void TenseurQ2geneHHBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.19 operator=()

```
TenseurHHBB & TenseurQ2geneHHBB::operator= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.20 operator==()

```
int TenseurQ2geneHHBB::operator== (
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.21 Prod\_gauche()

```
TenseurBB & TenseurQ2geneHHBB::Prod_gauche (
    const TenseurBB & F ) const [protected], [virtual]
```

Implémente [TenseurHHBB](#).

### 6.853.1.22 Transpose1et2avec3et4()

```
TenseurBBHH & TenseurQ2geneHHBB::Transpose1et2avec3et4 ( ) const [virtual]
```

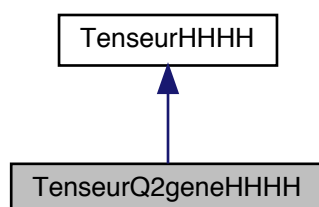
Implémente [TenseurHHBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

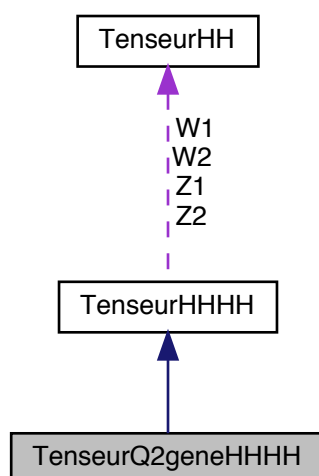
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ2gene.h

## 6.854 Référence de la classe TenseurQ2geneHHHH

Graphe d'héritage de TenseurQ2geneHHHH:



Graph de collaboration de TenseurQ2geneHHHH:



## Fonctions membres publiques

- `TenseurQ2geneHHHH` (const double val)
- `TenseurQ2geneHHHH` (int cas, const `TenseurHH` &aHH, const `TenseurHH` &bHH)
- `TenseurQ2geneHHHH` (int cas, const `Tenseur2HH` &aHH, const `Tenseur2HH` &bHH)
- `TenseurQ2geneHHHH` (const `TenseurHHHH` &)
- `TenseurQ2geneHHHH` (const `TenseurQ2geneHHHH` &)
- void `Inita` (double val)
- `TenseurHHHH` & `operator+` (const `TenseurHHHH` &) const
- void `operator+=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator-` () const
- `TenseurHHHH` & `operator-` (const `TenseurHHHH` &) const
- void `operator-=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator=` (const `TenseurQ2geneHHHH` &B)
- `TenseurHHHH` & `operator*` (const double &) const
- void `operator*=` (const double &)
- `TenseurHHHH` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurHH` & `operator&&` (const `TenseurBB` &) const
- `TenseurHHHH` & `operator&&` (const `TenseurBBHH` &) const
- `TenseurHHBB` & `operator&&` (const `TenseurBBBB` &) const
- `TenseurHHHH` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurHHHH` &B, bool plusZero)
- `TenseurHHHH` & `Symetrise1et2_3et4` () const
- int `operator==` (const `TenseurHHHH` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres publiques statiques

- static `TenseurHHHH` & `Prod_tensoriel` (const `TenseurHH` &aHH, const `TenseurHH` &bHH)
- static `TenseurHHHH` & `Prod_tensoriel_barre` (const `TenseurHH` &aHH, const `TenseurHH` &bHH)
- static `TenseurHHHH` & `Prod_tensoriel_under_barre` (const `TenseurHH` &aHH, const `TenseurHH` &bHH)

## Fonctions membres protégées

- [TenseurHH](#) & [Prod\\_gauche](#) (const [TenseurBB](#) &F) const
- [TenseurBBHH](#) & [Prod\\_gauche](#) (const [TenseurBBBB](#) &F) const
- [TenseurHHHH](#) & [Prod\\_gauche](#) (const [TenseurHHBB](#) &F) const

## Attributs protégés

- listdouble16Iter [ipointe](#)

## Amis

- istream & [operator](#)>> (istream &, [TenseurQ2geneHHHH](#) &)
- ostream & [operator](#)<< (ostream &, const [TenseurQ2geneHHHH](#) &)

## Membres hérités additionnels

### 6.854.1 Documentation des fonctions membres

#### 6.854.1.1 Affectation\_trans\_dimension()

```
void TenseurQ2geneHHHH::Affectation_trans_dimension (
    const TenseurHHHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.854.1.2 Change()

```
void TenseurQ2geneHHHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.854.1.3 ChangePlus()

```
void TenseurQ2geneHHHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.854.1.4 Ecriture()

```
ostream & TenseurQ2geneHHHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.5 Inita()**

```
void TenseurQ2geneHHHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.6 Lecture()**

```
istream & TenseurQ2geneHHHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.7 MaxiComposante()**

```
double TenseurQ2geneHHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.8 operator&&()**

```
TenseurHH & TenseurQ2geneHHHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.9 operator>()()**

```
double TenseurQ2geneHHHH::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.10 operator\*()**

```
TenseurHHHH & TenseurQ2geneHHHH::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.11 operator\*=( )**

```
void TenseurQ2geneHHHH::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.12 operator+( )**

```
TenseurHHHH & TenseurQ2geneHHHH::operator+ (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.13 operator+=()**

```
void TenseurQ2geneHHHH::operator+= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.14 operator-() [1/2]**

```
TenseurHHHH & TenseurQ2geneHHHH::operator- ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.15 operator-() [2/2]**

```
TenseurHHHH & TenseurQ2geneHHHH::operator- (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.16 operator-=()**

```
void TenseurQ2geneHHHH::operator-= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.17 operator/()**

```
TenseurHHHH & TenseurQ2geneHHHH::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.18 operator/=()**

```
void TenseurQ2geneHHHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.19 operator=()**

```
TenseurHHHH & TenseurQ2geneHHHH::operator= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.20 operator==()**

```
int TenseurQ2geneHHHH::operator== (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.854.1.21 Prod\_gauche()**

```
TenseurHH & TenseurQ2geneHHHH::Prod_gauche (
    const TenseurBB & F ) const [protected], [virtual]
```

Implémente [TenseurHHHH](#).

### 6.854.1.22 Transpose1et2avec3et4()

`TenseurHHHH & TenseurQ2geneHHHH::Transpose1et2avec3et4 ( ) const [virtual]`

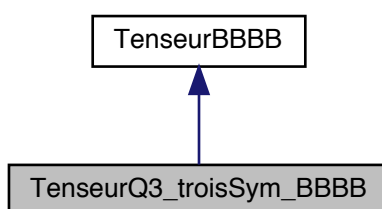
Implémente [TenseurHHHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

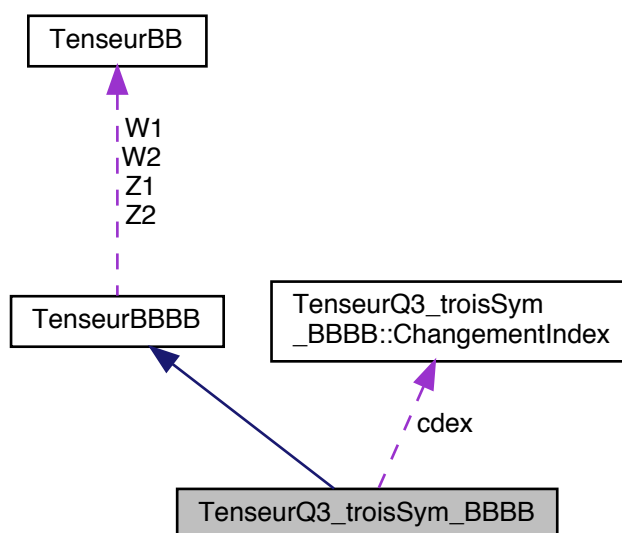
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ2gene.h

## 6.855 Référence de la classe TenseurQ3\_troisSym\_BBBB

Graphe d'héritage de TenseurQ3\_troisSym\_BBBB:



Graphe de collaboration de TenseurQ3\_troisSym\_BBBB:



### Classes

— class [ChangementIndex](#)

## Fonctions membres publiques

- `TenseurQ3_troisSym_BBBB` (const double &val)
- `TenseurQ3_troisSym_BBBB` (const double &x1111, const double &x2222, const double &x3333, const double &x1122, const double &x1133, const double &x2233, const double &x1211, const double &x1222, const double &x1233, const double &x1311, const double &x1322, const double &x1333, const double &x2311, const double &x2322, const double &x2333, const double &x1212, const double &x1313, const double &x2323, const double &x1213, const double &x1223, const double &x1323)
- `TenseurQ3_troisSym_BBBB` (const `TenseurBBBB` &)
- `TenseurQ3_troisSym_BBBB` (const `TenseurQ3_troisSym_BBBB` &)
- void `Inita` (double val)
- `TenseurBBBB` & `operator+` (const `TenseurBBBB` &) const
- void `operator+=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator-` () const
- `TenseurBBBB` & `operator-` (const `TenseurBBBB` &) const
- void `operator-=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator=` (const `TenseurBBBB` &)
- `TenseurBBBB` & `operator*` (const double &) const
- void `operator*+=` (const double &)
- `TenseurBBBB` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurBB` & `operator&&` (const `TenseurHH` &) const
- `TenseurBBHH` & `operator&&` (const `TenseurHHHH` &) const
- `TenseurBBBB` & `operator&&` (const `TenseurHHBB` &) const
- `TenseurBBBB` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurBBBB` &B, bool plusZero)
- int `operator==` (const `TenseurBBBB` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres protégées

- `TenseurBB` & `Prod_gauche` (const `TenseurHH` &F) const
- `TenseurHHBB` & `Prod_gauche` (const `TenseurHHHH` &) const
- `TenseurBBBB` & `Prod_gauche` (const `TenseurBBHH` &) const

## Attributs protégés

- listdouble21Iter `ipointe`

## Attributs protégés statiques

- static const `ChangementIndex` `cdex`  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3*

## Amis

- istream & `operator>>` (istream &, `TenseurQ3_troisSym_BBBB` &)
- ostream & `operator<<` (ostream &, const `TenseurQ3_troisSym_BBBB` &)

## Membres hérités additionnels

### 6.855.1 Documentation des fonctions membres



### 6.855.1.1 Affectation\_trans\_dimension()

```
void TenseurQ3_troisSym_BBBB::Affectation_trans_dimension (
    const TenseurBBBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.2 Change()

```
void TenseurQ3_troisSym_BBBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.3 ChangePlus()

```
void TenseurQ3_troisSym_BBBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.4 Ecriture()

```
ostream & TenseurQ3_troisSym_BBBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.5 Inita()

```
void TenseurQ3_troisSym_BBBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.6 Lecture()

```
istream & TenseurQ3_troisSym_BBBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.7 MaxiComposante()

```
double TenseurQ3_troisSym_BBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.8 operator&&()

```
TenseurBB & TenseurQ3_troisSym_BBBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.9 operator>()()

```
double TenseurQ3_troisSym_BBBB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.10 operator\*()

```
TenseurBBBB & TenseurQ3_troisSym_BBBB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.11 operator\*=( )

```
void TenseurQ3_troisSym_BBBB::operator*= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.12 operator+( )

```
TenseurBBBB & TenseurQ3_troisSym_BBBB::operator+ (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.13 operator+=( )

```
void TenseurQ3_troisSym_BBBB::operator+= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.14 operator-() [1/2]

```
TenseurBBBB & TenseurQ3_troisSym_BBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

### 6.855.1.15 operator-() [2/2]

```
TenseurBBBB & TenseurQ3_troisSym_BBBB::operator- (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.855.1.16 operator--()**

```
void TenseurQ3_troisSym_BBBB::operator-- (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.855.1.17 operator/()**

```
TenseurBBBB & TenseurQ3_troisSym_BBBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.855.1.18 operator/=( )**

```
void TenseurQ3_troisSym_BBBB::operator/=(
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.855.1.19 operator=( )**

```
TenseurBBBB & TenseurQ3_troisSym_BBBB::operator=(
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.855.1.20 operator==( )**

```
int TenseurQ3_troisSym_BBBB::operator==(
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.855.1.21 Prod\_gauche()**

```
TenseurBB & TenseurQ3_troisSym_BBBB::Prod_gauche (
    const TenseurHH & F ) const [inline], [protected], [virtual]
```

Implémente [TenseurBBBB](#).

**6.855.1.22 Transpose1et2avec3et4()**

```
TenseurBBBB & TenseurQ3_troisSym_BBBB::Transpose1et2avec3et4 ( ) const [virtual]
```

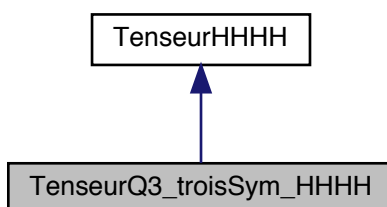
Implémente [TenseurBBBB](#).

La documentation de cette classe a été générée à partir du fichier suivant :

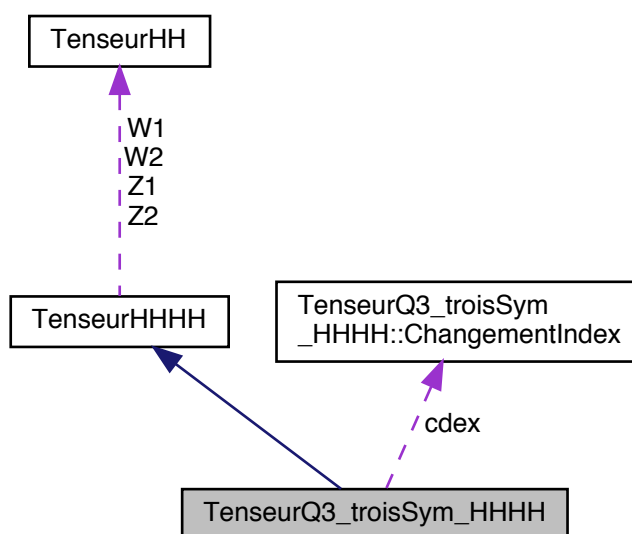
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3\_TroisSym.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

## 6.856 Référence de la classe TenseurQ3\_troisSym\_HHHH

Grphe d'héritage de TenseurQ3\_troisSym\_HHHH:



Grphe de collaboration de TenseurQ3\_troisSym\_HHHH:



### Classes

- class [ChangementIndex](#)

### Fonctions membres publiques

- **TenseurQ3\_troisSym\_HHHH** (const double &val)
- **TenseurQ3\_troisSym\_HHHH** (const double &x1111, const double &x2222, const double &x3333, const double &x1122, const double &x1133, const double &x2233, const double &x1211, const double &x1222, const double &x1233, const double &x1311, const double &x1322, const double &x1333, const double &x2311, const double &x2322, const double &x2333, const double &x1212, const double &x1313, const double &x2323, const double &x1213, const double &x1223, const double &x1323)

- `TenseurQ3_troisSym_HHHH` (const `TenseurHHHH` &)
- `TenseurQ3_troisSym_HHHH` (const `TenseurQ3_troisSym_HHHH` &)
- void `Inita` (double val)
- `TenseurHHHH` & `operator+` (const `TenseurHHHH` &) const
- void `operator+=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator-` () const
- `TenseurHHHH` & `operator-` (const `TenseurHHHH` &) const
- void `operator-=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator=` (const `TenseurHHHH` &)
- `TenseurHHHH` & `operator*` (const double &) const
- void `operator*=` (const double &)
- `TenseurHHHH` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurHH` & `operator&&` (const `TenseurBB` &) const
- `TenseurHHHH` & `operator&&` (const `TenseurBBHH` &) const
- `TenseurHHBB` & `operator&&` (const `TenseurBBBB` &) const
- `TenseurHHHH` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurHHHH` &B, bool plusZero)
- int `operator==` (const `TenseurHHHH` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres protégées

- `TenseurHH` & `Prod_gauche` (const `TenseurBB` &F) const
- `TenseurBBHH` & `Prod_gauche` (const `TenseurBBBB` &) const
- `TenseurHHHH` & `Prod_gauche` (const `TenseurHHBB` &) const

## Attributs protégés

- listdouble21Iter `ipointe`

## Attributs protégés statiques

- static const `ChangementIndex` `cdex`  
*initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3*

## Amis

- istream & `operator>>` (istream &, `TenseurQ3_troisSym_HHHH` &)
- ostream & `operator<<` (ostream &, const `TenseurQ3_troisSym_HHHH` &)

## Membres hérités additionnels

### 6.856.1 Documentation des fonctions membres

#### 6.856.1.1 `Affectation_trans_dimension()`

```
void TenseurQ3_troisSym_HHHH::Affectation_trans_dimension (
    const TenseurHHHH & B,
    bool plusZero ) [virtual]
```

Implémente `TenseurHHHH`.

### 6.856.1.2 Change()

```
void TenseurQ3_troisSym_HHHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.856.1.3 ChangePlus()

```
void TenseurQ3_troisSym_HHHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.856.1.4 Ecriture()

```
ostream & TenseurQ3_troisSym_HHHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.856.1.5 Inita()

```
void TenseurQ3_troisSym_HHHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.856.1.6 Lecture()

```
istream & TenseurQ3_troisSym_HHHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.856.1.7 MaxiComposante()

```
double TenseurQ3_troisSym_HHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.856.1.8 operator&&()

```
TenseurHH & TenseurQ3_troisSym_HHHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.856.1.9 operator>()()

```
double TenseurQ3_troisSym_HHHH::operator() (
    int i,
```

```
int j,  
int k,  
int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.856.1.10 operator\*()

```
TenseurHHHH & TenseurQ3_troisSym_HHHH::operator* (  
const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.856.1.11 operator\*=( )

```
void TenseurQ3_troisSym_HHHH::operator*= (  
const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.856.1.12 operator+()

```
TenseurHHHH & TenseurQ3_troisSym_HHHH::operator+ (  
const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.856.1.13 operator+=( )

```
void TenseurQ3_troisSym_HHHH::operator+= (  
const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.856.1.14 operator-() [1/2]

```
TenseurHHHH & TenseurQ3_troisSym_HHHH::operator- ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.856.1.15 operator-() [2/2]

```
TenseurHHHH & TenseurQ3_troisSym_HHHH::operator- (  
const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.856.1.16 operator-=( )

```
void TenseurQ3_troisSym_HHHH::operator-= (  
const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.856.1.17 operator/()

```
TenseurHHHH & TenseurQ3_troisSym_HHHH::operator/ (  
const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.856.1.18 operator/=()**

```
void TenseurQ3_troisSym_HHHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.856.1.19 operator=()**

```
TenseurHHHH & TenseurQ3_troisSym_HHHH::operator= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.856.1.20 operator==()**

```
int TenseurQ3_troisSym_HHHH::operator== (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.856.1.21 Prod\_gauche()**

```
TenseurHH & TenseurQ3_troisSym_HHHH::Prod_gauche (
    const TenseurBB & F ) const [inline], [protected], [virtual]
```

Implémente [TenseurHHHH](#).

**6.856.1.22 Transpose1et2avec3et4()**

```
TenseurHHHH & TenseurQ3_troisSym_HHHH::Transpose1et2avec3et4 ( ) const [virtual]
```

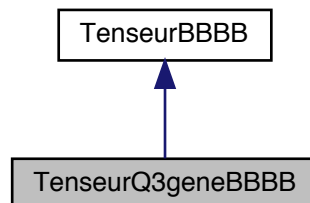
Implémente [TenseurHHHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/Tenseur3\_TroisSym.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

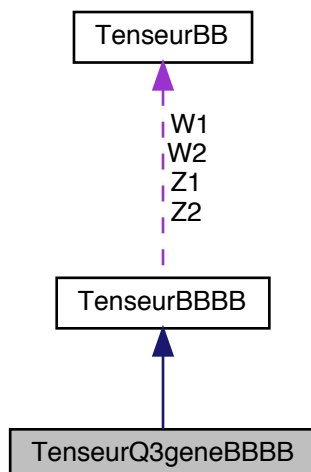
**6.857 Référence de la classe TenseurQ3geneBBBB**

Graphes d'héritage de TenseurQ3geneBBBB:





Graph de collaboration de TenseurQ3geneBBBB:



## Fonctions membres publiques

- **TenseurQ3geneBBBB** (const double val)
- **TenseurQ3geneBBBB** (int cas, const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- **TenseurQ3geneBBBB** (int cas, const [Tenseur3BB](#) &aBB, const [Tenseur3BB](#) &bBB)
- **TenseurQ3geneBBBB** (const [TenseurBBBB](#) &)
- **TenseurQ3geneBBBB** (const [TenseurQ3geneBBBB](#) &)
- void **Inita** (double val)
- [TenseurBBBB](#) & **operator+** (const [TenseurBBBB](#) &) const
- void **operator+=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator-** () const
- [TenseurBBBB](#) & **operator-** (const [TenseurBBBB](#) &) const
- void **operator-=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator=** (const [TenseurBBBB](#) &)
- [TenseurBBBB](#) & **operator=** (const [TenseurQ3geneBBBB](#) &B)
- [TenseurBBBB](#) & **operator\*** (const double &) const
- void **operator\*=** (const double &)
- [TenseurBBBB](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurBB](#) & **operator&&** (const [TenseurHH](#) &) const
- [TenseurBBBB](#) & **operator&&** (const [TenseurHHBB](#) &) const
- [TenseurBBHH](#) & **operator&&** (const [TenseurHHHH](#) &) const
- [TenseurBBBB](#) & **Transpose1et2avec3et4** () const
- void **Affectation\_trans\_dimension** (const [TenseurBBBB](#) &B, bool plusZero)
- [TenseurBBBB](#) & **Symetrise1et2\_3et4** () const
- int **operator==** (const [TenseurBBBB](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort) const

## Fonctions membres publiques statiques

- static [TenseurBBBB](#) & **Prod\_tensoriel** (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- static [TenseurBBBB](#) & **Prod\_tensoriel\_barre** (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)
- static [TenseurBBBB](#) & **Prod\_tensoriel\_under\_barre** (const [TenseurBB](#) &aBB, const [TenseurBB](#) &bBB)

## Fonctions membres protégées

- [TenseurBB](#) & [Prod\\_gauche](#) (const [TenseurHH](#) &F) const
- [TenseurHHBB](#) & [Prod\\_gauche](#) (const [TenseurHHHH](#) &F) const
- [TenseurBBBB](#) & [Prod\\_gauche](#) (const [TenseurBBHH](#) &F) const

## Attributs protégés

- listdouble81Iter [ipointe](#)

## Amis

- istream & [operator](#)>> (istream &, [TenseurQ3geneBBBB](#) &)
- ostream & [operator](#)<< (ostream &, const [TenseurQ3geneBBBB](#) &)

## Membres hérités additionnels

### 6.857.1 Documentation des fonctions membres

#### 6.857.1.1 Affectation\_trans\_dimension()

```
void TenseurQ3geneBBBB::Affectation_trans_dimension (
    const TenseurBBBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.857.1.2 Change()

```
void TenseurQ3geneBBBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.857.1.3 ChangePlus()

```
void TenseurQ3geneBBBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBBB](#).

#### 6.857.1.4 Ecriture()

```
ostream & TenseurQ3geneBBBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.5 Inita()**

```
void TenseurQ3geneBBBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.6 Lecture()**

```
istream & TenseurQ3geneBBBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.7 MaxiComposante()**

```
double TenseurQ3geneBBBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.8 operator&&()**

```
TenseurBB & TenseurQ3geneBBBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.9 operator>()()**

```
double TenseurQ3geneBBBB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.10 operator\*()**

```
TenseurBBBB & TenseurQ3geneBBBB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.11 operator\*=( )**

```
void TenseurQ3geneBBBB::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.12 operator+()**

```
TenseurBBBB & TenseurQ3geneBBBB::operator+ (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.13 operator+=()**

```
void TenseurQ3geneBBBB::operator+= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.14 operator-() [1/2]**

```
TenseurBBBB & TenseurQ3geneBBBB::operator- ( ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.15 operator-() [2/2]**

```
TenseurBBBB & TenseurQ3geneBBBB::operator- (
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.16 operator-=()**

```
void TenseurQ3geneBBBB::operator-= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.17 operator/()**

```
TenseurBBBB & TenseurQ3geneBBBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.18 operator/=()**

```
void TenseurQ3geneBBBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.19 operator=()**

```
TenseurBBBB & TenseurQ3geneBBBB::operator= (
    const TenseurBBBB & ) [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.20 operator==(())**

```
int TenseurQ3geneBBBB::operator==(
    const TenseurBBBB & ) const [virtual]
```

Implémente [TenseurBBBB](#).

**6.857.1.21 Prod\_gauche()**

```
TenseurBB & TenseurQ3geneBBBB::Prod_gauche (
    const TenseurHH & F ) const [protected], [virtual]
```

Implémente [TenseurBBBB](#).

### 6.857.1.22 Transpose1et2avec3et4()

`TenseurBBBB` & `TenseurQ3geneBBBB::Transpose1et2avec3et4 ( ) const` [virtual]

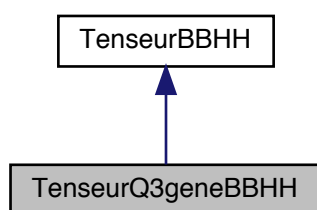
Implémente `TenseurBBBB`.

La documentation de cette classe a été générée à partir du fichier suivant :

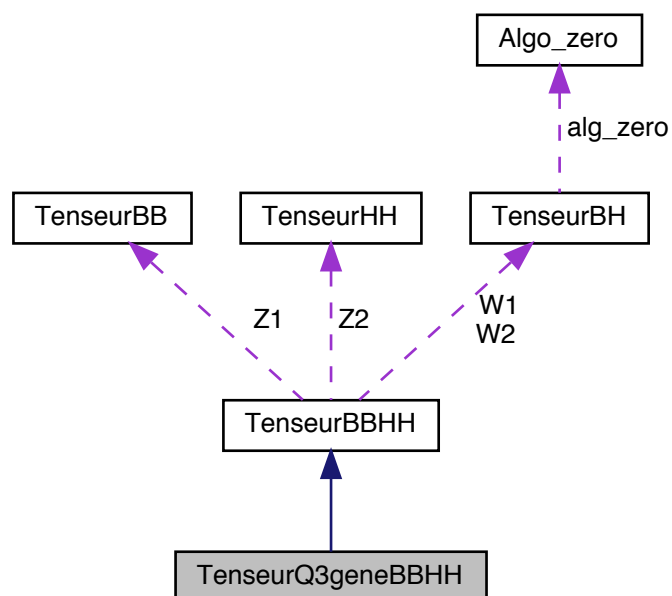
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3gene.h

## 6.858 Référence de la classe TenseurQ3geneBBHH

Graphe d'héritage de TenseurQ3geneBBHH:



Graphe de collaboration de TenseurQ3geneBBHH:



## Fonctions membres publiques

- `TenseurQ3geneBBHH` (const double val)
- `TenseurQ3geneBBHH` (const `TenseurBB` &aBB, const `TenseurHH` &bHH)
- `TenseurQ3geneBBHH` (const `Tenseur3BB` &aBB, const `Tenseur3HH` &bHH)
- `TenseurQ3geneBBHH` (const `TenseurBBHH` &)
- `TenseurQ3geneBBHH` (const `TenseurQ3geneBBHH` &)
- void `Inita` (double val)
- `TenseurBBHH` & `operator+` (const `TenseurBBHH` &) const
- void `operator+=` (const `TenseurBBHH` &)
- `TenseurBBHH` & `operator-` () const
- `TenseurBBHH` & `operator-` (const `TenseurBBHH` &) const
- void `operator-=` (const `TenseurBBHH` &)
- `TenseurBBHH` & `operator=` (const `TenseurBBHH` &)
- `TenseurBBHH` & `operator*` (const double &) const
- void `operator*=` (const double &)
- `TenseurBBHH` & `operator/` (const double &) const
- void `operator/=` (const double &)
- `TenseurBB` & `operator&&` (const `TenseurBB` &) const
- `TenseurBBBB` & `operator&&` (const `TenseurBBBB` &) const
- `TenseurBBHH` & `operator&&` (const `TenseurBBHH` &) const
- `TenseurHHBB` & `Transpose1et2avec3et4` () const
- void `Affectation_trans_dimension` (const `TenseurBBHH` &B, bool plusZero)
- int `operator==` (const `TenseurBBHH` &) const
- void `Change` (int i, int j, int k, int l, const double &val)
- void `ChangePlus` (int i, int j, int k, int l, const double &val)
- double `operator()` (int i, int j, int k, int l) const
- double `MaxiComposante` () const
- istream & `Lecture` (istream &entree)
- ostream & `Ecriture` (ostream &sort) const

## Fonctions membres publiques statiques

- static `TenseurBBHH` & `Prod_tensoriel` (const `TenseurBB` &aBB, const `TenseurHH` &bHH)

## Fonctions membres protégées

- `TenseurHH` & `Prod_gauche` (const `TenseurHH` &F) const
- `TenseurBBHH` & `Prod_gauche` (const `TenseurBBHH` &F) const
- `TenseurHHHH` & `Prod_gauche` (const `TenseurHHHH` &F) const

## Attributs protégés

- listdouble81iter `ipointe`

## Amis

- istream & `operator>>` (istream &, `TenseurQ3geneBBHH` &)
- ostream & `operator<<` (ostream &, const `TenseurQ3geneBBHH` &)

## Membres hérités additionnels

### 6.858.1 Documentation des fonctions membres

#### 6.858.1.1 `Affectation_trans_dimension()`

```
void TenseurQ3geneBBHH::Affectation_trans_dimension (
    const TenseurBBHH & B,
    bool plusZero ) [virtual]
```

Implémente `TenseurBBHH`.

### 6.858.1.2 Change()

```
void TenseurQ3geneBBHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.858.1.3 ChangePlus()

```
void TenseurQ3geneBBHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.858.1.4 Ecriture()

```
ostream & TenseurQ3geneBBHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.858.1.5 Inita()

```
void TenseurQ3geneBBHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.858.1.6 Lecture()

```
istream & TenseurQ3geneBBHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurBBHH](#).

### 6.858.1.7 MaxiComposante()

```
double TenseurQ3geneBBHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.858.1.8 operator&&()

```
TenseurBB & TenseurQ3geneBBHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurBBHH](#).

### 6.858.1.9 operator>()()

```
double TenseurQ3geneBBHH::operator() (
    int i,
```

```
int j,  
int k,  
int l ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.858.1.10 operator\*()

```
TenseurBBHH & TenseurQ3geneBBHH::operator* (  
const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.858.1.11 operator\*=( )

```
void TenseurQ3geneBBHH::operator*= (  
const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.858.1.12 operator+( )

```
TenseurBBHH & TenseurQ3geneBBHH::operator+ (  
const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.858.1.13 operator+=( )

```
void TenseurQ3geneBBHH::operator+= (  
const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.858.1.14 operator-( ) [1/2]

```
TenseurBBHH & TenseurQ3geneBBHH::operator- ( ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.858.1.15 operator-( ) [2/2]

```
TenseurBBHH & TenseurQ3geneBBHH::operator- (  
const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.858.1.16 operator-=( )

```
void TenseurQ3geneBBHH::operator-=  
const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

#### 6.858.1.17 operator/( )

```
TenseurBBHH & TenseurQ3geneBBHH::operator/  
const double & ) const [virtual]
```

Implémente [TenseurBBHH](#).



**6.858.1.18 operator/=()**

```
void TenseurQ3geneBBHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.858.1.19 operator=()**

```
TenseurBBHH & TenseurQ3geneBBHH::operator= (
    const TenseurBBHH & ) [virtual]
```

Implémente [TenseurBBHH](#).

**6.858.1.20 operator==()**

```
int TenseurQ3geneBBHH::operator== (
    const TenseurBBHH & ) const [virtual]
```

Implémente [TenseurBBHH](#).

**6.858.1.21 Prod\_gauche()**

```
TenseurHH & TenseurQ3geneBBHH::Prod_gauche (
    const TenseurHH & F ) const [protected], [virtual]
```

Implémente [TenseurBBHH](#).

**6.858.1.22 Transpose1et2avec3et4()**

```
TenseurHHBB & TenseurQ3geneBBHH::Transpose1et2avec3et4 ( ) const [virtual]
```

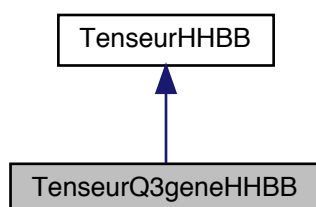
Implémente [TenseurBBHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

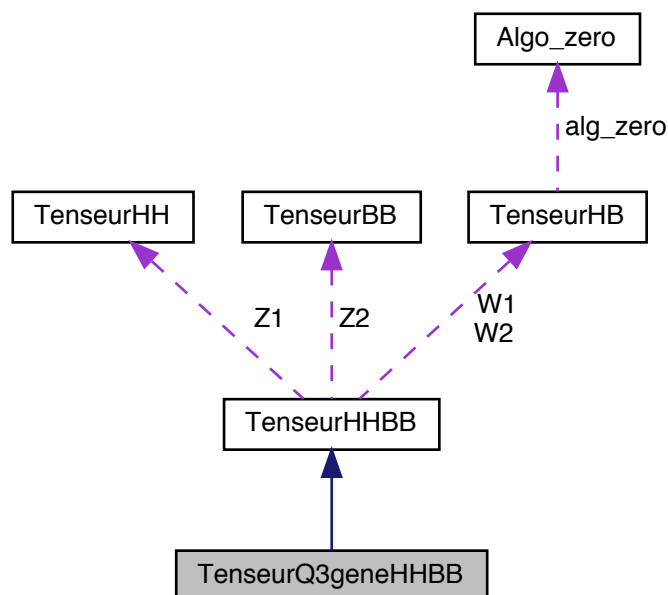
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3gene.h

**6.859 Référence de la classe TenseurQ3geneHHBB**

Graphe d'héritage de TenseurQ3geneHHBB:



Grphe de collaboration de TenseurQ3geneHHBB:



## Fonctions membres publiques

- **TenseurQ3geneHHBB** (const double val)
- **TenseurQ3geneHHBB** (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)
- **TenseurQ3geneHHBB** (const [Tenseur3HH](#) &aHH, const [Tenseur3BB](#) &bBB)
- **TenseurQ3geneHHBB** (const [TenseurHHBB](#) &)
- **TenseurQ3geneHHBB** (const [TenseurQ3geneHHBB](#) &)
- void **Inita** (double val)
- [TenseurHHBB](#) & **operator+** (const [TenseurHHBB](#) &) const
- void **operator+=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator-** () const
- [TenseurHHBB](#) & **operator-** (const [TenseurHHBB](#) &) const
- void **operator-=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator=** (const [TenseurHHBB](#) &)
- [TenseurHHBB](#) & **operator\*** (const double &) const
- void **operator\*= **(const double &)****
- [TenseurHHBB](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurHH](#) &) const
- [TenseurHHHH](#) & **operator&&** (const [TenseurHHHH](#) &) const
- [TenseurHHBB](#) & **operator&&** (const [TenseurHHBB](#) &) const
- [TenseurBBHH](#) & **Transpose1et2avec3et4** () const
- void **Affectation\_trans\_dimension** (const [TenseurHHBB](#) &B, bool plusZero)
- int **operator==** (const [TenseurHHBB](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort) const

## Fonctions membres publiques statiques

- static [TenseurHHBB](#) & [Prod\\_tensoriel](#) (const [TenseurHH](#) &aHH, const [TenseurBB](#) &bBB)

## Fonctions membres protégées

- [TenseurBB](#) & [Prod\\_gauche](#) (const [TenseurBB](#) &F) const
- [TenseurHHBB](#) & [Prod\\_gauche](#) (const [TenseurHHBB](#) &F) const
- [TenseurBBBB](#) & [Prod\\_gauche](#) (const [TenseurBBBB](#) &F) const

## Attributs protégés

- listdouble81Iter [ipointe](#)

## Amis

- istream & [operator](#)>> (istream &, [TenseurQ3geneHHBB](#) &)
- ostream & [operator](#)<< (ostream &, const [TenseurQ3geneHHBB](#) &)

## Membres hérités additionnels

### 6.859.1 Documentation des fonctions membres

#### 6.859.1.1 Affectation\_trans\_dimension()

```
void TenseurQ3geneHHBB::Affectation_trans_dimension (
    const TenseurHHBB & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.859.1.2 Change()

```
void TenseurQ3geneHHBB::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.859.1.3 ChangePlus()

```
void TenseurQ3geneHHBB::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHBB](#).

#### 6.859.1.4 Ecriture()

```
ostream & TenseurQ3geneHHBB::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.5 Inita()

```
void TenseurQ3geneHHBB::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.6 Lecture()

```
istream & TenseurQ3geneHHBB::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.7 MaxiComposante()

```
double TenseurQ3geneHHBB::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.8 operator&&()

```
TenseurHH & TenseurQ3geneHHBB::operator&& (
    const TenseurHH & ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.9 operator()()

```
double TenseurQ3geneHHBB::operator() (
    int i,
    int j,
    int k,
    int l ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.10 operator\*()

```
TenseurHHBB & TenseurQ3geneHHBB::operator* (
    const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.11 operator\*=( )

```
void TenseurQ3geneHHBB::operator*=(
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.12 operator+()

```
TenseurHHBB & TenseurQ3geneHHBB::operator+ (
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.13 operator+=()**

```
void TenseurQ3geneHHBB::operator+= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.14 operator-() [1/2]**

```
TenseurHHBB & TenseurQ3geneHHBB::operator- ( ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.15 operator-() [2/2]**

```
TenseurHHBB & TenseurQ3geneHHBB::operator- (
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.16 operator-=()**

```
void TenseurQ3geneHHBB::operator-= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.17 operator/()**

```
TenseurHHBB & TenseurQ3geneHHBB::operator/ (
    const double & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.18 operator/=()**

```
void TenseurQ3geneHHBB::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.19 operator=()**

```
TenseurHHBB & TenseurQ3geneHHBB::operator= (
    const TenseurHHBB & ) [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.20 operator==()**

```
int TenseurQ3geneHHBB::operator== (
    const TenseurHHBB & ) const [virtual]
```

Implémente [TenseurHHBB](#).

**6.859.1.21 Prod\_gauche()**

```
TenseurBB & TenseurQ3geneHHBB::Prod_gauche (
    const TenseurBB & F ) const [protected], [virtual]
```

Implémente [TenseurHHBB](#).

### 6.859.1.22 Transpose1et2avec3et4()

`TenseurBBHH` & `TenseurQ3geneHHBB::Transpose1et2avec3et4 ( ) const [virtual]`

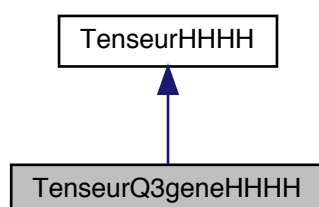
Implémente `TenseurHHBB`.

La documentation de cette classe a été générée à partir du fichier suivant :

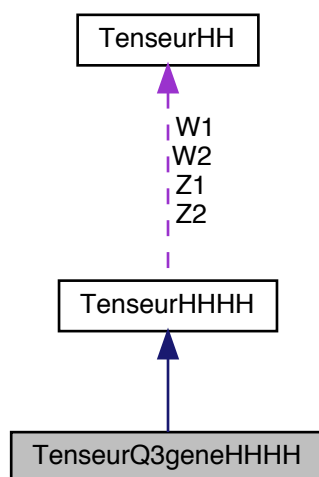
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3gene.h

## 6.860 Référence de la classe TenseurQ3geneHHHH

Graphe d'héritage de TenseurQ3geneHHHH:



Graphe de collaboration de TenseurQ3geneHHHH:



### Fonctions membres publiques

- `TenseurQ3geneHHHH` (const double val)
- `TenseurQ3geneHHHH` (int cas, const `TenseurHH` &aHH, const `TenseurHH` &bHH)

- **TenseurQ3geneHHHH** (int cas, const [Tenseur3HH](#) &aHH, const [Tenseur3HH](#) &bHH)
- **TenseurQ3geneHHHH** (const [TenseurHHHH](#) &)
- **TenseurQ3geneHHHH** (const [TenseurQ3geneHHHH](#) &)
- void **Inita** (double val)
- [TenseurHHHH](#) & **operator+** (const [TenseurHHHH](#) &) const
- void **operator+=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator-** () const
- [TenseurHHHH](#) & **operator-** (const [TenseurHHHH](#) &) const
- void **operator-=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator=** (const [TenseurHHHH](#) &)
- [TenseurHHHH](#) & **operator=** (const [TenseurQ3geneHHHH](#) &B)
- [TenseurHHHH](#) & **operator\*** (const double &) const
- void **operator\*=(** (const double &)
- [TenseurHHHH](#) & **operator/** (const double &) const
- void **operator/=** (const double &)
- [TenseurHH](#) & **operator&&** (const [TenseurBB](#) &) const
- [TenseurHHHH](#) & **operator&&** (const [TenseurBBHH](#) &) const
- [TenseurHHBB](#) & **operator&&** (const [TenseurBBBB](#) &) const
- [TenseurHHHH](#) & **Transpose1et2avec3et4** () const
- void **Affectation\_trans\_dimension** (const [TenseurHHHH](#) &B, bool plusZero)
- [TenseurHHHH](#) & **Symetrise1et2\_3et4** () const
- int **operator==** (const [TenseurHHHH](#) &) const
- void **Change** (int i, int j, int k, int l, const double &val)
- void **ChangePlus** (int i, int j, int k, int l, const double &val)
- double **operator()** (int i, int j, int k, int l) const
- double **MaxiComposante** () const
- istream & **Lecture** (istream &entree)
- ostream & **Ecriture** (ostream &sort) const

## Fonctions membres publiques statiques

- static [TenseurHHHH](#) & **Prod\_tensoriel** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & **Prod\_tensoriel\_barre** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)
- static [TenseurHHHH](#) & **Prod\_tensoriel\_under\_barre** (const [TenseurHH](#) &aHH, const [TenseurHH](#) &bHH)

## Fonctions membres protégées

- [TenseurHH](#) & **Prod\_gauche** (const [TenseurBB](#) &F) const
- [TenseurBBHH](#) & **Prod\_gauche** (const [TenseurBBBB](#) &F) const
- [TenseurHHHH](#) & **Prod\_gauche** (const [TenseurHHBB](#) &F) const

## Attributs protégés

- listdouble81iter **ipointe**

## Amis

- istream & **operator>>** (istream &, [TenseurQ3geneHHHH](#) &)
- ostream & **operator<<** (ostream &, const [TenseurQ3geneHHHH](#) &)

## Membres hérités additionnels

### 6.860.1 Documentation des fonctions membres

#### 6.860.1.1 Affectation\_trans\_dimension()

```
void TenseurQ3geneHHHH::Affectation_trans_dimension (
    const TenseurHHHH & B,
    bool plusZero ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.860.1.2 Change()

```
void TenseurQ3geneHHHH::Change (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.860.1.3 ChangePlus()

```
void TenseurQ3geneHHHH::ChangePlus (
    int i,
    int j,
    int k,
    int l,
    const double & val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.860.1.4 Ecriture()

```
ostream & TenseurQ3geneHHHH::Ecriture (
    ostream & sort ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.860.1.5 Inita()

```
void TenseurQ3geneHHHH::Inita (
    double val ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.860.1.6 Lecture()

```
istream & TenseurQ3geneHHHH::Lecture (
    istream & entree ) [virtual]
```

Implémente [TenseurHHHH](#).

### 6.860.1.7 MaxiComposante()

```
double TenseurQ3geneHHHH::MaxiComposante ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.860.1.8 operator&&()

```
TenseurHH & TenseurQ3geneHHHH::operator&& (
    const TenseurBB & ) const [virtual]
```

Implémente [TenseurHHHH](#).

### 6.860.1.9 operator>()()

```
double TenseurQ3geneHHHH::operator() (
    int i,
```



```
int j,  
int k,  
int l ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.860.1.10 operator\*()

```
TenseurHHHH & TenseurQ3geneHHHH::operator* (  
const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.860.1.11 operator\*=( )

```
void TenseurQ3geneHHHH::operator*= (  
const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.860.1.12 operator+( )

```
TenseurHHHH & TenseurQ3geneHHHH::operator+ (  
const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.860.1.13 operator+=( )

```
void TenseurQ3geneHHHH::operator+= (  
const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.860.1.14 operator-( ) [1/2]

```
TenseurHHHH & TenseurQ3geneHHHH::operator- ( ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.860.1.15 operator-( ) [2/2]

```
TenseurHHHH & TenseurQ3geneHHHH::operator- (  
const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.860.1.16 operator-=( )

```
void TenseurQ3geneHHHH::operator-=(  
const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

#### 6.860.1.17 operator/( )

```
TenseurHHHH & TenseurQ3geneHHHH::operator/ (  
const double & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.860.1.18 operator/=()**

```
void TenseurQ3geneHHHH::operator/= (
    const double & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.860.1.19 operator=()**

```
TenseurHHHH & TenseurQ3geneHHHH::operator= (
    const TenseurHHHH & ) [virtual]
```

Implémente [TenseurHHHH](#).

**6.860.1.20 operator==()**

```
int TenseurQ3geneHHHH::operator== (
    const TenseurHHHH & ) const [virtual]
```

Implémente [TenseurHHHH](#).

**6.860.1.21 Prod\_gauche()**

```
TenseurHH & TenseurQ3geneHHHH::Prod_gauche (
    const TenseurBB & F ) const [protected], [virtual]
```

Implémente [TenseurHHHH](#).

**6.860.1.22 Transpose1et2avec3et4()**

```
TenseurHHHH & TenseurQ3geneHHHH::Transpose1et2avec3et4 ( ) const [virtual]
```

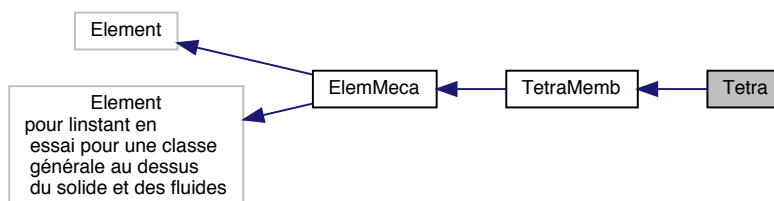
Implémente [TenseurHHHH](#).

La documentation de cette classe a été générée à partir du fichier suivant :

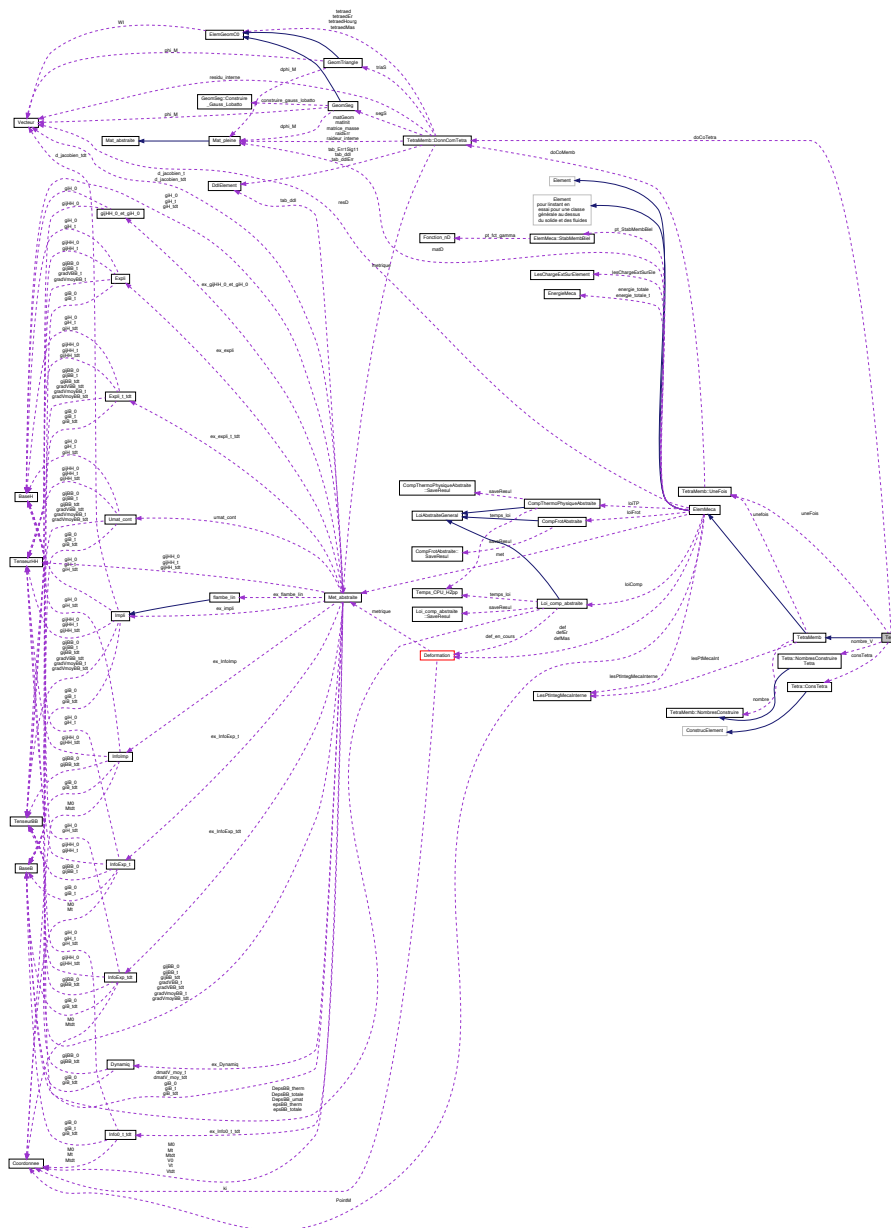
— /Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ3gene.h

**6.861 Référence de la classe Tetra**

Graphe d'héritage de Tetra:



Graphe de collaboration de Tetra:



## Classes

- class [ConsTetra](#)
- class [NombresConstruireTetra](#)

## Fonctions membres publiques

- **Tetra** (int num\_mail, int num\_id)
- **Tetra** (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **Tetra** (const [Tetra](#) &tetra)
- [Element](#) \* **Nevez\_copie** () const
- [Tetra](#) & **operator=** ([Tetra](#) &tetra)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `TetraMemb::DonnComTetra` \* `doCoTetra` = NULL
- static `TetraMemb::UneFois` `uneFois`
- static `NombresConstruireTetra` `nombre_V`
- static `ConstTetra` `consTetra`
- static int `bidon`

## Membres hérités additionnels

### 6.861.1 Documentation des fonctions membres

#### 6.861.1.1 new\_frontiere\_lin()

```
ElFrontiere * Tetra::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.861.1.2 new\_frontiere\_surf()

```
ElFrontiere * Tetra::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

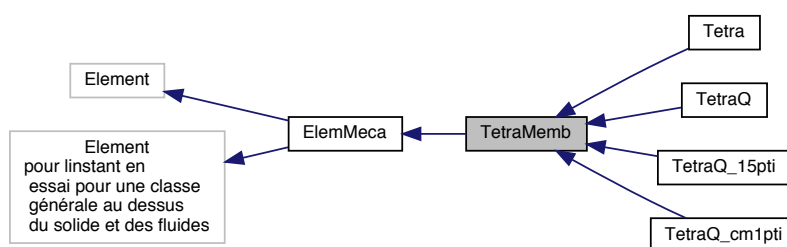
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

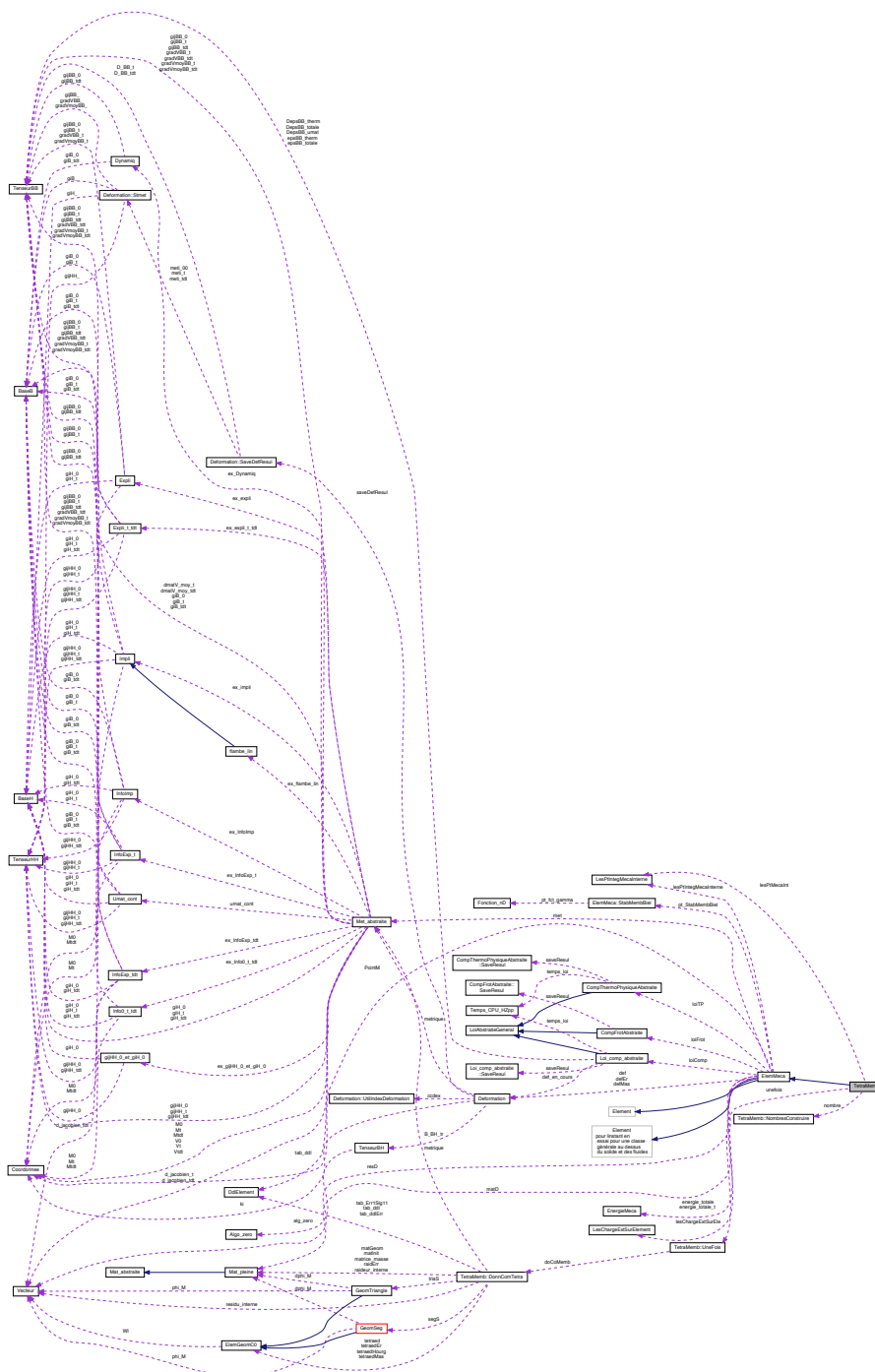
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/Tetra.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/Tetra.cc`

## 6.862 Référence de la classe TetraMemb

Graphe d'héritage de TetraMemb:



Graphe de collaboration de TetraMemb:



**Classes**

- class [DonnComTetra](#)
- class [NombresConstruire](#)
- class [UneFois](#)

**Fonctions membres publiques**

- **TetraMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")

- **TetraMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, const [Tableau](#)< [Noeud](#) \* > &tab, string info="")
- **TetraMemb** (const [TetraMemb](#) &tetraMem)
- **TetraMemb** & **operator=** ([TetraMemb](#) &tetraMem)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- [ElemGeomC0](#) & [ElementGeometrique](#) () const
- const [ElemGeomC0](#) & [ElementGeometrique\\_const](#) () const
- [Coordonnee](#) & [Point\\_physique](#) (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComple** ()
- Element \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- Element \* **Comple** **Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- bool **SurfExiste** (int ns) const
- bool **AreteExiste** (int na) const
- const [DdlElement](#) & [TableauDdl](#) () const
- Element::ResRaid **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- [DdlElement](#) & [Tableau\\_de\\_Sig1](#) () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- ResRaid **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_presUniDir\_E\_t** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_presUniDir\_E\_tdt** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)

- ResRaid **SMR\_charge\_presUniDir\_I** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_lineique\\_E\\_t](#) (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_lineique\\_E\\_tdt](#) (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_pression\\_E\\_t](#) (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_pression\\_E\\_tdt](#) (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_hydrostatique\\_E\\_t](#) (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sans\_←\_limitation)
- [Vecteur SM\\_charge\\_hydrostatique\\_E\\_tdt](#) (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- [Vecteur SM\\_charge\\_hydrodynamique\\_E\\_t](#) ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Vecteur SM\\_charge\\_hydrodynamique\\_E\\_tdt](#) ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Tableau< EIFrontiere \\* >](#) const & **Frontiere** (bool force=false)
- void **ConstTabDdl** ()

## Fonctions membres protégées

- [TetraMemb::DonnComTetra](#) \* **Init** ([ElemGeomC0](#) \*tetra, [ElemGeomC0](#) \*tetraEr, [ElemGeomC0](#) \*tetraMas, [ElemGeomC0](#) \*tetraeHourg, bool sans\_init\_noeud=false)
- void **Destruction** ()
- int [Dim\\_sig\\_eps](#) () const

## Attributs protégés

- [UneFois](#) \* **unefois**
- [LesPtIntegMecalInterne](#) **lesPtMecalnt**
- [NombresConstruire](#) \* **nombre**

## Membres hérités additionnels

### 6.862.1 Documentation des fonctions membres

#### 6.862.1.1 Active\_ddl\_Sigma()

void TetraMemb::Active\_ddl\_Sigma ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

#### 6.862.1.2 Active\_premier\_ddl\_Sigma()

void TetraMemb::Active\_premier\_ddl\_Sigma ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

### 6.862.1.3 ContraintesAbsolues()

```
bool TetraMemb::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.862.1.4 Dim\_sig\_eps()

```
int TetraMemb::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.862.1.5 ErreurElement()

```
void TetraMemb::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

### 6.862.1.6 Inactive\_ddl\_Sigma()

```
void TetraMemb::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.862.1.7 LectureContraintes()

```
void TetraMemb::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.862.1.8 Long\_arrete\_mini\_sur\_c()

```
double TetraMemb::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.862.1.9 Plus\_ddl\_Sigma()

```
void TetraMemb::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.862.1.10 Tableau\_de\_Sig1()

```
DdlElement & TetraMemb::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

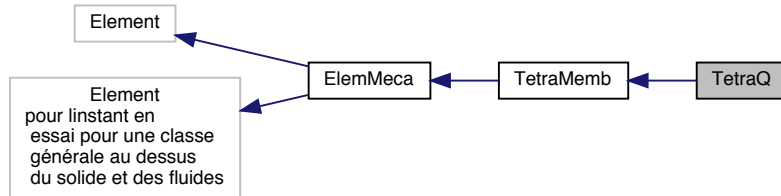
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraMemb.cc

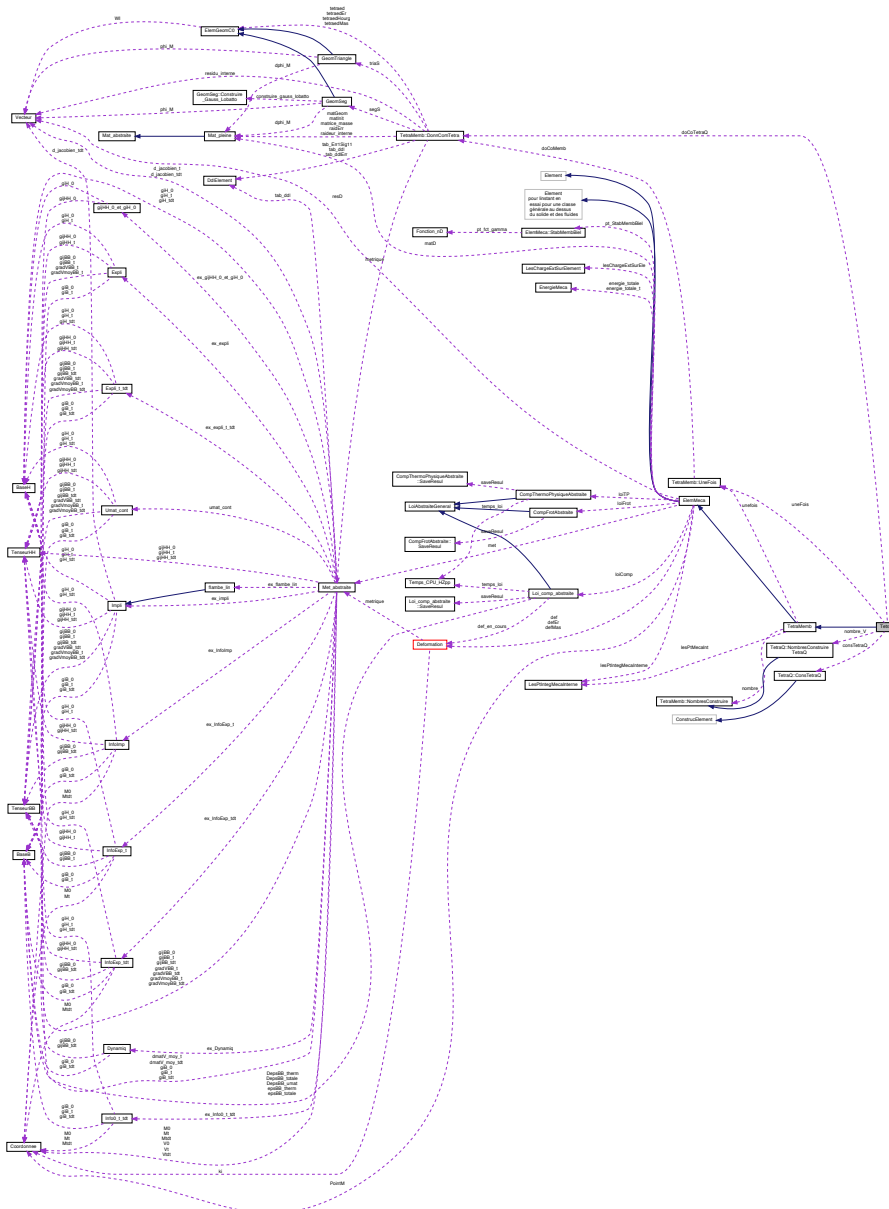


## 6.863 Référence de la classe TetraQ

Graphe d'héritage de TetraQ:



Graphe de collaboration de TetraQ:



## Classes

- class [ConsTetraQ](#)
- class [NombresConstruireTetraQ](#)

## Fonctions membres publiques

- [TetraQ](#) (int num\_mail, int num\_id)
- [TetraQ](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TetraQ](#) (const [TetraQ](#) &tetra)
- [Element](#) \* [Nevez\\_copie](#) () const
- [TetraQ](#) & [operator=](#) ([TetraQ](#) &tetra)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [TetraMemb::DonnComTetra](#) \* [doCoTetraQ](#) = NULL
- static [TetraMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireTetraQ](#) [nombre\\_V](#)
- static [ConsTetraQ](#) [consTetraQ](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.863.1 Documentation des fonctions membres

#### 6.863.1.1 new\_frontiere\_lin()

```
ElFrontiere * TetraQ::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.863.1.2 new\_frontiere\_surf()

```
ElFrontiere * TetraQ::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

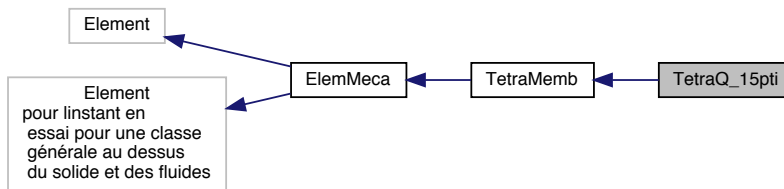
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

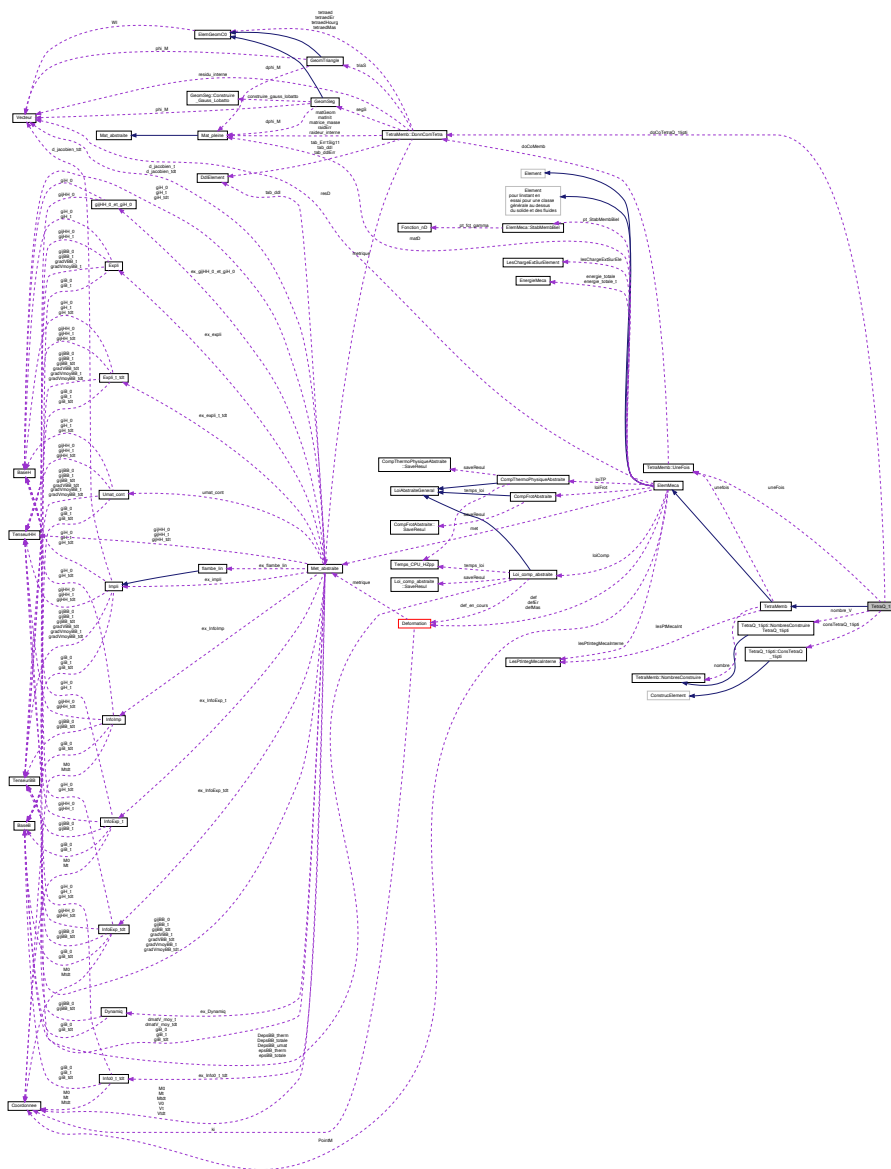
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ.cc

### 6.864 Référence de la classe TetraQ\_15pti

Grphe d'héritage de TetraQ\_15pti:



Grphe de collaboration de TetraQ\_15pti:



## Classes

- class [ConsTetraQ\\_15pti](#)
- class [NombresConstruireTetraQ\\_15pti](#)

## Fonctions membres publiques

- [TetraQ\\_15pti](#) (int num\_mail, int num\_id)
- [TetraQ\\_15pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TetraQ\\_15pti](#) (const [TetraQ\\_15pti](#) &tetra)
- [Element](#) \* [Nevez\\_copie](#) () const
- [TetraQ\\_15pti](#) & [operator=](#) ([TetraQ\\_15pti](#) &tetra)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [TetraMemb::DonnComTetra](#) \* [doCoTetraQ\\_15pti](#) = NULL
- static [TetraMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireTetraQ\\_15pti](#) [nombre\\_V](#)
- static [ConsTetraQ\\_15pti](#) [consTetraQ\\_15pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.864.1 Documentation des fonctions membres

#### 6.864.1.1 new\_frontiere\_lin()

```
ElFrontiere * TetraQ_15pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.864.1.2 new\_frontiere\_surf()

```
ElFrontiere * TetraQ_15pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

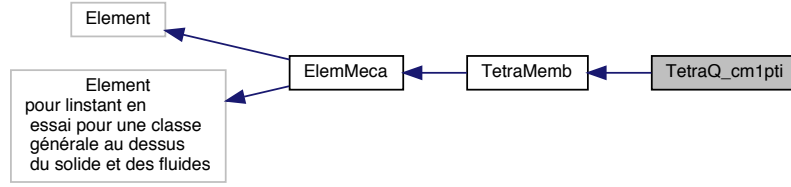
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

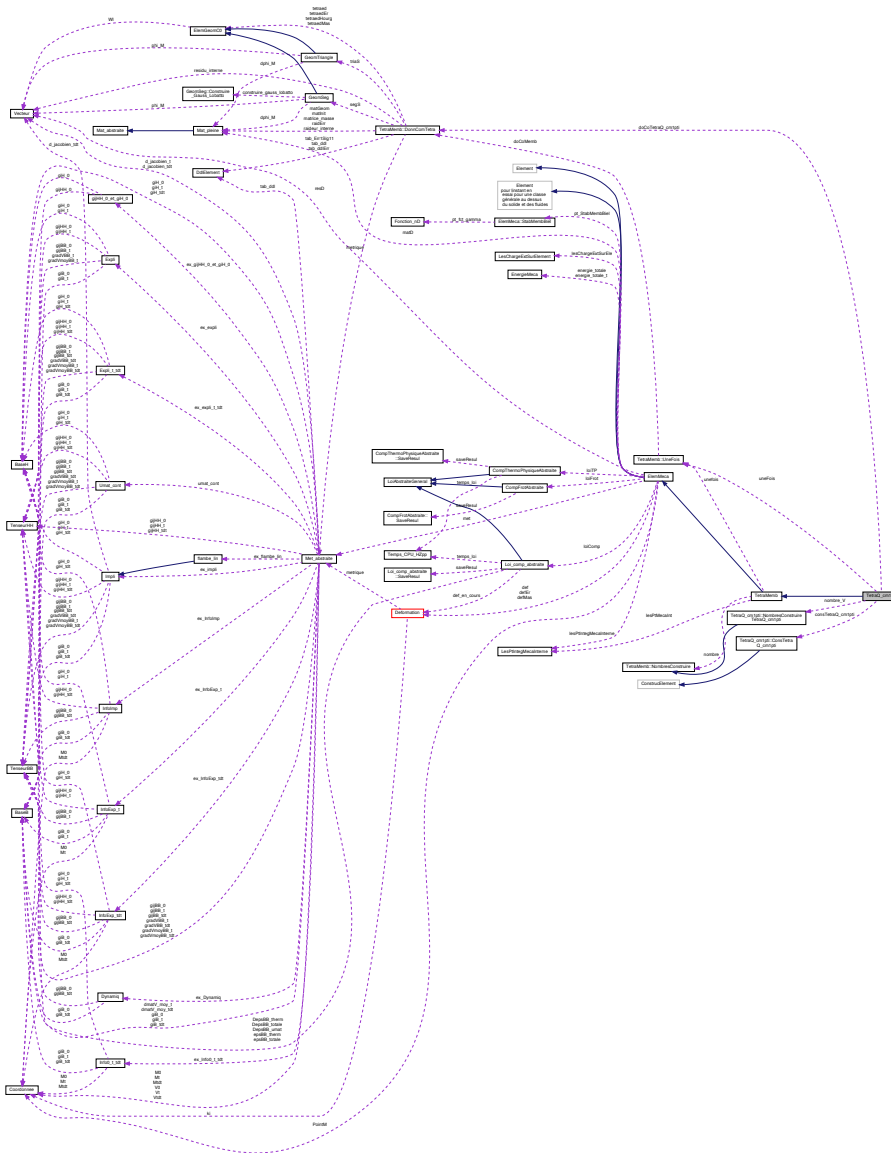
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm15pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm15pti.cc

## 6.865 Référence de la classe TetraQ\_cm1pti

Graphe d'héritage de TetraQ\_cm1pti:



Graphe de collaboration de TetraQ\_cm1pti:



## Classes

- class [ConsTetraQ\\_cm1pti](#)
- class [NombresConstruireTetraQ\\_cm1pti](#)

## Fonctions membres publiques

- [TetraQ\\_cm1pti](#) (int num\_mail, int num\_id)
- [TetraQ\\_cm1pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TetraQ\\_cm1pti](#) (const [TetraQ\\_cm1pti](#) &tetra)
- [Element](#) \* [Nevez\\_copie](#) () const
- [TetraQ\\_cm1pti](#) & [operator=](#) ([TetraQ\\_cm1pti](#) &tetra)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [TetraMemb::DonnComTetra](#) \* [doCoTetraQ\\_cm1pti](#) = NULL
- static [TetraMemb::UneFois](#) [uneFois](#)
- static [NombresConstruireTetraQ\\_cm1pti](#) [nombre\\_V](#)
- static [ConsTetraQ\\_cm1pti](#) [consTetraQ\\_cm1pti](#)
- static int [bidon](#)

## Membres hérités additionnels

### 6.865.1 Documentation des fonctions membres

#### 6.865.1.1 new\_frontiere\_lin()

```
ElFrontiere * TetraQ\_cm1pti::new\_frontiere\_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.865.1.2 new\_frontiere\_surf()

```
ElFrontiere * TetraQ\_cm1pti::new\_frontiere\_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraQ\_cm1pti.cc

## 6.866 Référence de la classe ThermoDonnee

### Fonctions membres publiques

- [ThermoDonnee](#) (const [ThermoDonnee](#) &co)
- const double & [Dilatation](#) () const
- const double & [Conductivite](#) () const

- const double & **CapaciteCalorifique** () const
- const double & **Compressibilite** () const
- bool **ActiveDilatation** () const
- const double \* **TauxCrista** () const
- void **ChangeDilatation** (double dila)
- void **ChangeAlphaTLambdaCp** (const double &alph, const double &lamb, const double &ccc)
- void **ChangeCompressibilite** (const double compress)
- void **ChangeTauxCrista** (const double taux\_cris)

### Amis

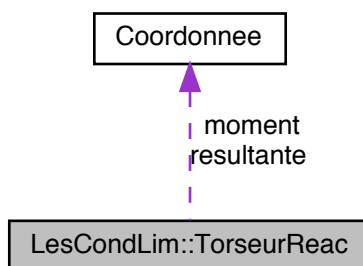
- istream & **operator**>> (istream &, [ThermoDonnee](#) &)
- ostream & **operator**<< (ostream &, const [ThermoDonnee](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/thermique/ThermoDonnee.h

## 6.867 Référence de la classe LesCondLim::TorseurReac

Graphe de collaboration de LesCondLim::TorseurReac:



### Fonctions membres publiques

- **TorseurReac** (const [TorseurReac](#) &a)
- void **Activation** (int dima)
- void **Zero\_init\_torseur** ()

### Attributs publics

- bool **existe\_torseur\_reac**
- [Coordonnee](#) **resultante**
- [Coordonnee](#) **moment**
- bool **bloque\_ou\_CLL**

### Amis

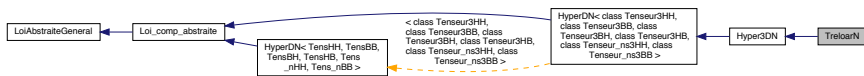
- istream & **operator**>> (istream &, [TorseurReac](#) &)
- ostream & **operator**<< (ostream &, const [TorseurReac](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

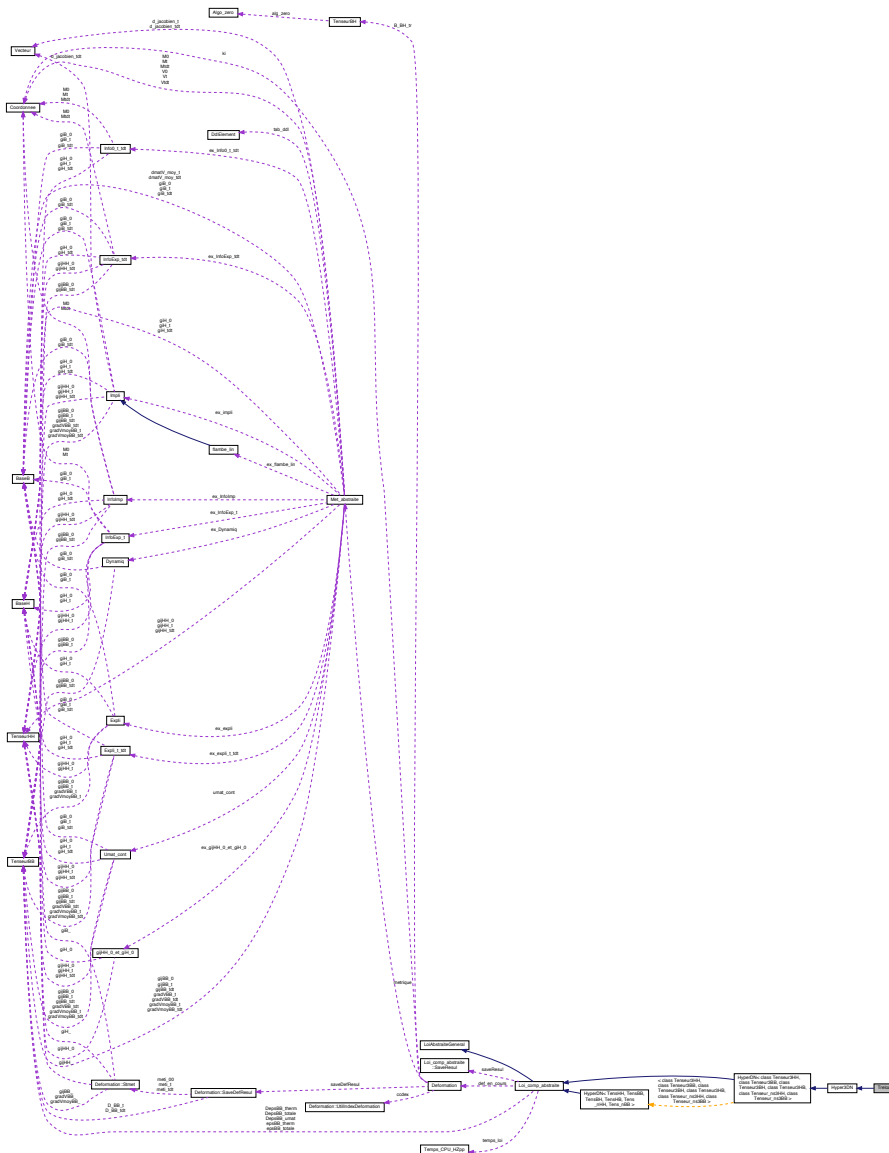
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/LesCondLim.h

## 6.868 Référence de la classe TreloarN

Graphe d'héritage de TreloarN:



Graphe de collaboration de TreloarN:



### Fonctions membres publiques

- **TreloarN** (const [TreloarN](#) &loi)
- void [LectureDonneesParticulieres](#) ([UtilLecture](#) \*, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void [Affiche](#) () const
- int [TestComple](#) ()



- void `Lecture_base_info_loi` (ifstream &ent, const int cas, [LesReferences](#) &lesRef, [LesCourbes1D](#) &lesCourbes1D, [LesFonctions\\_nD](#) &lesFonctionsnD)
- void `Ecriture_base_info_loi` (ofstream &sort, const int cas)
- void `Info_commande_LoisDeComp` ([UtilLecture](#) &lec)
- `Loi_comp_abstraite * Nouvelle_loi_identique` () const
- void `Potentiel` (double &jacobien\_0, double &leps, double &V, double &b1lb, double &E, double &EV, double &Eb1lb, double &E1leps)
- void `Potentiel_et_var` (double &jacobien\_0, double &leps, double &V, double &b1lb, double &E, double &EV, double &Eb1lb, double &E1leps, double &EVV, double &Eb1lb2, double &E1leps2, double &EVb1lb, double &EV1leps, double &Eb1lb1leps)

## Attributs publics

- double `K`
- double `C`

## Membres hérités additionnels

### 6.868.1 Documentation des fonctions membres

#### 6.868.1.1 Affiche()

```
void TreloarN::Affiche ( ) const [virtual]
Implémente LoiAbstraiteGeneral.
```

#### 6.868.1.2 Ecriture\_base\_info\_loi()

```
void TreloarN::Ecriture_base_info_loi (
    ofstream & sort,
    const int cas ) [virtual]
Implémente LoiAbstraiteGeneral.
```

#### 6.868.1.3 Info\_commande\_LoisDeComp()

```
void TreloarN::Info_commande_LoisDeComp (
    UtilLecture & lec ) [virtual]
Implémente LoiAbstraiteGeneral.
```

#### 6.868.1.4 Lecture\_base\_info\_loi()

```
void TreloarN::Lecture_base_info_loi (
    ifstream & ent,
    const int cas,
    LesReferences & lesRef,
    LesCourbes1D & lesCourbes1D,
    LesFonctions\_nD & lesFonctionsnD ) [virtual]
Implémente LoiAbstraiteGeneral.
```

#### 6.868.1.5 LectureDonneesParticulieres()

```
void TreloarN::LectureDonneesParticulieres (
    UtilLecture * entreePrinc,
    LesCourbes1D & lesCourbes1D,
    LesFonctions\_nD & lesFonctionsnD ) [virtual]
```

Implémente [LoiAbstraiteGeneral](#).

#### 6.868.1.6 Nouvelle\_loi\_identique()

```
Loi_comp_abstraite * TreloarN::Nouvelle_loi_identique ( ) const [inline], [virtual]
```

Implémente [Loi\\_comp\\_abstraite](#).

#### 6.868.1.7 Potentiel()

```
void TreloarN::Potentiel (
    double & jacobien_0,
    double & Ieps,
    double & V,
    double & bIIb,
    double & E,
    double & EV,
    double & EbIIb,
    double & EIeps ) [virtual]
```

Implémente [HyperDN< class Tenseur3HH, class Tenseur3BB, class Tenseur3BH, class Tenseur3HB, class Tenseur\\_ns3HH, class T](#)

#### 6.868.1.8 Potentiel\_et\_var()

```
void TreloarN::Potentiel_et_var (
    double & jacobien_0,
    double & Ieps,
    double & V,
    double & bIIb,
    double & E,
    double & EV,
    double & EbIIb,
    double & EIeps,
    double & EVV,
    double & EbIIb2,
    double & EIeps2,
    double & EVbIIb,
    double & EVIeps,
    double & EbIIbIeps ) [virtual]
```

Implémente [HyperDN< class Tenseur3HH, class Tenseur3BB, class Tenseur3BH, class Tenseur3HB, class Tenseur\\_ns3HH, class T](#)

#### 6.868.1.9 TestComplet()

```
int TreloarN::TestComplet ( ) [virtual]
```

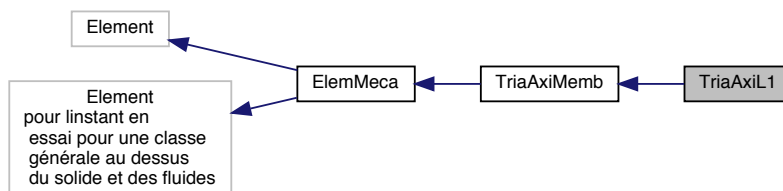
Réimplémentée à partir de [LoiAbstraiteGeneral](#).

La documentation de cette classe a été générée à partir du fichier suivant :

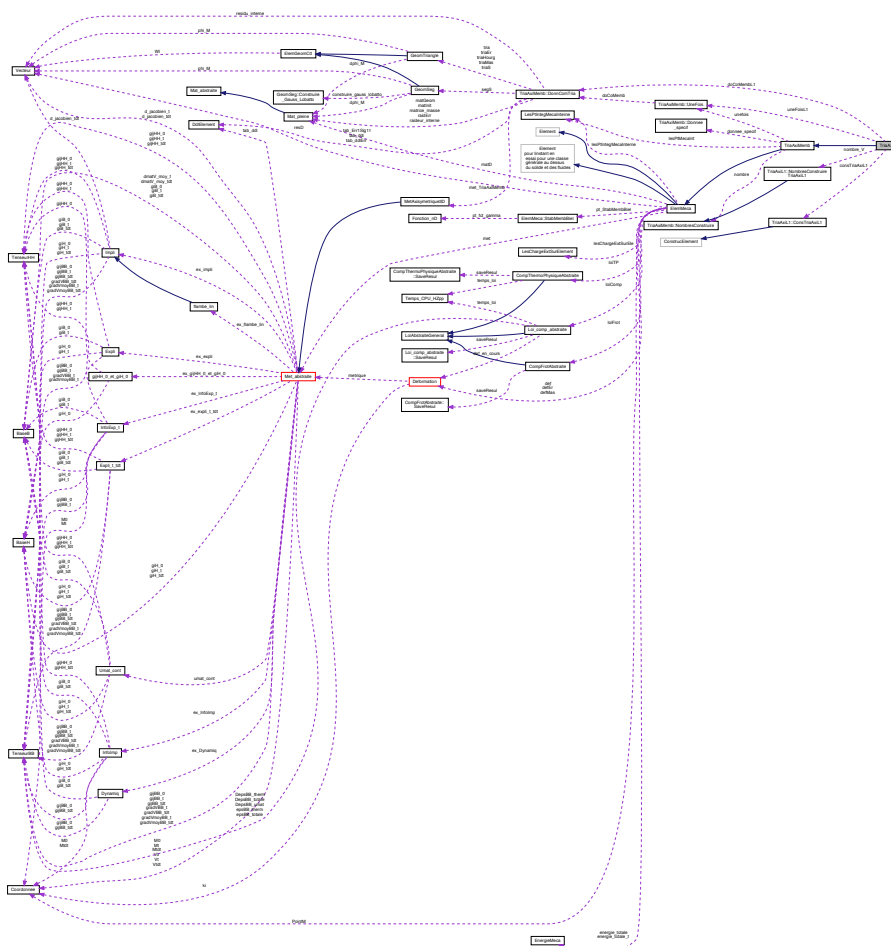
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/TreloarN.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/Hyper\_elastique/TreloarN.cc

## 6.869 Référence de la classe TriaAxiL1

Graphe d'héritage de TriaAxiL1:



Graphe de collaboration de TriaAxiL1:



### Classes

- class [ConsTriaAxiL1](#)
- class [NombresConstruireTriaAxiL1](#)

### Fonctions membres publiques

- [TriaAxiL1](#) (int num\_mail, int num\_id)

- `TriaAxiL1` (int num\_mail, int num\_id, const `Tableau< Noeud * >` &tab)
- `TriaAxiL1` (const `TriaAxiL1` &tria)
- `Element * Nevez_copie` () const
- `TriaAxiL1 & operator=` (`TriaAxiL1` &tria)
- void `AfficheVarDual` (ofstream &sort, `Tableau< string >` &nom)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `TriaAxiMemb::DonnComTria * doCoMembL1` = NULL
- static `TriaAxiMemb::UneFois uneFoisL1`
- static `NombresConstruireTriaAxiL1 nombre_V`
- static `ConsTriaAxiL1 consTriaAxiL1`

## Membres hérités additionnels

### 6.869.1 Documentation des fonctions membres

#### 6.869.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaAxiL1::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.869.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaAxiL1::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

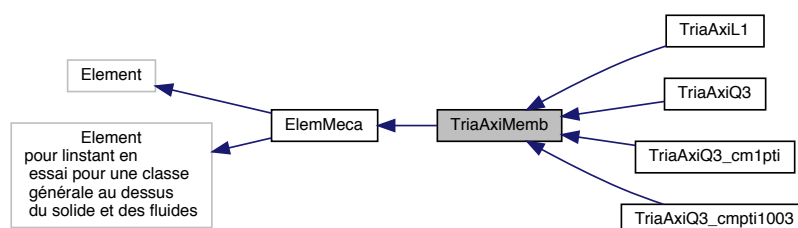
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

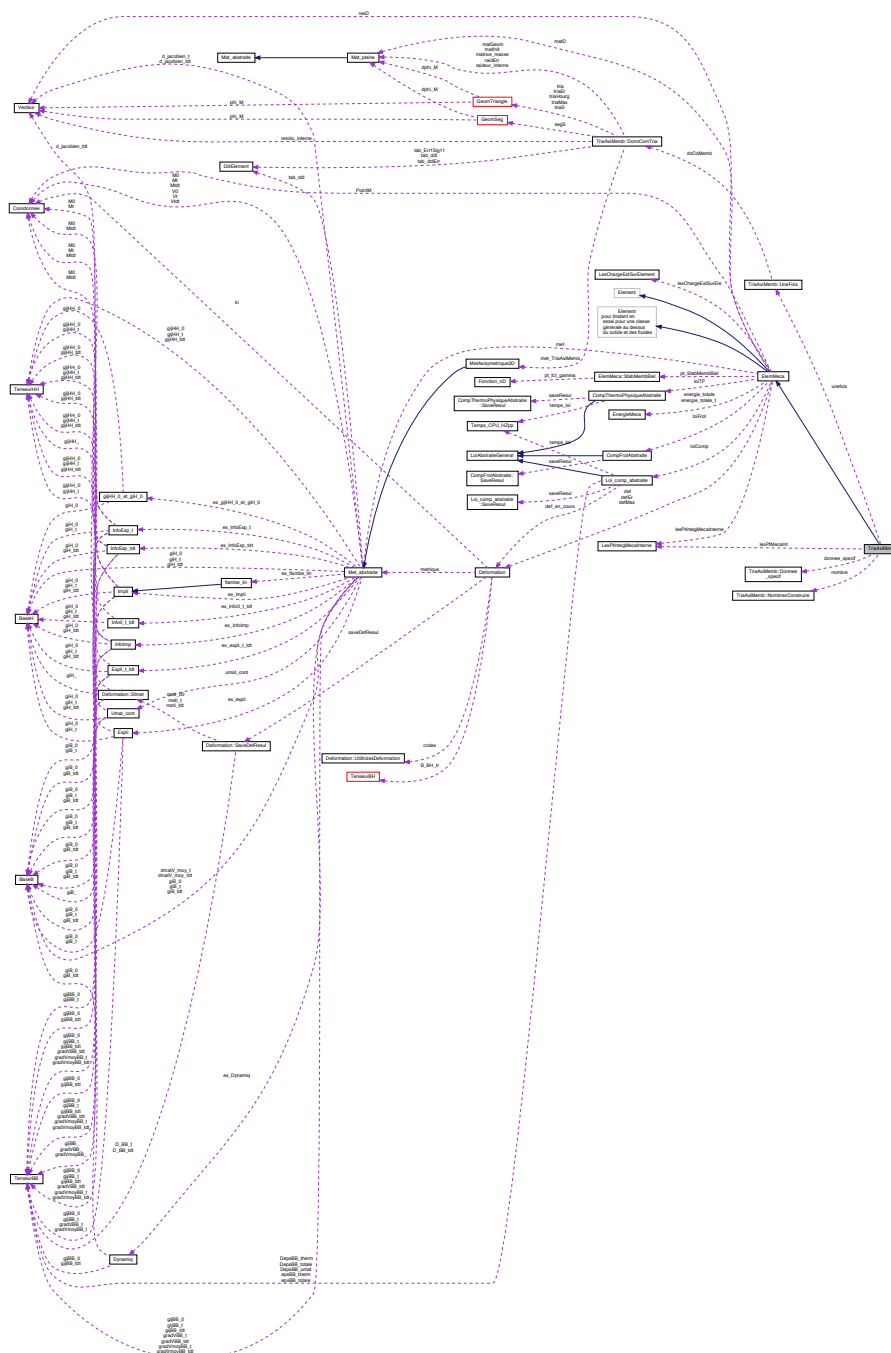
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiL1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiL1.cc

## 6.870 Référence de la classe TriaAxiMemb

Graphe d'héritage de `TriaAxiMemb`:



Graphe de collaboration de TriaAxiMemb:



## Classes

- class [DonnComTria](#)
- class [Donne\\_specif](#)
- class [NombresConstruire](#)
- class [UneFois](#)

## Fonctions membres publiques

- **TriaAxiMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")

- **TriaAxiMemb** (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, const [Tableau](#)< [Noeud](#) \* > &tab, string info="")
- **TriaAxiMemb** (const [TriaAxiMemb](#) &tria)
- **TriaAxiMemb** & **operator=** (const [TriaAxiMemb](#) &tria)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- [ElemGeomC0](#) & **ElementGeometrique** () const
- const [ElemGeomC0](#) & **ElementGeometrique\_const** () const
- [Coordonnee](#) & **Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComple** ()
- Element \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- Element \* **Comple** **Hourglass** ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- bool **SurfExiste** (int ns) const
- bool **AreteExiste** (int na) const
- const [DdlElement](#) & **TableauDdl** () const
- Element::ResRaid **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- [DdlElement](#) & **Tableau\_de\_Sig1** () const
  - !!!!!!! pour l'instant en virtuelle il faudra après en
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
  - !!!!!!! pour l'instant en virtuelle il faudra après en
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- ResRaid **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_lineique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_lineique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)

- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- **Vecteur SM\_charge\_lineique\_Suiv\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- **Vecteur SM\_charge\_lineique\_Suiv\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- **Vecteur SM\_charge\_pression\_E\_t** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- **Vecteur SM\_charge\_pression\_E\_tdt** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- **Vecteur SM\_charge\_hydrostatique\_E\_t** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- **Vecteur SM\_charge\_hydrostatique\_E\_tdt** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- **Vecteur SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- **Vecteur SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- [Tableau](#)< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- void **ConstTabDdl** ()

## Fonctions membres protégées

- [TriaAxiMemb::DonnComTria](#) \* **Init** (bool sans\_init\_noeud=false)
- [TriaAxiMemb::DonnComTria](#) \* **Init** ([Donnee\\_specif](#) donnee\_specif, bool sans\_init\_noeud=false)
- void **Destruction** ()
- int **Dim\_sig\_eps** () const

## Attributs protégés

- [UneFois](#) \* **unefois**
- [Donnee\\_specif](#) **donnee\_specif**
- [LesPtIntegMecalInterne](#) **lesPtMecalnt**
- [NombresConstruire](#) \* **nombre**

## Membres hérités additionnels

### 6.870.1 Documentation des fonctions membres

#### 6.870.1.1 Active\_ddl\_Sigma()

void [TriaAxiMemb::Active\\_ddl\\_Sigma](#) ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

#### 6.870.1.2 Active\_premier\_ddl\_Sigma()

void [TriaAxiMemb::Active\\_premier\\_ddl\\_Sigma](#) ( ) [inline], [virtual]  
 Implémente [ElemMeca](#).

### 6.870.1.3 ContraintesAbsolues()

```
bool TriaAxiMemb::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.870.1.4 Dim\_sig\_eps()

```
int TriaAxiMemb::Dim_sig_eps ( ) const [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

### 6.870.1.5 ErreurElement()

```
void TriaAxiMemb::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

### 6.870.1.6 Inactive\_ddl\_Sigma()

```
void TriaAxiMemb::Inactive_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.870.1.7 LectureContraintes()

```
void TriaAxiMemb::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.870.1.8 Long\_arrete\_mini\_sur\_c()

```
double TriaAxiMemb::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.870.1.9 Plus\_ddl\_Sigma()

```
void TriaAxiMemb::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

### 6.870.1.10 Tableau\_de\_Sig1()

```
DdlElement & TriaAxiMemb::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

Réimplémentée à partir de [ElemMeca](#).

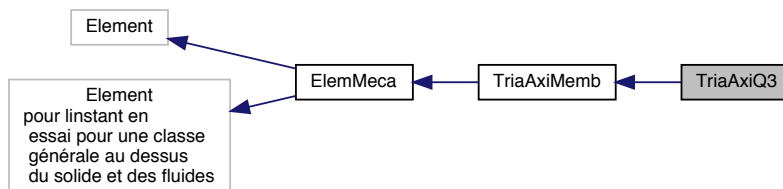
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxi↔Memb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxi↔Memb.cc

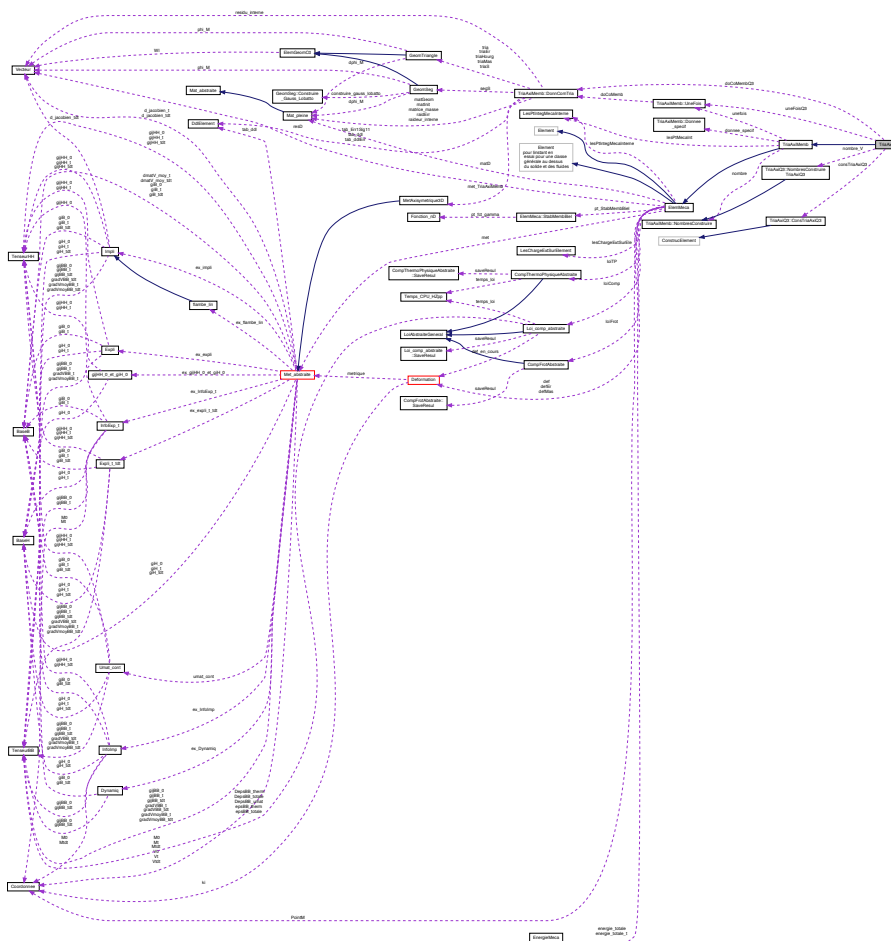


## 6.871 Référence de la classe TriaAxiQ3

Graphe d'héritage de TriaAxiQ3:



Graphe de collaboration de TriaAxiQ3:



### Classes

- class [ConsTriaAxiQ3](#)
- class [NombresConstruireTriaAxiQ3](#)

### Fonctions membres publiques

- [TriaAxiQ3](#) (int num\_mail, int num\_id)

- `TriaAxiQ3` (int num\_mail, int num\_id, const `Tableau< Noeud * >` &tab)
- `TriaAxiQ3` (const `TriaAxiQ3` &tria)
- `Element * Nevez_copie` () const
- `TriaAxiQ3 & operator=` (`TriaAxiQ3` &tria)
- void `AfficheVarDual` (ofstream &sort, `Tableau< string >` &nom)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau< Noeud * >` &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `TriaAxiMemb::DonnComTria * doCoMembQ3` = NULL
- static `TriaAxiMemb::UneFois uneFoisQ3`
- static `NombresConstruireTriaAxiQ3 nombre_V`
- static `ConsTriaAxiQ3 consTriaAxiQ3`

## Membres hérités additionnels

### 6.871.1 Documentation des fonctions membres

#### 6.871.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaAxiQ3::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.871.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaAxiQ3::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

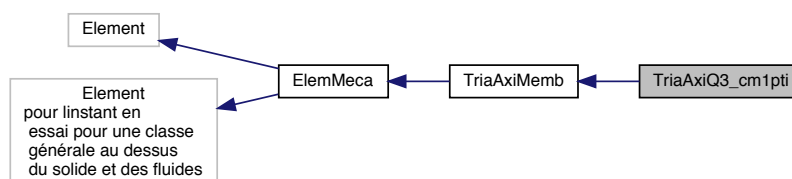
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

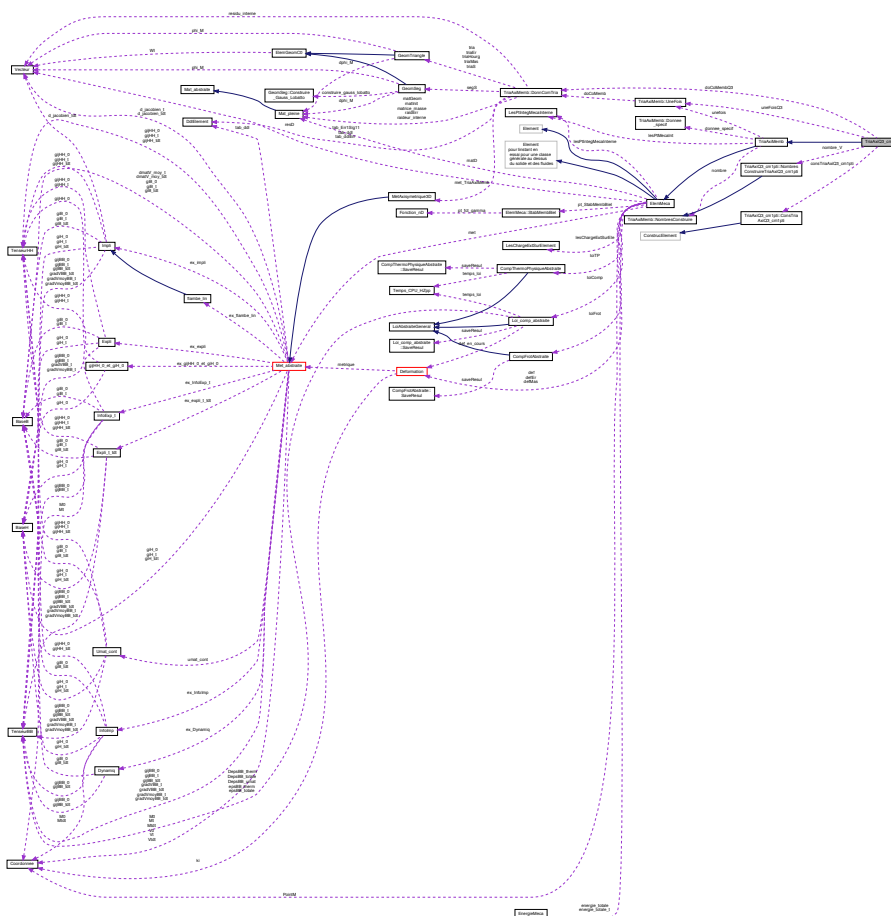
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3.cc

## 6.872 Référence de la classe TriaAxiQ3\_cm1pti

Grappe d'héritage de `TriaAxiQ3_cm1pti`:



Graphe de collaboration de TriaAxiQ3\_cm1pti:



## Classes

- class [ConsTriaAxiQ3\\_cm1pti](#)
- class [NombresConstruireTriaAxiQ3\\_cm1pti](#)

## Fonctions membres publiques

- [TriaAxiQ3\\_cm1pti](#) (int num\_mail, int num\_id)
- [TriaAxiQ3\\_cm1pti](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaAxiQ3\\_cm1pti](#) (const [TriaAxiQ3\\_cm1pti](#) &tria)
- [Element](#) \* [Nevez\\_copie](#) () const
- [TriaAxiQ3\\_cm1pti](#) & [operator=](#) ([TriaAxiQ3\\_cm1pti](#) &tria)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [TriaAxiMemb::DonnComTria](#) \* [doCoMembQ3](#) = NULL
- static [TriaAxiMemb::UneFois](#) [uneFoisQ3](#)
- static [NombresConstruireTriaAxiQ3\\_cm1pti](#) [nombre\\_V](#)
- static [ConsTriaAxiQ3\\_cm1pti](#) [consTriaAxiQ3\\_cm1pti](#)

## Membres hérités additionnels

### 6.872.1 Documentation des fonctions membres

#### 6.872.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaAxiQ3_cmlpti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.872.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaAxiQ3_cmlpti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

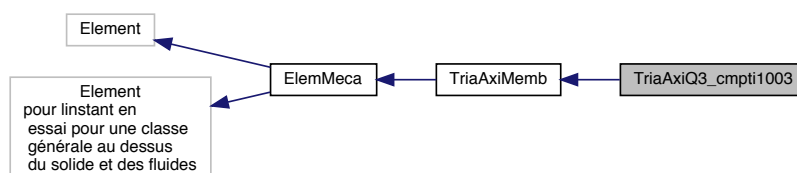
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

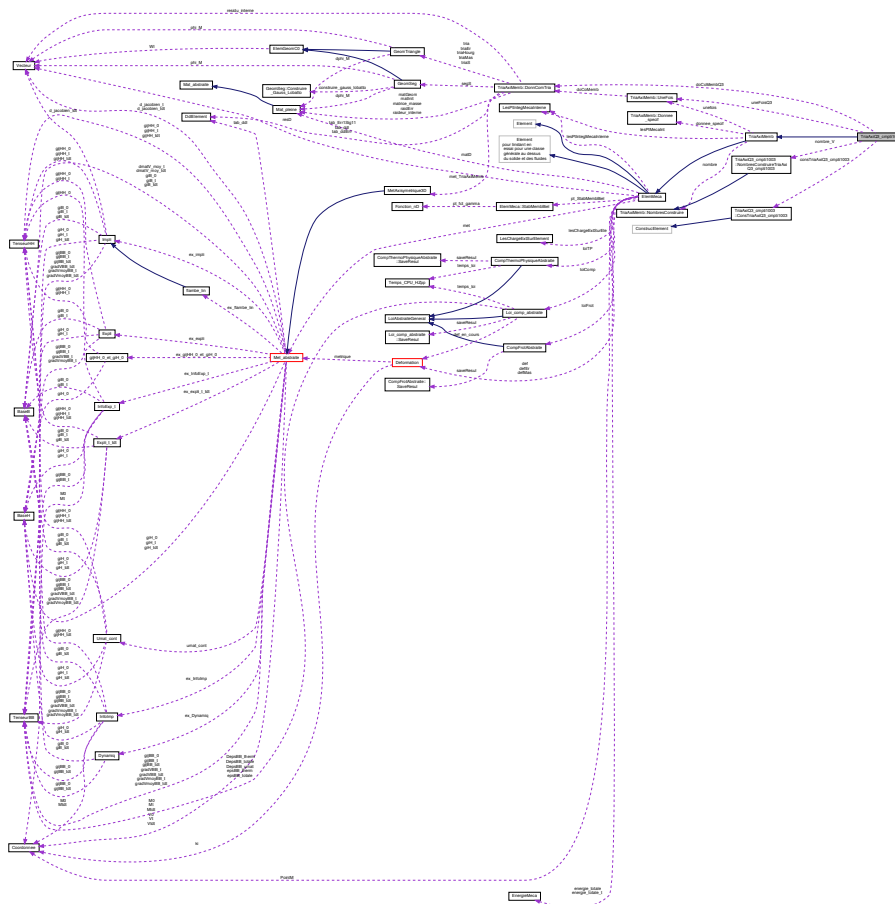
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_↔cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_↔cm1pti.cc

## 6.873 Référence de la classe TriaAxiQ3\_cmpti1003

Graphe d'héritage de TriaAxiQ3\_cmpti1003:



Graphe de collaboration de TriaAxiQ3\_cmpti1003:



## Classes

- class [ConsTriaAxiQ3\\_cmpti1003](#)
- class [NombresConstruireTriaAxiQ3\\_cmpti1003](#)

## Fonctions membres publiques

- [TriaAxiQ3\\_cmpti1003](#) (int num\_mail, int num\_id)
- [TriaAxiQ3\\_cmpti1003](#) (int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaAxiQ3\\_cmpti1003](#) (const [TriaAxiQ3\\_cmpti1003](#) &tria)
- [Element](#) \* [Nevez\\_copie](#) () const
- [TriaAxiQ3\\_cmpti1003](#) & [operator=](#) ([TriaAxiQ3\\_cmpti1003](#) &tria)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [TriaAxiMemb::DonnComTria](#) \* [doCoMembQ3](#) = NULL
- static [TriaAxiMemb::UneFois](#) [uneFoisQ3](#)
- static [NombresConstruireTriaAxiQ3\\_cmpti1003](#) [nombre\\_V](#)
- static [ConsTriaAxiQ3\\_cmpti1003](#) [consTriaAxiQ3\\_cmpti1003](#)

## Membres hérités additionnels

### 6.873.1 Documentation des fonctions membres

#### 6.873.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaAxiQ3_cmpti1003::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.873.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaAxiQ3_cmpti1003::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

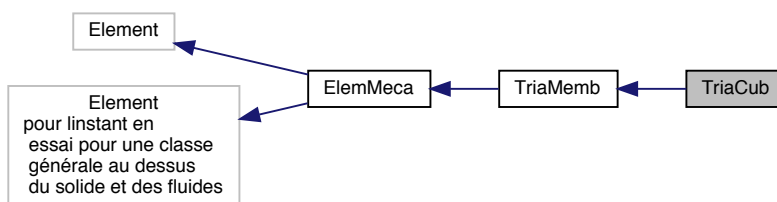
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

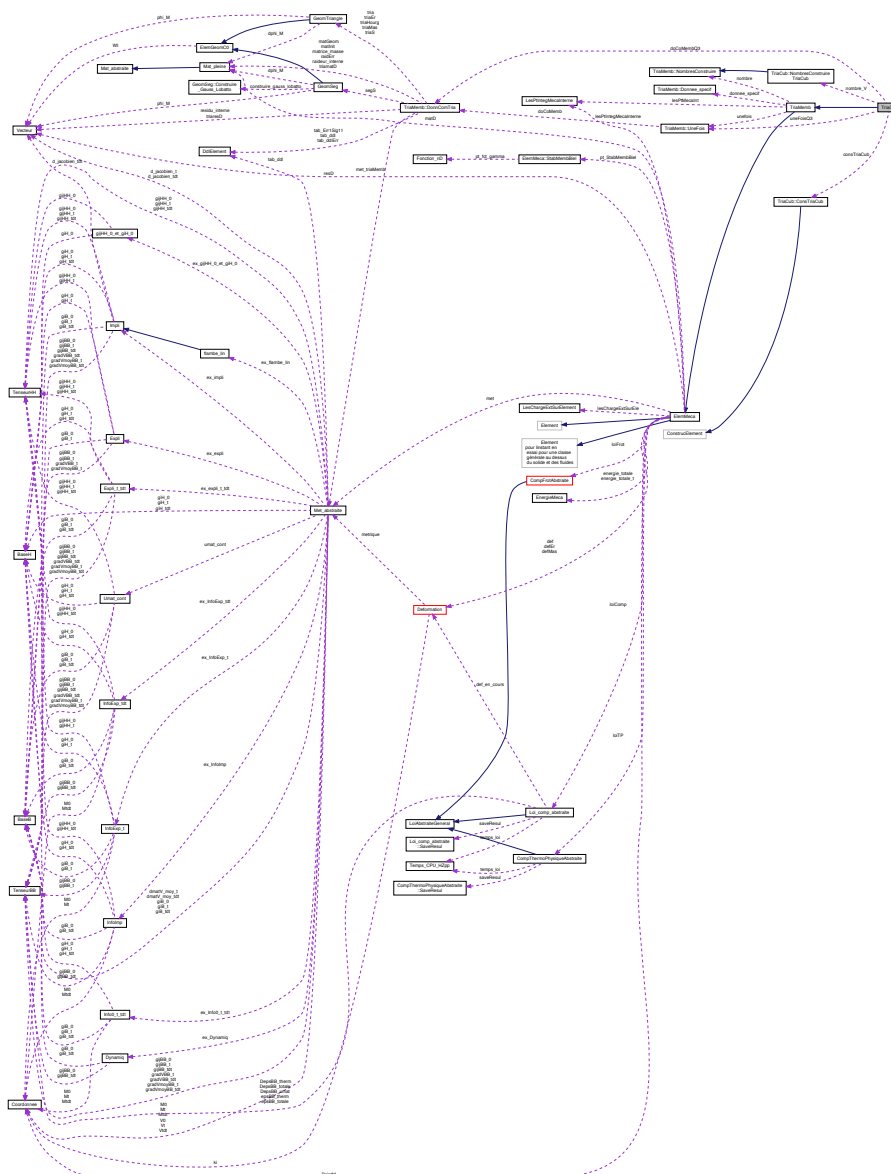
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_↔  
cmpti1003.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiQ3\_↔  
cmpti1003.cc

## 6.874 Référence de la classe TriaCub

Graphe d'héritage de TriaCub:



Graphe de collaboration de TriaCub:



## Classes

- class [ConsTriaCub](#)
- class [NombresConstruireTriaCub](#)

## Fonctions membres publiques

- **TriaCub** (double epaiss, int num\_mail=0, int num\_id=-3)
- **TriaCub** (int num\_mail, int num\_id)
- **TriaCub** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **TriaCub** (const [TriaCub](#) &tria)
- Element \* **Nevez\_copie** () const
- [TriaCub](#) & **operator=** ([TriaCub](#) &tria)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

— [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

### Attributs protégés statiques

— static [TriaMemb::DonnComTria](#) \* [doCoMembQ3](#) = NULL  
 — static [TriaMemb::UneFois](#) [uneFoisQ3](#)  
 — static [NombresConstruireTriaCub](#) [nombre\\_V](#)  
 — static [ConstriaCub](#) [consTriaCub](#)

### Membres hérités additionnels

#### 6.874.1 Documentation des fonctions membres

##### 6.874.1.1 [new\\_frontiere\\_lin\(\)](#)

```
ElFrontiere * TriaCub::new\_frontiere\_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.874.1.2 [new\\_frontiere\\_surf\(\)](#)

```
ElFrontiere * TriaCub::new\_frontiere\_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

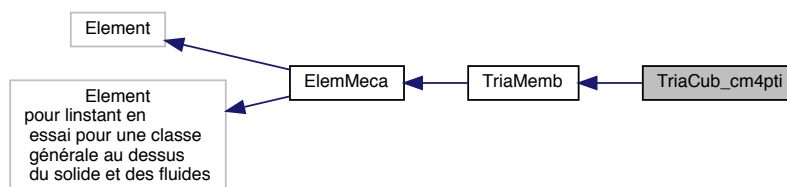
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub.cc

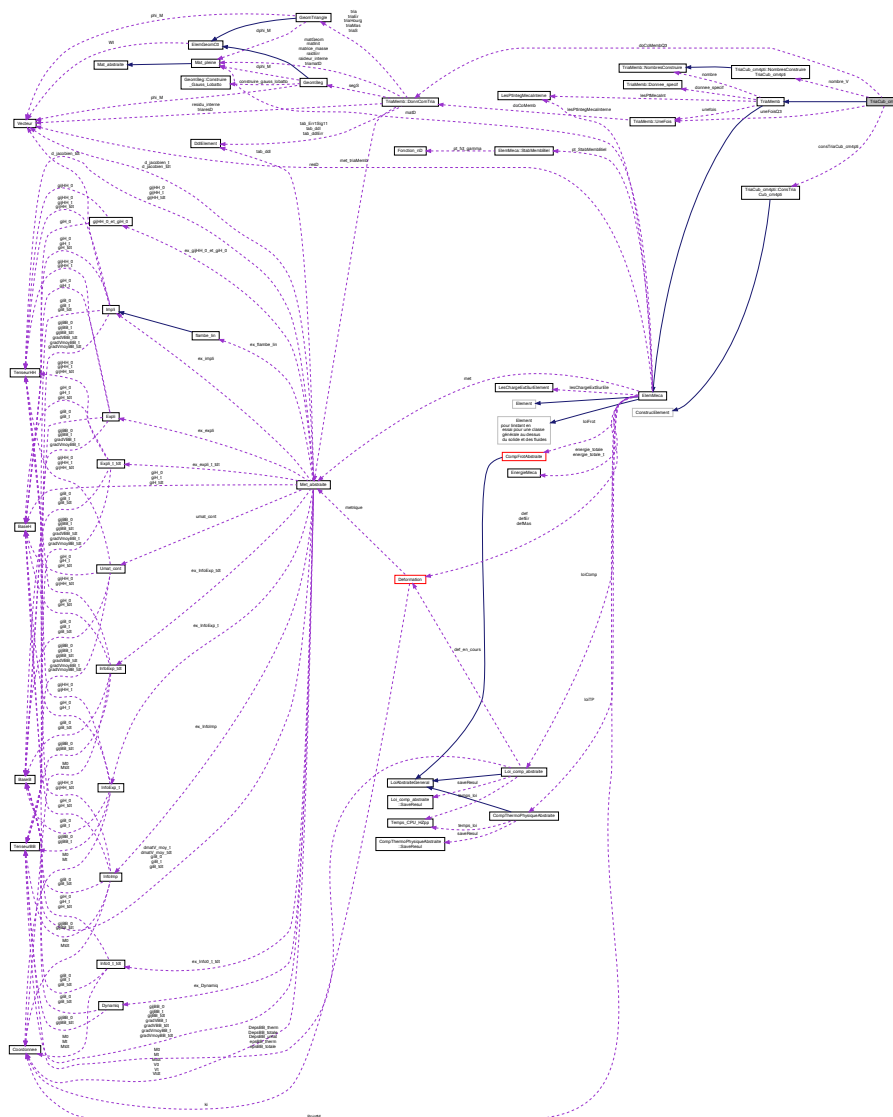
## 6.875 Référence de la classe [TriaCub\\_cm4pti](#)

Graphe d'héritage de [TriaCub\\_cm4pti](#):





Graphe de collaboration de TriaCub\_cm4pti:



## Classes

- class [ConsTriaCub\\_cm4pti](#)
- class [NombresConstruireTriaCub\\_cm4pti](#)

## Fonctions membres publiques

- [TriaCub\\_cm4pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaCub\\_cm4pti](#) (int num\_mail, int num\_id)
- [TriaCub\\_cm4pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaCub\\_cm4pti](#) (const [TriaCub\\_cm4pti](#) &tria)
- [Element](#) \* [Nevez\\_copie](#) () const
- [TriaCub\\_cm4pti](#) & [operator=](#) ([TriaCub\\_cm4pti](#) &tria)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [TriaMemb::DonnComTria](#) \* **doCoMembQ3** = NULL
- static [TriaMemb::UneFois](#) **uneFoisQ3**
- static [NombresConstruireTriaCub\\_cm4pti](#) **nombre\_V**
- static [ConstTriaCub\\_cm4pti](#) **consTriaCub\_cm4pti**

## Membres hérités additionnels

### 6.875.1 Documentation des fonctions membres

#### 6.875.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaCub_cm4pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.875.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaCub_cm4pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

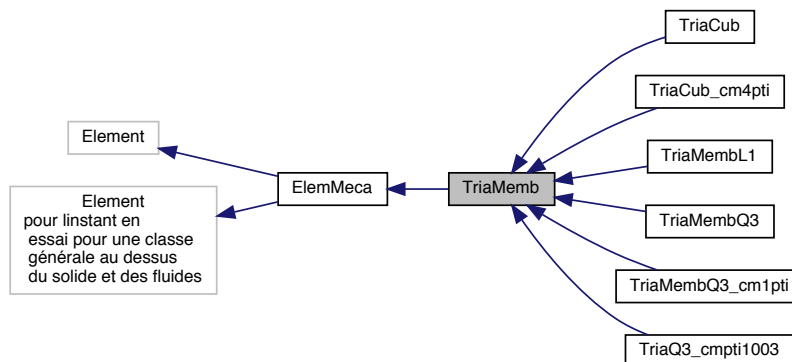
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

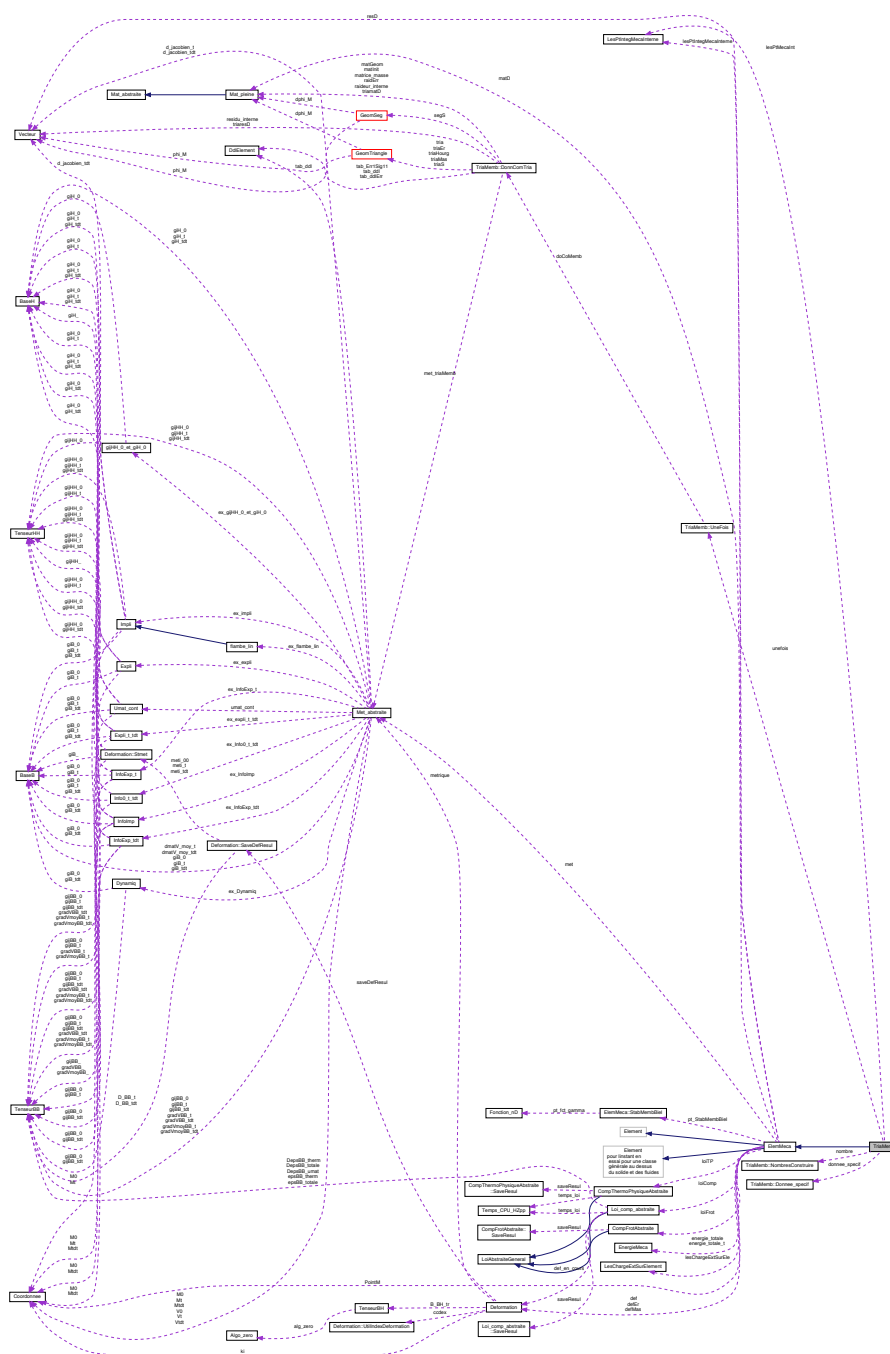
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub\_cm4pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaCub\_cm4pti.cc

## 6.876 Référence de la classe TriaMemb

Graphe d'héritage de TriaMemb:



Graphe de collaboration de TriaMemb:



## Classes

- class [DonnComTria](#)
- class [Donnee\\_specif](#)
- class [NombresConstruire](#)
- class [UneFois](#)

## Fonctions membres publiques

- [TriaMemb](#) (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, string info="")
- [TriaMemb](#) (int num\_mail, int num\_id, [Enum\\_interpol](#) id\_interp\_elt, [Enum\\_geom](#) id\_geom\_elt, const [Tableau](#)<[Noeud](#) \* > &tab, string info="")

- **TriaMemb** (const [TriaMemb](#) &tria)
- **TriaMemb** & **operator=** (const [TriaMemb](#) &tria)
- void **LectureDonneesParticulieres** ([UtilLecture](#) \*, [Tableau](#)< [Noeud](#) \* > \*)
- void **Info\_com\_Element** ([UtilLecture](#) \*entreePrinc, string &ordre, [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud)
- **ElemGeomC0** & **ElementGeometrique** () const
- const **ElemGeomC0** & **ElementGeometrique\_const** () const
- **Coordonnee** & **Point\_physique** (const [Coordonnee](#) &c\_int, [Coordonnee](#) &co, [Enum\\_dure](#) temps)
- void **Point\_physique** (const [Coordonnee](#) &c\_int, [Tableau](#)< [Coordonnee](#) > &t\_co)
- virtual double **Epaisseurs** ([Enum\\_dure](#) enu, const [Coordonnee](#) &)
- virtual double **EpaisseurMoyenne** ([Enum\\_dure](#) enu)
- [List\\_io](#)< [TypeQuelconque](#) > **Les\_types\_particuliers\_interne**s (bool absolue) const
- void **Grandeur\_particuliere** (bool absolue, [List\\_io](#)< [TypeQuelconque](#) > &litq, int iteg)
- void **Inactive\_ddl\_primaire** ()
- void **Active\_ddl\_primaire** ()
- void **Plus\_ddl\_Sigma** ()
- void **Inactive\_ddl\_Sigma** ()
- void **Active\_ddl\_Sigma** ()
- void **Active\_premier\_ddl\_Sigma** ()
- void **LectureContraintes** ([UtilLecture](#) \*entreePrinc)
- bool **ContraintesAbsolues** ([Tableau](#)< [Vecteur](#) > &tabSig)
- void **Libere** ()
- void **DefLoi** ([LoiAbstraiteGeneral](#) \*NouvelleLoi)
- int **TestComple**t ()
- Element \* **Complete** ([BlocGen](#) &bloc, [LesFonctions\\_nD](#) \*lesFonctionsnD)
- Element \* **Comple**t\_Hourglass ([LoiAbstraiteGeneral](#) \*NouvelleLoi, const [BlocGen](#) &bloc)
- double **H** (int ni, [Enum\\_dure](#) enu=TEMPS\_tdt)
- double **H\_moy** ([Enum\\_dure](#) enu)
- bool **SurfExiste** (int ns) const
- bool **AreteExiste** (int na) const
- const **DdlElement** & **TableauDdl** () const
- Element::ResRaid **Calcul\_implicit** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_t** (const [ParaAlgoControle](#) &pa)
- [Vecteur](#) \* **CalculResidu\_tdt** (const [ParaAlgoControle](#) &pa)
- [Mat\\_pleine](#) \* **CalculMatriceMasse** ([Enum\\_calcul\\_masse](#) id\_calcul\_masse)
- double **Long\_arrete\_mini\_sur\_c** ([Enum\\_dure](#) temps)
- Element::Er\_ResRaid **ContrainteAuNoeud\_ResRaid** ()
- Element::Er\_ResRaid **ErreurAuNoeud\_ResRaid** ()
- int **PointLePlusPres** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, const [Coordonnee](#) &M)
- [Coordonnee](#) **CoordPtInteg** ([Enum\\_dure](#) temps, [Enum\\_ddl](#) enu, int iteg, bool &erreur)
- [Tableau](#)< double > **Valeur\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, const [List\\_io](#)< [Ddl\\_enum\\_etendu](#) > &enu, int iteg)
- void **ValTensorielle\_a\_diff\_temps** (bool absolue, [Enum\\_dure](#) enu\_t, [List\\_io](#)< [TypeQuelconque](#) > &enu, int iteg)
- void **Lecture\_base\_info** (ifstream &ent, const [Tableau](#)< [Noeud](#) \* > \*tabMaillageNoeud, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)
- **DdlElement** & **Tableau\_de\_Sig1** () const
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- void **TdtversT** ()
- void **TversTdt** ()
- void **ErreurElement** (int type, double &errElemRelative, double &numérateur, double &denominateur)
- *!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en*
- virtual const [DeuxCoordonnees](#) & **Boite\_encembre\_element** ([Enum\\_dure](#) temps)
- [Vecteur](#) **SM\_charge\_volumique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_volumique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- ResRaid **SMR\_charge\_volumique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, const [ParaAlgoControle](#) &pa, bool sur\_volume\_finale\_)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- [Vecteur](#) **SM\_charge\_surfacique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_surfacique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)

- Vecteur **SM\_charge\_lineique\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_t** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_lineique\_Suiv\_E\_tdt** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_lineique\_Suiv\_I** (const [Coordonnee](#) &force, [Fonction\\_nD](#) \*pt\_fonct, int numArete, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E\_t** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_pression\_E\_tdt** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_pression\_I** (double pression, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_presUniDir\_E\_t** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_presUniDir\_E\_tdt** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_presUniDir\_I** (const [Coordonnee](#) &presUniDir, [Fonction\\_nD](#) \*pt\_fonct, int numFace, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrostatique\_E\_t** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_hydrostatique\_E\_tdt** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- ResRaid **SMR\_charge\_hydrostatique\_I** (const [Coordonnee](#) &dir\_normal\_liquide, const double &poidvol, int numFace, const [Coordonnee](#) &M\_liquide, const [ParaAlgoControle](#) &pa, bool sans\_limitation)
- Vecteur **SM\_charge\_hydrodynamique\_E\_t** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Vecteur **SM\_charge\_hydrodynamique\_E\_tdt** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- ResRaid **SMR\_charge\_hydrodynamique\_I** ([Courbe1D](#) \*frot\_fluid, const double &poidvol, [Courbe1D](#) \*coef\_aero\_n, int numFace, const double &coef\_mul, [Courbe1D](#) \*coef\_aero\_t, const [ParaAlgoControle](#) &pa)
- Tableau< [EIFrontiere](#) \* > const & **Frontiere** (bool force=false)
- void **ConstTabDdl** ()

### Fonctions membres protégées

- [TriaMemb::DonnComTria](#) \* **Init** (bool sans\_init\_noeud=false)
- [TriaMemb::DonnComTria](#) \* **Init** ([Donnee\\_specif](#) donnee\_specif, bool sans\_init\_noeud=false)
- void **Destruction** ()
- int **Dim\_sig\_eps** () const

### Attributs protégés

- [UneFois](#) \* **unefois**
- [NombresConstruire](#) \* **nombre**
- [Donnee\\_specif](#) **donnee\_specif**
- [LesPtIntegMecalInterne](#) **lesPtMecalInt**

### Membres hérités additionnels

#### 6.876.1 Documentation des fonctions membres

### 6.876.1.1 Active\_ddl\_Sigma()

```
void TriaMemb::Active_ddl_Sigma ( ) [inline], [virtual]
Implémente ElemMeca.
```

### 6.876.1.2 Active\_premier\_ddl\_Sigma()

```
void TriaMemb::Active_premier_ddl_Sigma ( ) [inline], [virtual]
Implémente ElemMeca.
```

### 6.876.1.3 ContraintesAbsolues()

```
bool TriaMemb::ContraintesAbsolues (
    Tableau< Vecteur > & tabSig ) [inline], [virtual]
Implémente ElemMeca.
```

### 6.876.1.4 Dim\_sig\_eps()

```
int TriaMemb::Dim_sig_eps ( ) const [inline], [protected], [virtual]
Implémente ElemMeca.
```

### 6.876.1.5 EpaisseurMoyenne()

```
virtual double TriaMemb::EpaisseurMoyenne (
    Enum_dure enu ) [inline], [virtual]
Réimplémentée à partir de ElemMeca.
```

### 6.876.1.6 Epaisseurs()

```
double TriaMemb::Epaisseurs (
    Enum_dure enu,
    const Coordonnee & M ) [virtual]
Réimplémentée à partir de ElemMeca.
```

### 6.876.1.7 ErreurElement()

```
void TriaMemb::ErreurElement (
    int type,
    double & errElemRelative,
    double & numerateur,
    double & denominateur ) [virtual]
```

!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en  
Réimplémentée à partir de [ElemMeca](#).

### 6.876.1.8 Inactive\_ddl\_Sigma()

```
void TriaMemb::Inactive_ddl_Sigma ( ) [inline], [virtual]
Implémente ElemMeca.
```

**6.876.1.9 LectureContraintes()**

```
void TriaMemb::LectureContraintes (
    UtilLecture * entreePrinc ) [inline], [virtual]
```

Implémente [ElemMeca](#).

**6.876.1.10 Les\_types\_particuliers\_internes()**

```
List_io< TypeQuelconque > TriaMemb::Les_types_particuliers_internes (
    bool absolue ) const [virtual]
```

Réimplémentée à partir de [ElemMeca](#).

**6.876.1.11 Long\_arrete\_mini\_sur\_c()**

```
double TriaMemb::Long_arrete_mini_sur_c (
    Enum_dure temps ) [inline], [virtual]
```

Implémente [ElemMeca](#).

**6.876.1.12 Plus\_ddl\_Sigma()**

```
void TriaMemb::Plus_ddl_Sigma ( ) [inline], [virtual]
```

Implémente [ElemMeca](#).

**6.876.1.13 Tableau\_de\_Sig1()**

```
DdlElement & TriaMemb::Tableau_de_Sig1 ( ) const [inline], [virtual]
```

!!!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en

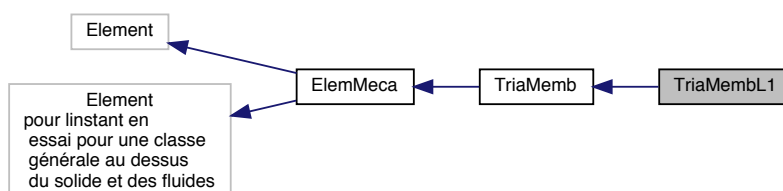
Réimplémentée à partir de [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

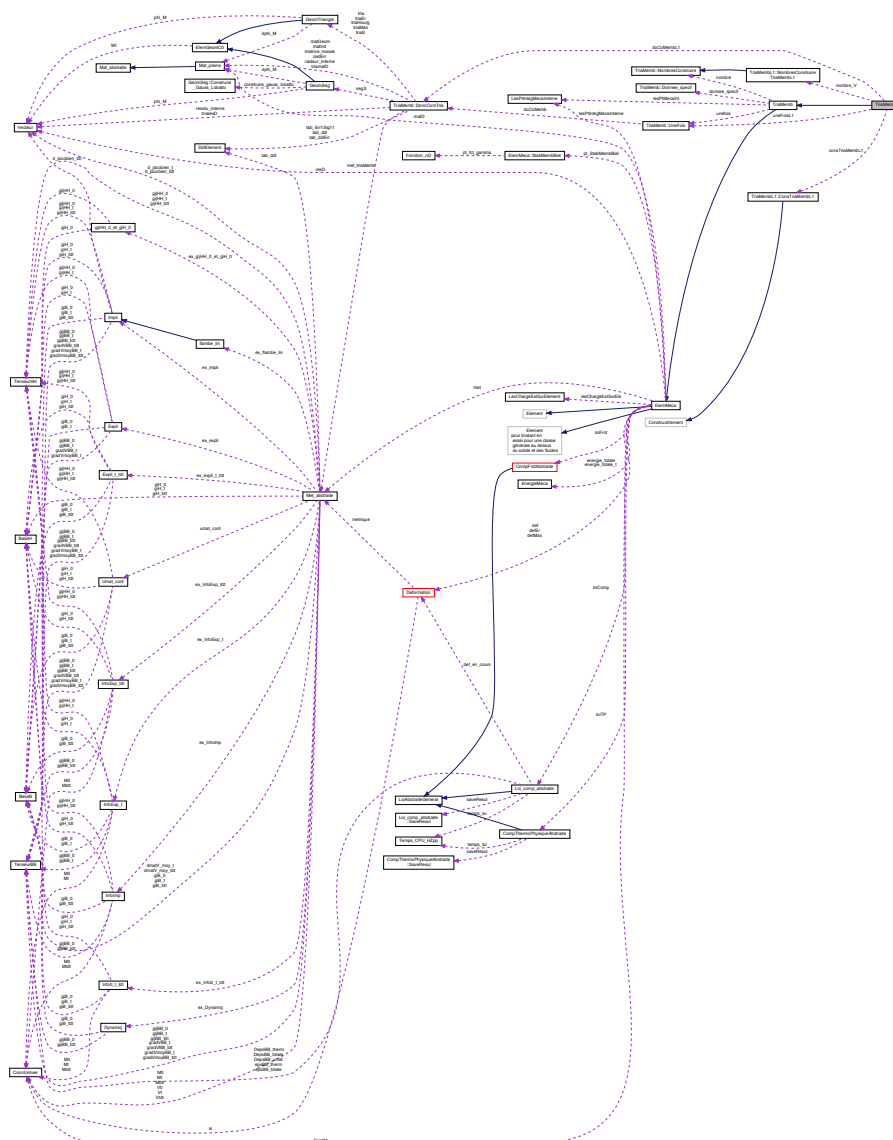
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.cc

**6.877 Référence de la classe TriaMembL1**

Graphe d'héritage de TriaMembL1:



Graphe de collaboration de TriaMembL1:



## Classes

- class [ConsTriaMembL1](#)
- class [NombresConstruireTriaMembL1](#)

## Fonctions membres publiques

- [TriaMembL1](#) (double epais, int num\_maill=0, int num\_id=-3)
- [TriaMembL1](#) (int num\_mail, int num\_id)
- [TriaMembL1](#) (double epais, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaMembL1](#) (const [TriaMembL1](#) &tria)
- [Element](#) \* [Nevez\\_copie](#) () const
- [TriaMembL1](#) & [operator=](#) ([TriaMembL1](#) &tria)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EiFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdiElement](#) &ddelem)
- [EiFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdiElement](#) &ddelem)



## Attributs protégés statiques

- static [TriaMemb::DonnComTria](#) \* **doCoMembL1** = NULL
- static [TriaMemb::UneFois](#) **uneFoisL1**
- static [NombresConstruireTriaMembL1](#) **nombre\_V**
- static [ConstTriaMembL1](#) **consTriaMembL1**

## Membres hérités additionnels

### 6.877.1 Documentation des fonctions membres

#### 6.877.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaMembL1::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.877.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaMembL1::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

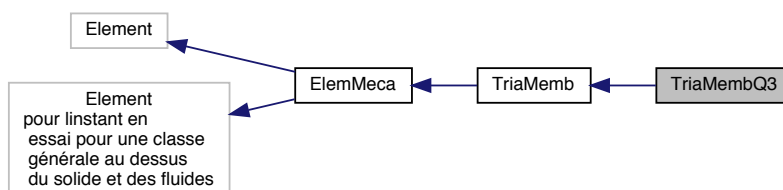
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

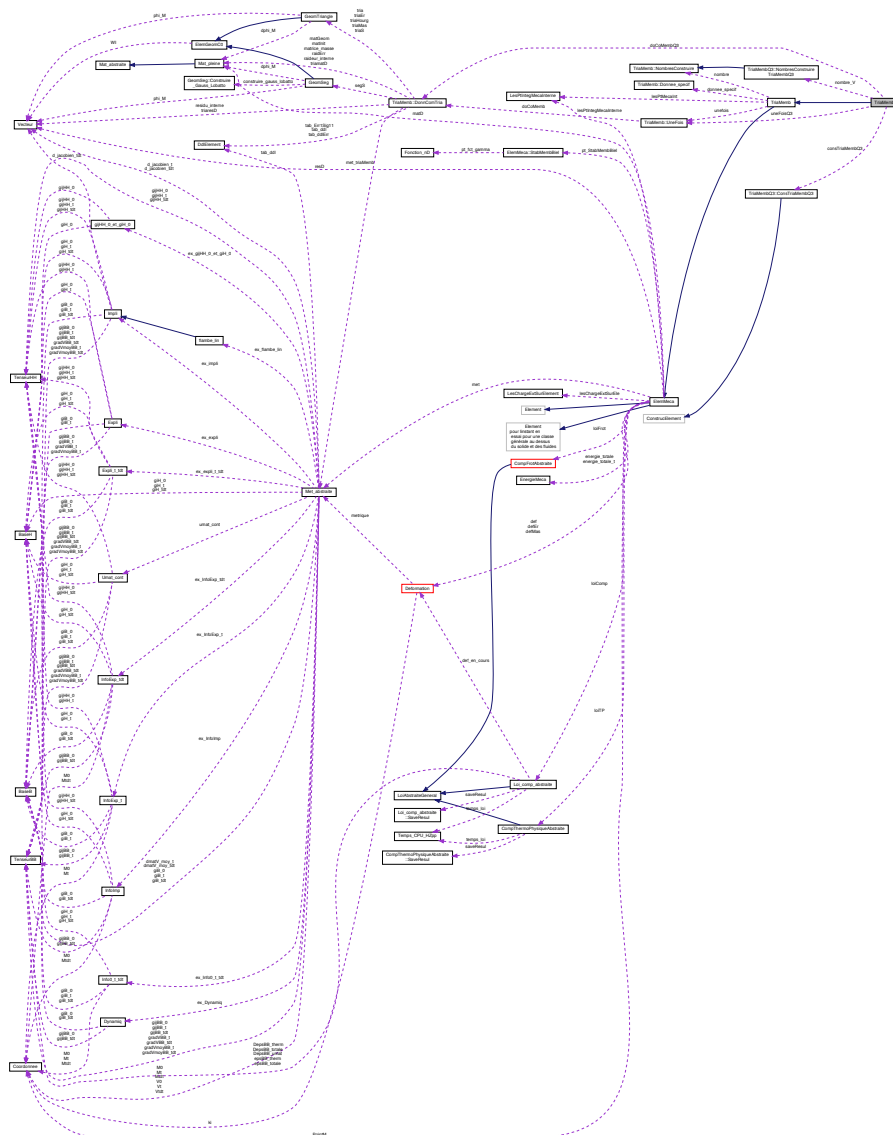
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembL1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembL1.cc

## 6.878 Référence de la classe TriaMembQ3

Graphe d'héritage de TriaMembQ3:



Graphe de collaboration de TriaMembQ3:



## Classes

- class [ConsTriaMembQ3](#)
- class [NombresConstruireTriaMembQ3](#)

## Fonctions membres publiques

- **TriaMembQ3** (double epaiss, int num\_maill=0, int num\_id=-3)
- **TriaMembQ3** (int num\_mail, int num\_id)
- **TriaMembQ3** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **TriaMembQ3** (const [TriaMembQ3](#) &tria)
- [Element](#) \* **Nevez\_copie** () const
- [TriaMembQ3](#) & **operator=** ([TriaMembQ3](#) &tria)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdElement](#) &ddelem)

## Attributs protégés statiques

- static [TriaMemb::DonnComTria](#) \* **doCoMembQ3** = NULL
- static [TriaMemb::UneFois](#) **uneFoisQ3**
- static [NombresConstruireTriaMembQ3](#) **nombre\_V**
- static [ConstTriaMembQ3](#) **consTriaMembQ3**

## Membres hérités additionnels

### 6.878.1 Documentation des fonctions membres

#### 6.878.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaMembQ3::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.878.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaMembQ3::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

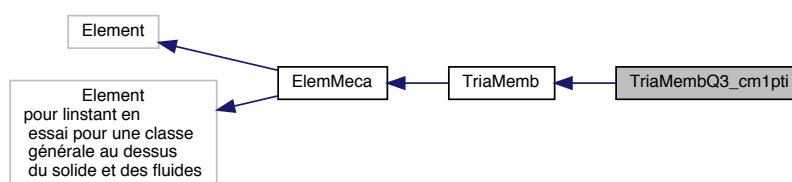
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

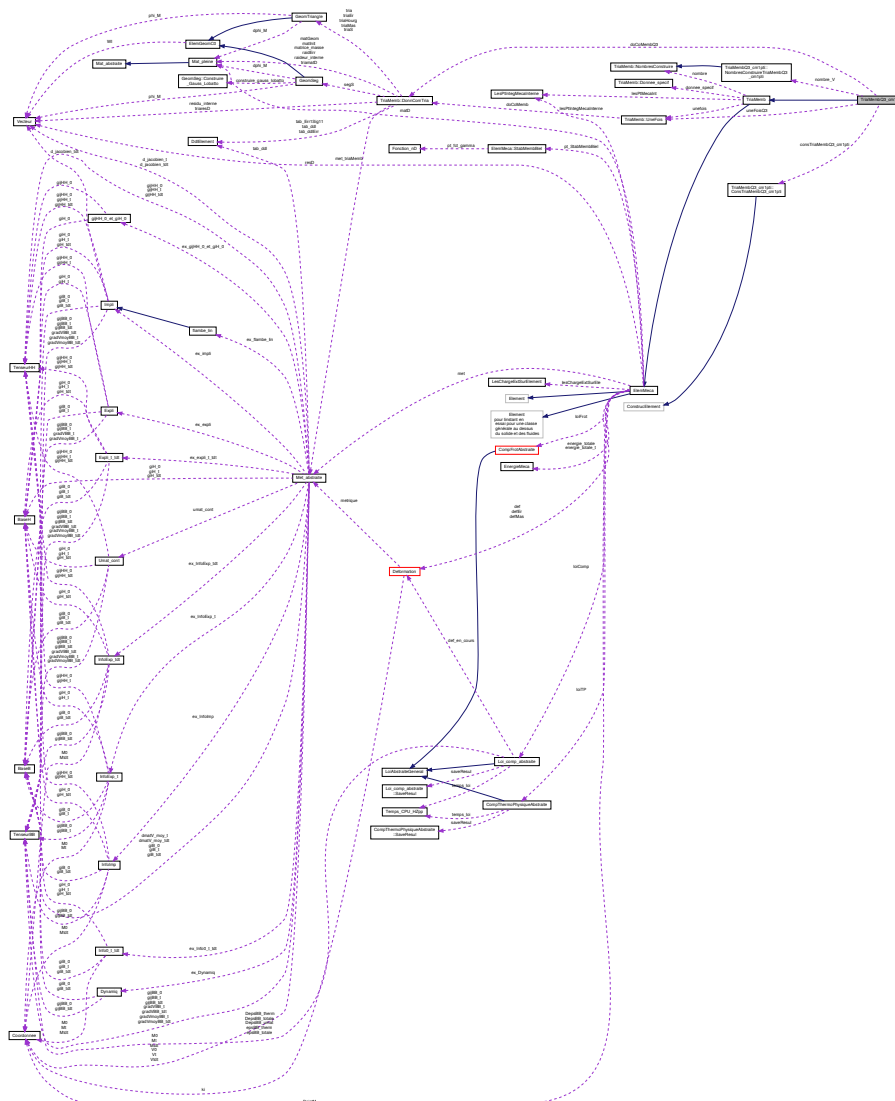
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3.cc

## 6.879 Référence de la classe TriaMembQ3\_cm1pti

Graphe d'héritage de TriaMembQ3\_cm1pti:



Graphe de collaboration de TriaMembQ3\_cm1pti:



## Classes

- class [ConsTriaMembQ3\\_cm1pti](#)
- class [NombresConstruireTriaMembQ3\\_cm1pti](#)

## Fonctions membres publiques

- [TriaMembQ3\\_cm1pti](#) (double epaiss, int num\_maill=0, int num\_id=-3)
- [TriaMembQ3\\_cm1pti](#) (int num\_mail, int num\_id)
- [TriaMembQ3\\_cm1pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaMembQ3\\_cm1pti](#) (const [TriaMembQ3\\_cm1pti](#) &tria)
- [Element](#) \* [Nevez\\_copie](#) () const
- [TriaMembQ3\\_cm1pti](#) & [operator=](#) ([TriaMembQ3\\_cm1pti](#) &tria)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [TriaMemb::DonnComTria](#) \* **doCoMembQ3** = NULL
- static [TriaMemb::UneFois](#) **uneFoisQ3**
- static [NombresConstruireTriaMembQ3\\_cm1pti](#) **nombre\_V**
- static [ConstTriaMembQ3\\_cm1pti](#) **constTriaMembQ3\_cm1pti**

## Membres hérités additionnels

### 6.879.1 Documentation des fonctions membres

#### 6.879.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaMembQ3_cm1pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.879.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaMembQ3_cm1pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

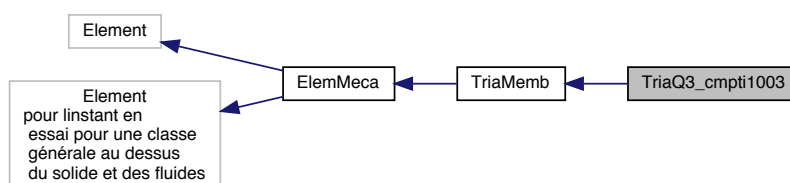
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

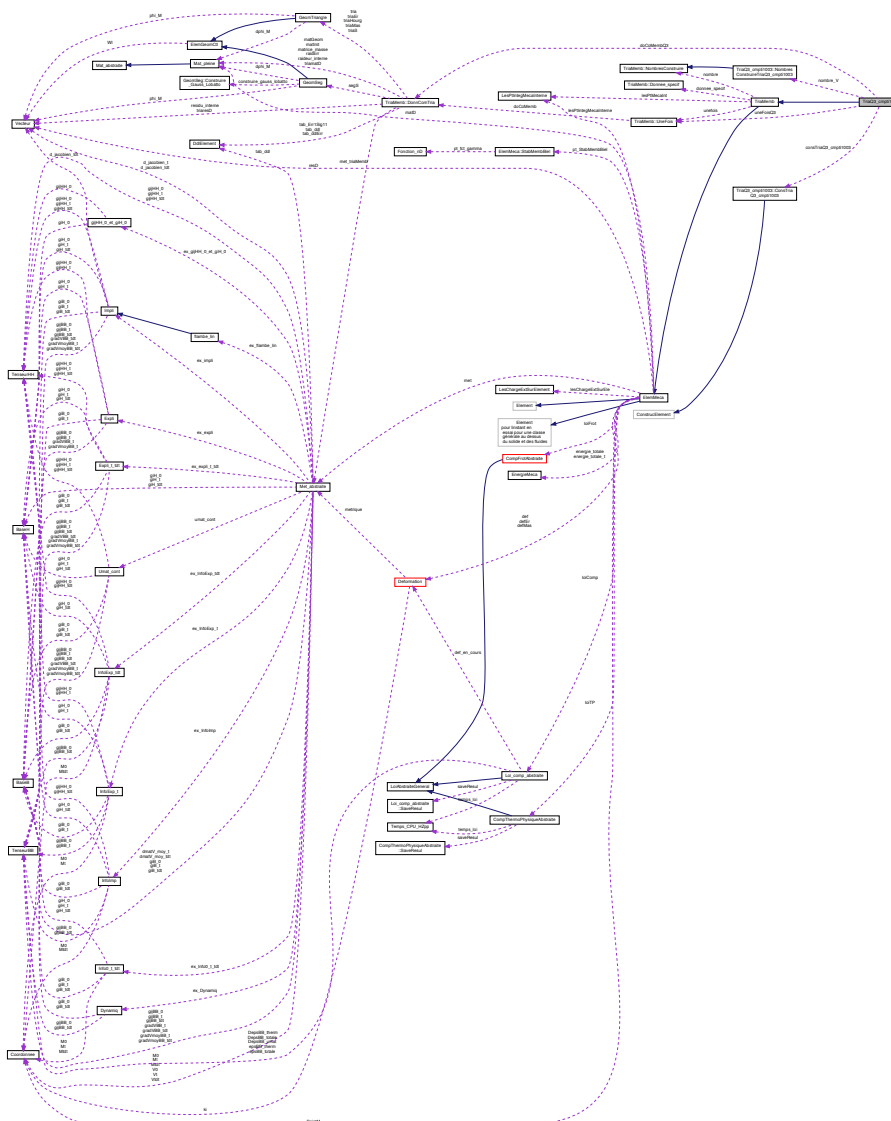
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3\_↵  
cm1pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMembQ3\_↵  
cm1pti.cc

## 6.880 Référence de la classe TriaQ3\_cmpti1003

Graphe d'héritage de TriaQ3\_cmpti1003:



Graphe de collaboration de TriaQ3\_cmpti1003:



## Classes

- class [ConsTriaQ3\\_cmpti1003](#)
- class [NombresConstruireTriaQ3\\_cmpti1003](#)

## Fonctions membres publiques

- [TriaQ3\\_cmpti1003](#) (double epaiss, int num\_maill=0, int num\_id=-3)
- [TriaQ3\\_cmpti1003](#) (int num\_mail, int num\_id)
- [TriaQ3\\_cmpti1003](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaQ3\\_cmpti1003](#) (const [TriaQ3\\_cmpti1003](#) &tria)
- Element \* [Nevez\\_copie](#) () const
- [TriaQ3\\_cmpti1003](#) & [operator=](#) ([TriaQ3\\_cmpti1003](#) &tria)
- void [AfficheVarDual](#) (ofstream &sort, [Tableau](#)< string > &nom)

## Fonctions membres protégées

- [EIFrontiere](#) \* [new\\_frontiere\\_lin](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [EIFrontiere](#) \* [new\\_frontiere\\_surf](#) (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [TriaMemb::DonnComTria](#) \* **doCoMembQ3** = NULL
- static [TriaMemb::UneFois](#) **uneFoisQ3**
- static [NombresConstruireTriaQ3\\_cmpti1003](#) **nombre\_V**
- static [ConstTriaQ3\\_cmpti1003](#) **consTriaQ3\_cmpti1003**

## Membres hérités additionnels

### 6.880.1 Documentation des fonctions membres

#### 6.880.1.1 new\_frontiere\_lin()

```
ElFrontiere * TriaQ3_cmpti1003::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.880.1.2 new\_frontiere\_surf()

```
ElFrontiere * TriaQ3_cmpti1003::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

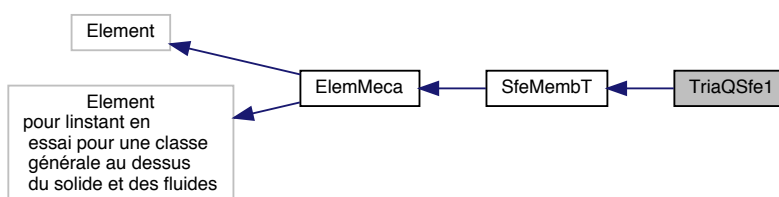
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

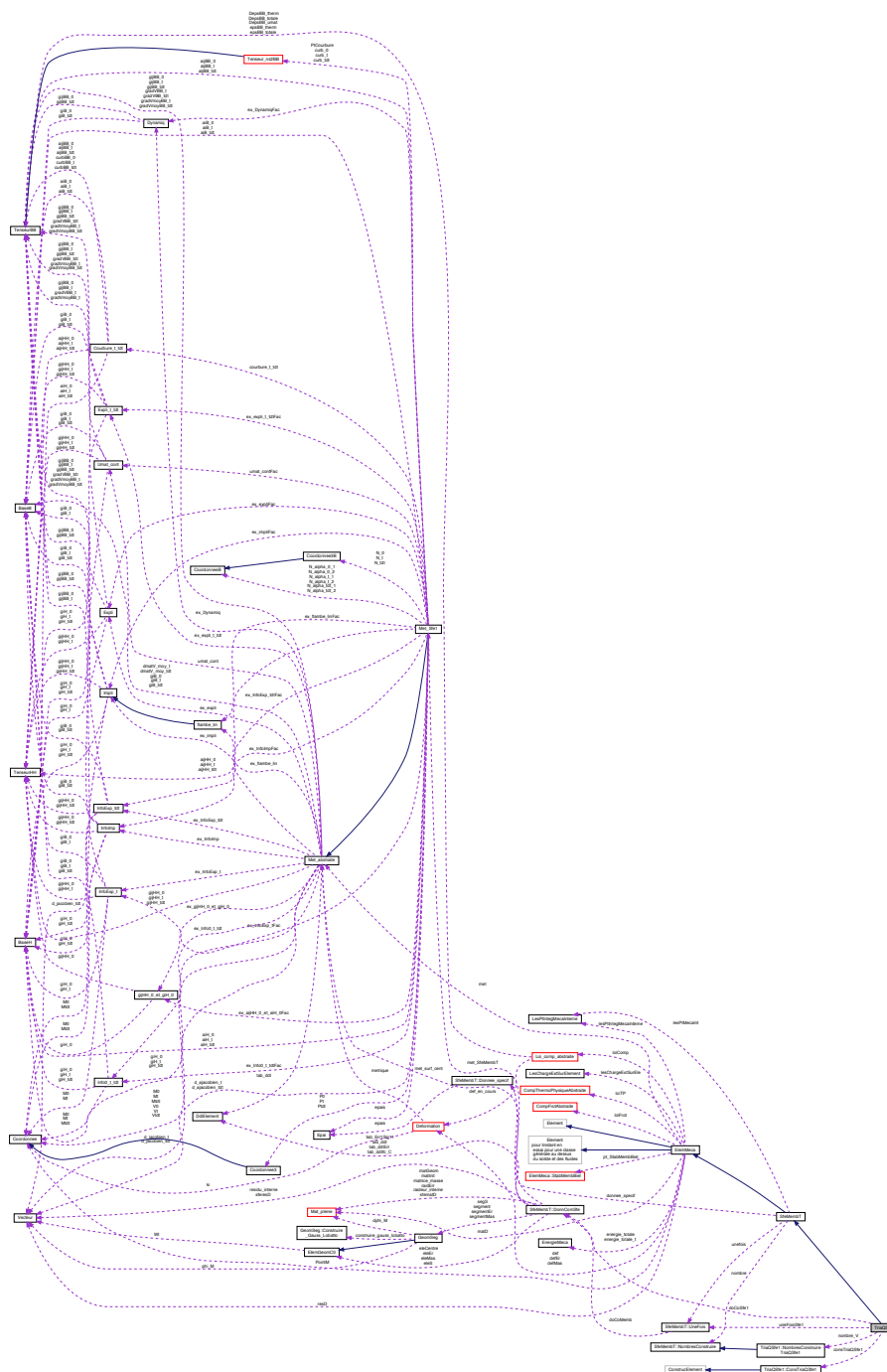
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaQ3\_cmpti1003.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaQ3\_cmpti1003.cc

## 6.881 Référence de la classe TriaQSfe1

Graphe d'héritage de TriaQSfe1:



Graphe de collaboration de TriaQSfe1:



## Classes

- class [ConsTriaQSfe1](#)
- class [NombresConstruireTriaQSfe1](#)

## Fonctions membres publiques

- **TriaQSfe1** (double epais, int num\_mail=0, int num\_id=-3)
- **TriaQSfe1** (int num\_mail, int num\_id)
- **TriaQSfe1** (double epais, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **TriaQSfe1** (const [TriaQSfe1](#) &tria)



- Element \* **Nevez\_copie** () const
- [TriaQSfe1](#) & **operator=** ([TriaQSfe1](#) &tria)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- double **KSI** (int i)

### Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

### Attributs protégés statiques

- static [SfeMembT::DonnComSfe](#) \* **doCoSfe1** = NULL
- static [SfeMembT::UneFois](#) **uneFoisSfe1**
- static [NombresConstruireTriaQSfe1](#) **nombre\_V**
- static [ConstriaQSfe1](#) **consTriaQSfe1**

### Membres hérités additionnels

#### 6.881.1 Documentation des fonctions membres

##### 6.881.1.1 KSI()

```
double TriaQSfe1::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.881.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaQSfe1::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.881.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaQSfe1::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

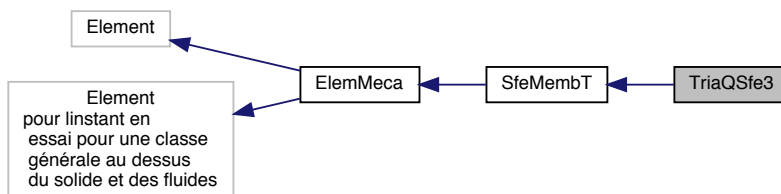
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

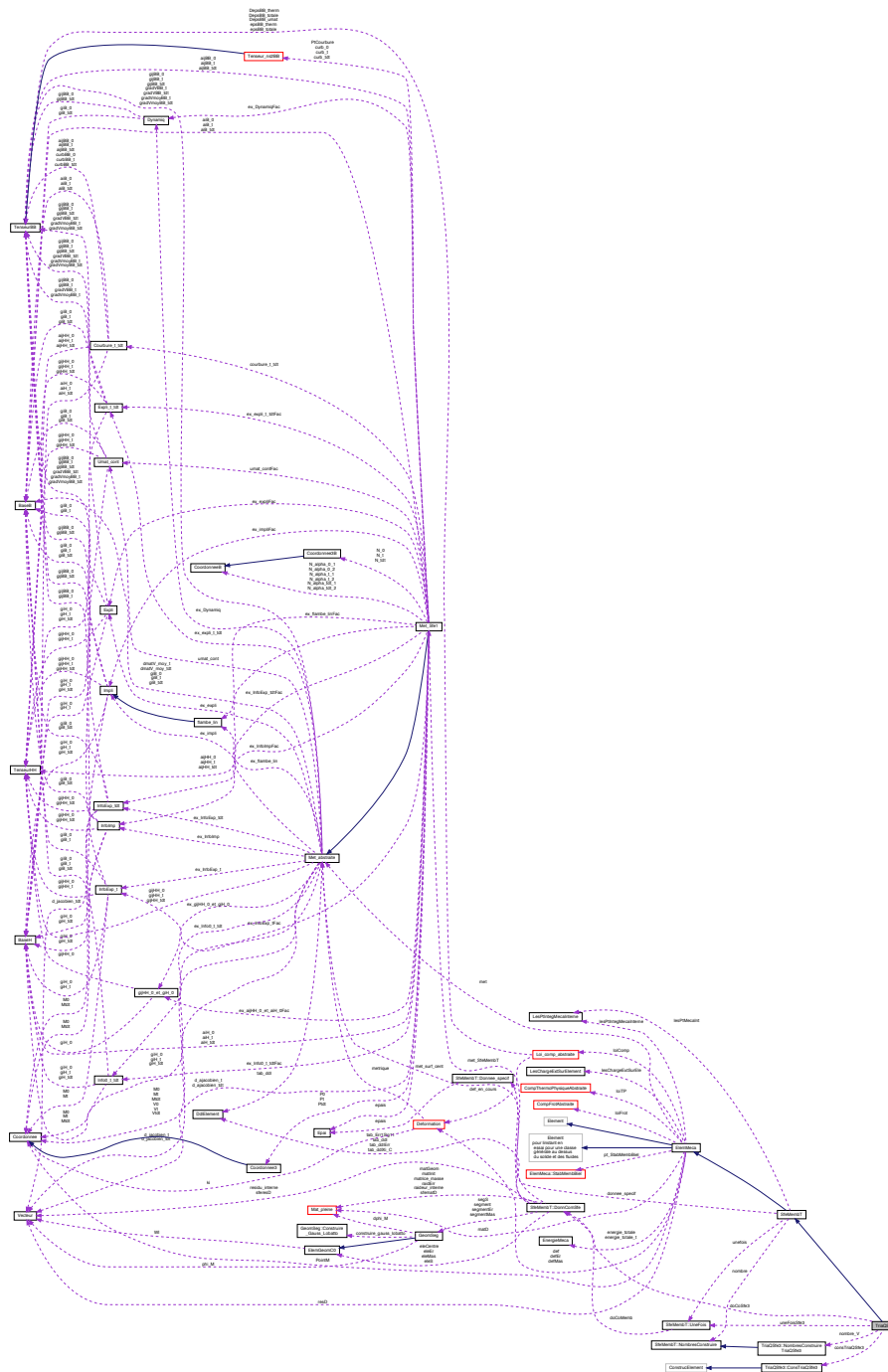
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe1.cc

## 6.882 Référence de la classe TriaQSfe3

Graphe d'héritage de TriaQSfe3:



Graphe de collaboration de TriaQSfe3:



## Classes

- class [ConsTriaQSfe3](#)
- class [NombresConstruireTriaQSfe3](#)

## Fonctions membres publiques

- **TriaQSfe3** (double epaiss, int num\_mail=0, int num\_id=-3)
- **TriaQSfe3** (int num\_mail, int num\_id)
- **TriaQSfe3** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **TriaQSfe3** (const [TriaQSfe3](#) &tria)

- `Element * Nevez_copie () const`
- `TriaQSfe3 & operator= (TriaQSfe3 &tria)`
- `void AfficheVarDual (ofstream &sort, Tableau< string > &nom)`
- `double KSI (int i)`

### Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- `static SfeMembT::DonnComSfe * doCoSfe3 = NULL`
- `static SfeMembT::UneFois uneFoisSfe3`
- `static NombresConstruireTriaQSfe3 nombre_V`
- `static ConsTriaQSfe3 consTriaQSfe3`

### Membres hérités additionnels

#### 6.882.1 Documentation des fonctions membres

##### 6.882.1.1 KSI()

```
double TriaQSfe3::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.882.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaQSfe3::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.882.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaQSfe3::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

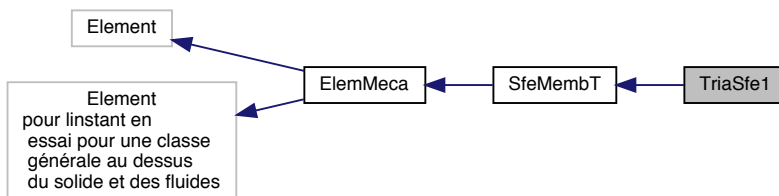
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

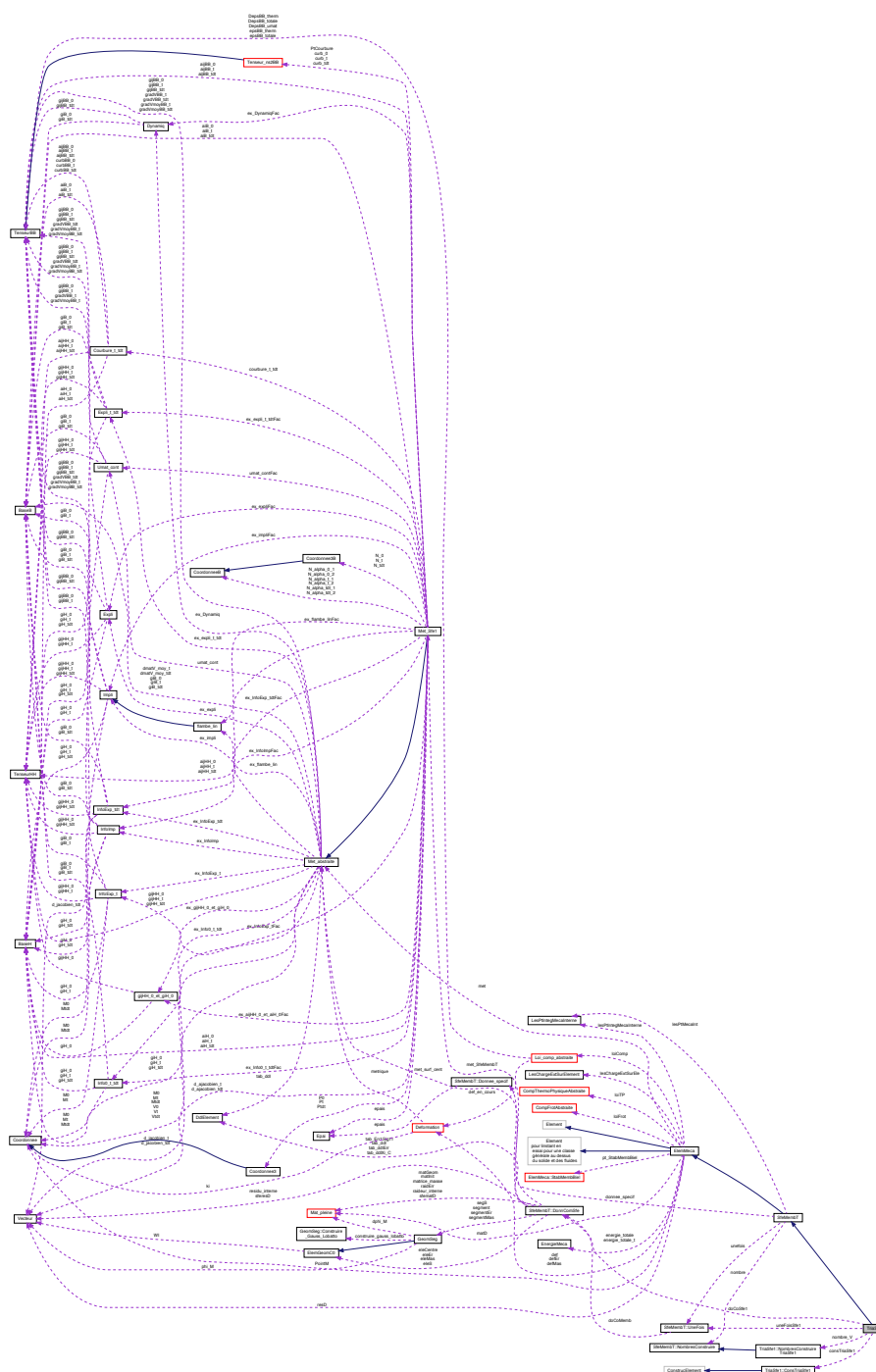
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe3.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaQSfe3.cc`

## 6.883 Référence de la classe TriaSfe1

Graphe d'héritage de TriaSfe1:



Graphe de collaboration de TriaSfe1:



## Classes

- class [ConsTriaSfe1](#)
- class [NombresConstruireTriaSfe1](#)

## Fonctions membres publiques

- **TriaSfe1** (double epaiss, int num\_mail=0, int num\_id=-3)
- **TriaSfe1** (int num\_mail, int num\_id)
- **TriaSfe1** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)

- `TriaSfe1` (const `TriaSfe1` &tria)
- `Element * Nevez_copie` () const
- `TriaSfe1 & operator=` (`TriaSfe1` &tria)
- void `AfficheVarDual` (ofstream &sort, `Tableau`< string > &nom)
- double `KSI` (int i)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau`< `Noeud` \* > &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau`< `Noeud` \* > &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `SfeMembT::DonnComSfe * doCoSfe1` = NULL
- static `SfeMembT::UneFois uneFoisSfe1`
- static `NombresConstruireTriaSfe1 nombre_V`
- static `ConsTriaSfe1 consTriaSfe1`

## Membres hérités additionnels

### 6.883.1 Documentation des fonctions membres

#### 6.883.1.1 KSI()

```
double TriaSfe1::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

#### 6.883.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe1::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.883.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe1::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

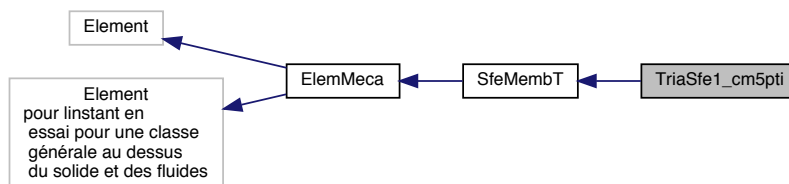
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1.cc

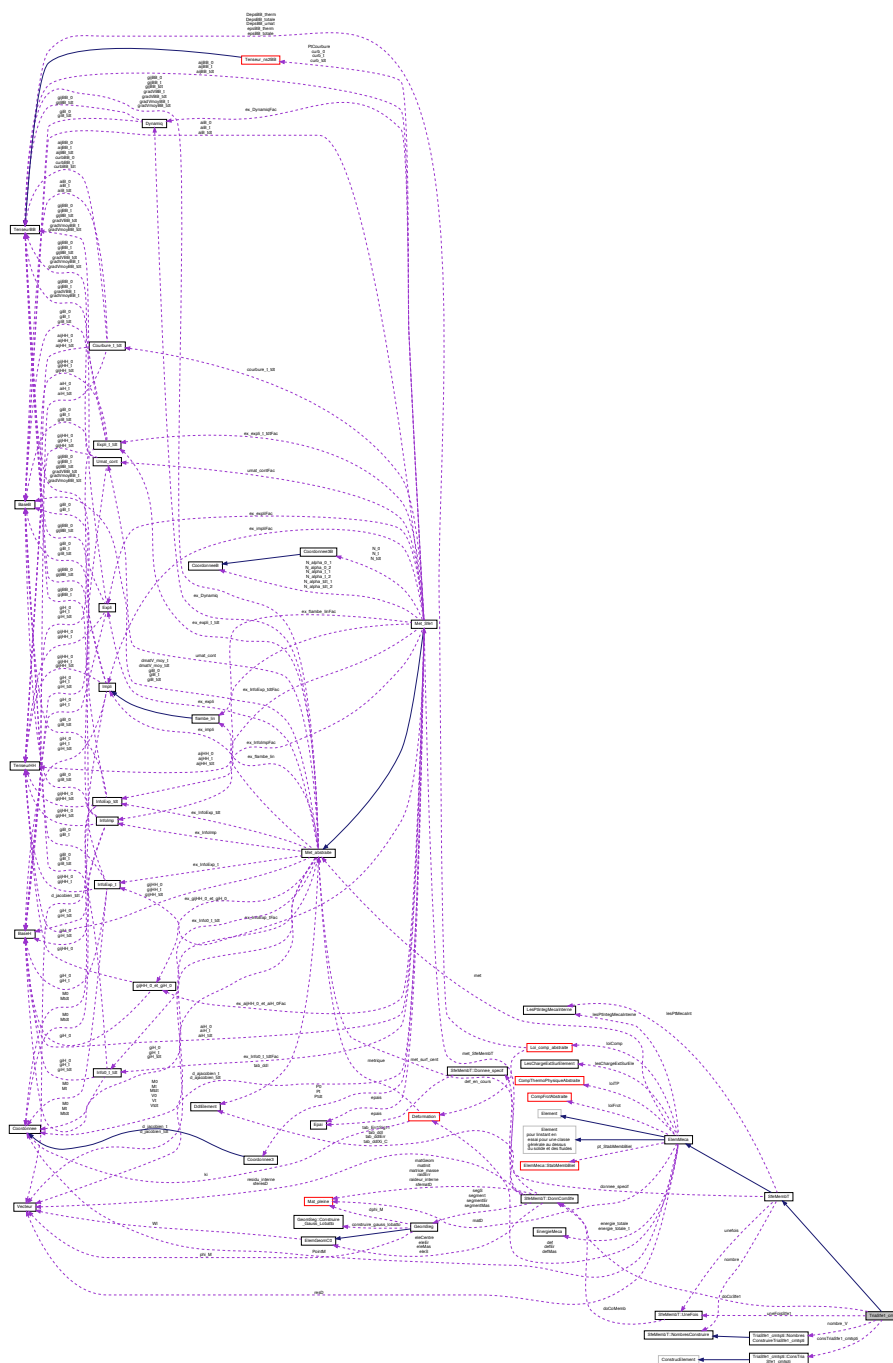
## 6.884 Référence de la classe TriaSfe1\_cm5pti

Graphe d'héritage de TriaSfe1\_cm5pti:





Graphe de collaboration de TriaSfe1\_cm5pti:



## Classes

- class [ConsTriaSfe1\\_cm5pti](#)
- class [NombresConstruireTriaSfe1\\_cm5pti](#)

## Fonctions membres publiques

- [TriaSfe1\\_cm5pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaSfe1\\_cm5pti](#) (int num\_mail, int num\_id)
- [TriaSfe1\\_cm5pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaSfe1\\_cm5pti](#) (const [TriaSfe1\\_cm5pti](#) &tria)

- `Element * Nevez_copie () const`
- `TriaSfe1_cm5pti & operator= (TriaSfe1_cm5pti &tria)`
- `void AfficheVarDual (ofstream &sort, Tableau< string > &nom)`
- `double KSI (int i)`

### Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- `static SfeMembT::DonnComSfe * doCoSfe1 = NULL`
- `static SfeMembT::UneFois uneFoisSfe1`
- `static NombresConstruireTriaSfe1_cm5pti nombre_V`
- `static ConsTriaSfe1_cm5pti consTriaSfe1_cm5pti`

### Membres hérités additionnels

#### 6.884.1 Documentation des fonctions membres

##### 6.884.1.1 KSI()

```
double TriaSfe1_cm5pti::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.884.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe1_cm5pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.884.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe1_cm5pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

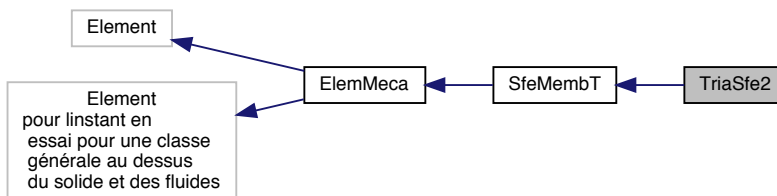
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

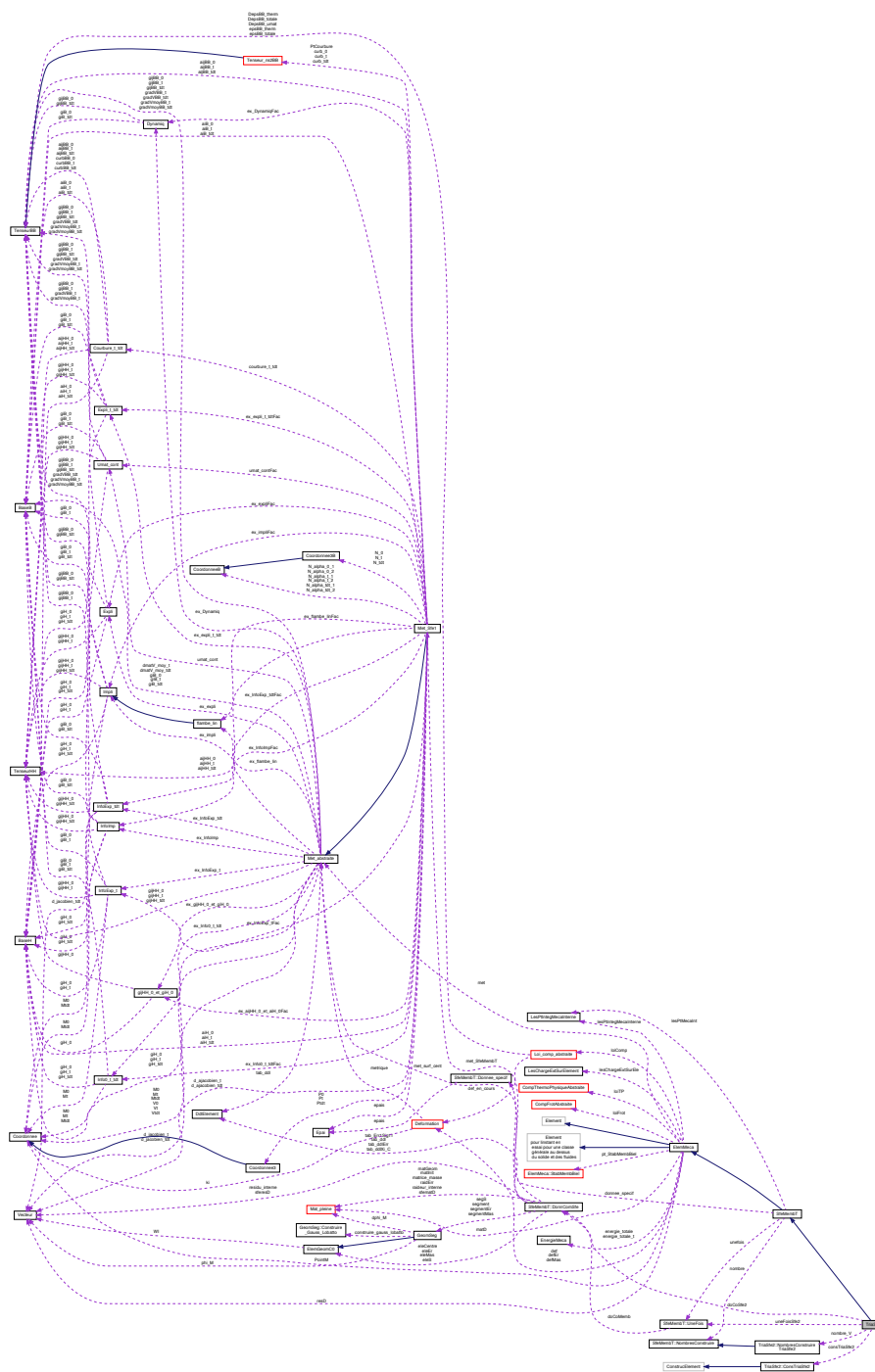
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1_cm5pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe1_cm5pti.cc`

## 6.885 Référence de la classe TriaSfe2

Graphe d'héritage de TriaSfe2:



Graphe de collaboration de TriaSfe2:



## Classes

- class [ConsTriaSfe2](#)
- class [NombresConstruireTriaSfe2](#)

## Fonctions membres publiques

- **TriaSfe2** (double epaiss, int num\_mail=0, int num\_id=-3)
- **TriaSfe2** (int num\_mail, int num\_id)
- **TriaSfe2** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)

- `TriaSfe2` (const `TriaSfe2` &tria)
- `Element * Nevez_copie` () const
- `TriaSfe2 & operator=` (`TriaSfe2` &tria)
- void `AfficheVarDual` (ofstream &sort, `Tableau`< string > &nom)
- double `KSI` (int i)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau`< `Noeud` \* > &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau`< `Noeud` \* > &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `SfeMembT::DonnComSfe * doCoSfe2` = NULL
- static `SfeMembT::UneFois uneFoisSfe2`
- static `NombresConstruireTriaSfe2 nombre_V`
- static `ConsTriaSfe2 consTriaSfe2`

## Membres hérités additionnels

### 6.885.1 Documentation des fonctions membres

#### 6.885.1.1 KSI()

```
double TriaSfe2::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

#### 6.885.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe2::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.885.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe2::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

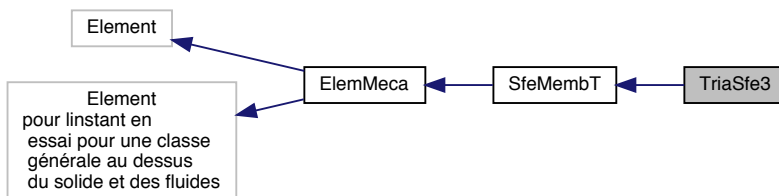
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

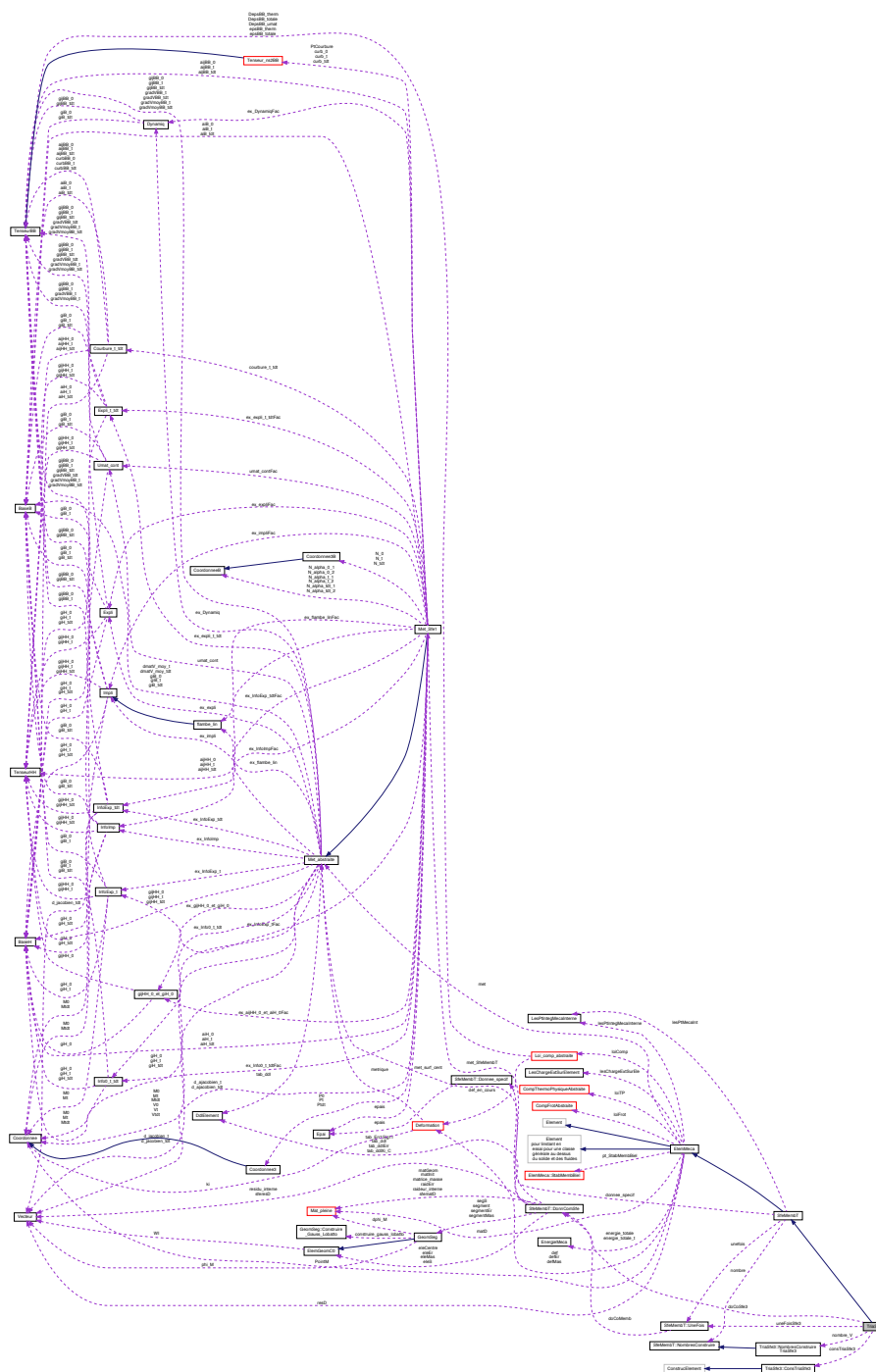
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe2.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe2.cc

## 6.886 Référence de la classe TriaSfe3

Graphe d'héritage de TriaSfe3:



Graphe de collaboration de TriaSfe3:



## Classes

- class [ConsTriaSfe3](#)
- class [NombresConstruireTriaSfe3](#)

## Fonctions membres publiques

- **TriaSfe3** (double epaiss, int num\_mail=0, int num\_id=-3)
- **TriaSfe3** (int num\_mail, int num\_id)
- **TriaSfe3** (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)

- `TriaSfe3` (const `TriaSfe3` &tria)
- `Element * Nevez_copie` () const
- `TriaSfe3 & operator=` (`TriaSfe3` &tria)
- void `AfficheVarDual` (ofstream &sort, `Tableau`< string > &nom)
- double `KSI` (int i)

## Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau`< `Noeud` \* > &tab, `DdlElement` &ddelem)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau`< `Noeud` \* > &tab, `DdlElement` &ddelem)

## Attributs protégés statiques

- static `SfeMembT::DonnComSfe * doCoSfe3` = NULL
- static `SfeMembT::UneFois uneFoisSfe3`
- static `NombresConstruireTriaSfe3 nombre_V`
- static `ConsTriaSfe3 consTriaSfe3`

## Membres hérités additionnels

### 6.886.1 Documentation des fonctions membres

#### 6.886.1.1 KSI()

```
double TriaSfe3::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

#### 6.886.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.886.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

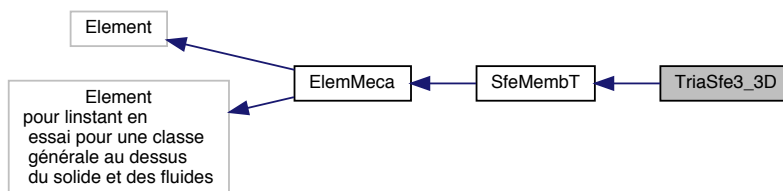
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3.cc

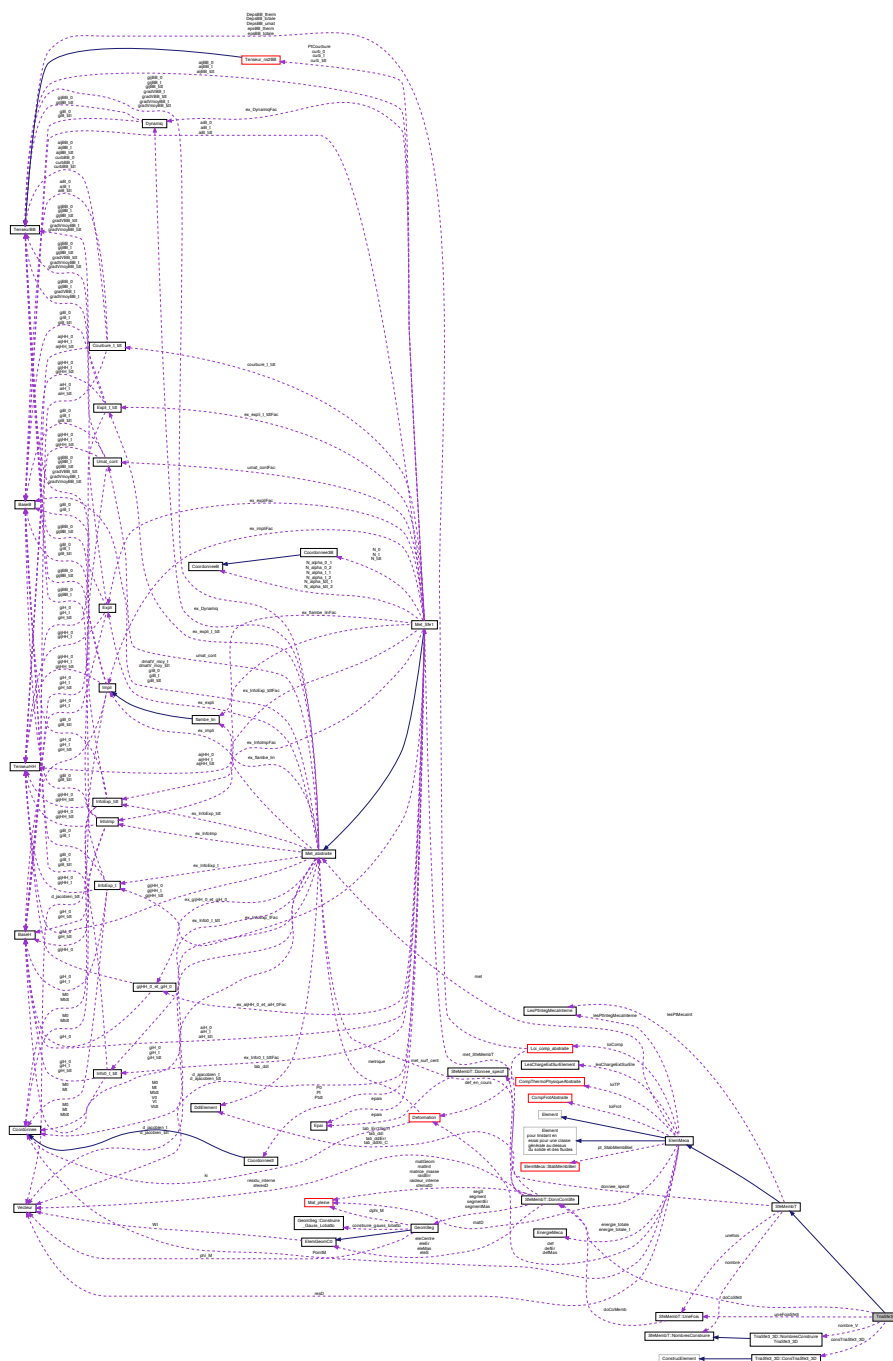


## 6.887 Référence de la classe TriaSfe3\_3D

Graphe d'héritage de TriaSfe3\_3D:



Graphe de collaboration de TriaSfe3\_3D:



## Classes

- class [ConsTriaSfe3\\_3D](#)
- class [NombresConstruireTriaSfe3\\_3D](#)

## Fonctions membres publiques

- [TriaSfe3\\_3D](#) (int num\_mail, int num\_id)
- [TriaSfe3\\_3D](#) (int num\_mail, int num\_id, const [Tableau< Noeud \\* >](#) &tab)
- [TriaSfe3\\_3D](#) (const [TriaSfe3\\_3D](#) &tria)
- [Element \\* Nevez\\_copie](#) () const

- [TriaSfe3\\_3D](#) & **operator=** ([TriaSfe3\\_3D](#) &tria)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- double **KSI** (int i)

### Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

### Attributs protégés statiques

- static [SfeMembT::DonnComSfe](#) \* **doCoSfe3** = NULL
- static [SfeMembT::UneFois](#) **uneFoisSfe3**
- static [NombresConstruireTriaSfe3\\_3D](#) **nombre\_V**
- static [ConsTriaSfe3\\_3D](#) **consTriaSfe3\_3D**

### Membres hérités additionnels

#### 6.887.1 Documentation des fonctions membres

##### 6.887.1.1 KSI()

```
double TriaSfe3_3D::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.887.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3_3D::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.887.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3_3D::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

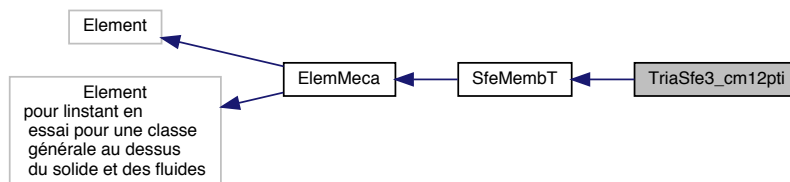
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

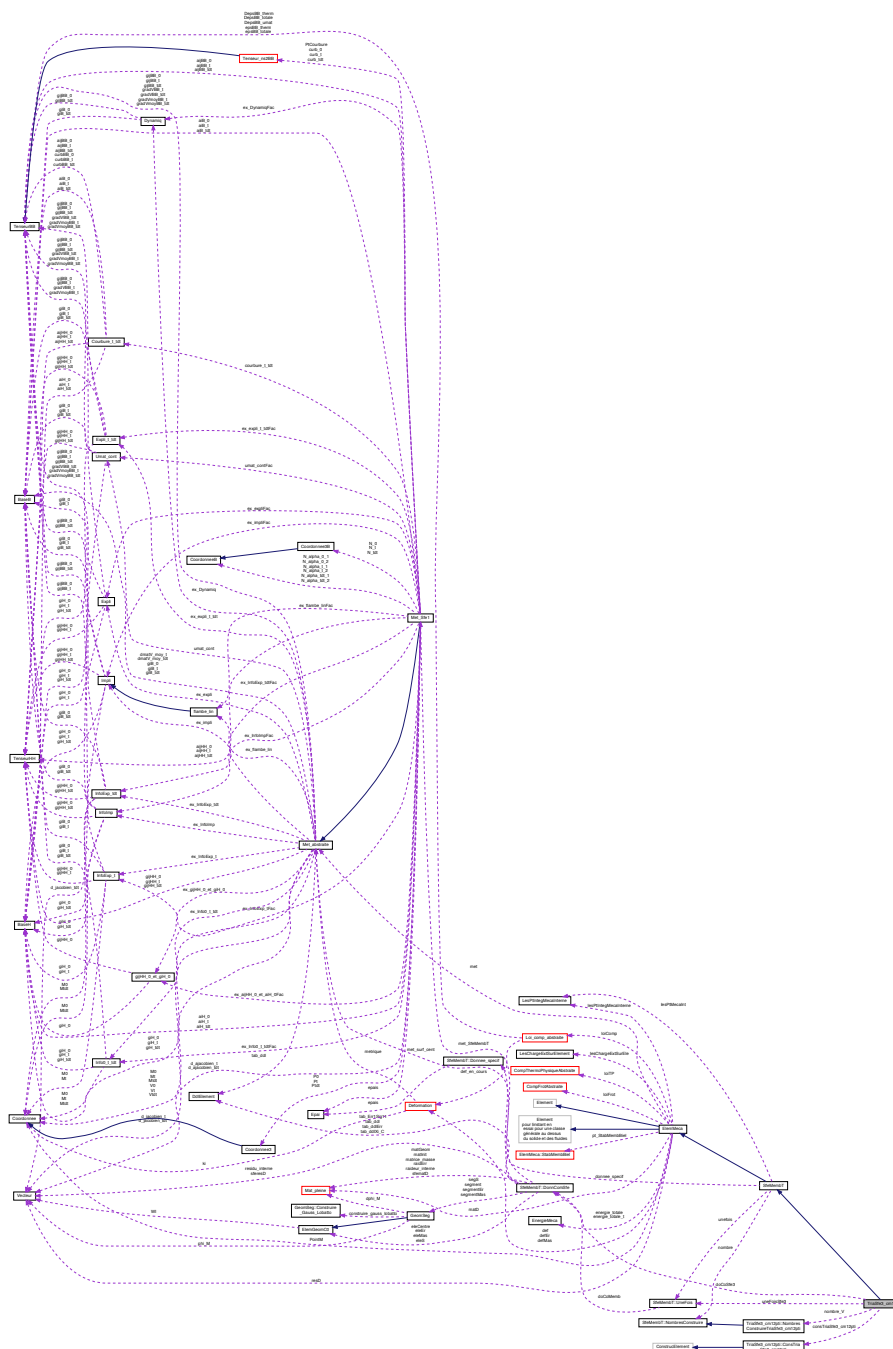
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_3D.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_3D.cc

## 6.888 Référence de la classe TriaSfe3\_cm12pti

Graphe d'héritage de TriaSfe3\_cm12pti:



Graphe de collaboration de TriaSfe3\_cm12pti:



## Classes

- class [ConsTriaSfe3\\_cm12pti](#)
- class [NombresConstruireTriaSfe3\\_cm12pti](#)

## Fonctions membres publiques

- [TriaSfe3\\_cm12pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaSfe3\\_cm12pti](#) (int num\_mail, int num\_id)
- [TriaSfe3\\_cm12pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaSfe3\\_cm12pti](#) (const [TriaSfe3\\_cm12pti](#) &tria)
- [Element](#) \* [Nevez\\_copie](#) () const

- [TriaSfe3\\_cm12pti](#) & **operator=** ([TriaSfe3\\_cm12pti](#) &tria)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- double **KSI** (int i)

## Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

## Attributs protégés statiques

- static [SfeMembT::DonnComSfe](#) \* **doCoSfe3** = NULL
- static [SfeMembT::UneFois](#) **uneFoisSfe3**
- static [NombresConstruireTriaSfe3\\_cm12pti](#) **nombre\_V**
- static [ConsTriaSfe3\\_cm12pti](#) **consTriaSfe3\_cm12pti**

## Membres hérités additionnels

### 6.888.1 Documentation des fonctions membres

#### 6.888.1.1 KSI()

```
double TriaSfe3_cm12pti::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

#### 6.888.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3_cm12pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

#### 6.888.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3_cm12pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

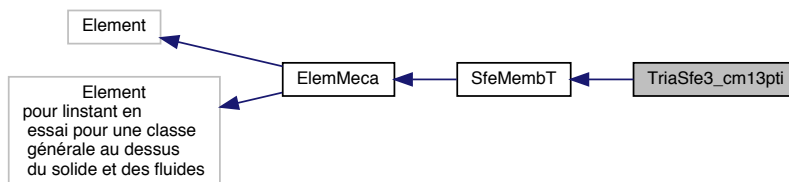
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

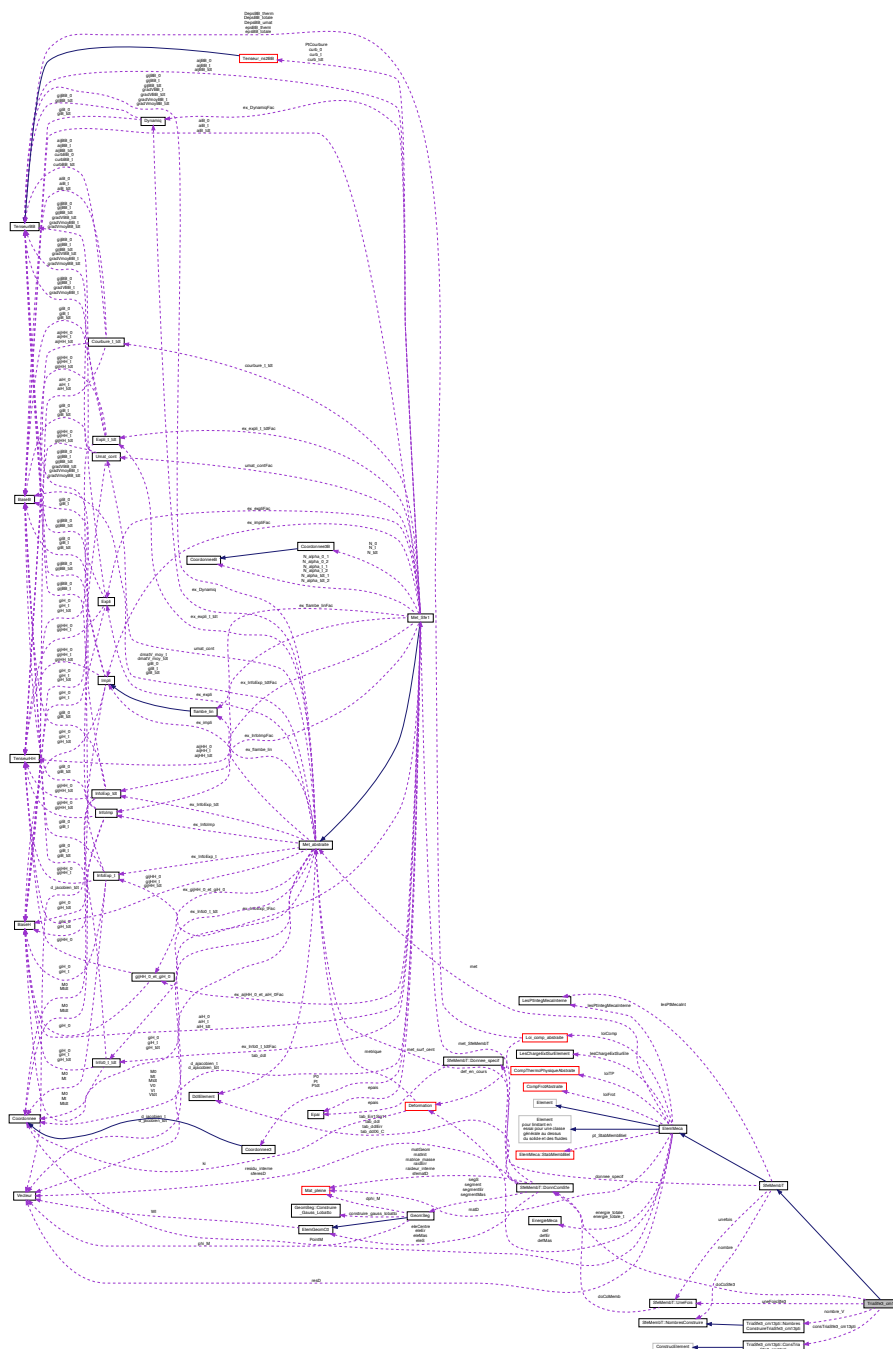
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm12pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm12pti.cc

## 6.889 Référence de la classe TriaSfe3\_cm13pti

Graphe d'héritage de TriaSfe3\_cm13pti:



Graphe de collaboration de TriaSfe3\_cm13pti:



## Classes

- class [ConsTriaSfe3\\_cm13pti](#)
- class [NombresConstruireTriaSfe3\\_cm13pti](#)

## Fonctions membres publiques

- [TriaSfe3\\_cm13pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaSfe3\\_cm13pti](#) (int num\_mail, int num\_id)
- [TriaSfe3\\_cm13pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaSfe3\\_cm13pti](#) (const [TriaSfe3\\_cm13pti](#) &tria)
- [Element](#) \* [Nevez\\_copie](#) () const



- [TriaSfe3\\_cm13pti](#) & `operator=` ([TriaSfe3\\_cm13pti](#) &tria)
- void `AfficheVarDual` (ofstream &sort, [Tableau](#)< string > &nom)
- double `KSI` (int i)

### Fonctions membres protégées

- [ElFrontiere](#) \* `new_frontiere_lin` (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* `new_frontiere_surf` (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

### Attributs protégés statiques

- static [SfeMembT::DonnComSfe](#) \* `doCoSfe3` = NULL
- static [SfeMembT::UneFois](#) `uneFoisSfe3`
- static [NombresConstruireTriaSfe3\\_cm13pti](#) `nombre_V`
- static [ConsTriaSfe3\\_cm13pti](#) `consTriaSfe3_cm13pti`

### Membres hérités additionnels

#### 6.889.1 Documentation des fonctions membres

##### 6.889.1.1 KSI()

```
double TriaSfe3_cm13pti::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.889.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3_cm13pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.889.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3_cm13pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

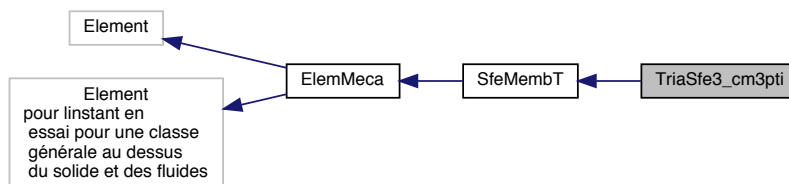
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

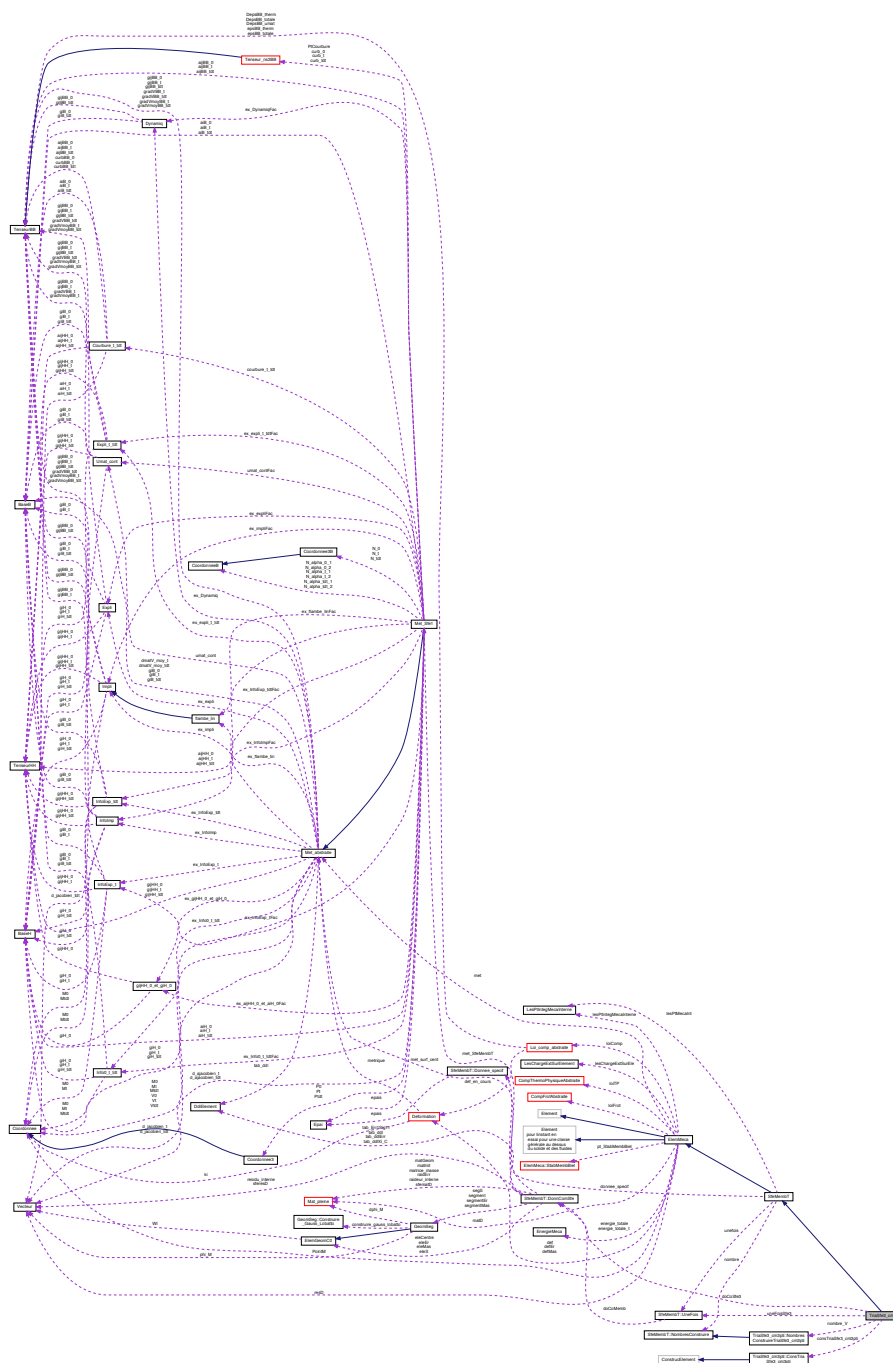
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm13pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm13pti.cc

## 6.890 Référence de la classe TriaSfe3\_cm3pti

Graphe d'héritage de TriaSfe3\_cm3pti:



Graphe de collaboration de TriaSfe3\_cm3pti:



## Classes

- class [ConsTriaSfe3\\_cm3pti](#)
- class [NombresConstruireTriaSfe3\\_cm3pti](#)

## Fonctions membres publiques

- [TriaSfe3\\_cm3pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaSfe3\\_cm3pti](#) (int num\_mail, int num\_id)
- [TriaSfe3\\_cm3pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaSfe3\\_cm3pti](#) (const [TriaSfe3\\_cm3pti](#) &tria)

- `Element * Nevez_copie () const`
- `TriaSfe3_cm3pti & operator= (TriaSfe3_cm3pti &tria)`
- `void AfficheVarDual (ofstream &sort, Tableau< string > &nom)`
- `double KSI (int i)`

### Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- `static SfeMembT::DonnComSfe * doCoSfe3 = NULL`
- `static SfeMembT::UneFois uneFoisSfe3`
- `static NombresConstruireTriaSfe3_cm3pti nombre_V`
- `static ConsTriaSfe3_cm3pti consTriaSfe3_cm3pti`

### Membres hérités additionnels

#### 6.890.1 Documentation des fonctions membres

##### 6.890.1.1 KSI()

```
double TriaSfe3_cm3pti::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.890.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3_cm3pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.890.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3_cm3pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

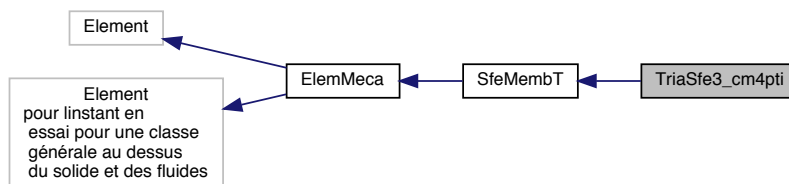
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

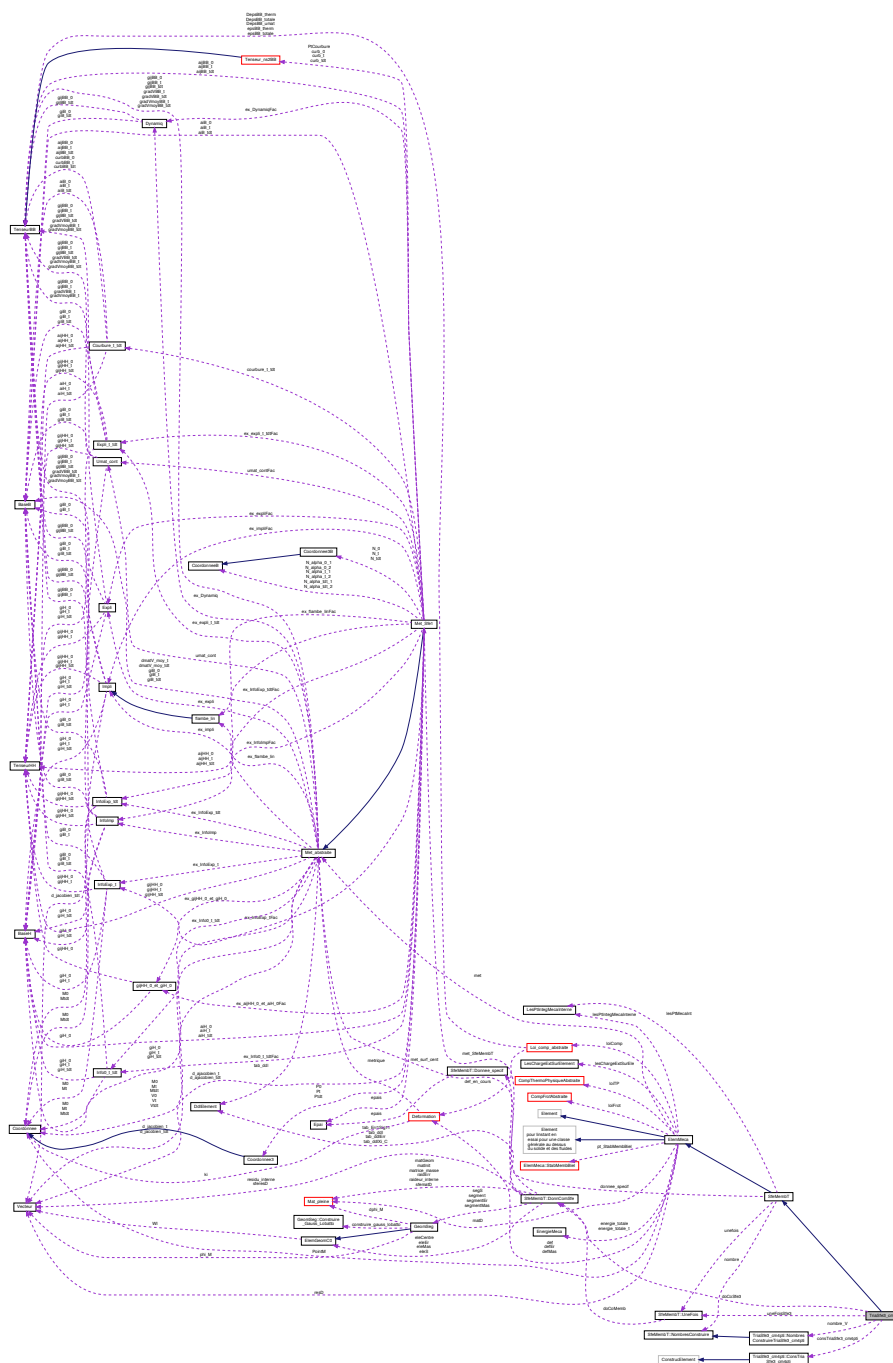
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm3pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm3pti.cc`

## 6.891 Référence de la classe TriaSfe3\_cm4pti

Graphe d'héritage de TriaSfe3\_cm4pti:



Graphe de collaboration de TriaSfe3\_cm4pti:



## Classes

- class [ConsTriaSfe3\\_cm4pti](#)
- class [NombresConstruireTriaSfe3\\_cm4pti](#)

## Fonctions membres publiques

- [TriaSfe3\\_cm4pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaSfe3\\_cm4pti](#) (int num\_mail, int num\_id)
- [TriaSfe3\\_cm4pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaSfe3\\_cm4pti](#) (const [TriaSfe3\\_cm4pti](#) &tria)

- Element \* **Nevez\_copie** () const
- [TriaSfe3\\_cm4pti](#) & **operator=** ([TriaSfe3\\_cm4pti](#) &tria)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- double **KSI** (int i)

### Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

### Attributs protégés statiques

- static [SfeMembT::DonnComSfe](#) \* **doCoSfe3** = NULL
- static [SfeMembT::UneFois](#) **uneFoisSfe3**
- static [NombresConstruireTriaSfe3\\_cm4pti](#) **nombre\_V**
- static [ConsTriaSfe3\\_cm4pti](#) **consTriaSfe3\_cm4pti**

### Membres hérités additionnels

#### 6.891.1 Documentation des fonctions membres

##### 6.891.1.1 KSI()

```
double TriaSfe3_cm4pti::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.891.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3_cm4pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.891.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3_cm4pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

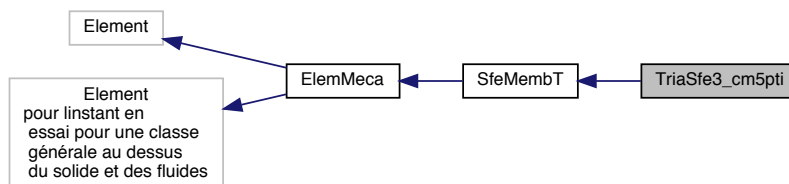
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm4pti.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3\_cm4pti.cc

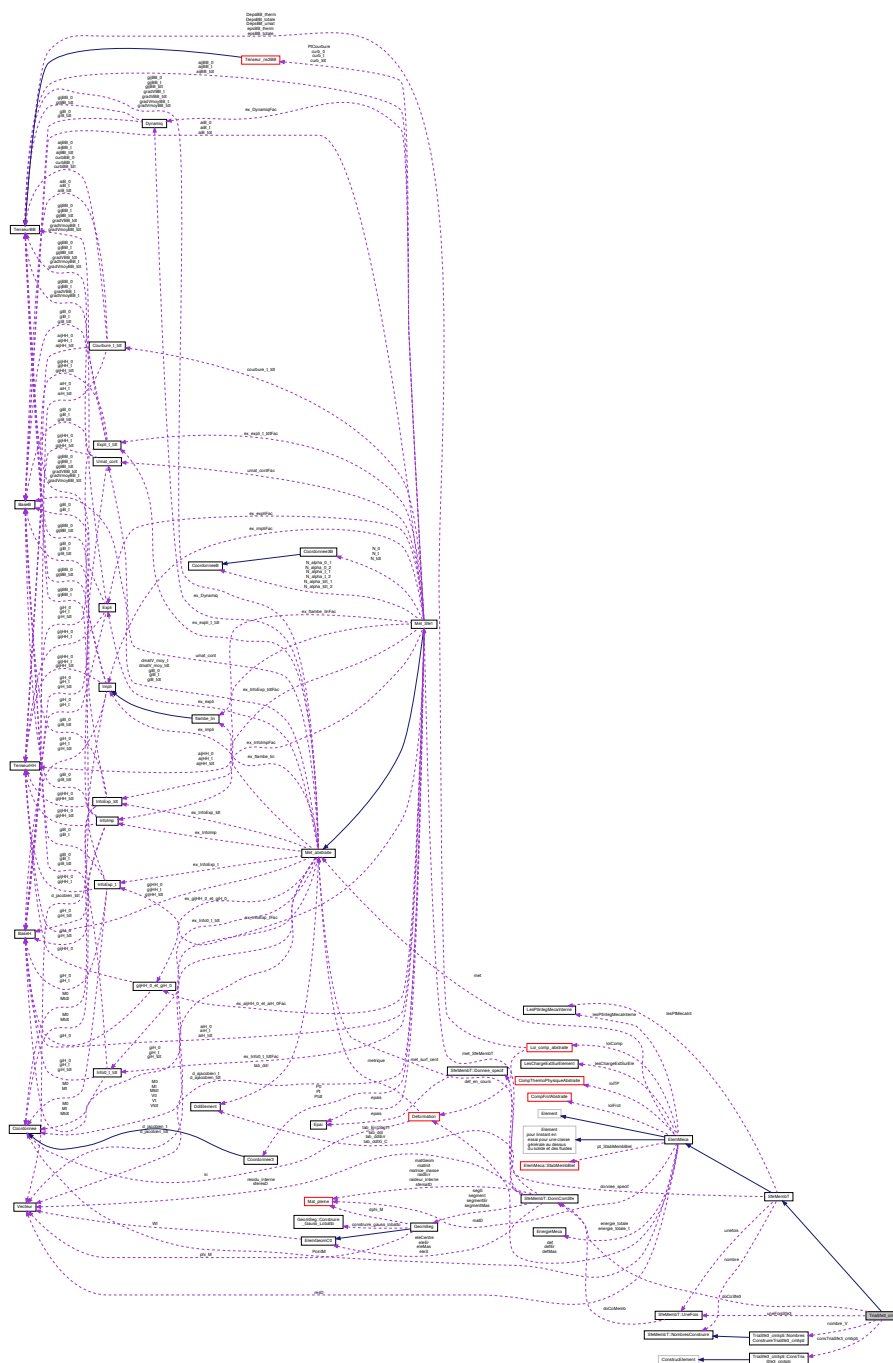
## 6.892 Référence de la classe TriaSfe3\_cm5pti

Graphe d'héritage de TriaSfe3\_cm5pti:





Graphe de collaboration de TriaSfe3\_cm5pti:



## Classes

- class [ConsTriaSfe3\\_cm5pti](#)
- class [NombresConstruireTriaSfe3\\_cm5pti](#)

## Fonctions membres publiques

- [TriaSfe3\\_cm5pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaSfe3\\_cm5pti](#) (int num\_mail, int num\_id)
- [TriaSfe3\\_cm5pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaSfe3\\_cm5pti](#) (const [TriaSfe3\\_cm5pti](#) &tria)

- `Element * Nevez_copie () const`
- `TriaSfe3_cm5pti & operator= (TriaSfe3_cm5pti &tria)`
- `void AfficheVarDual (ofstream &sort, Tableau< string > &nom)`
- `double KSI (int i)`

### Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- `static SfeMembT::DonnComSfe * doCoSfe3 = NULL`
- `static SfeMembT::UneFois uneFoisSfe3`
- `static NombresConstruireTriaSfe3_cm5pti nombre_V`
- `static ConsTriaSfe3_cm5pti consTriaSfe3_cm5pti`

### Membres hérités additionnels

#### 6.892.1 Documentation des fonctions membres

##### 6.892.1.1 KSI()

```
double TriaSfe3_cm5pti::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.892.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3_cm5pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.892.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3_cm5pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

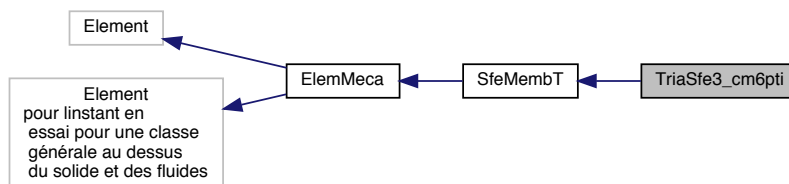
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

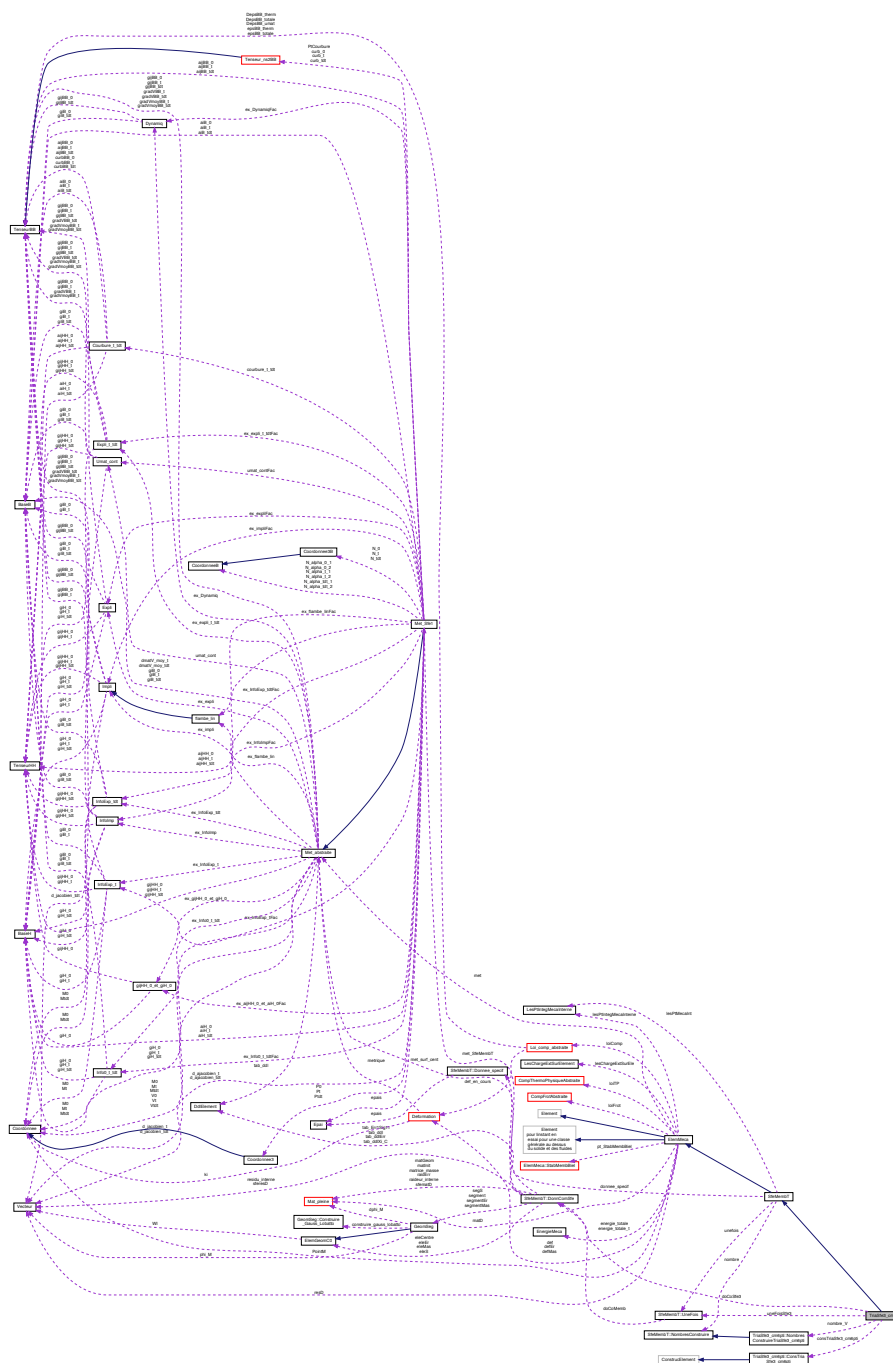
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm5pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm5pti.cc`

## 6.893 Référence de la classe TriaSfe3\_cm6pti

Graphe d'héritage de TriaSfe3\_cm6pti:



Graphe de collaboration de TriaSfe3\_cm6pti:



## Classes

- class [ConsTriaSfe3\\_cm6pti](#)
- class [NombresConstruireTriaSfe3\\_cm6pti](#)

## Fonctions membres publiques

- [TriaSfe3\\_cm6pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaSfe3\\_cm6pti](#) (int num\_mail, int num\_id)
- [TriaSfe3\\_cm6pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaSfe3\\_cm6pti](#) (const [TriaSfe3\\_cm6pti](#) &tria)

- `Element * Nevez_copie ()` const
- `TriaSfe3_cm6pti & operator= (TriaSfe3_cm6pti &tria)`
- void `AfficheVarDual` (ofstream &sort, `Tableau< string > &nom`)
- double `KSI` (int i)

### Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin` (int, `Tableau< Noeud * > &tab`, `DdlElement &ddelem`)
- `ElFrontiere * new_frontiere_surf` (int, `Tableau< Noeud * > &tab`, `DdlElement &ddelem`)

### Attributs protégés statiques

- static `SfeMembT::DonnComSfe * doCoSfe3` = NULL
- static `SfeMembT::UneFois uneFoisSfe3`
- static `NombresConstruireTriaSfe3_cm6pti nombre_V`
- static `ConsTriaSfe3_cm6pti consTriaSfe3_cm6pti`

### Membres hérités additionnels

#### 6.893.1 Documentation des fonctions membres

##### 6.893.1.1 `KSI()`

```
double TriaSfe3_cm6pti::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.893.1.2 `new_frontiere_lin()`

```
ElFrontiere * TriaSfe3_cm6pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.893.1.3 `new_frontiere_surf()`

```
ElFrontiere * TriaSfe3_cm6pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

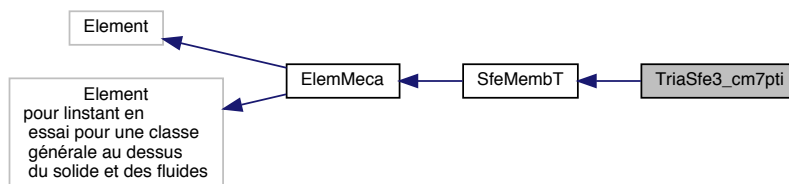
Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

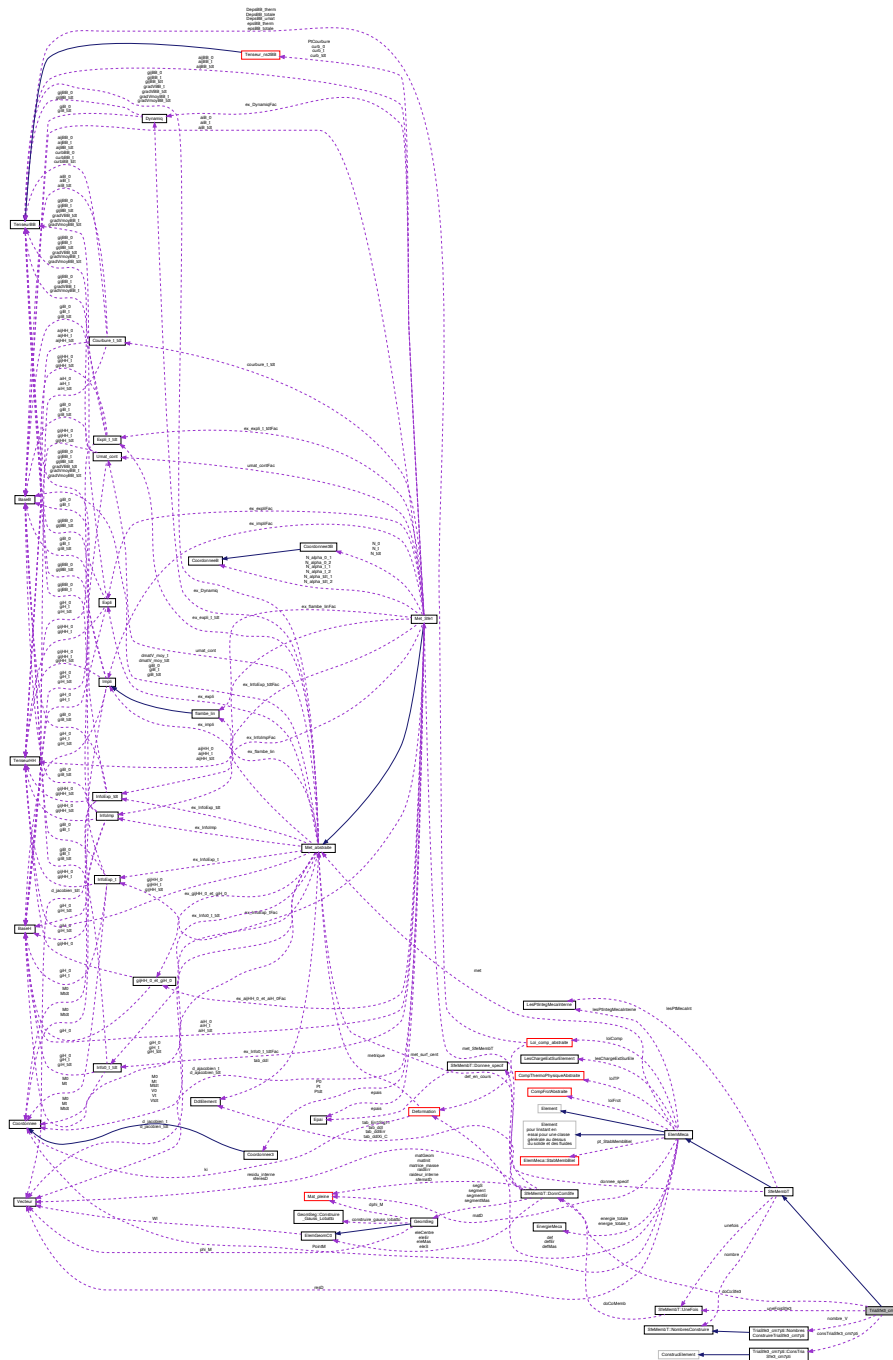
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm6pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm6pti.cc`

## 6.894 Référence de la classe TriaSfe3\_cm7pti

Graphe d'héritage de TriaSfe3\_cm7pti:



Graphe de collaboration de TriaSfe3\_cm7pti:



## Classes

- class [ConsTriaSfe3\\_cm7pti](#)
- class [NombresConstruireTriaSfe3\\_cm7pti](#)

## Fonctions membres publiques

- [TriaSfe3\\_cm7pti](#) (double epaiss, int num\_mail=0, int num\_id=-3)
- [TriaSfe3\\_cm7pti](#) (int num\_mail, int num\_id)
- [TriaSfe3\\_cm7pti](#) (double epaiss, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- [TriaSfe3\\_cm7pti](#) (const [TriaSfe3\\_cm7pti](#) &tria)

- `Element * Nevez_copie () const`
- `TriaSfe3_cm7pti & operator= (TriaSfe3_cm7pti &tria)`
- `void AfficheVarDual (ofstream &sort, Tableau< string > &nom)`
- `double KSI (int i)`

### Fonctions membres protégées

- `ElFrontiere * new_frontiere_lin (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`
- `ElFrontiere * new_frontiere_surf (int, Tableau< Noeud * > &tab, DdlElement &ddelem)`

### Attributs protégés statiques

- `static SfeMembT::DonnComSfe * doCoSfe3 = NULL`
- `static SfeMembT::UneFois uneFoisSfe3`
- `static NombresConstruireTriaSfe3_cm7pti nombre_V`
- `static ConsTriaSfe3_cm7pti consTriaSfe3_cm7pti`

### Membres hérités additionnels

#### 6.894.1 Documentation des fonctions membres

##### 6.894.1.1 KSI()

```
double TriaSfe3_cm7pti::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.894.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3_cm7pti::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.894.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3_cm7pti::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

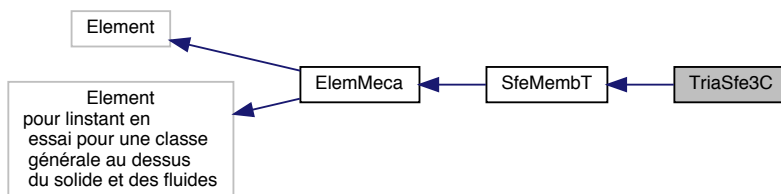
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm7pti.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3_cm7pti.cc`

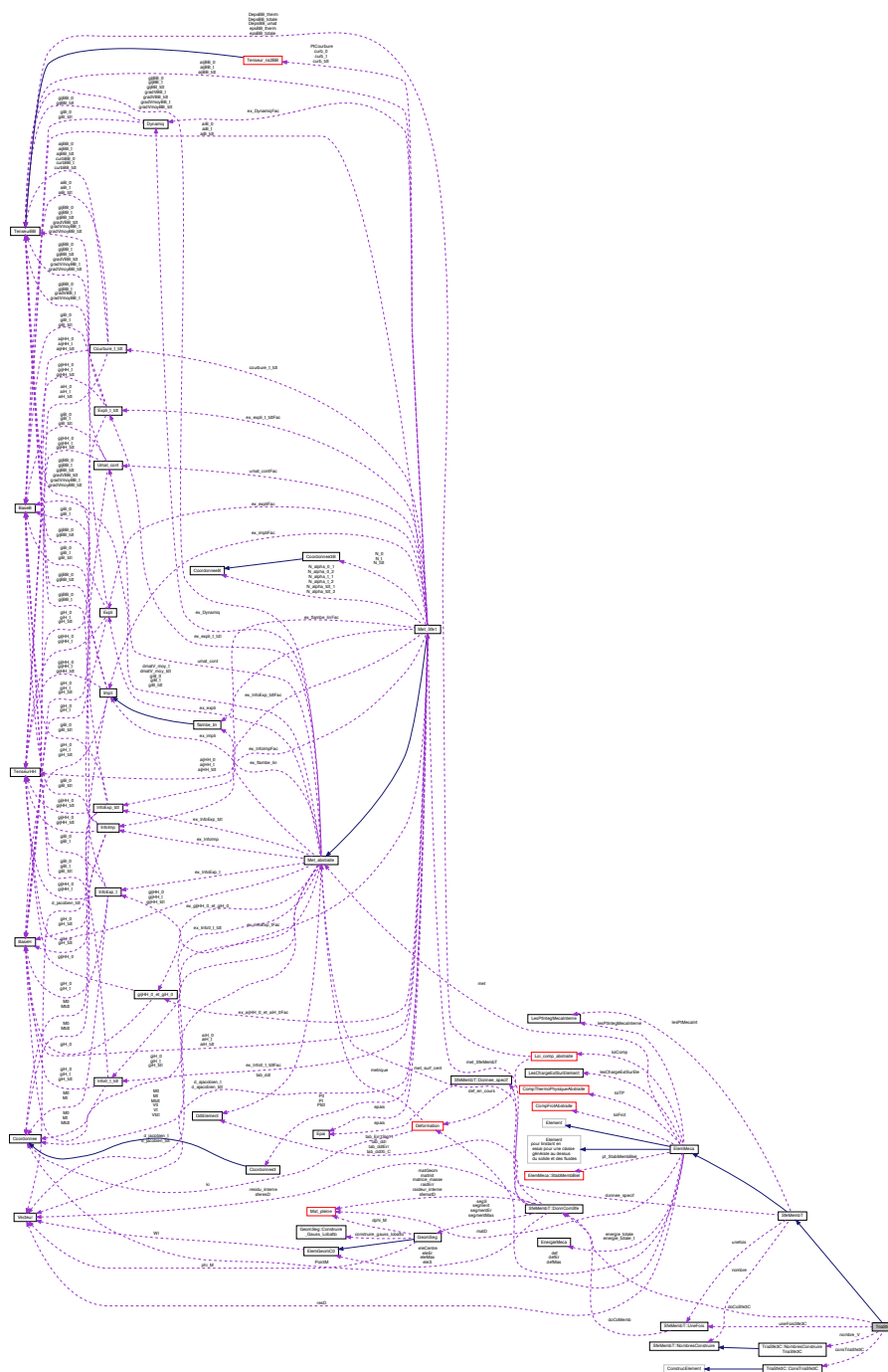


## 6.895 Référence de la classe TriaSfe3C

Graphe d'héritage de TriaSfe3C:



Graphe de collaboration de TriaSfe3C:



## Classes

- class [ConsTriaSfe3C](#)
- class [NombresConstruireTriaSfe3C](#)

## Fonctions membres publiques

- **TriaSfe3C** (double epais, int num\_mail=0, int num\_id=-3)
- **TriaSfe3C** (int num\_mail, int num\_id)
- **TriaSfe3C** (double epais, int num\_mail, int num\_id, const [Tableau](#)< [Noeud](#) \* > &tab)
- **TriaSfe3C** (const [TriaSfe3C](#) &tria)

- Element \* **Nevez\_copie** () const
- [TriaSfe3C](#) & **operator=** ([TriaSfe3C](#) &tria)
- void **AfficheVarDual** (ofstream &sort, [Tableau](#)< string > &nom)
- double **KSI** (int i)

### Fonctions membres protégées

- [ElFrontiere](#) \* **new\_frontiere\_lin** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)
- [ElFrontiere](#) \* **new\_frontiere\_surf** (int, [Tableau](#)< [Noeud](#) \* > &tab, [DdlElement](#) &ddelem)

### Attributs protégés statiques

- static [SfeMembT::DonnComSfe](#) \* **doCoSfe3C** = NULL
- static [SfeMembT::UneFois](#) **uneFoisSfe3C**
- static [NombresConstruireTriaSfe3C](#) **nombre\_V**
- static [ConsTriaSfe3C](#) **consTriaSfe3C**

### Membres hérités additionnels

#### 6.895.1 Documentation des fonctions membres

##### 6.895.1.1 KSI()

```
double TriaSfe3C::KSI (
    int i ) [inline], [virtual]
```

Implémente [SfeMembT](#).

##### 6.895.1.2 new\_frontiere\_lin()

```
ElFrontiere * TriaSfe3C::new_frontiere_lin (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

##### 6.895.1.3 new\_frontiere\_surf()

```
ElFrontiere * TriaSfe3C::new_frontiere_surf (
    int ,
    Tableau< Noeud * > & tab,
    DdlElement & ddelem ) [inline], [protected], [virtual]
```

Implémente [ElemMeca](#).

La documentation de cette classe a été générée à partir du fichier suivant :

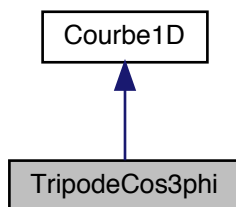
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3C.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/TriaSfe3C.cc

## 6.896 Référence de la classe TripodeCos3phi

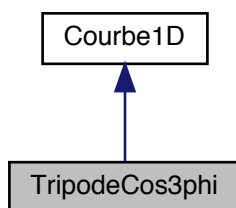
Classe permettant le calcul d'une fonction 1D de type :  $f(x) = 1./(1.+gamma*cos(3*phi))^n$ .

```
#include <TripodeCos3phi.h>
```

Grappe d'héritage de TripodeCos3phi:



Grappe de collaboration de TripodeCos3phi:



## Fonctions membres publiques

- **TripodeCos3phi** (string nom="")
- **TripodeCos3phi** (const [TripodeCos3phi](#) &Co)
- **TripodeCos3phi** (const [Courbe1D](#) &Co)
- void [Affiche](#) () const  
*affichage de la courbe*
- bool [Compleet\\_courbe](#) () const  
*vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon*
- void [LectDonnParticulieres\\_courbes](#) (const string &nom, [UtilLecture](#) \*)  
*Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.*
- void [Info\\_commande\\_Courbes1D](#) ([UtilLecture](#) &entreePrinc)  
*def info fichier de commande*
- double [Valeur](#) (double x)  
*ramène la valeur*
- [Courbe1D::ValDer Valeur\\_Et\\_derivee](#) (double x)  
*ramène la valeur et la dérivée en paramètre*
- double [Derivee](#) (double x)  
*ramène la dérivée*
- [Courbe1D::ValDer2 Valeur\\_Et\\_der12](#) (double x)  
*ramène la valeur et les dérivées première et seconde en paramètre*
- double [Der\\_sec](#) (double x)  
*ramène la dérivée seconde*
- [Courbe1D::Valbool Valeur\\_stricte](#) (double x)

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au  $x_{\text{mini}}$ , ramène la valeur minimale possible de  $y$  si supérieur au  $x_{\text{maxi}}$ , ramène la valeur maximale possible de  $y$

- [Courbe1D::ValDerbool Valeur\\_Et\\_derivee\\_stricte](#) (double  $x$ )

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au  $x_{\text{mini}}$ , ramène la valeur minimale possible de  $y$  et  $Y'$  correspondant si supérieur au  $x_{\text{maxi}}$ , ramène la valeur maximale possible de  $y$  et  $Y'$  correspondant

- void [Lecture\\_base\\_info](#) (ifstream &ent, const int cas)

— cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

- void [Ecriture\\_base\\_info](#) (ofstream &sort, const int cas)

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

- void [SchemaXML\\_Courbes1D](#) (ofstream &sort, const [Enum\\_IO\\_XML](#) enu)

sortie du schemaXML: en fonction de enu

## Attributs protégés

- double **xn**
- double **gamma**
- bool **val\_absolu**

## Membres hérités additionnels

### 6.896.1 Description détaillée

Classe permettant le calcul d'une fonction 1D de type :  $f(x) = 1./(1.+gamma*cos(3*phi))^n$ .

BUT: Classe permettant le calcul d'une fonction 1D de type :  $f(x) = 1./(1.+gamma*cos(3*phi))^n$  ainsi qu'un certain nombre d'informations supplémentaires telles que dérivées.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

30/03/2008

### 6.896.2 Documentation des fonctions membres

#### 6.896.2.1 Affiche()

```
void TripodeCos3phi::Affiche ( ) const [virtual]
```

affichage de la courbe

Implémente [Courbe1D](#).

#### 6.896.2.2 Complet\_courbe()

```
bool TripodeCos3phi::Complet_courbe ( ) const [virtual]
```

vérification que tout est ok, pres à l'emploi ramène true si ok, false sinon

Implémente [Courbe1D](#).

### 6.896.2.3 Der\_sec()

```
double TripodeCos3phi::Der_sec (
    double x ) [virtual]
```

ramène la dérivée seconde

Implémente [Courbe1D](#).

### 6.896.2.4 Derivee()

```
double TripodeCos3phi::Derivee (
    double x ) [virtual]
```

ramène la dérivée

Implémente [Courbe1D](#).

### 6.896.2.5 Ecriture\_base\_info()

```
void TripodeCos3phi::Ecriture_base_info (
    ofstream & sort,
    const int cas ) [virtual]
```

cas donne le niveau de sauvegarde = 1 : on sauvegarde tout = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.896.2.6 Info\_commande\_Courbes1D()

```
void TripodeCos3phi::Info_commande_Courbes1D (
    UtilLecture & entreePrinc ) [virtual]
```

def info fichier de commande

Implémente [Courbe1D](#).

### 6.896.2.7 LectDonnParticulieres\_courbes()

```
void TripodeCos3phi::LectDonnParticulieres_courbes (
    const string & nom,
    UtilLecture * ) [virtual]
```

Lecture des donnees de la classe sur fichier le nom passé en paramètre est le nom de la courbe s'il est vide c-a-d = "", la methode commence par lire le nom sinon ce nom remplace le nom actuel.

Implémente [Courbe1D](#).

### 6.896.2.8 Lecture\_base\_info()

```
void TripodeCos3phi::Lecture_base_info (
    ifstream & ent,
    const int cas ) [virtual]
```

--- lecture écriture de restart --- cas donne le niveau de la récupération = 1 : on récupère tout = 2 : on récupère uniquement les données variables (supposées comme telles)

Implémente [Courbe1D](#).

### 6.896.2.9 SchemaXML\_Courbes1D()

```
void TripodeCos3phi::SchemaXML_Courbes1D (
    ofstream & sort,
    const Enum_IO_XML enu ) [virtual]
```

sortie du schemaXML: en fonction de enu

Implémente [Courbe1D](#).

#### 6.896.2.10 Valeur()

```
double TripodeCos3phi::Valeur (
    double x ) [virtual]
```

ramène la valeur

Implémente [Courbe1D](#).

#### 6.896.2.11 Valeur\_Et\_der12()

```
Courbe1D::ValDer2 TripodeCos3phi::Valeur_Et_der12 (
    double x ) [virtual]
```

ramène la valeur et les dérivées première et seconde en paramètre

Implémente [Courbe1D](#).

#### 6.896.2.12 Valeur\_Et\_derivee()

```
Courbe1D::ValDer TripodeCos3phi::Valeur_Et_derivee (
    double x ) [virtual]
```

ramène la valeur et la dérivée en paramètre

Implémente [Courbe1D](#).

#### 6.896.2.13 Valeur\_Et\_derivee\_stricte()

```
Courbe1D::ValDerbool TripodeCos3phi::Valeur_Et_derivee_stricte (
    double x ) [virtual]
```

ramène la valeur et la dérivée si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant

Implémente [Courbe1D](#).

#### 6.896.2.14 Valeur\_stricte()

```
Courbe1D::Valbool TripodeCos3phi::Valeur_stricte (
    double x ) [virtual]
```

ramène la valeur si dans le domaine strictement de définition si c'est inférieur au x mini, ramène la valeur minimale possible de y si supérieur au x maxi , ramène le valeur maximale possible de y

Implémente [Courbe1D](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/TripodeCos3phi.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/TripodeCos3phi.cc

## 6.897 Référence de la classe Trois\_String

cas de trois String

```
#include <Basiques.h>
```

### Fonctions membres publiques

- **Trois\_String** (const string &n1, const string &n2, const string &n3)
- **Trois\_String** (const [Trois\\_String](#) &de)
- **Trois\_String & operator=** (const [Trois\\_String](#) &de)

- `istream & LectXML_Trois_String` (`istream &ent`)
- `ostream & EcritXML_Trois_String` (`ostream &sort`)
- `bool operator==` (`const Trois_String &a`) `const`
- `bool operator!=` (`const Trois_String &a`) `const`
- `void SchemaXML_Trois_String` (`ofstream &sort`, `const Enum_IO_XML enu`) `const`
- `bool operator>` (`const Trois_String &a`) `const`
- `bool operator>=` (`const Trois_String &a`) `const`
- `bool operator<` (`const Trois_String &a`) `const`
- `bool operator<=` (`const Trois_String &a`) `const`

### Attributs publics

- `string nom1`
- `string nom2`
- `string nom3`

### Attributs publics statiques

- `static short int impre_schem_XML =0`

### Amis

- `istream & operator>>` (`istream &ent`, `Trois_String &de`)
- `ostream & operator<<` (`ostream &sort`, `const Trois_String &de`)

## 6.897.1 Description détaillée

cas de trois String

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.cc`

## 6.898 Référence de la classe TroisEntiers

cas de 3 entiers

```
#include <Basiques.h>
```

### Fonctions membres publiques

- `TroisEntiers` (`int u`, `int v`, `int w`)
- `TroisEntiers` (`const TroisEntiers &de`)
- `TroisEntiers & operator=` (`const TroisEntiers &de`)
- `istream & LectXML_TroisEntiers` (`istream &ent`)
- `ostream & EcritXML_TroisEntiers` (`ostream &sort`)
- `bool operator==` (`const TroisEntiers &a`) `const`
- `bool operator!=` (`const TroisEntiers &a`) `const`
- `bool operator>` (`const TroisEntiers &a`) `const`
- `bool operator>=` (`const TroisEntiers &a`) `const`
- `bool operator<` (`const TroisEntiers &a`) `const`
- `bool operator<=` (`const TroisEntiers &a`) `const`
- `void SchemaXML_TroisEntiers` (`ofstream &sort`, `const Enum_IO_XML enu`) `const`

### Attributs publics

- `int un`
- `int deux`
- `int trois`

### Attributs publics statiques

- `static short int impre_schem_XML =0`



## Amis

- `istream & operator>>` (`istream &ent`, [TroisEntiers](#) &de)
- `ostream & operator<<` (`ostream &sort`, const [TroisEntiers](#) &de)

### 6.898.1 Description détaillée

cas de 3 entiers

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Basiques.cc`

## 6.899 Référence de la classe `Type_Calcul`

### Fonctions membres publiques

- void `LectureTypeCalcul` ([UtilLecture](#) &lec)
- bool `SousType` ()
- bool `Remonte_in` ()
- bool `Avec_in` (const [EnumSousTypeCalcul](#) sousTypeCalcul)
- bool `Erreur_in` ()

## Amis

- `istream & operator>>` (`istream &entree`, [Type\\_Calcul](#) &a)
- `ostream & operator<<` (`ostream &sort`, const [Type\\_Calcul](#) &a)

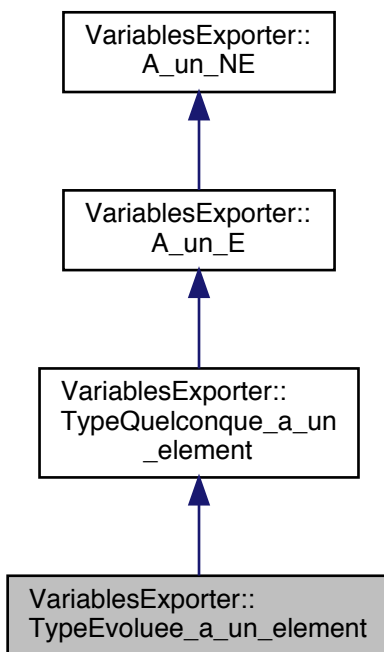
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/Type_Calcul.h`

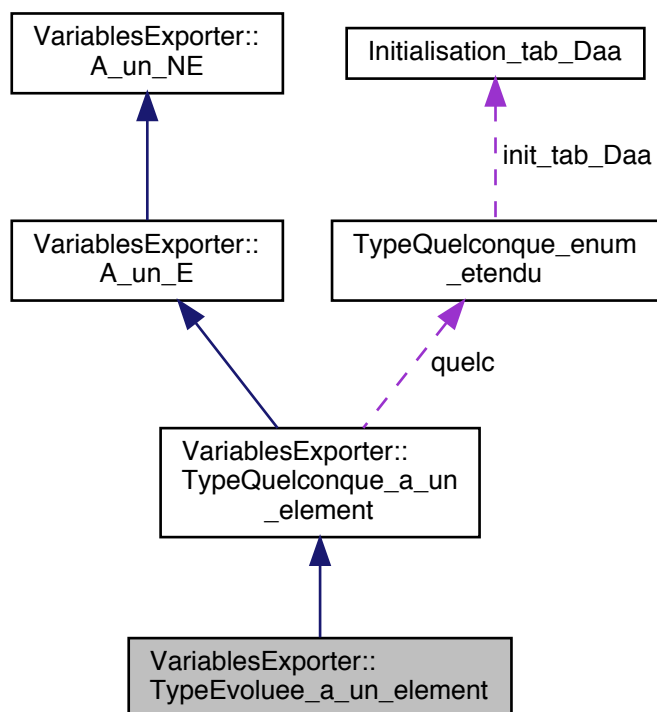
## 6.900 Référence de la classe

### VariablesExporter::TypeEvoluee\_a\_un\_element

Graphe d'héritage de VariablesExporter::TypeEvoluee\_a\_un\_element:



Graphe de collaboration de VariablesExporter::TypeEvoluee\_a\_un\_element:



## Fonctions membres publiques

- `TypeEvoluee_a_un_element` (int absolu\_, `TypeQuelconque_enum_etendu` e, string ref\_noeud, string nom\_mail\_, `Enum_dure` tps, string nom\_var\_, int nbpti, int num\_ord)
- `TypeEvoluee_a_un_element` (const `TypeEvoluee_a_un_element` &a)
- `TypeEvoluee_a_un_element` & `operator=` (const `TypeEvoluee_a_un_element` &a)
- bool `operator==` (const `TypeEvoluee_a_un_element` &a) const
- bool `operator!=` (const `TypeEvoluee_a_un_element` &a) const
- bool `operator<` (const `TypeEvoluee_a_un_element` &a) const
- bool `operator<=` (const `TypeEvoluee_a_un_element` &a) const
- bool `operator>` (const `TypeEvoluee_a_un_element` &a) const
- bool `operator>=` (const `TypeEvoluee_a_un_element` &a) const
- void `Affiche` ()

## Amis

- istream & `operator>>` (istream &, `TypeEvoluee_a_un_element` &)
- ostream & `operator<<` (ostream &, const `TypeEvoluee_a_un_element` &)

## Membres hérités additionnels

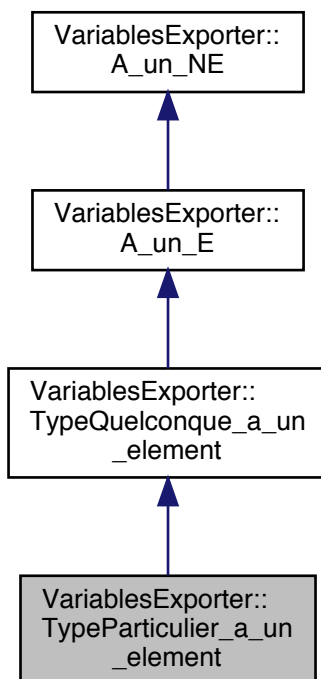
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

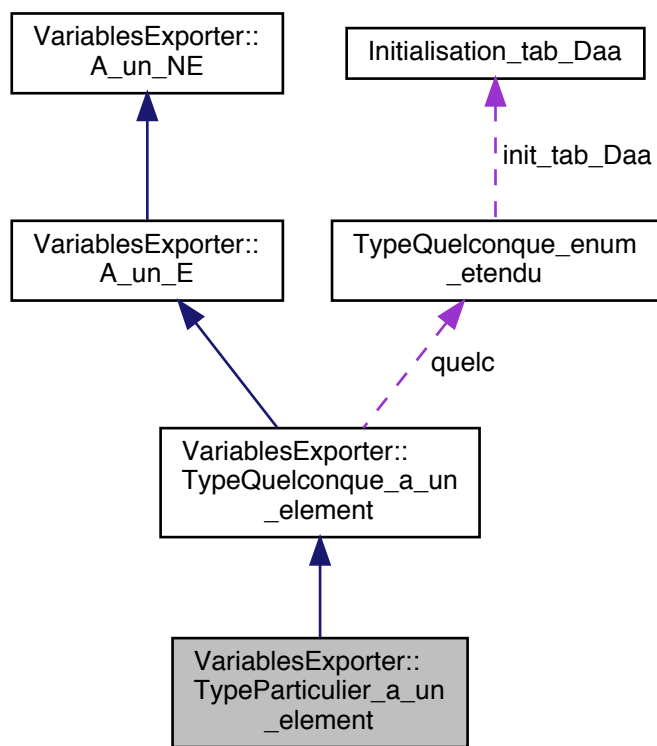
## 6.901 Référence de la classe

### VariablesExporter::TypeParticulier\_a\_un\_element

Graphe d'héritage de VariablesExporter::TypeParticulier\_a\_un\_element:



Graphe de collaboration de VariablesExporter::TypeParticulier\_a\_un\_element:



## Fonctions membres publiques

- **TypeParticulier\_a\_un\_element** (int absolu\_, [TypeQuelconque\\_enum\\_etendu](#) e, string ref\_noeud, string nom\_mail\_, [Enum\\_dure](#) tps, string nom\_var\_, int nbpti, int num\_ord)
- **TypeParticulier\_a\_un\_element** (const [TypeParticulier\\_a\\_un\\_element](#) &a)
- **TypeParticulier\_a\_un\_element &operator=** (const [TypeParticulier\\_a\\_un\\_element](#) &a)
- bool **operator==** (const [TypeParticulier\\_a\\_un\\_element](#) &a) const
- bool **operator!=** (const [TypeParticulier\\_a\\_un\\_element](#) &a) const
- bool **operator<** (const [TypeParticulier\\_a\\_un\\_element](#) &a) const
- bool **operator<=** (const [TypeParticulier\\_a\\_un\\_element](#) &a) const
- bool **operator>** (const [TypeParticulier\\_a\\_un\\_element](#) &a) const
- bool **operator>=** (const [TypeParticulier\\_a\\_un\\_element](#) &a) const
- void **Affiche** ()

## Amis

- istream & **operator>>** (istream &, [TypeParticulier\\_a\\_un\\_element](#) &)
- ostream & **operator<<** (ostream &, const [TypeParticulier\\_a\\_un\\_element](#) &)

## Membres hérités additionnels

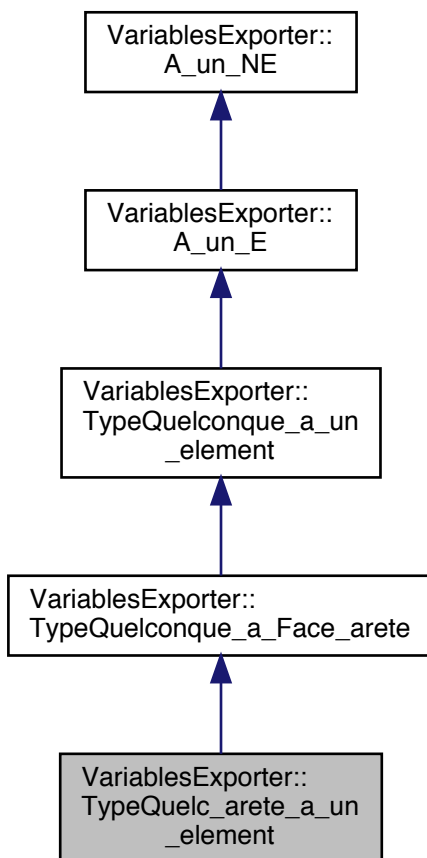
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

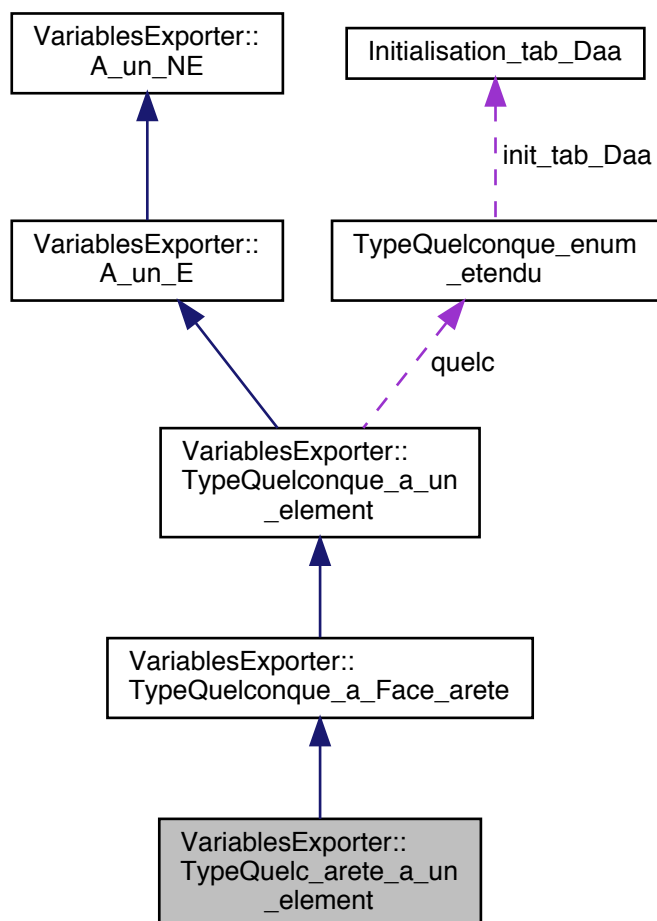
## 6.902 Référence de la classe

### VariablesExporter::TypeQuelc\_arete\_a\_un\_element

Grphe d'héritage de VariablesExporter::TypeQuelc\_arete\_a\_un\_element:



Graphe de collaboration de VariablesExporter::TypeQuelc\_arete\_a\_un\_element:



## Fonctions membres publiques

- `TypeQuelc_arete_a_un_element` (int absolu\_, `TypeQuelconque_enum_etendu` e, string ref\_element, string nom\_mail\_, int nbFA, `Enum_dure` tps, string nom\_var\_, int nbpti, int num\_ord)
- `TypeQuelc_arete_a_un_element` (const `TypeQuelc_arete_a_un_element` &a)
- `TypeQuelc_arete_a_un_element` & **operator=** (const `TypeQuelc_arete_a_un_element` &a)
- bool **operator==** (const `TypeQuelc_arete_a_un_element` &a) const
- bool **operator!=** (const `TypeQuelc_arete_a_un_element` &a) const
- bool **operator<** (const `TypeQuelc_arete_a_un_element` &a) const
- bool **operator<=** (const `TypeQuelc_arete_a_un_element` &a) const
- bool **operator>** (const `TypeQuelc_arete_a_un_element` &a) const
- bool **operator>=** (const `TypeQuelc_arete_a_un_element` &a) const
- void **Affiche** ()

## Amis

- istream & **operator>>** (istream &, `TypeQuelc_arete_a_un_element` &)
- ostream & **operator<<** (ostream &, const `TypeQuelc_arete_a_un_element` &)

## Membres hérités additionnels

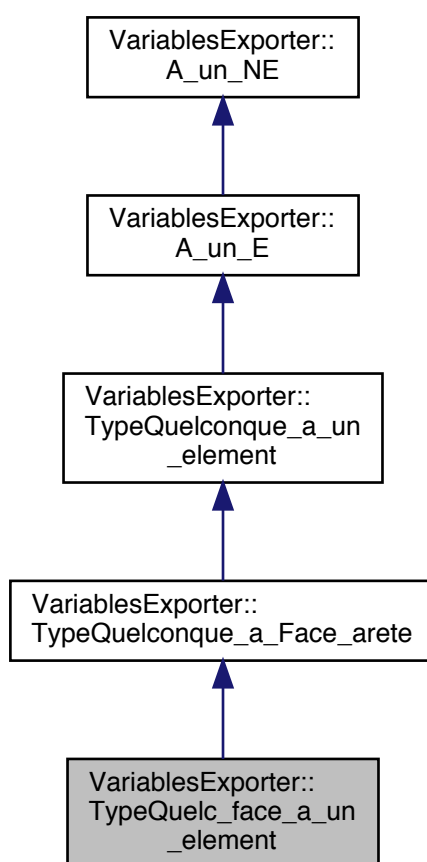
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

## 6.903 Référence de la classe

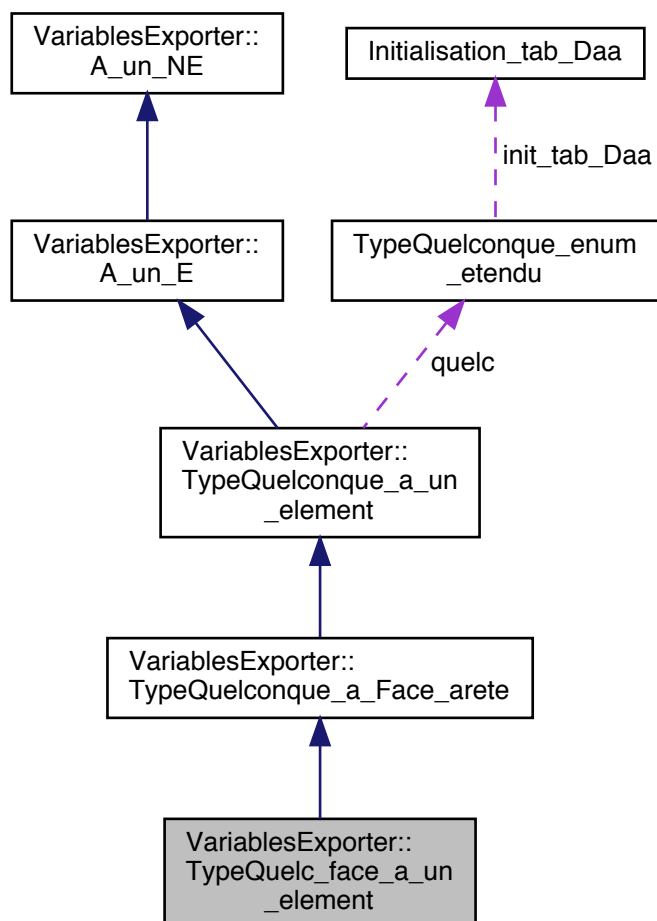
### VariablesExporter::TypeQuelc\_face\_a\_un\_element

Grphe d'héritage de VariablesExporter::TypeQuelc\_face\_a\_un\_element:





Graphe de collaboration de VariablesExporter::TypeQuelc\_face\_a\_un\_element:



## Fonctions membres publiques

- `TypeQuelc_face_a_un_element` (int absolu\_, `TypeQuelconque_enum_etendu` e, string ref\_element, string nom\_mail\_, int nbFA, `Enum_dure` tps, string nom\_var\_, int nbpti, int num\_ord)
- `TypeQuelc_face_a_un_element` (const `TypeQuelc_face_a_un_element` &a)
- `TypeQuelc_face_a_un_element` & **operator=** (const `TypeQuelc_face_a_un_element` &a)
- bool **operator==** (const `TypeQuelc_face_a_un_element` &a) const
- bool **operator!=** (const `TypeQuelc_face_a_un_element` &a) const
- bool **operator<** (const `TypeQuelc_face_a_un_element` &a) const
- bool **operator<=** (const `TypeQuelc_face_a_un_element` &a) const
- bool **operator>** (const `TypeQuelc_face_a_un_element` &a) const
- bool **operator>=** (const `TypeQuelc_face_a_un_element` &a) const
- void **Affiche** ()

## Amis

- `istream` & **operator>>** (`istream` &, `TypeQuelc_face_a_un_element` &)
- `ostream` & **operator<<** (`ostream` &, const `TypeQuelc_face_a_un_element` &)

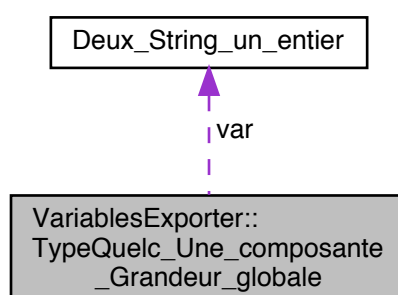
## Membres hérités additionnels

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

## 6.904 Référence de la classe VariablesExporter::TypeQuelc\_Une\_composante\_Grandeur\_globale

Graphe de collaboration de VariablesExporter::TypeQuelc\_Une\_composante\_Grandeur\_globale:



## Fonctions membres publiques

- `TypeQuelc_Une_composante_Grandeur_globale` (string nom\_var, string nom\_glob, int num\_ord)
- `TypeQuelc_Une_composante_Grandeur_globale` (const `TypeQuelc_Une_composante_Grandeur_globale` &a)
- `TypeQuelc_Une_composante_Grandeur_globale` & `operator=` (const `TypeQuelc_Une_composante_Grandeur_globale` &a)
- bool `operator==` (const `TypeQuelc_Une_composante_Grandeur_globale` &a) const
- bool `operator!=` (const `TypeQuelc_Une_composante_Grandeur_globale` &a) const
- bool `operator<` (const `TypeQuelc_Une_composante_Grandeur_globale` &a) const
- bool `operator<=` (const `TypeQuelc_Une_composante_Grandeur_globale` &a) const
- bool `operator>` (const `TypeQuelc_Une_composante_Grandeur_globale` &a) const
- bool `operator>=` (const `TypeQuelc_Une_composante_Grandeur_globale` &a) const
- void `Affiche` ()
- const string & `Nom_var_const` () const
- string & `Nom_var` ()
- const string & `Nom_grandeur_globale_const` () const
- string & `Nom_grandeur_globale` ()
- const int & `Indice_const` () const
- int & `Indice` ()
- `Deux_String_un_entier` `Deux_String_un_entier_const` () const
- `Deux_String_un_entier` & `Deux_String_un_entier_` ()

## Attributs protégés

- `Deux_String_un_entier` var

## Amis

- `istream & operator>>` (`istream &`, `TypeQuelc_Une_composante_Grandeur_globale &`)
- `ostream & operator<<` (`ostream &`, `const TypeQuelc_Une_composante_Grandeur_globale &`)

La documentation de cette classe a été générée à partir du fichier suivant :

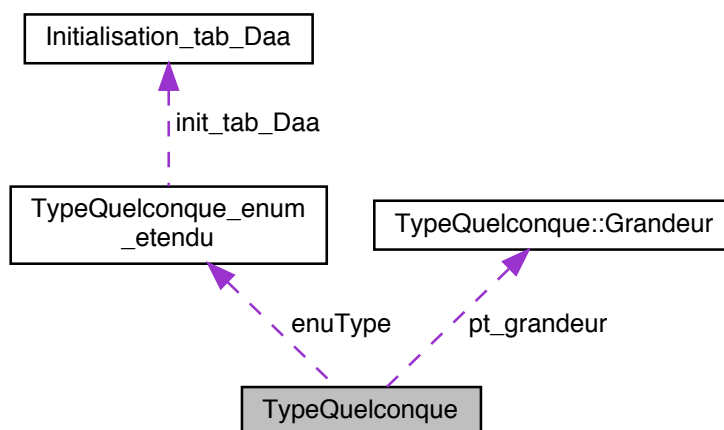
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc`

## 6.905 Référence de la classe TypeQuelconque

`TypeQuelconque` : def de la classe générique globale qui sert a gérer la grandeur particulière qui est attachée via un pointeur.

```
#include <TypeQuelconque.h>
```

Graphe de collaboration de `TypeQuelconque`:



## Classes

- class `Grandeur`  
*définition de la classe virtuelle qui serait déclinée en une grandeur particulière il s'agit ici de la classe conteneur*

## Fonctions membres publiques

- `TypeQuelconque` (`TypeQuelconque_enum_etendu enuType`, `Enum_ddl posi_ddl`, `const TypeQuelconque::Grandeur &pt_gr`)
- `TypeQuelconque` (`EnumTypeQuelconque enuType`, `Enum_ddl posi_ddl`, `const TypeQuelconque::Grandeur &pt_gr`)
- `TypeQuelconque` (`TypeQuelconque_enum_etendu enuType`)
- `TypeQuelconque` (`EnumTypeQuelconque enuType`)
- `TypeQuelconque` (`const TypeQuelconque &a`)
- `void ChangeGrandeur` (`const TypeQuelconque &a`)
- `TypeQuelconque & operator=` (`const TypeQuelconque &a`)
- `bool operator==` (`const TypeQuelconque &a`) `const`
- `bool operator!=` (`const TypeQuelconque &a`) `const`
- `bool operator>` (`const TypeQuelconque &a`) `const`
- `bool operator>=` (`const TypeQuelconque &a`) `const`
- `bool operator<` (`const TypeQuelconque &a`) `const`
- `bool operator<=` (`const TypeQuelconque &a`) `const`

- [EnumTypeGrandeur](#) **EnuTypeGrandeur** () const
- const [TypeQuelconque\\_enum\\_etendu](#) & **EnuTypeQuelconque** () const
- int **TypeExpressionGrandeur** () const
- void **Grandeur\_brut** (ostream &sort, int nbcarr) const
- Enum\_ddl **Enum** () const
- void **Change\_repere** (const [Mat\\_pleine](#) &beta0, const [Mat\\_pleine](#) &betat, const [Mat\\_pleine](#) &betatdt, const [Mat\\_pleine](#) &gamma0, const [Mat\\_pleine](#) &gammat, const [Mat\\_pleine](#) &gammatdt)
- bool **Type\_numerique** () const
- int **NbMaxiNumeroOrdre** () const
- double **GrandeurNumOrdre** (int num) const
- [Grandeur](#) \* **Grandeur\_pointee** ()
- const [Grandeur](#) \* **Const\_Grandeur\_pointee** () const
- void **Ecriture\_entete\_grandeur** (ostream &sort)
- [EnumTypeQuelParticulier](#) **Lecture\_un** (istream &ent)
- [EnumTypeQuelParticulier](#) **Lecture\_un\_avec\_creation\_TypeQuelconque\_enum\_etendu** (istream &ent)
- void **Active** ()
- void **Inactive** ()
- bool **Activite** () const

### Fonctions membres protégées

- void **Gestion\_erreur** () const

### Attributs protégés

- [TypeQuelconque\\_enum\\_etendu](#) **enuType**
- Enum\_ddl **enu\_ddl\_posi**
- [Grandeur](#) \* **pt\_grandeur**
- bool **actif**

### Amis

- istream & **operator**>> (istream &ent, [TypeQuelconque](#) &a)
- ostream & **operator**<< (ostream &sort, const [TypeQuelconque](#) &a)

### 6.905.1 Description détaillée

[TypeQuelconque](#) : def de la classe générique globale qui sert à gérer la grandeur particulière qui est attachée à un pointeur.

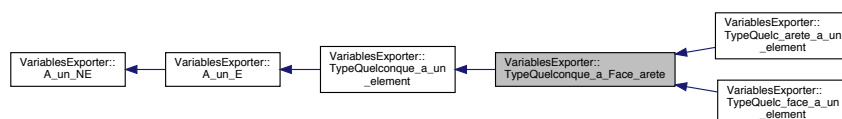
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque.cc

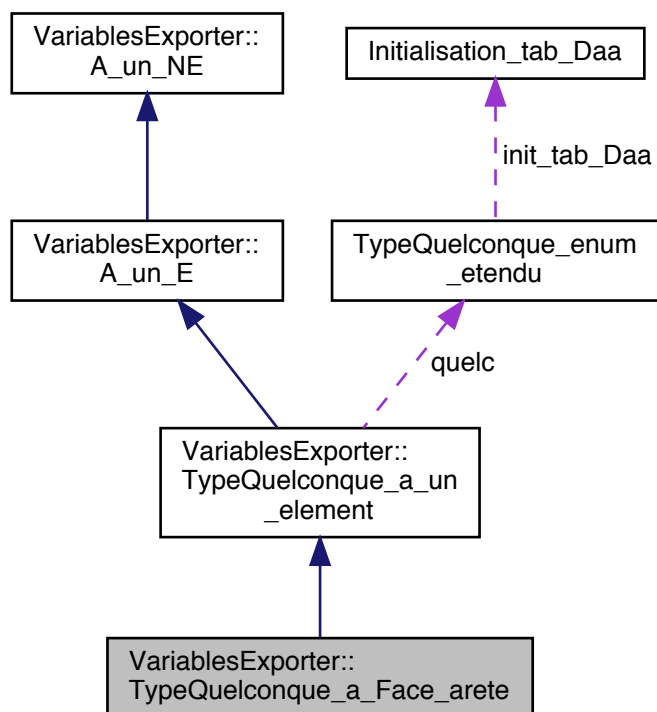
## 6.906 Référence de la classe

### VariablesExporter::TypeQuelconque\_a\_Face\_arete

Graphe d'héritage de VariablesExporter::TypeQuelconque\_a\_Face\_arete:



Graphe de collaboration de VariablesExporter::TypeQuelconque\_a\_Face\_arete:



## Fonctions membres publiques

- **TypeQuelconque\_a\_Face\_arete** (int absolu\_, [TypeQuelconque\\_enum\\_etendu](#) e, string ref\_element, string nom\_mail\_, int nbFA, [Enum\\_dure](#) tps, string nom\_var\_, int nbpti, int num\_ord)
- **TypeQuelconque\_a\_Face\_arete** (const [TypeQuelconque\\_a\\_Face\\_arete](#) &a)
- **TypeQuelconque\_a\_Face\_arete** & **operator=** (const [TypeQuelconque\\_a\\_Face\\_arete](#) &a)
- bool **operator==** (const [TypeQuelconque\\_a\\_Face\\_arete](#) &a) const
- bool **operator!=** (const [TypeQuelconque\\_a\\_Face\\_arete](#) &a) const
- bool **operator<** (const [TypeQuelconque\\_a\\_Face\\_arete](#) &a) const
- bool **operator<=** (const [TypeQuelconque\\_a\\_Face\\_arete](#) &a) const
- bool **operator>** (const [TypeQuelconque\\_a\\_Face\\_arete](#) &a) const
- bool **operator>=** (const [TypeQuelconque\\_a\\_Face\\_arete](#) &a) const
- const [TypeQuelconque\\_a\\_Face\\_arete](#) \* **PointClass\_QuelcFA\_const** () const
- [TypeQuelconque\\_a\\_Face\\_arete](#) \* **PointClass\_QuelcFA** ()
- const int & **Num\_FA\_const** () const
- int & **Num\_FA** ()

## Attributs protégés

- int num\_FA

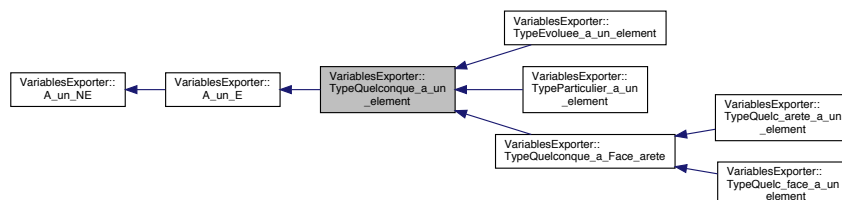
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

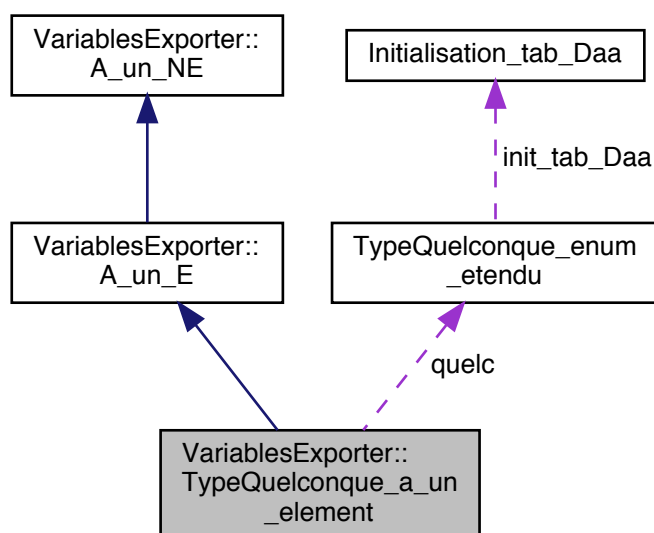
## 6.907 Référence de la classe

### VariablesExporter::TypeQuelconque\_a\_un\_element

Graphe d'héritage de VariablesExporter::TypeQuelconque\_a\_un\_element:



Graphe de collaboration de VariablesExporter::TypeQuelconque\_a\_un\_element:



## Fonctions membres publiques

- `TypeQuelconque_a_un_element` (int absolu\_, `TypeQuelconque_enum_etendu` e, string ref\_NE, string nom\_mail\_, `Enum_dure` tps, string nom\_var\_, int nbpti, int num\_ord)
- `TypeQuelconque_a_un_element` (const `TypeQuelconque_a_un_element` &a)
- `TypeQuelconque_a_un_element` & `operator=` (const `TypeQuelconque_a_un_element` &a)
- bool `operator==` (const `TypeQuelconque_a_un_element` &a) const
- bool `operator!=` (const `TypeQuelconque_a_un_element` &a) const
- bool `operator<` (const `TypeQuelconque_a_un_element` &a) const
- bool `operator<=` (const `TypeQuelconque_a_un_element` &a) const
- bool `operator>` (const `TypeQuelconque_a_un_element` &a) const
- bool `operator>=` (const `TypeQuelconque_a_un_element` &a) const
- void `Affiche` ()
- const `TypeQuelconque_a_un_element` \* `PointeurClass_Quelc_const` () const
- `TypeQuelconque_a_un_element` \* `PointeurClass_Quelc` ()
- const `TypeQuelconque_enum_etendu` & `Quelc_const` () const

- [TypeQuelconque\\_enum\\_etendu](#) & **Quelc** ()
- const int & **Num\_ordre\_const** () const
- int & **Num\_ordre** ()

### Attributs protégés

- [TypeQuelconque\\_enum\\_etendu](#) **quelc**
- int **num\_ordre**

### Amis

- istream & **operator**>> (istream &, [TypeQuelconque\\_a\\_un\\_element](#) &)
- ostream & **operator**<< (ostream &, const [TypeQuelconque\\_a\\_un\\_element](#) &)

La documentation de cette classe a été générée à partir du fichier suivant :

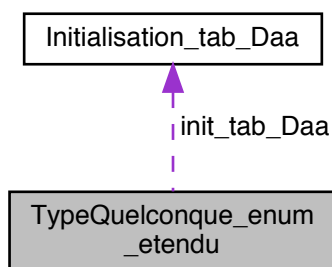
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc

## 6.908 Référence de la classe TypeQuelconque\_enum\_etendu

[TypeQuelconque\\_enum\\_etendu](#): la classe qui gère les types quelconques étendus.

```
#include <TypeQuelconque_enum_etendu.h>
```

Grphe de collaboration de [TypeQuelconque\\_enum\\_etendu](#):



### Fonctions membres publiques

- [TypeQuelconque\\_enum\\_etendu](#) ([EnumTypeQuelconque](#) en=RIEN\_TYPEQUELCONQUE, string no="-")
- [TypeQuelconque\\_enum\\_etendu](#) (string no)
- [TypeQuelconque\\_enum\\_etendu](#) (char \*no)
- [TypeQuelconque\\_enum\\_etendu](#) (const [TypeQuelconque\\_enum\\_etendu](#) &a)
- [TypeQuelconque\\_enum\\_etendu](#) & **operator=** (const [TypeQuelconque\\_enum\\_etendu](#) &a)
- bool **operator==** (const [TypeQuelconque\\_enum\\_etendu](#) &a) const
- bool **operator!=** (const [TypeQuelconque\\_enum\\_etendu](#) &a) const
- bool **operator>** (const [TypeQuelconque\\_enum\\_etendu](#) &a) const
- bool **operator>=** (const [TypeQuelconque\\_enum\\_etendu](#) &a) const
- bool **operator<** (const [TypeQuelconque\\_enum\\_etendu](#) &a) const
- bool **operator<=** (const [TypeQuelconque\\_enum\\_etendu](#) &a) const
- [EnumTypeQuelconque](#) **EnumTQ** () const
- string **Nom** () const
- string **NomPlein** () const
- string **NomGenerique** () const
- int **Position** () const
- bool **Nom\_vide** () const
- [EnumTypeGrandeur](#) **TypeDeGrandeurTG** () const

## Fonctions membres publiques statiques

- static [TypeQuelconque\\_enum\\_etendu](#) **Ajout\_un\_TypeQuelconque\_enum\_etendu** ([EnumTypeQuelconque](#) enu, const string no, [EnumTypeGrandeur](#) type\_grandeur)
- static bool **VerifExistence** (string no)
- static [TypeQuelconque\\_enum\\_etendu](#) **RecupTypeQuelconque\_enum\_etendu** (string to)
- static [List\\_io](#)< [TypeQuelconque\\_enum\\_etendu](#) > **TransfoList\_io** (const [List\\_io](#)< [EnumTypeQuelconque](#) > &li)
- static [Tableau](#)< [TypeQuelconque\\_enum\\_etendu](#) > **TransfoTableau** (const [Tableau](#)< [EnumTypeQuelconque](#) > &tab)
- static bool **Existe\_dans\_la\_liste** (const [List\\_io](#)< [TypeQuelconque\\_enum\\_etendu](#) > &lis, const [TypeQuelconque\\_enum\\_etendu](#) &dd)
- static int **NBmax\_TypeQuelconque\_enum\_etendu** ()
- static void **Lecture\_element\_distinctes** (istream &ent, string &no, [EnumTypeQuelconque](#) &enu, [EnumTypeGrandeur](#) &type\_grandeur)
- static [TypeQuelconque\\_enum\\_etendu](#) **Lecture\_avec\_creation\_eventuelle** (istream &ent)

## Attributs protégés

- string **nom**
- [EnumTypeQuelconque](#) **enu**
- int **posi\_nom**
- [EnumTypeGrandeur](#) **type\_grandeur**

## Attributs protégés statiques

- static [Tableau](#)< [TypeQuelconque\\_enum\\_etendu](#) > **tab\_Daa**
- static [Initialisation\\_tab\\_Daa](#) **init\_tab\_Daa**
- static int **tailTab** = 0

## Amis

- class **Initialisation\_tab\_Daa**
- istream & **operator**>> (istream &ent, [TypeQuelconque\\_enum\\_etendu](#) &a)
- ostream & **operator**<< (ostream &sort, const [TypeQuelconque\\_enum\\_etendu](#) &a)

### 6.908.1 Description détaillée

[TypeQuelconque\\_enum\\_etendu](#): la classe qui gère les types quelconques étendus.

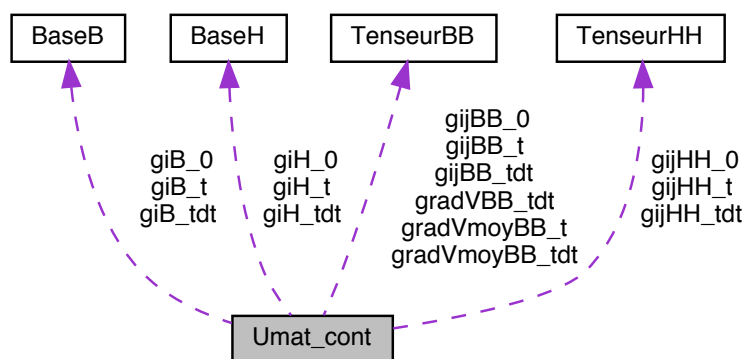
La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque\_enum\_etendu.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/TypeQuelconque\_enum\_etendu.cc



## 6.909 Référence de la classe Umat\_cont

Grphe de collaboration de Umat\_cont:



### Fonctions membres publiques

- **Umat\_cont** (`BaseB *ggiB_0`, `BaseH *ggiH_0`, `BaseB *ggiB_t`, `BaseH *ggiH_t`, `BaseB *ggiB_tdt`, `BaseH *ggiH_tdt`, `TenseurBB *ggijBB_0`, `TenseurHH *ggijHH_0`, `TenseurBB *ggijBB_t`, `TenseurHH *ggijHH_t`, `TenseurBB *ggijBB_tdt`, `TenseurHH *ggijHH_tdt`, `TenseurBB *ggradVmoyBB_t`, `TenseurBB *ggradVmoyBB_tdt`, `TenseurBB *ggradVBB_tdt`, `double *gjacobien_tdt`, `double *gjacobien_t`, `double *gjacobien_0`)
- **Umat\_cont** (`const Umat_cont &ex`)
- **Umat\_cont &operator=** (`const Umat_cont &ex`)
- **void Mise\_a\_jour\_grandeur** (`BaseB *ggiB_0`, `BaseH *ggiH_0`, `BaseB *ggiB_t`, `BaseH *ggiH_t`, `BaseB *ggiB_tdt`, `BaseH *ggiH_tdt`, `TenseurBB *ggijBB_0`, `TenseurHH *ggijHH_0`, `TenseurBB *ggijBB_t`, `TenseurHH *ggijHH_t`, `TenseurBB *ggijBB_tdt`, `TenseurHH *ggijHH_tdt`, `TenseurBB *ggradVmoyBB_t`, `TenseurBB *ggradVmoyBB_tdt`, `TenseurBB *ggradVBB_tdt`, `double *gjacobien_tdt`, `double *gjacobien_t`, `double *gjacobien_0`)
- **void Passage\_de\_Ordre2\_vers\_Ordre3** (`const Umat_cont &ex`, `bool plusZero`, `int type_recopie`)
- **void Passage\_de\_Ordre3\_vers\_Ordre2** (`const Umat_cont &ex`, `int type_recopie`)
- **void Passage\_de\_Ordre1\_vers\_Ordre3** (`const Umat_cont &ex`, `bool plusZero`, `int type_recopie`)
- **void Passage\_de\_Ordre3\_vers\_Ordre1** (`const Umat_cont &ex`, `int type_recopie`)
- **void Passage\_de\_Ordre1\_vers\_Ordre2** (`const Umat_cont &ex`, `bool plusZero`, `int type_recopie`)
- **void Passage\_de\_Ordre2\_vers\_Ordre1** (`const Umat_cont &ex`, `int type_recopie`)

### Attributs publics

- `BaseB * giB_0`
- `BaseH * giH_0`
- `BaseB * giB_t`
- `BaseH * giH_t`
- `BaseB * giB_tdt`
- `BaseH * giH_tdt`
- `TenseurBB * gjjBB_0`
- `TenseurHH * gjjHH_0`
- `TenseurBB * gjjBB_t`
- `TenseurHH * gjjHH_t`
- `TenseurBB * gjjBB_tdt`
- `TenseurHH * gjjHH_tdt`
- `TenseurBB * gradVmoyBB_t`
- `TenseurBB * gradVmoyBB_tdt`
- `TenseurBB * gradVBB_tdt`
- `double * jacobien_tdt`

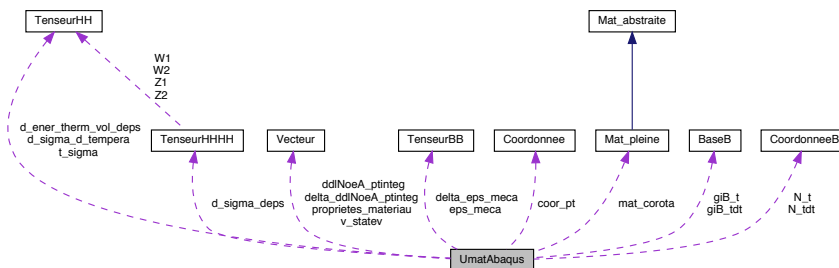
- double \* **jacobien\_t**
- double \* **jacobien\_0**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Met\_↔ abstraite\_struc\_donnees.h

## 6.910 Référence de la classe UmatAbaqus

Graphe de collaboration de UmatAbaqus:



### Fonctions membres publiques

- **UmatAbaqus** (int dimension\_espace=3, int nb\_vecteur=3)
- **UmatAbaqus** (const **UmatAbaqus** &a)
- **UmatAbaqus** & **operator=** (**UmatAbaqus** &a)
- void **LectureDonneesUmat** (bool utilisation\_umat\_interne, const int niveau)
- void **EcritureDonneesUmat** (bool utilisation\_umat\_interne, const int niveau)
- void **EcritureDonneesPourUmat** (bool utilisation\_umat\_interne, const int niveau)
- void **LectureResultatUmat** (bool utilisation\_umat\_interne, const int niveau)
- void **Change\_nom\_pipe\_envoi** (const string &env)
- void **Change\_nom\_pipe\_reception** (const string &recep)
- void **Init\_zero** ()
- void **Init\_un** (int dim\_reel\_tens=3, int nb\_reel\_tau\_ij=3)
- void **Transfert\_Umat\_ptInteg\_t** (**PtIntegMecalInterne** &ptIntegMeca)
- void **Transfert\_Umat\_ptInteg\_tdt** (**PtIntegMecalInterne** &ptIntegMeca)
- void **Affiche** (ofstream &sort, const int niveau)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

### Attributs publics

- **TenseurHH** \* **t\_sigma**
- **Vecteur** **v\_statev**
- **TenseurHHH** \* **d\_sigma\_deps**
- double **energie\_elastique**
- double **dissipation\_plastique**
- double **dissipation\_visqueuse**
- double **ener\_therm\_vol\_par\_utemps**
- **TenseurHH** \* **d\_sigma\_d\_tempera**
- **TenseurHH** \* **d\_ener\_therm\_vol\_deps**
- double **d\_ener\_therm\_dtemper**
- double **pnewdt**
- **TenseurBB** \* **eps\_meca**
- **TenseurBB** \* **delta\_eps\_meca**
- double **temps\_t**
- double **temps\_tdt**
- double **delta\_t**

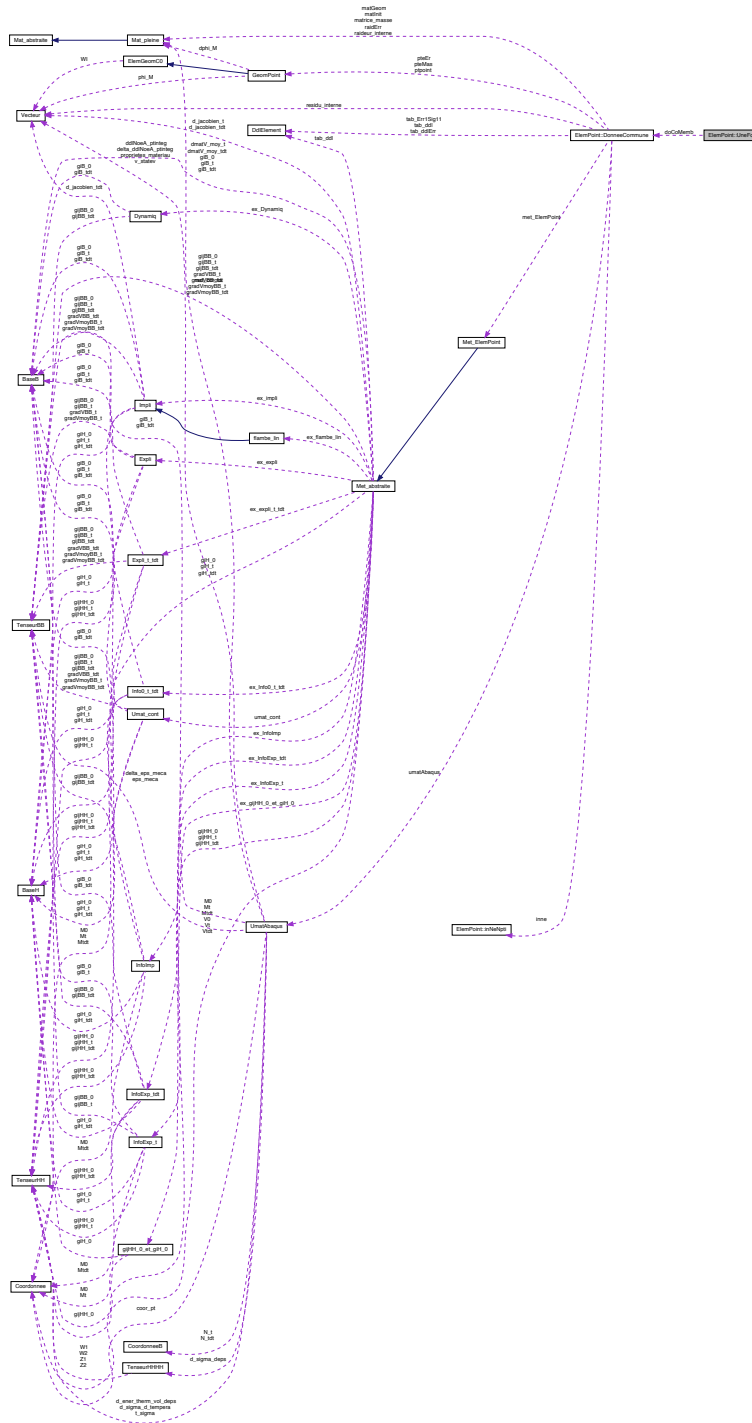
- double **temper\_t**
- double **delta\_temper**
- [Vecteur](#) **ddlNoeA\_ptinteg**
- [Vecteur](#) **delta\_ddlNoeA\_ptinteg**
- string **nom\_materiau**
- int **dim\_tens**
- int **nb\_tau\_ij**
- int **nb\_ij\_tenseur**
- int **nb\_variables\_etat**
- [Vecteur](#) **proprietes\_materiau**
- int **nb\_proprietes\_materiau**
- [Coordonnee](#) **coor\_pt**
- [Mat\\_pleine](#) **mat\_corota**
- double **longueur\_caracteristique**
- [BaseB](#) **giB\_t**
- [BaseB](#) **giB\_tdt**
- int **nb\_elem**
- int **nb\_pt\_integ**
- int **nb\_du\_plis**
- int **nb\_pt\_dans\_le\_plis**
- int **nb\_step**
- int **nb\_increment**
- [CoordonneeB](#) **N\_t**
- [CoordonneeB](#) **N\_tdt**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/UmatAbaqus.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/UmatAbaqus.cc

### 6.911 Référence de la classe ElemPoint::UneFois

Graphe de collaboration de ElemPoint::UneFois:



#### Attributs publics

- `DonneeCommune` \* `doCoMemb`
- `int CalResPrem_t`
- `int CalResPrem_tdt`
- `int CalImpPrem`
- `int dualSortbiel`

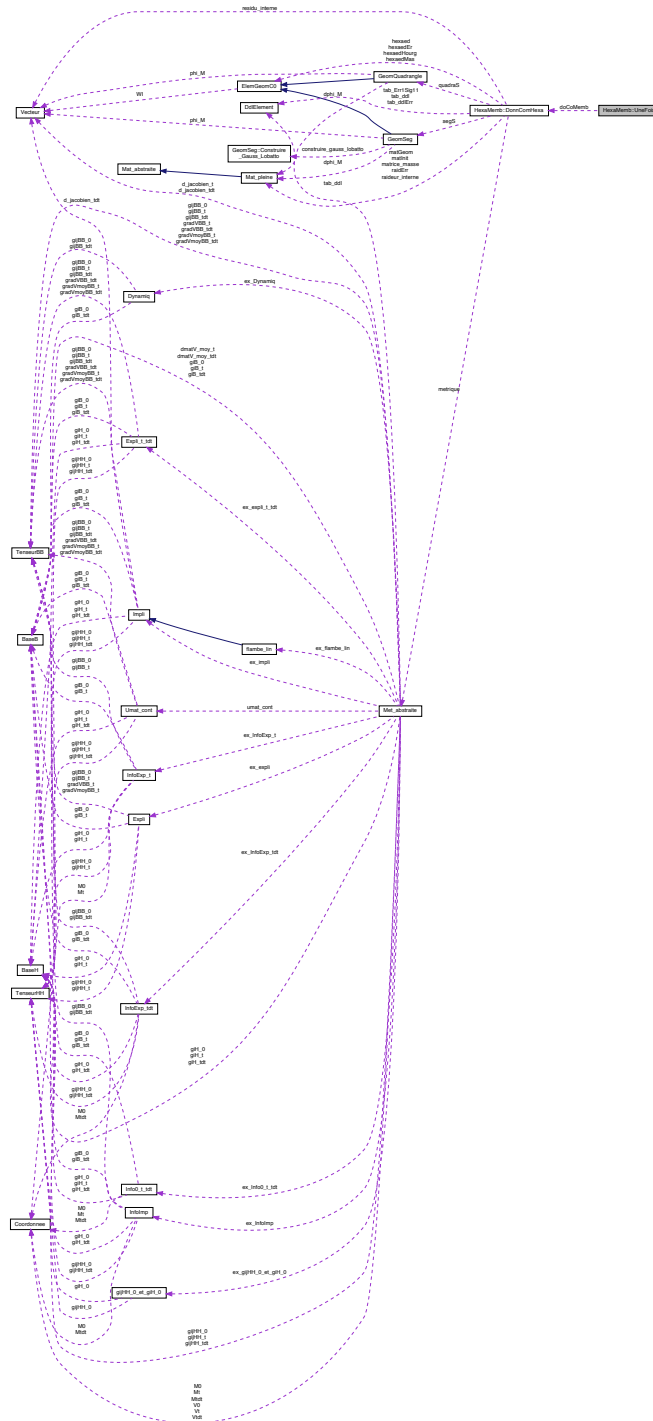
- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/ElemPoint/ElemPoint.cc

## 6.912 Référence de la classe HexaMemb::UneFois

Graphe de collaboration de HexaMemb::UneFois:



### Attributs publics

- HexaMemb::DonnComHexa \* doCoMemb
- int CalResPrem\_t
- int CalResPrem\_tdt
- int CalimpPrem
- int dualSortHexa

- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- int **CalSMsurf\_t**
- int **CalSMsurf\_tdt**
- int **CalSMRsurf**
- int **CalSMvol\_t**
- int **CalSMvol\_tdt**
- int **CalSMvol**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Hexaedre/HexaMemb.cc





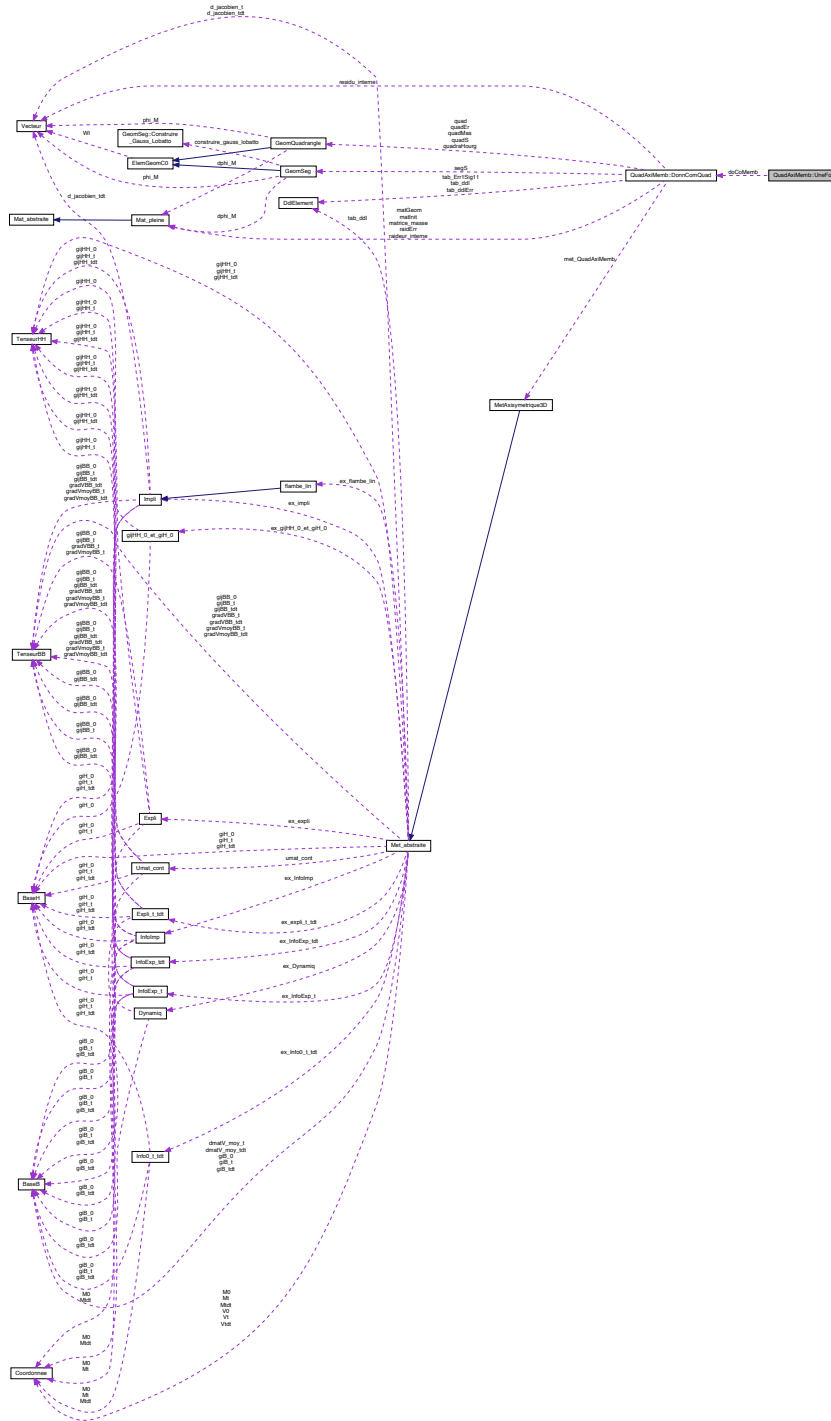
- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- [Tableau](#)< int > **CalSMsurf\_t**
- [Tableau](#)< int > **CalSMsurf\_tdt**
- [Tableau](#)< int > **CalSMRsurf**
- int **CalSMvol\_t**
- int **CalSMvol\_tdt**
- int **CalSMvol**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Pentaedre/PentaMemb.cc

## 6.914 Référence de la classe QuadAxiMemb::UneFois

Graphe de collaboration de QuadAxiMemb::UneFois:



### Attributs publics

- [QuadAxiMemb::DonnComQuad](#) \* [doCoMemb](#)
- int [CalResPrem\\_t](#)
- int [CalResPrem\\_tdt](#)
- int [CalimpPrem](#)
- int [dualSortQuad](#)

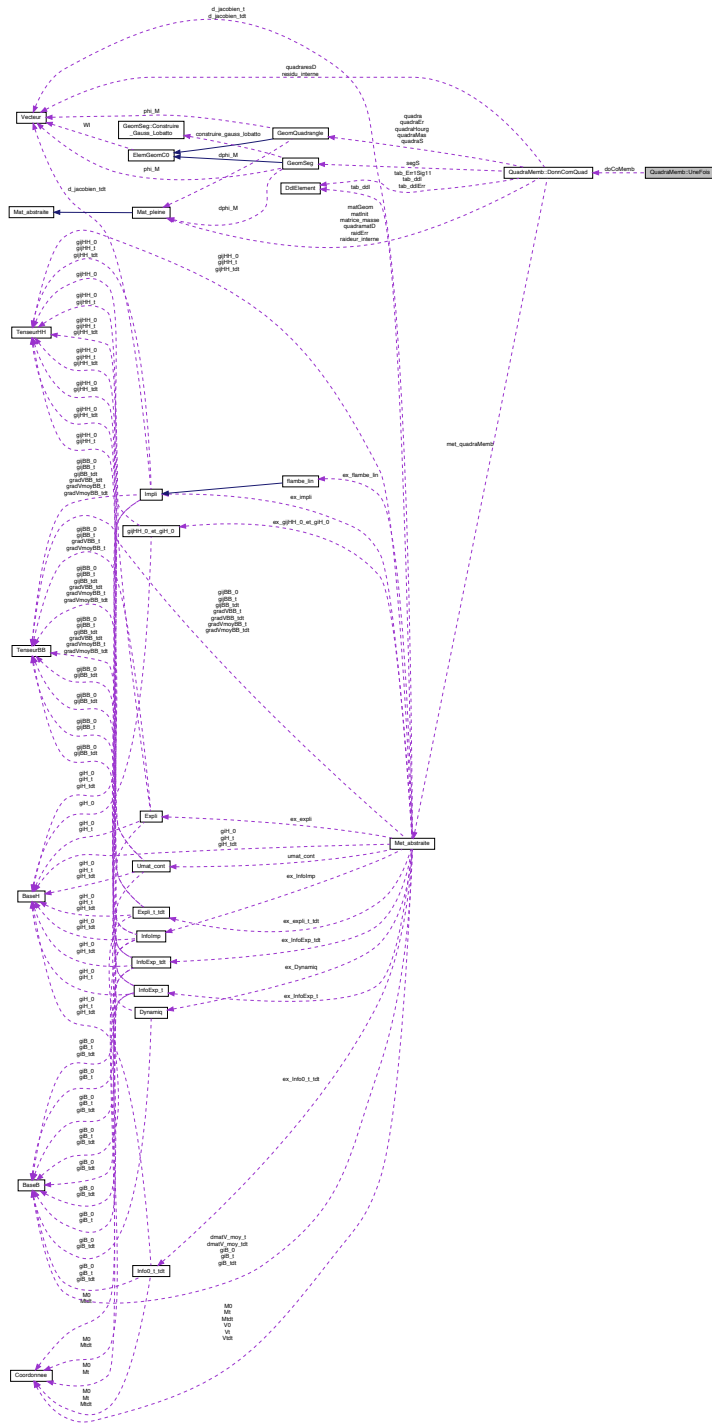
- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- int **CalSMsurf\_t**
- int **CalSMsurf\_tdt**
- int **CalSMRsurf**
- int **CalSMvol\_t**
- int **CalSMvol\_tdt**
- int **CalSMvol**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Quad\_asisymetrie/QuadAxiMemb.cc

## 6.915 Référence de la classe QuadraMemb::UneFois

Graphe de collaboration de QuadraMemb::UneFois:



### Attributs publics

- `QuadraMemb::DonnComQuad` \* `doCoMemb`
- `int CalResPrem_t`
- `int CalResPrem_tdt`
- `int CalImpPrem`
- `int dualSortQuad`

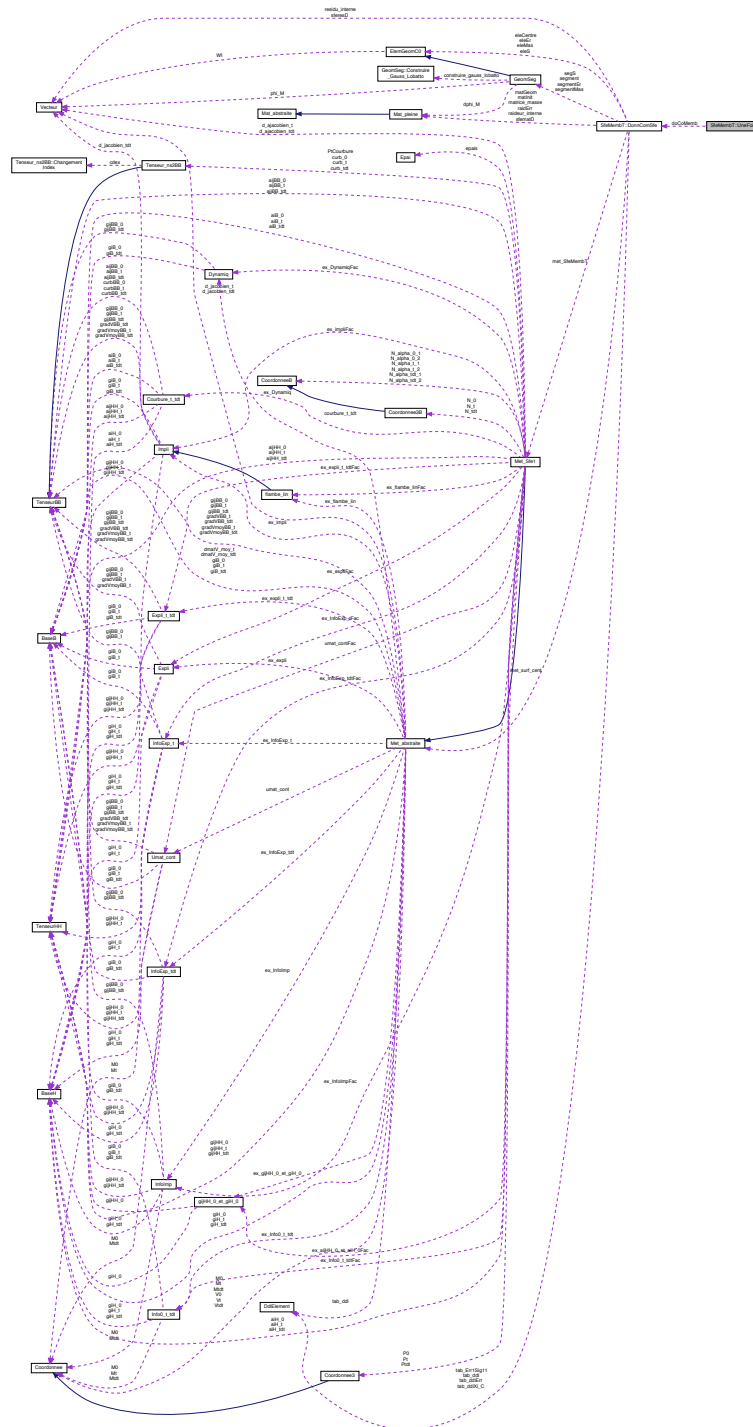
- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- int **CalSMsurf\_t**
- int **CalSMsurf\_tdt**
- int **CalSMRsurf**
- int **CalSMvol\_t**
- int **CalSMvol\_tdt**
- int **CalSMvol**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/quadrangle/QuadraMemb.cc

## 6.916 Référence de la classe SfeMembT::UneFois

Grphe de collaboration de SfeMembT::UneFois:



### Attributs publics

- SfeMembT::DonnComSfe \* doCoMemb
- int CalResPrem\_t
- int CalResPrem\_tdt
- int CalimpPrem
- int dualSortSfe

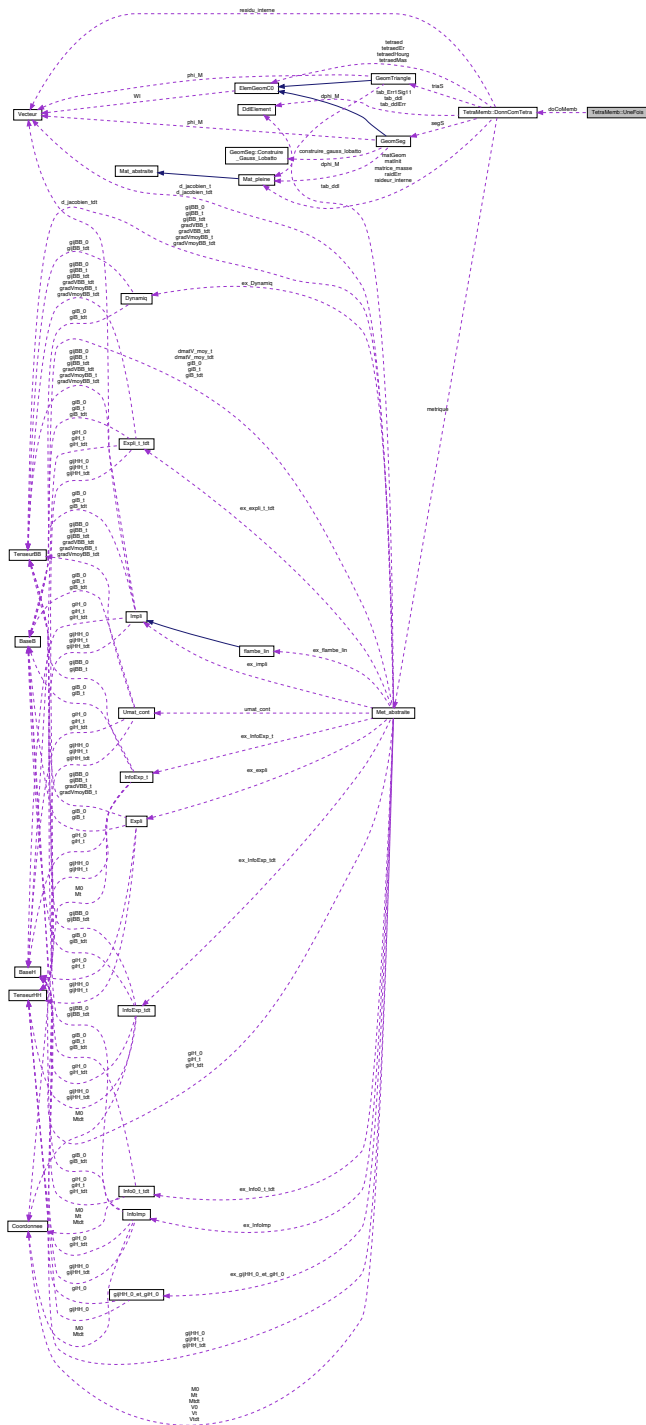
- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- int **CalSMsurf\_t**
- int **CalSMsurf\_tdt**
- int **CalSMRsurf**
- int **CalSMvol\_t**
- int **CalSMvol\_tdt**
- int **CalSMvol**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/SFE/SfeMembT.cc

### 6.917 Référence de la classe TetraMemb::UneFois

Grphe de collaboration de TetraMemb::UneFois:



#### Attributs publics

- `TetraMemb::DonnComTetra` \* `doCoMemb`
- `int CalResPrem_t`
- `int CalResPrem_tdt`
- `int CalimpPrem`
- `int dualSortTetra`



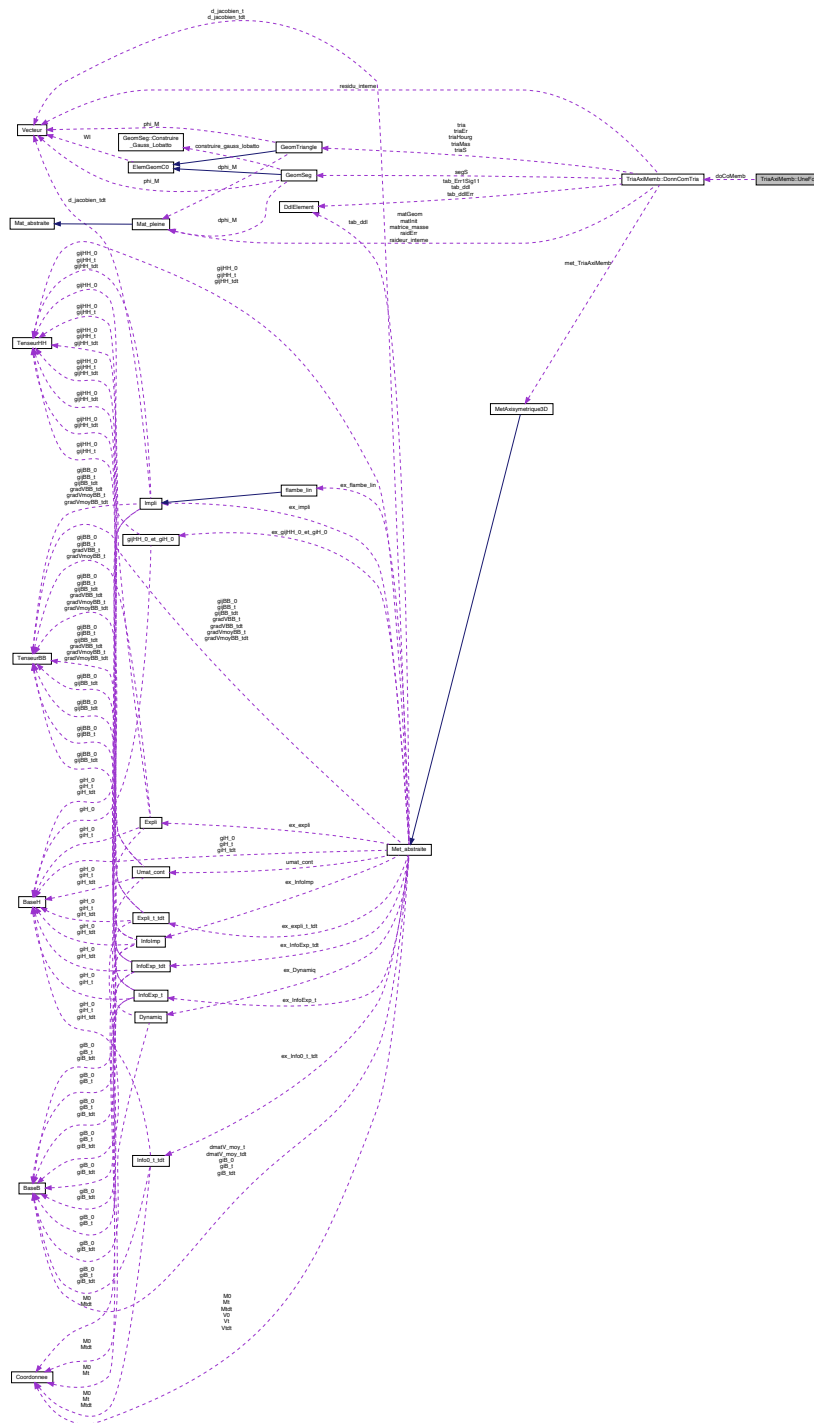
- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- int **CalSMsurf\_t**
- int **CalSMsurf\_tdt**
- int **CalSMRsurf**
- int **CalSMvol\_t**
- int **CalSMvol\_tdt**
- int **CalSMvol**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tetraedre/TetraMemb.cc

## 6.918 Référence de la classe TriaAxiMemb::UneFois

Graphe de collaboration de TriaAxiMemb::UneFois:



### Attributs publics

- [TriaAxiMemb::DonnComTri](#) \* doCoMemb
- int CalResPrem\_t
- int CalResPrem\_tdt
- int CalimpPrem
- int dualSortTria

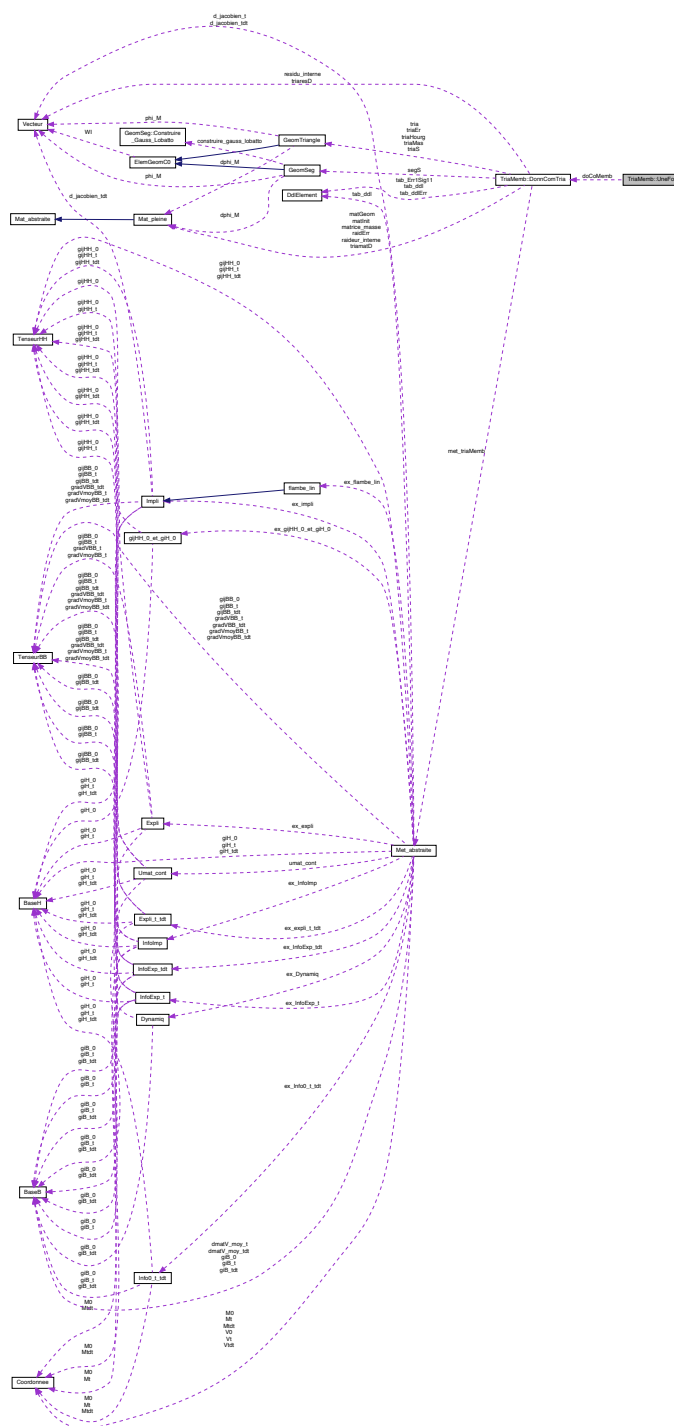
- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- int **CalSMsurf\_t**
- int **CalSMsurf\_tdt**
- int **CalSMRsurf**
- int **CalSMvol\_t**
- int **CalSMvol\_tdt**
- int **CalSMvol**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Tria\_axisymetrie/TriaAxiMemb.cc

## 6.919 Référence de la classe TriAMemb::UneFois

Grphe de collaboration de TriAMemb::UneFois:



### Attributs publics

- TriAMemb::DonnComTriA \* doCoMemb
- int CalResPrem\_t
- int CalResPrem\_tdt
- int CalimpPrem
- int dualSortTriA

- int **CalSMlin\_t**
- int **CalSMlin\_tdt**
- int **CalSMRlin**
- int **CalSMsurf\_t**
- int **CalSMsurf\_tdt**
- int **CalSMRsurf**
- int **CalSMvol\_t**
- int **CalSMvol\_tdt**
- int **CalSMvol**
- int **CalDynamique**
- int **CalPt\_0\_t\_tdt**
- int **nbelem\_in\_Prog**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Triangle/TriaMemb.cc

## 6.920 Référence de la classe Util

**Util**: divers utilitaires sur vecteurs, coordonnées, matrices ...

```
#include <Util.h>
```

### Fonctions membres publiques statiques

- static **Coordonnee ProdVec\_coor** (const **Coordonnee** &v1, const **Coordonnee** &v2)  
*PRODUIT VECTORIEL DE DEUX VECTEURS EN COORDONNEES ASBOLUS on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les coordonnées static **Vecteur ProdVec**( const **Vecteur** & v1, const **Vecteur** & v2); idem en coordonnées absolues avec le type **Coordonnee**.*
- static **CoordonneeH ProdVec\_coorH** (const **CoordonneeH** &v1, const **CoordonneeH** &v2)  
*idem en coordonnées contravariantes avec le type **CoordonneeH***
- static **CoordonneeB ProdVec\_coorB** (const **CoordonneeB** &v1, const **CoordonneeB** &v2)  
*idem en coordonnées covariantes avec le type **CoordonneeB***
- static **Coordonnee ProdVec\_coorBN** (const **CoordonneeB** &v1, const **CoordonneeB** &v2)  
*idem en coordonnées covariantes avec le type **CoordonneeB**, et en retour un coordonnee normal*
- static double **DeterminantB** (const **CoordonneeB** &v1, const **CoordonneeB** &v2, const **CoordonneeB** &v3)  
*CALCUL DU DETERMINANT DE TROIS VECTEURS on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les coordonnées static double **Determinant**( const **Vecteur** & v1, const **Vecteur** & v2, const **Vecteur** & v3); idem en coordonnées covariantes.*
- static double **Determinant** (const **Coordonnee** &v1, const **Coordonnee** &v2, const **Coordonnee** &v3)  
*idem en coordonnées absolues avec le type **Coordonnee***
- static double **DeterminantB** (const **CoordonneeB** &v1, const **CoordonneeB** &v2)  
*CALCUL DU DETERMINANT DE DEUX VECTEURS on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les coordonnées static double **Determinant**( const **Vecteur** & v1, const **Vecteur** & v2); idem en coordonnées covariantes.*
- static double **Determinant** (const **Coordonnee** &v1, const **Coordonnee** &v2)  
*idem en coordonnées absolues avec le type **Coordonnee***
- static double **DeterminantB** (const **CoordonneeB** &v1)  
*CALCUL DU DETERMINANT DE UN VECTEUR on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les coordonnées static double **Determinant**( const **Vecteur** & v1); idem en coordonnées covariantes.*
- static double **Determinant** (const **Coordonnee** &v1)  
*idem en coordonnées absolues avec le type **Coordonnee***
- static **Vecteur VarUnVect** (const **Vecteur** &v, const **Vecteur** &Dv, double nor)  
*calcul de la variation d'un vecteur unitaire connaissant la variation du vecteur non norme v : le vecteur non norme, Dv : la variation de v, nor : la norme de v en retour : la variation de vecteur : le vecteur peut-être de dimension > 3!!!!!!!!!!!! très important: il doit s'agir de vecteur exprimé dans un repère orthonormé ceci est vrai quelque soit la variance affichée: car ici on ne prend pas en compte la variation d'une métrique associée à un repère non orthonormé ex: les variations des gi sont ok car en fait les gi représentent les coordonnées dans un repère absolu*
- static **Coordonnee VarUnVect\_coor** (const **Coordonnee** &v, const **Coordonnee** &Dv, double nor)  
*idem avec des coordonnées, donc dim <= 3*
- static **CoordonneeB VarUnVect\_coorB** (const **CoordonneeB** &v, const **CoordonneeB** &Dv, double nor)

- *idem avec des coordonnéesB, donc dim <= 3*
- static **CoordonneeH VarUnVect\_coorH** (const **CoordonneeH** &v, const **CoordonneeH** &Dv, double nor)
- *idem avec des coordonnéesH, donc dim <= 3*
- static **Tableau< Vecteur > VarUnVect** (const **Vecteur** &v, const **Tableau< Vecteur >** &Dv, double nor)
- *calcul du tableau de variation d'un vecteur unitaire connaissant le tableau de variation du vecteur non norme v : le vecteur non norme, Dv : la variation de v, nor : la norme de v en retour : le tableau de variation*
- static **Tableau< Coordonnee > VarUnVect\_coor** (const **Coordonnee** &v, const **Tableau< Coordonnee >** &Dv, double nor)
- static **Tableau< CoordonneeB > VarUnVect\_coorB** (const **CoordonneeB** &v, const **Tableau< CoordonneeB >** &Dv, double nor)
- static **Tableau< Vecteur > & VarUnVect** (const **Vecteur** &v, const **Tableau< Vecteur >** &Dv, double nor, **Tableau< Vecteur >** &retour)
- *idem et le tableau de retour passé en paramètre*
- static **Tableau< Coordonnee > & VarUnVect\_coor** (const **Coordonnee** &v, const **Tableau< Coordonnee >** &Dv, double nor, **Tableau< Coordonnee >** &retour)
- static **Tableau< CoordonneeB > & VarUnVect\_coorB** (const **CoordonneeB** &v, const **Tableau< CoordonneeB >** &Dv, double nor, **Tableau< CoordonneeB >** &retour)
- static **Tableau< Coordonnee > & VarUnVect\_coorBN** (const **CoordonneeB** &v, const **Tableau< BaseB >** &Dv, double nor, **Tableau< Coordonnee >** &retour)
- *la variation du vecteur est supposé se trouver dans le premier vecteur de la base*
- static **Tableau< Coordonnee > VarProdVect\_coor** (const **Coordonnee** &v1, const **Coordonnee** &v2, const **Tableau< Coordonnee >** &Dv1, const **Tableau< Coordonnee >** &Dv2)
- *calcul de la variation d'un produit vectoriel vi et Dvi les vecteurs du produit vectoriel et leurs variations on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les coordonnées static Tableau< Vecteur> VarProdVect( const Vecteur & v1, const Vecteur & v2, const Tableau< Vecteur > & Dv1, const Tableau< Vecteur > & Dv2);*
- static **Tableau< Coordonnee > VarProdVect\_coor** (const **Coordonnee** &v1, const **Coordonnee** &v2, const **Tableau< BaseB >** &Dv)
- *on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les coordonnées // idem que le précédent module mais avec Dv qui est la référence des vecteurs Dv1 et Dv2 static Tableau< Vecteur> VarProdVect( const Vecteur & v1, const Vecteur & v2, const Tableau< BaseB > & Dv); idem que le précédent module mais avec un retour en coordonnée, et un tableau de BaseB*
- static **Tableau< CoordonneeB > VarProdVect\_coorB** (const **CoordonneeB** &v1, const **CoordonneeB** &v2, const **Tableau< BaseB >** &Dv)
- *idem que le précédent module mais avec in-out en coordonnéeB, et un tableau de BaseB*
- static **Tableau< Coordonnee > & VarProdVect\_coor** (const **Coordonnee** &v1, const **Coordonnee** &v2, const **Tableau< Coordonnee >** &Dv1, const **Tableau< Coordonnee >** &Dv2, **Tableau< Coordonnee >** &retour)
- *idem avec le tableau de retour passé en paramètre*
- static **Tableau< Coordonnee > & VarProdVect\_coor** (const **Coordonnee** &v1, const **Coordonnee** &v2, const **Tableau< BaseB >** &Dv, **Tableau< Coordonnee >** &retour)
- static **Tableau< CoordonneeB > & VarProdVect\_coorB** (const **CoordonneeB** &v1, const **CoordonneeB** &v2, const **Tableau< BaseB >** &Dv, **Tableau< CoordonneeB >** &retour)
- static **Tableau< Coordonnee > & VarProdVect\_coorBN** (const **CoordonneeB** &v1, const **CoordonneeB** &v2, const **Tableau< BaseB >** &Dv, **Tableau< Coordonnee >** &retour)
- static **Tableau< double > VarNorme** (const **Tableau< Coordonnee >** &Dv, const **Coordonnee** &V, const double &norme)
- *calcul de la variation de la norme d'un vecteur, connaissant la variation du vecteur, le vecteur, et sa norme, si la norme est trop petite on met à 0 la variation*
- static **Tableau< double > VarNorme** (const **Tableau< Vecteur >** &Dv, const **Vecteur** &V, const double &norme)
- *ici le vecteur peut-être de dimension quelconque > 3 par exemple*
- static int **Existe** (const **Tableau< Ddl >** &tab\_ddl, Enum\_ddl en)
- *retourne le numero du ddl recherche identifie par en, dans le tableau passé en paramètre s'il existe sinon 0*
- static double **ProduitMixte** (const **BaseB** &tab\_v)
- *calcul du produit mixte des vecteurs d'une base en coordonnées covariantes*
- static void **Inverse\_mat3x3** (listdouble9lter &i9lter)
- *calcul de l'inverse d'une matrice 3x3 donnée par ces coordonnées, en retour la matrice d'entrée est remplacée par la matrice inverse 1) cas d'une matrice non symétrique (quelconque), rangement des valeurs: (1,1); (1,2); (1,3); (2,1); (2,2); (2,3); (3,1); (3,2); (3,3)*
- static void **Inverse\_mat3x3** (listdouble6lter &i6lter)
- *2) cas d'une matrice symétrique, rangement des valeurs: (1,1); (2,2); (3,3); (2,1) = (1,2); (3,2) = (2,3); (3,1) = (1,3);*

- static void **Inverse\_mat2x2** (listdouble4Iter &i4Iter)  
calcul de l'inverse d'une matrice 2x2 donnée par ces coordonnées, en retour la matrice d'entrée est remplacée par la matrice inverse 1) cas d'une matrice non symétrique (quelconque), rangement des valeurs: (1,1); (2,2); (2,1); (1,2);
- static void **Inverse\_mat2x2** (listdouble3Iter &i3Iter)  
2) cas d'une matrice symétrique, rangement des valeurs: (1,1); (2,2); (2,1) = (1,2);

### 6.920.1 Description détaillée

**Util**: divers utilitaires sur vecteurs, coordonnées, matrices ...

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Util.h

## 6.921 Référence de la classe Deformation::UtilIndexDeformation

### Attributs publics

- [Tableau](#)< int > **iii**
- [Tableau](#)< int > **jjj**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation.↵  
h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Elements/Mecanique/Deformation\_gene/Deformation.↵  
cc

## 6.922 Référence de la classe UtilLecture

### Classes

- class [ErrNouvelEnreg](#)
- class [ErrNouvelEnregCVisu](#)
- class [ErrNouvelleDonnee](#)
- class [ErrNouvelleDonneeCVisu](#)
- class [PointFich](#)
- class [Position\\_BI](#)

### Fonctions membres publiques

- ofstream \* **Commande\_pointInfo** ()
- ofstream \* **SchemaXML** ()
- **UtilLecture** (int argc=0, const char \*argv[]=NULL)
- int **OuvrirFichier** ()
- void **Raffraichir\_pointInfo** ()
- void **fermeture\_pointInfo** ()
- void **fermetureSchemaXML** ()
- void **OuvrirCopie** ()
- void **FermerCopie** ()
- void **NouveauFichier** (char \*nevezFich)
- void **FermetureFichier** (int &indic)
- void **NouvelleDonnee** ()
- void **NouvelleDonneeSansInf** ()
- void **FlotDebutDonnee** ()
- void **ErreurLecture** ([UtilLecture::ErrNouvelleDonnee](#), char \*)
- void **MessageBuffer** (string)
- void **AfficheEnreg** () const
- char \* **RacineNom** ()
- int **Lec\_ent\_info** ()
- ifstream \* **Ent\_BI** ()
- ofstream \* **Sort\_BI** ()

- void **Ouverture\_base\_info** (string type\_entree="lecture")
- void **Fermeture\_base\_info** ()
- bool **Positionnement\_base\_info** (int inc\_voulu)
- bool **Increment\_suivant\_base\_info** (streampos &debut\_increment, int &inc\_lu)
- void **Enregistrement\_position\_increment\_base\_info** (int incr)
- list< int > **Liste\_increment** ()
- void **Ouverture\_fichier\_temps\_cpu** ()
- void **Fermeture\_fichier\_temps\_cpu** ()
- ofstream & **Sort\_temps\_cpu** ()
- void **Ouverture\_fichier\_principal\_vrml** ()
- void **Fermeture\_fichier\_principal\_vrml** ()
- ofstream & **Sort\_princ\_vrml** ()
- void **Ouverture\_fichier\_legende\_vrml** ()
- void **Fermeture\_fichier\_legende\_vrml** ()
- ofstream & **Sort\_legende\_vrml** ()
- bool **Existe\_princ\_vrml** () const
- void **Ouverture\_fichier\_principal\_maple** ()
- void **Fermeture\_fichier\_principal\_maple** ()
- ofstream & **Sort\_princ\_maple** ()
- bool **Existe\_princ\_maple** () const
- void **Ouverture\_fichier\_principal\_geomview** ()
- void **Fermeture\_fichier\_principal\_geomview** ()
- ofstream & **Sort\_princ\_geomview** ()
- void **Ouverture\_fichier\_legende\_geomview** ()
- void **Fermeture\_fichier\_legende\_geomview** ()
- ofstream & **Sort\_legende\_geomview** ()
- bool **Existe\_princ\_geomview** () const
- bool **Existe\_legende\_geomview** () const
- void **Ouverture\_fichier\_initial\_Gid** ()
- void **Fermeture\_fichier\_initial\_Gid** ()
- ofstream & **Sort\_initial\_Gid** ()
- void **Ouverture\_fichier\_resultat\_Gid** ()
- void **Fermeture\_fichier\_resultat\_Gid** ()
- ofstream & **Sort\_resultat\_Gid** ()
- bool **Existe\_initial\_Gid** () const
- bool **Existe\_resultat\_Gid** () const
- void **Ouverture\_fichier\_initial\_Gmsh** ()
- void **Fermeture\_fichier\_initial\_Gmsh** ()
- ofstream & **Sort\_initial\_Gmsh** ()
- void **CreationRepertoireResultat** ()
- void **Ouverture\_fichier\_resultat\_Gmsh** (const string &nom)
- void **Ouverture\_fichier\_resultat\_Gmsh** (const list< string > &list\_nom)
- const list< string > & **Noms\_base\_fichiers\_gmsh** () const
- void **Fermeture\_fichier\_resultat\_Gmsh** (const string &nom)
- void **Fermeture\_TousLesFichiersResultats\_Gmsh** ()
- ofstream & **Sort\_resultat\_Gmsh** (const string &nom)
- bool **Existe\_initial\_Gmsh** () const
- void **Ouverture\_CommandeVisu** (string type\_entree)
- void **Fermeture\_CommandeVisu** ()
- void **Changement\_NomCVisu** (string nouveauNomCvisu)
- ofstream \* **Sort\_CommandeVisu** ()
- void **NouvelleDonneeCVisu** ()
- bool **PositionEtExisteCVisu** (const string chaine)
- bool **Lecture\_un\_parametre\_int** (int val\_defaut, const string &nom\_class\_methode, int min, int max, string &mot\_cle, int &parametre) const
- bool **Lecture\_un\_parametre\_double** (double val\_defaut, const string &nom\_class\_methode, double min, double max, const string &mot\_cle, double &parametre) const
- bool **Lecture\_et\_verif\_mot\_cle** (const string &nom\_class\_methode, const string &mot\_cle) const
- bool **Lecture\_mot\_cle\_et\_string** (const string &nom\_class\_methode, const string &mot\_cle, string &nom) const
- double **lect\_avec\_const\_double\_utilisateur** (string erreur) const

## Fonctions membres publiques statiques

- static void **Modif\_interactive\_constante\_utilisateur** ()



### Attributs publics

- char \* **tablcar**
- const int **longueur**
- istream \* **entree**
- ostream \* **sortie**
- ofstream \* **copie**
- char \* **tablcarCVisu**
- istream \* **entCVisu**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/nouvelle\_enreg.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilLecture.cc
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/utilLecture2.cc

## 6.923 Référence de la classe UtilXML

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Lecture/UtilXML.h

## 6.924 Référence de la classe Courbe1D::Valbool

conteneur public pour une valeur et un booléen pour le strictement inclus

```
#include <Courbe1D.h>
```

### Attributs publics

- double **valeur**
- bool **dedans**

#### 6.924.1 Description détaillée

conteneur public pour une valeur et un booléen pour le strictement inclus

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe1D.h

## 6.925 Référence de la classe Courbe1D::ValDer

conteneur public pour une valeur et une dérivée

```
#include <Courbe1D.h>
```

### Attributs publics

- double **valeur**
- double **derivee**

#### 6.925.1 Description détaillée

conteneur public pour une valeur et une dérivée

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe1D.h

## 6.926 Référence de la classe Courbe1D::ValDer2

conteneur public pour une valeur, une dérivée première et une dérivée seconde

```
#include <Courbe1D.h>
```

## Attributs publics

- double **valeur**
- double **derivee**
- double **der\_sec**

### 6.926.1 Description détaillée

conteneur public pour une valeur, une dérivée première et une dérivée seconde

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe1D.h

## 6.927 Référence de la classe Courbe1D::ValDerbool

conteneur public pour une valeur et une dérivée et un booléen pour le strictement inclus

```
#include <Courbe1D.h>
```

## Attributs publics

- double **valeur**
- double **derivee**
- bool **dedans**

### 6.927.1 Description détaillée

conteneur public pour une valeur et une dérivée et un booléen pour le strictement inclus

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Courbes/Courbe1D.h

## 6.928 Référence de la classe VariablesExporter

### Classes

- class [A\\_un\\_E](#)
- class [A\\_un\\_NE](#)
- class [Ddl\\_a\\_un\\_element](#)
- class [Ddl\\_a\\_un\\_noeud](#)
- class [Ddl\\_etendu\\_a\\_un\\_noeud](#)
- class [Quelconque\\_a\\_un\\_noeud](#)
- class [TypeEvoluee\\_a\\_un\\_element](#)
- class [TypeParticulier\\_a\\_un\\_element](#)
- class [TypeQuelc\\_arete\\_a\\_un\\_element](#)
- class [TypeQuelc\\_face\\_a\\_un\\_element](#)
- class [TypeQuelc\\_Une\\_composante\\_Grandeur\\_globale](#)
- class [TypeQuelconque\\_a\\_Face\\_arete](#)
- class [TypeQuelconque\\_a\\_un\\_element](#)

## Fonctions membres publiques

- **VariablesExporter** (const [VariablesExporter](#) &var)
- void **LectureVariablesExporter** ([UtilLecture](#) \*entreePrinc)
- void **InsertConstVarUtilisateur\_dans\_globale** ()
- void **InitialisationConteneursQuelconques** ([LesMaillages](#) &lesMail, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob, const [LesReferences](#) &lesRef)
- const [List\\_io](#)< [Ddl\\_a\\_un\\_noeud](#) > & **List\_noeud\_type\_ddl** () const
- const [List\\_io](#)< [Ddl\\_etendu\\_a\\_un\\_noeud](#) > & **List\_noeud\_type\_ddlEtendu** () const
- const [List\\_io](#)< [Quelconque\\_a\\_un\\_noeud](#) > & **List\_noeud\_type\_quelconque** () const
- void **Info\_commande\_VariablesExporters** ([UtilLecture](#) \*entreePrinc)
- void **Affiche** () const

- [VariablesExporter](#) & **operator=** ([VariablesExporter](#) &var)
- bool **Complet\_VariablesExporter** (const [LesReferences](#) &lesRef)
- void **MiseAJourConstantesUtilisateur** ([UtilLecture](#) &lec)
- void **RenseigneVarUtilisateur** ([LesMaillages](#) &lesMail, const [LesReferences](#) &lesRef)
- void **Lecture\_base\_info** (ifstream &ent, const int cas)
- void **Ecriture\_base\_info** (ofstream &sort, const int cas)

### Fonctions membres publiques statiques

- static void **SchemaXML\_VariablesExporters** ([UtilLecture](#) \*entreePrinc, const [Enum\\_IO\\_XML](#) enu)

### Fonctions membres protégées

- void **LecConstantesUtilisateur** ([UtilLecture](#) &lec)
- void **InsertConstUtilisateur\_dans\_globale** ()
- void **Info\_commande\_ConstantesUtilisateur** ([UtilLecture](#) \*lec)
- void **LecVariablesUtilisateur** ([UtilLecture](#) &lec)
- void **Info\_commande\_VariablesUtilisateur** ([UtilLecture](#) \*lec)
- void **Insert\_VarUtilisateur\_dans\_globale** ()
- bool **VerifNomVariable** (const string &nom, bool avec\_sortie) const
- string **Lect\_interactive\_nom\_var** ()
- string **Lect\_interactive\_nom\_var\_relais** ()
- void **InsertCompGrandeurQuelc** (const string &non\_var, const [TypeQuelconque\\_enum\\_etendu](#) &enu)

### Attributs protégés

- list< [TypeQuelconque](#) > **li\_Q**
- list< string > **li\_nom**
- List\_io< [Ddl\\_a\\_un\\_noeud](#) > **list\_noeud\_type\_ddl**
- List\_io< [Ddl\\_etendu\\_a\\_un\\_noeud](#) > **list\_noeud\_type\_ddlEtendu**
- List\_io< [Quelconque\\_a\\_un\\_noeud](#) > **list\_noeud\_type\_quelconque**
- List\_io< [Ddl\\_a\\_un\\_element](#) > **list\_element\_type\_ddl**
- List\_io< [TypeParticulier\\_a\\_un\\_element](#) > **list\_element\_type\_particulier**
- List\_io< [TypeQuelconque](#) > **list\_quelc\_element\_type\_particulier**
- List\_io< [TypeEvoluee\\_a\\_un\\_element](#) > **list\_element\_type\_evoluee**
- List\_io< [TypeQuelconque](#) > **list\_quelc\_element\_type\_evoluee**
- List\_io< [TypeQuelc\\_face\\_a\\_un\\_element](#) > **list\_face\_element\_type\_quelc**
- List\_io< [TypeQuelconque](#) > **list\_quelc\_face\_element\_type\_quelc**
- List\_io< [TypeQuelc\\_arete\\_a\\_un\\_element](#) > **list\_arete\_element\_type\_quelc**
- List\_io< [TypeQuelconque](#) > **list\_quelc\_arete\_element\_type\_quelc**
- bool **initiaConteneurQuelconque**
- List\_io< [TypeQuelc\\_Une\\_composante\\_Grandeur\\_globale](#) > **list\_var\_glob\_sur\_grandeur\_globale**

### Amis

- class **LesMaillages**

La documentation de cette classe a été générée à partir du fichier suivant :

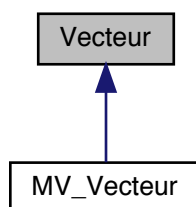
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.h](#)
- [/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Maillage/VariablesExporter.cc](#)

## 6.929 Référence de la classe Vecteur

La classe [Vecteur](#) permet de déclarer des vecteurs d'une longueur prédefinie par une allocation dynamique de mémoire. Les composantes d'un vecteur de cette classe sont de type double. Correspond à un vecteur absolu ( par opposition avec des vecteurs avec des coordonnées covariantes ou contravariantes.

```
#include <Vecteur.h>
```

Grapshe d'héritage de Vecteur:



## Fonctions membres publiques

- **Vecteur** (int n)
- **Vecteur** (int n, double val)
- **Vecteur** (int n, double \*vec)
- **Vecteur** (const [Coordonnee](#) &a)
- **Vecteur** (const [Vecteur](#) &vec)
- int **Taille** () const
- void **Affiche** () const
- void **Change\_taille** (int n)
- void **Libere** ()
- double **Max\_val\_abs** () const
- double **Max\_val\_abs** (int &i) const
- double **Norme** () const
- [Vecteur](#) & **Normer** ()
- double **Prod\_scal** (const [Vecteur](#) &vec) const
- double **Somme** () const
- [Vecteur](#) **operator+** (const [Vecteur](#) &vec) const
- [Vecteur](#) **operator-** (const [Vecteur](#) &vec) const
- [Vecteur](#) **operator-** () const
- void **operator+=** (const [Vecteur](#) &vec)
- void **operator-=** (const [Vecteur](#) &vec)
- void **operator\*=** (double val)
- int **operator==** (const [Vecteur](#) &vec) const
- int **operator!=** (const [Vecteur](#) &vec) const
- [Vecteur](#) & **operator=** (const [Vecteur](#) &vec)
- [Vecteur](#) & **operator=** (const [Coordonnee](#) &point)
- [Vecteur](#) **operator\*** (double val) const
- [Vecteur](#) **operator/** (const double val) const
- void **operator/=** (const double val)
- double **operator\*** (const [Vecteur](#) &vec) const
- double & **operator()** (int i)
- double **operator()** (int i) const
- double **operator!** () const
- void **Zero** ()
- **operator Coordonnee** ()
- [Vecteur](#) & **Vect** ()
- **Vecteur** ()
  - Constructeur par default.*
- **Vecteur** (int n)
  - Constructeur fonction de la taille du vecteur initialisation à zéro par défaut.*
- **Vecteur** (int n, double val)
  - Constructeur fonction de la taille du vecteur et d'une valeur d'initialisation pour les composantes.*
- **Vecteur** (int n, double \*vec)

- Constructeur fonction d'un tableau de composantes de type T aucun test n'est fait concernant la taille disponible de vec routine dangereuse !!!!*
- **Vecteur** (const **MV\_Vecteur**< double > &a)  
*constructeur fonction d'un **MV\_Vecteur** : oui car c'est un cas particulier (différent des coordonnées)*
  - **Vecteur** (const **Vecteur** &vec)  
*Constructeur de copie.*
  - **~Vecteur** ()  
*DESTRUCTEUR :*
  - int **Taille** () const  
*Retourne la taille du vecteur.*
  - void **Affiche** () const  
*Affichage des composantes du vecteur.*
  - void **Affiche** (int min\_i, int max\_i, int pas\_i, int nbdigit) const  
*Affichage des composantes du vecteur avec paramètres de contrôle.*
  - void **Affichage\_ecran** (string entete) const  
*affichage à l'écran après une demande interactive du vecteur entete: une chaine explicative, a afficher en entête*
  - void **Change\_taille** (int n)  
*Change la taille du vecteur (la nouvelle taille est n) si la nouvelle taille est inférieur, on garde les valeurs existantes du vecteur si la nouvelle taille est supérieur, on garde les valeurs existantes et les nouvelles valeurs sont mises à 0.*
  - void **Change\_taille** (int n, const double &val\_init)  
*Change la taille du vecteur avec initialisation de tous le nouveau vecteur.*
  - void **Libere** ()  
*Permet de desallouer l'ensemble des elements du vecteur.*
  - double **Max\_val\_abs** () const  
*Calcul du maximum en valeur absolu des composantes du vecteur.*
  - double **Max\_val\_abs** (int &i) const  
*Calcul du maximum en valeur absolu des composantes du vecteur ramene également l'indice de tableau du maximum.*
  - double **Max\_val\_abs\_signe** () const  
*Calcul du maximum en valeur absolu des composantes du vecteur mais ramène la grandeur signée (avec son signe)*
  - double **Max\_val\_abs\_signe** (int &i) const  
*Calcul du maximum en valeur absolu des composantes du vecteur mais ramène la grandeur signée (avec son signe) ramene également l'indice de tableau du maximum.*
  - **Coordonnee MinMaxMoy** (bool affiche) const  
*calcul, récupération et affichage éventuelle des mini, maxi, et en valeur absolue la moyenne des composantes du vecteur en retour: le min, le max et la moyenne en valeur absolue*
  - double **Norme** () const  
*Calcul de la norme euclidienne des composantes du vecteur.*
  - **Vecteur & Normer** ()  
*norme le vecteur*
  - double **Prod\_scal** (const **Vecteur** &vec) const  
*Calcul du produit scalaire entre deux vecteurs.*
  - double **Somme** () const  
*somme de tous les composantes du vecteur*
  - double **SommeAbs** () const  
*somme de tous les valeurs absolues des composantes du vecteur*
  - double **Moyenne** () const  
*moyenne de tous les composantes du vecteur*
  - **Vecteur operator+** (const **Vecteur** &vec) const  
*Surcharge de l'operateur + : addition entre deux vecteurs.*
  - **Vecteur operator-** (const **Vecteur** &vec) const  
*Surcharge de l'operateur - : soustraction entre deux vecteurs.*
  - **Vecteur operator-** () const  
*Surcharge de l'operateur - : oppose d'un vecteur.*
  - void **operator+=** (const **Vecteur** &vec)  
*Surcharge de l'operateur +=.*
  - void **operator-=** (const **Vecteur** &vec)  
*Surcharge de l'operateur -=.*
  - void **operator\*=(double val)**  
*Surcharge de l'operateur \*= : multiplication d'un scalaire par un vecteur.*

- int **operator==** (const [Vecteur](#) &vec) const  
*Surcharge de l'opérateur == : test d'égalité entre deux vecteurs.*
- int **operator!=** (const [Vecteur](#) &vec) const  
*Surcharge de l'opérateur != Renvoie 1 si les deux vecteurs ne sont pas égaux Renvoie 0 sinon.*
- [Vecteur](#) & **operator=** (const [Vecteur](#) &vec)  
*Surcharge de l'opérateur = : réalise l'égalité de deux vecteurs.*
- [Vecteur](#) & **operator=** (const [MV\\_Vector](#)< double > &A)  
*affectation avec un [MV\\_Vector](#)*
- [Vecteur](#) **operator\*** (double val) const  
*Surcharge de l'opérateur \* : multiplication d'un vecteur par un scalaire.*
- [Vecteur](#) **operator/** (const double val) const  
*Surcharge de l'opérateur / : division des composantes d'un vecteur par un scalaire.*
- void **operator/=** (const double val)  
*Surcharge de l'opérateur /= : division des composantes d'un vecteur par un scalaire.*
- double **operator\*** (const [Vecteur](#) &vec) const  
*Surcharge de l'opérateur \* : multiplication entre deux vecteurs Permet de réaliser le produit scalaire entre deux vecteurs.*
- double **operator()** (int i) const  
*Retourne la ième composante du vecteur : accès en lecture uniquement.*
- double & **operator()** (int i)  
*Retourne la ième composante du vecteur : accès en lecture et en écriture.*
- double **operator!** () const  
*Surcharge de l'opérateur ! : renvoie la norme d'un vecteur.*
- void **Zero** ()  
*mise à zéro d'un vecteur*
- void **Inita** (double val)  
*initialise toutes les composantes à val*
- [Vecteur](#) & **Prod\_ii** (const [Vecteur](#) &A, [Vecteur](#) &P) const  
*calcul le produit  $P(i) = (*this)(i) * A(i)$ ; ramène P qui est passé en paramètre*
- [Coordonnee](#) **Coordo** () const  
*— création explicite de coordonnées équivalentes*
- [CoordonneeH](#) **CoordoH** () const  
*— création explicite de coordonnées équivalentes*
- [CoordonneeB](#) **CoordoB** () const  
*— création explicite de coordonnées équivalentes*
- [Vecteur](#) & **Egale\_une\_partie\_de** (const [Vecteur](#) &vec, int indice)  
*affectation à une partie d'un vecteur qui doit avoir une taille donc plus grande  $(*this)(i) = vec(i+indice-1)$ ;  $i= 1$  à la taille de  $(*this)$  ramène  $*this$*
- [Vecteur](#) & **Une\_partie\_egale** (int indice, const [Vecteur](#) &vec)  
*affectation d'une partie de  $(*this)$  au vecteur passé en paramètre  $(*this)(i+indice-1) = vec(i)$ ;  $i= 1$  à la taille de vec ramène  $*this$*
- [Vecteur](#) & **Vect** ()  
*methode pour la compatibilité avec des classes dérivées*
- [MV\\_Vector](#)< double > **MV\_vecteur\_double** ()  
*ramène un vecteur [MV\\_Vector](#)<double> qui utilise la même place mémoire que this ce qui permet d'utiliser le type [MV\\_Vector](#)*
- const [MV\\_Vector](#)< double > **MV\_vecteur\_double** () const
- [MV\\_Vector](#)< double > \* **Nouveau\_MV\_Vector\_double** ()  
*spécifiquement à la classe [MV\\_Vector](#)<double> ramène un pointeur de nouveau vecteur [MV\\_Vector](#)<double> qui utilise la même place mémoire que this ce qui permet d'utiliser le type [MV\\_Vector](#)*
- const [MV\\_Vector](#)< double > \* **Nouveau\_MV\_Vector\_double\_const** () const  
*idem en const*

## Fonctions membres protégées

- double \* **Pointeur\_vect** ()

## Attributs protégés

- int **taille**
- double \* **v**

## Amis

- class `MatLapack`  
*utilisation directe du pointeur de double par la classe `MatLapack`*
- `Vecteur operator*` (double val, const `Vecteur` &vec)  
*Surcharge de l'opérateur \* : multiplication entre un scalaire et un vecteur.*
- `istream & operator>>` (istream &, `Vecteur` &)  
*surcharge de l'opérateur de lecture les informations sont le type puis la taille puis les datas*
- `ostream & operator<<` (ostream &, const `Vecteur` &)  
*surcharge de l'opérateur d'écriture non formatée les informations sont le type puis la taille puis les datas séparées par un espace*

### 6.929.1 Description détaillée

La classe `Vecteur` permet de déclarer des vecteurs d'une longueur prédefinie par une allocation dynamique de mémoire. Les composantes d'un vecteur de cette classe sont de type double. Correspond à un vecteur absolu ( par opposition avec des vecteurs avec des coordonnées covariantes ou contravariantes.

La documentation de cette classe a été générée à partir du fichier suivant :

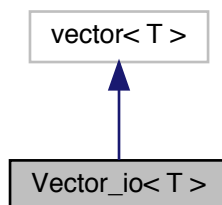
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Vecteurs/Vecteur.cpp`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Vecteurs/Vecteur.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Vecteurs/Vecteur2.cc`

## 6.930 Référence du modèle de la classe `Vector_io< T >`

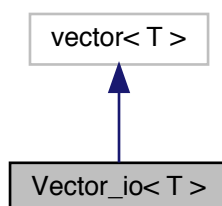
création de conteneurs type vector stl avec surcharge de lecture écriture

```
#include <Vector_io.h>
```

Graphe d'héritage de `Vector_io< T >`:



Graphe de collaboration de `Vector_io< T >`:



## Amis

- `istream & operator>>` (`istream &entree`, [Vector\\_io](#) &)
- `ostream & operator<<` (`ostream &sort`, const [Vector\\_io](#) &)

### 6.930.1 Description détaillée

```
template<class T>
class Vector_io< T >
```

création de conteneurs type vector stl avec surcharge de lecture écriture  
BUT: création d'un conteneur vector stl, qui comporte en plus une surcharge de lecture écriture.

#### Auteur

Gérard Rio

#### Version

1.0

#### Date

23/01/97

La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/TypeBase/Vector_io.h`

## 6.931 Référence de la classe VeurPropre

### Fonctions membres publiques

- `VeurPropre` (int n)
- `VeurPropre` (double val, [Vecteur](#) v)
- `VeurPropre` (const [VeurPropre](#) &a)
- `VeurPropre & operator=` (const [VeurPropre](#) &a)
- `double & Val` ()
- `Vecteur & Pv` ()
- `void Affiche` () const

## Amis

- `istream & operator>>` (`istream &`, [VeurPropre](#) &)
- `ostream & operator<<` (`ostream &`, const [VeurPropre](#) &)

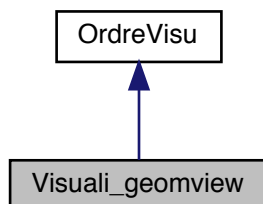
La documentation de cette classe a été générée à partir du fichier suivant :

- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Vecteurs/VeurPropre.h`
- `/Volumes/linux/calcul/partageMacLinux/Herezh++2009/tenseurs_mai99/Vecteurs/VeurPropre.cc`

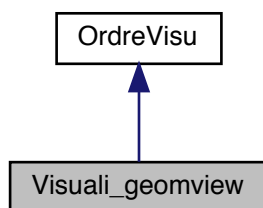


## 6.932 Référence de la classe Visuali\_geomview

Graphe d'héritage de Visuali\_geomview:



Graphe de collaboration de Visuali\_geomview:



### Fonctions membres publiques

- **Visuali\_geomview** (const [Visuali\\_geomview](#) &algo)
- void [ExeOrdre](#) (ParaGlob \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, Charge \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#)<::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)

### Membres hérités additionnels

#### 6.932.1 Documentation des fonctions membres

##### 6.932.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Visuali_geomview::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.932.1.2 ExeOrdre()

```
void Visuali_geomview::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
    const map< string, const double *, std::less< string > > & ,
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.932.1.3 Lecture\_parametres\_OrdreVisu()

```
void Visuali_geomview::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

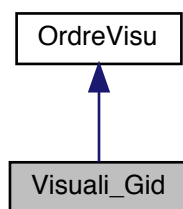
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

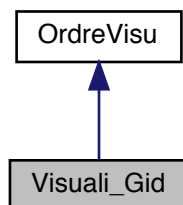
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Visuali\_geomview.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Visuali\_geomview.cc

## 6.933 Référence de la classe Visuali\_Gid

Graphe d'héritage de Visuali\_Gid:



Graphe de collaboration de Visuali\_Gid:



## Fonctions membres publiques

- **Visuali\_Gid** (const [Visuali\\_Gid](#) &algo)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#)::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

## Membres hérités additionnels

### 6.933.1 Documentation des fonctions membres

#### 6.933.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Visuali_Gid::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.933.1.2 ExeOrdre()

```
void Visuali_Gid::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
```

```
const map< string, const double *, std::less< string > & ,
const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.933.1.3 Lecture\_parametres\_OrdreVisu()

```
void Visuali_Gid::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

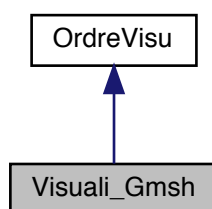
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

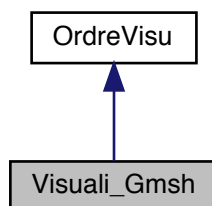
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Visuali\_Gid.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Visuali\_Gid.cc

## 6.934 Référence de la classe Visuali\_Gmsh

Grappe d'héritage de Visuali\_Gmsh:



Grappe de collaboration de Visuali\_Gmsh:



## Fonctions membres publiques

- **Visuali\_Gmsh** (const [Visuali\\_Gmsh](#) &algo)
- void [ExeOrdre](#) ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#) ↔ ::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)

## Membres hérités additionnels

### 6.934.1 Documentation des fonctions membres

#### 6.934.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Visuali_Gmsh::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.934.1.2 ExeOrdre()

```
void Visuali_Gmsh::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
    const map< string, const double *, std::less< string > > & ,
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.934.1.3 Lecture\_parametres\_OrdreVisu()

```
void Visuali_Gmsh::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

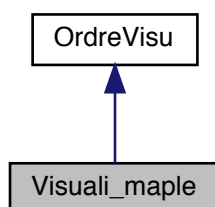
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

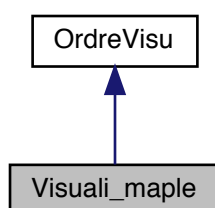
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Visuali\_Gmsh.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Visuali\_Gmsh.cc

## 6.935 Référence de la classe Visuali\_maple

Graphe d'héritage de Visuali\_maple:



Graphe de collaboration de Visuali\_maple:



### Fonctions membres publiques

- **Visuali\_maple** (const [Visuali\\_maple](#) &algo)
- void [ExeOrdre](#) ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#)<::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void [Lecture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)
- void [Ecriture\\_parametres\\_OrdreVisu](#) ([UtilLecture](#) &entreePrinc)

### Membres hérités additionnels

#### 6.935.1 Documentation des fonctions membres

##### 6.935.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Visuali_maple::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.935.1.2 ExeOrdre()

```
void Visuali_maple::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
    const map< string, const double *, std::less< string > > & ,
    const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.935.1.3 Lecture\_parametres\_OrdreVisu()

```
void Visuali_maple::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

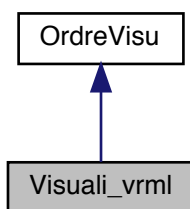
Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

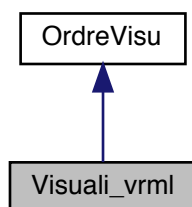
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Visuali\_maple.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Visuali\_maple.cc

## 6.936 Référence de la classe Visuali\_vrml

Graphe d'héritage de Visuali\_vrml:



Graphe de collaboration de Visuali\_vrml:



## Fonctions membres publiques

- **Visuali\_vrml** (const [Visuali\\_vrml](#) &algo)
- void **ExeOrdre** ([ParaGlob](#) \*, const [Tableau](#)< int > &, [LesMaillages](#) \*, bool, [LesReferences](#) \*, [LesLoisDeComp](#) \*, [DiversStockage](#) \*, [Charge](#) \*, [LesCondLim](#) \*, [LesContacts](#) \*, [Resultats](#) \*, [UtilLecture](#) &, [OrdreVisu](#)::EnumTypeIncre, int, bool, const map< string, const double \*, std::less< string > > &, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

## Membres hérités additionnels

### 6.936.1 Documentation des fonctions membres

#### 6.936.1.1 Ecriture\_parametres\_OrdreVisu()

```
void Visuali_vrml::Ecriture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

#### 6.936.1.2 ExeOrdre()

```
void Visuali_vrml::ExeOrdre (
    ParaGlob * ,
    const Tableau< int > & ,
    LesMaillages * ,
    bool ,
    LesReferences * ,
    LesLoisDeComp * ,
    DiversStockage * ,
    Charge * ,
    LesCondLim * ,
    LesContacts * ,
    Resultats * ,
    UtilLecture & ,
    OrdreVisu::EnumTypeIncre ,
    int ,
    bool ,
```



```
const map< string, const double *, std::less< string > > & ,
const List_io< TypeQuelconque > & listeVecGlob ) [inline], [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

### 6.936.1.3 Lecture\_parametres\_OrdreVisu()

```
void Visuali_vrml::Lecture_parametres_OrdreVisu (
    UtilLecture & entreePrinc ) [virtual]
```

Réimplémentée à partir de [OrdreVisu](#).

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Visuali\_vrml.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/VRML/Visuali\_vrml.cc

## 6.937 Référence de la classe Visualisation

### Fonctions membres publiques

- **Visualisation** ([UtilLecture](#) \*ent)
- int **OrdresPossible** ()
- void **List\_increment\_disponible** (list< int > &list\_incr)
- const list< int > & **List\_balaie\_init** (int interactif=1)
- void **List\_balaie\_init** (const list< int > &list\_init)
- const list< int > & **List\_balaie** ()
- void **List\_maillage\_disponible** (int nombre\_maillage\_dispo)
- void **Initialisation** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, [Charge](#) \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob, bool fil\_calcul)
- void **Visu** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, [Charge](#) \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **Contexte\_debut\_visualisation** ()
- void **Contexte\_fin\_visualisation** ()
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- bool **Visu\_vrml\_valide** ()
- void **Inactiv\_Visu\_vrml** ()
- void **EcritDebut\_fichier\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **EcritFin\_fichier\_OrdreVisu** ([UtilLecture](#) &entreePrinc)

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Visualisation.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Visualisation.cc

## 6.938 Référence de la classe Visualisation\_geomview

### Fonctions membres publiques

- **Visualisation\_geomview** ([UtilLecture](#) \*ent)
- int **OrdresPossible** ()
- void **List\_increment\_disponible** (list< int > &list\_incr)
- const list< int > & **List\_balaie\_init** ()
- void **List\_balaie\_init** (const list< int > &list\_init)
- const list< int > & **List\_balaie** ()
- void **List\_maillage\_disponible** (int nombre\_maillage\_dispo)

- void **Initialisation** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob, bool fil\_calcul)
- void **Visu** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **Contexte\_debut\_visualisation** ()
- void **Contexte\_fin\_visualisation** ()
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- bool **Visu\_geomview\_valide** ()
- void **Inactiv\_Visu\_geomview** ()

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Visualisation\_geomview.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Geomview/Visualisation\_geomview.cc

## 6.939 Référence de la classe [Visualisation\\_Gid](#)

### Fonctions membres publiques

- **Visualisation\_Gid** ([UtilLecture](#) \*ent)
- int **OrdresPossible** ()
- void **List\_increment\_disponible** (list< int > &list\_incr)
- const list< int > & **List\_balaie\_init** ()
- void **List\_balaie\_init** (const list< int > &list\_init)
- const list< int > & **List\_balaie** ()
- void **List\_maillage\_disponible** (int nombre\_maillage\_dispo)
- void **Initialisation** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob, bool fil\_calcul)
- void **Visu** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **Contexte\_debut\_visualisation** ()
- void **Contexte\_fin\_visualisation** ()
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- bool **Visu\_Gid\_valide** ()
- void **Inactiv\_Visu\_Gid** ()

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Visualisation\_Gid.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gid/Visualisation\_Gid.cc

## 6.940 Référence de la classe [Visualisation\\_Gmsh](#)

### Fonctions membres publiques

- **Visualisation\_Gmsh** ([UtilLecture](#) \*ent)
- int **OrdresPossible** ()
- void **List\_increment\_disponible** (list< int > &list\_incr)
- const list< int > & **List\_balaie\_init** ()

- void **List\_balaie\_init** (const list< int > &list\_init)
- const list< int > & **List\_balaie** ()
- void **List\_maillage\_disponible** (int nombre\_maillage\_dispo)
- void **Initialisation** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob, bool fil\_calcul)
- void **Visu** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)
- void **Contexte\_debut\_visualisation** ()
- void **Contexte\_fin\_visualisation** ()
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- bool **Visu\_Gmsh\_valide** ()
- void **Inactiv\_Visu\_Gmsh** ()
- const list< string > & **NomsTousLesGrandeursSortie** () const

### Fonctions membres publiques statiques

- static const [Tableau](#)< [Tableau](#)< int > > & **Tab\_HerGmsh** ()
- static const [Tableau](#)< [Tableau](#)< int > > & **Tab\_GmshHer** ()

### Fonctions membres protégées

- [OrdreVisu](#) \* **Existe** (string &reponse)
- void **Affiche\_options** ()

### Attributs protégés statiques

- static [Tableau](#)< [Tableau](#)< int > > **t\_HerGmsh**
- static [Tableau](#)< [Tableau](#)< int > > **t\_GmshHer**

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Visualisation\_Gmsh.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/Gmsh/Visualisation\_Gmsh.cc

## 6.941 Référence de la classe Visualisation\_maple

### Fonctions membres publiques

- **Visualisation\_maple** ([UtilLecture](#) \*ent)
- int **OrdresPossible** ()
- void **List\_increment\_disponible** (list< int > &list\_incr)
- const list< int > & **List\_balaie\_init** ()
- void **List\_balaie\_init** (const list< int > &list\_init)
- const list< int > & **List\_balaie** ()
- void **List\_maillage\_disponible** (int nombre\_maillage\_dispo)
- void **Initialisation** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob, bool fil\_calcul)
- void **Visu** ([ParaGlob](#) \*paraGlob, [LesMaillages](#) \*lesMaillages, [LesReferences](#) \*lesReferences, [LesLoisDeComp](#) \*lesLoisDeComp, [DiversStockage](#) \*diversStockage, Charge \*charge, [LesCondLim](#) \*lesCondLim, [LesContacts](#) \*lesContacts, [Resultats](#) \*resultats, [OrdreVisu::EnumTypeIncre](#) type\_incre, int incre, const map< string, const double \*, std::less< string > > &listeVarGlob, const [List\\_io](#)< [TypeQuelconque](#) > &listeVecGlob)

- void **Contexte\_debut\_visualisation** ()
- void **Contexte\_fin\_visualisation** ()
- void **Lecture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- void **Ecriture\_parametres\_OrdreVisu** ([UtilLecture](#) &entreePrinc)
- bool **Visu\_maple\_valide** ()
- void **Inactiv\_Visu\_maple** ()

La documentation de cette classe a été générée à partir du fichier suivant :

- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Visualisation\_maple.h
- /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Resultats/MAPLE/Visualisation\_maple.cc

# Chapitre 7

## Documentation des fichiers

### 7.1 Algori.h

```
1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *      DATE:      23/01/97
31 *
32 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
33 *
34 *      PROJET:    Herezh++
35 *
36 *
37 *      BUT:      Classe de base de Defintion des differents algorithmes
38 *               de resolution.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date !   auteur !           but
44 *      -----
45 *      !           !           !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date !   auteur !           but
50 *      -----
51 *
52 *****/
53 #ifndef ALGO_H
54 #define ALGO_H
55
56 #include <iostream>
57 #include "UtilLecture.h"
58 #include "EnumTypeCalcul.h"
59 #include "MotCle.h"
60 // #include "bool.h"
61 #include "string"
```

```

62 #include "Tableau_T.h"
63 #include "LesMaillages.h"
64 #include "LesReferences.h"
65 #include "LesCourbes1D.h"
66 #include "LesFonctions_nD.h"
67 #include "DiversStockage.h"
68 // #include "Charge.h"
69 #include "LesCondLim.h"
70 #include "ParaGlob.h"
71 #include "ParaAlgoControle.h"
72 #include "LesContacts.h"
73 #include "List_io.h"
74 #include "Visualisation.h"
75 #include "Visualisation_maple.h"
76 #include "Visualisation_geomview.h"
77 #include "Visualisation_Gid.h"
78 #include "Visualisation_Gmsh.h"
79 #include "Nb_assemb.h"
80 #include "Temps_CPU_HZpp.h"
81 #include "TypeQuelconqueParticulier.h"
82 #include "VariablesExporter.h"
83
84 // peut-être à modifier sur linux
85 #ifdef BIBLIOTHEQUE_STL_CODE_WARRIOR
86 #include <hash_map> // cas code warrior
87 #else
88 #ifdef SYSTEM_MAC_OS_X_unix
89 #include <ext/hash_map> // cas unix sur osX
90 #else
91 #include <ext/hash_map> // cas linux
92 #endif
93 #endif
94
95 class Resultats;
96 class Charge;
97 class AlgoriCombine;
98
99 /** @defgroup Les_algorithmes_de_resolutions_globales
100 *
101 * BUT: groupe des algorithmes de résolutions globales
102 *
103 *
104 * \author Gérard Rio
105 * \version 1.0
106 * \date 10/02/2004
107 * \brief groupe des algorithmes de résolutions globales
108 *
109 */
110
111 /// @addtogroup Les_algorithmes_de_resolutions_globales
112 /// @{
113 ///
114
115 /// BUT: Classe de base de Defintion des differents algorithmes
116 /// de resolution.
117
118 class Algori
119 { friend class Charge; friend class AlgoriCombine;
120 public :
121 // deux container de base qui servent pour les données externes
122 class DeuxString
123 { // surcharge de l'operator de lecture /écriture
124 friend istream & operator » (istream &, Algori::DeuxString &);
125 friend ostream & operator « (ostream &, const Algori::DeuxString &);
126 public :
127 string nomFichier; string nomMaillage;
128 };
129 class ListDeuxString
130 { // surcharge de l'operator de lecture /écriture
131 friend istream & operator » (istream &, Algori::ListDeuxString &);
132 friend ostream & operator « (ostream &, const Algori::ListDeuxString &);
133 public :
134 List_io<DeuxString > infos;
135 };
136
137 // surcharge de l'operator d'écriture pour avoir des tableaux mais
138 // non opérationnelle
139 friend ostream & operator « (ostream & sort, const Algori & )
140 { // tout d'abord un indicateur
141 cout « "\n **** écriture directe d'un algorithme non implante, il faut utiliser Affiche()";
142 Sortie(1);
143 return sort;
144 };
145
146 // CONSTRUCTEURS :
147 Algori (); // par default
148

```

```

149 // constructeur en fonction du type de calcul
150 // du sous type (pour les erreurs, remaillage etc...)
151 // il y a ici lecture des parametres attaches au type
152 Algori (EnumTypeCalcul type,const bool avec_typeDeCalcul
153         ,const list <EnumSousTypeCalcul>& soustype
154         ,const list <bool>& avec_soustypeDeCalcul
155         ,UtilLecture& entreePrinc);
156
157 // constructeur de copie
158 Algori (const Algori& algo);
159
160 // constructeur de copie à partie d'une instance indifférenciée
161 virtual Algori * New_idem(const Algori* algo) const =0 ;
162
163 // DESTRUCTEUR :
164 virtual ~Algori () ;
165
166 // METHODES PUBLIQUES :
167
168 // ----- static pour la création d'un algorithme particulier -----
169 // ramène un pointeur sur l'algorithme spécifique correspondant aux paramètres
170 // IMPORTANT : il y a création de l'algorithme correspondant (utilisation d'un new)
171 static Algori* New_Agori(EnumTypeCalcul type,const bool avec_typeDeCalcul
172                          ,const list <EnumSousTypeCalcul>& soustype
173                          ,const list <bool>& avec_soustypeDeCalcul
174                          ,UtilLecture& entreePrinc);
175 // ramène un tableau de pointeurs sur tous les algorithmes spécifique existants
176 // IMPORTANT : il y a création des algorithmes (utilisation d'un new)
177 static Tableau <Algori *> New_tous_les_Algo
178     (const bool avec_typeDeCalcul
179     ,const list <EnumSousTypeCalcul>& soustype
180     ,const list <bool>& avec_soustypeDeCalcul
181     ,UtilLecture& entreePrinc);
182
183 //lecture des parametres de controle
184 // peut être modifié dans le cas d'un algo particulier
185 virtual void Lecture(UtilLecture & entreePrinc,ParaGlob & paraGlob,LesMaillages& lesMail);
186 // cohérence du type de convergence adopté en fonction de l'algorithme: par exemple on vérifie
187 // que l'on n'utilise
188 // pas une convergence sur l'énergie cinétique avec un algo statique !!
189 void Coherence_Algo_typeConvergence();
190
191 // définition du nombre de cas d'assemblage, ce qui dépend du cas de calcul traité
192 int NbCasAssemblage() const {return nb_CasAssemblage;};
193
194 // execution de l'algorithme en fonction du type de calcul
195 virtual void Execution(ParaGlob *,LesMaillages *,LesReferences*,
196                      ,LesCourbesID*,LesFonctions_nD*,VariablesExporter*,
197                      ,LesLoisDeComp*, DiversStockage*,
198                      ,Charge*,LesCondLim*,LesContacts*,Resultats* ) = 0;
199 //----- décomposition en 3 du calcul d'équilibre -----
200 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
201 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
202 // calcul
203 // certaines variables ont-été changés
204
205 // initialisation
206 virtual void InitAlgorithme(ParaGlob *,LesMaillages *,LesReferences*,LesCourbesID*
207                            ,LesFonctions_nD* ,VariablesExporter*,LesLoisDeComp*
208                            ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* ) = 0;
209
210 // mise à jour
211 virtual void MiseAJourAlgo(ParaGlob *,LesMaillages *,LesReferences*,LesCourbesID*
212                            ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
213                            ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* ) = 0;
214
215 // calcul de l'équilibre
216 // si tb_combiner est non null -> un tableau de 2 fonctions
217 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
218 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
219 //
220 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
221 // en fonction de la demande de sauvegard,
222 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
223 // qui lui fonctionne indépendamment
224 virtual void CalEquilibre(ParaGlob *,LesMaillages *,LesReferences*,LesCourbesID*
225                          ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
226                          ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats*
227                          ,Tableau < Fonction_nD* > * tb_combiner) = 0;
228
229 // dernière passe
230 virtual void FinCalcul(ParaGlob *,LesMaillages *,LesReferences*,LesCourbesID*
231                       ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
232                       ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* ) = 0;
233
234 // sortie du schemaXML: en fonction de enu
235 virtual void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const = 0;
236
237 // affichage des parametres
238 void Affiche() const ; // affichage de tous les parametres

```

```

234 void Affiche1() const ; // affichage du type de calcul
235 void Affiche2() const ; // affichage des parametres de controle
236 // définition interactive exhaustive des paramètres de contrôle indépendamment de l'algorithme
237 // écriture du fichier de commande
238 void Info_commande_ParaAlgoControle(UtilLecture& lec)
239     {pa.Info_commande_ParaAlgoControle(lec);}
240
241 // sortie d'info sur un increment de calcul pour les ddl
242 // nb_casAssemb : donne le numéro du cas d'assemblage
243 void InfoIncrementDdl(LesMaillages * lesMail,
244     int inSol,double maxDeltaDdl,const Nb_assemb& nb_casAssemb);
245 // sortie d'info sur un increment de calcul pour les réaction
246 // dans le cas d'un compteur, par exemple dans le cas implicite
247 // nb_casAssemb : donne le numéro du cas d'assemblage
248 void InfoIncrementReac(LesMaillages * lesMail,int compteur,
249     int inreaction,double maxreaction,const Nb_assemb& nb_casAssemb);
250 // idem mais dans le cas où il n'y a pas de compteur, par exemple
251 // dans le cas explicite
252 // nb_casAssemb : donne le numéro du cas d'assemblage
253 void InfoIncrementReac(LesMaillages * lesMail,
254     int inreaction,double maxreaction,const Nb_assemb& nb_casAssemb);
255
256 // retourne le type de calcul
257 EnumTypeCalcul TypeDeCalcul() const { return typeCalcul;};
258
259 // dans le cas où un grand nombre d'information nécessaire pour l'algorithme
260 // est stocké de manière externe sur un fichier
261 // exemple : calcul d'erreur a partir
262 // d'un champ d'information stocké de manière externe sur un fichier
263 // l'acquisition du (ou des) lieu(s) (flot) où sont stocké ces infos est effectuée
264 // ce lieu fait parti des paramètres de l'algorithme
265 void DefFlotExterne(UtilLecture* entreePrinc);
266
267 // indique en retour s'il y a des données externes ou non
268 bool ExisteDonneesExternes() const
269     { if (noms_fichier.Taille() != 0) return true; else return false;};
270
271 // retourne le nombre total de flot externe où il faut lire
272 // des données externes
273 int NbTypeFlotExt() const {return noms_fichier.Taille();};
274
275 // retourne la liste des noms du fichier et des noms de maillage correspondant
276 // correspondant aux données externes nbff
277 const ListDeuxString& Noms_fichier(int nbff) const
278 // {return ((char*)(nom_fichier(nbff)).c_str());};
279     {return noms_fichier(nbff);};
280
281 // retourne le type de données externe pour le flot nbff
282 const int TypeDonneesExternes(int nbff) const { return typeFlotExterne(nbff);};
283
284 // retourne le numéro demandé de restart
285 const int Num_restart() const {return pa.Restart();};
286
287 // retourne le numéro d'incrément
288 const int Num_increment() const {return ichearge;};
289 // modif d'icharge, utilisé par les classes externes amies éventuellement
290 void Affectation_icharge(int ichearge_) {ichearge = ichearge_};
291
292 // retourne vraie si la sauvegarde est activée
293 const bool Active_sauvegarde() const {return pa.Sauvegarde();};
294
295 // sauvegarde générale sur base info
296 // cas donne le niveau de sauvegarde
297 // = 0 : initialisation de la sauvegarde -> c'est-à-dire de la sortie base info
298 // = 1 : on sauvegarde tout
299 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
300 // incre : numero d'incrément auquel on sauvegarde
301 // éventuellement est définit de manière spécifique pour chaque algorithme
302 // dans les classes filles
303 // type_incre :signal permet de localiser le dernier incrément
304 virtual void Ecriture_base_info
305     (int cas,LesMaillages *lesMaillages,
306     LesReferences* lesReferences,LesCourbes1D* lesCourbes1D,
307     LesFonctions_nD* lesFonctionsnD,
308     LesLoisDeComp* lesLoisDeComp,
309     DiversStockage* diversStockage,Charge* charge,
310     LesCondLim* lesCondlim,LesContacts* lesContacts,Resultats* resultats,
311     OrdreVisu::EnumTypeIncre type_incre,int incre = 0);
312 // cas d'un démarrage à partir de base_info
313 // éventuellement est définit de manière spécifique pour chaque algorithme
314 // dans les classes filles
315 // si inc_voulu est négatif cela signifie que l'on est déjà positionné sur un
316 // incrément voulu et qu'il faut simplement faire la lecture des infos
317 // cas = 3 : on met à jour uniquement les données variables, il y a modification des grandeurs,
318 // mais pas redimensionnement lorsque cela est viable (sinon on redimensionne)
319 virtual void Lecture_base_info(int cas,LesMaillages *lesMaillages,
320     LesReferences* lesReferences,LesCourbes1D* lesCourbes1D,

```



```

321         LesFonctions_nD* lesFonctionsnD,
322         LesLoisDeComp* lesLoisDeComp,
323         DiversStockage* diversStockage, Charge* charge,
324         LesCondLim* lesCondlim, LesContacts* lesContacts, Resultats* resultats,
325         int inc_voulu=0);
326
327 // visualisation interactive via le standard vrml
328 // la fonction est virtuelle ce qui lui permet d'être éventuellement facilement surchargée
329 // dans les algorithmes dérivées
330 virtual void Visu_vrml(ParaGlob * , LesMaillages * , LesReferences* , LesCourbes1D* ,
331         LesFonctions_nD* , LesLoisDeComp* , DiversStockage* ,
332         Charge* , LesCondLim* , LesContacts* , Resultats* );
333
334 // visualisation interactive par fichier de points, utilisant le format maple ou gnuplot
335 // la fonction est virtuelle ce qui lui permet d'être éventuellement facilement surchargée
336 // dans les algorithmes dérivées
337 virtual void Visu_maple(ParaGlob * , LesMaillages * , LesReferences* , LesCourbes1D* ,
338         LesFonctions_nD* , LesLoisDeComp* , DiversStockage* ,
339         Charge* , LesCondLim* , LesContacts* , Resultats* );
340
341 // visualisation interactive via geomview
342 // la fonction est virtuelle ce qui lui permet d'être éventuellement facilement surchargée
343 // dans les algorithmes dérivées
344 virtual void Visu_geomview(ParaGlob * , LesMaillages * , LesReferences* , LesCourbes1D* ,
345         LesFonctions_nD* , LesLoisDeComp* , DiversStockage* ,
346         Charge* , LesCondLim* , LesContacts* , Resultats* );
347
348 // visualisation interactive via Gid
349 // la fonction est virtuelle ce qui lui permet d'être éventuellement facilement surchargée
350 // dans les algorithmes dérivées
351 virtual void Visu_Gid(ParaGlob * , LesMaillages * , LesReferences* , LesCourbes1D* ,
352         LesFonctions_nD* , LesLoisDeComp* , DiversStockage* ,
353         Charge* , LesCondLim* , LesContacts* , Resultats* );
354
355 // visualisation interactive via Gmsh
356 // la fonction est virtuelle ce qui lui permet d'être éventuellement facilement surchargée
357 // dans les algorithmes dérivées
358 virtual void Visu_Gmsh(ParaGlob * , LesMaillages * , LesReferences* , LesCourbes1D* ,
359         LesFonctions_nD* , LesLoisDeComp* , DiversStockage* ,
360         Charge* , LesCondLim* , LesContacts* , Resultats* );
361
362 // lecture d'informations de visualisation dans un fichier de commande externe
363 void LectureCommandeVisu(ParaGlob * , LesMaillages * , LesReferences* , LesCourbes1D* ,
364         LesFonctions_nD* , LesLoisDeComp* , DiversStockage* ,
365         Charge* , LesCondLim* , LesContacts* , Resultats* );
366
367 // visualisation au fil du calcul à partir d'un fichier de commande externe
368 // type_incre signal ce qu'il y a à faire, il est modifié dans le programme !! donc le laisser en
variable
369 // en fait après le premier passage
370 void VisuAuFilDuCalcul(ParaGlob * , LesMaillages * , LesReferences* , LesCourbes1D*
371         , LesFonctions_nD* , LesLoisDeComp* , DiversStockage*
372         , Charge* , LesCondLim* , LesContacts* , Resultats* , OrdreVisu::EnumTypeIncre&
type_incre
373         , int num_incre);
374
375 // Ecriture des informations de visualisation dans un fichier de commande externe
376 void EcritureCommandeVisu();
377
378 // sortie sur fichier des temps cpu
379 virtual void Sortie_temps_cpu(const LesCondLim& lesCondlim
380         , const Charge& charge, const LesContacts & contact);
381
382 /* // visualisation automatique sans partie interactive
383 void VisualisationAutomatique(ParaGlob * , LesMaillages * , LesReferences* , LesCourbes1D*
, LesLoisDeComp* , DiversStockage* ,
384         Charge* , LesCondLim* , LesContacts* , Resultats* ); */
385
386 // une méthode qui a pour objectif de terminer tous les comptages, utile
387 // dans le cas d'un arrêt imprévu
388 virtual void Arret_du_comptage_CPU();
389
390 //----- protégé -----
391 protected :
392 // METHODES PROTEGEES :
393 // préparation des conteneurs interne, utilisés après la lecture
394 // où dans le cas où il n'y a pas de lecture directe
395 // (pour un sous algo via un algo combiné par exemple)
396 void Preparation_conteneurs_interne(LesMaillages& lesMail);
397 // sauvegarde sur base info
398 // cas donne le niveau de sauvegarde
399 // = 0 : initialisation de la sauvegarde -> c'est-à-dire de la sortie base info
400 // = 1 : on sauvegarde tout
401 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
402 // incre : numero d'incrément auquel on sauvegarde
403 // éventuellement est défini de manière spécifique pour chaque algorithme
404 // dans les classes filles

```

```

405 // si ptalgo est non nulle, on a une sortie des temps cpu, spécifique
406 // à la classe AlgoriCombine passé en paramètre
407 virtual void Ecriture_base_info
408     (ofstream& sort,const int cas);
409 // cas de la lecture spécifique à l'algorithme dans base_info
410 virtual void Lecture_base_info(ifstream& ent,const int cas);
411 // cas des paramètres spécifiques
412 // écriture des paramètres dans la base info
413 // = 1 : on écrit tout
414 // = 2 : on écrit uniquement les données variables (supposées comme telles)
415 virtual void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas) = 0;
416 // lecture des paramètres dans la base info
417 // = 1 : on récupère tout
418 // = 2 : on récupère uniquement les données variables (supposées comme telles)
419 // choix = true : fonctionnement normal
420 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
    défaut
421 //
    car la lecture est impossible
422 virtual void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix) = 0;
423
424 // création d'un fichier de commande: cas des paramètres spécifiques
425 virtual void Info_commande_parametres(UtilLecture& entreePrinc) = 0;
426
427 // mise à jour par une classe friend (attention) des pointeurs de sous types
428 void Change_affectation_pointeur_sous_type
429     (const list <EnumSousTypeCalcul>& soustype
430      ,const list <bool>& avec_soustypeDeCal)
431     {soustypeDeCalcul = &soustype; avec_soustypeDeCalcul = &avec_soustypeDeCal;};
432
433 // initialisation de tous les paramètres de contrôle aux valeurs transmises
434 void Init_ParaAlgoControle (ParaAlgoControle& paa)
435     {pa = paa;
436      if (type_calcul_visqu_critique) // cas particulier d'un amortissement critique
437          pa.Change_amort_visco_artificielle(4); // on indique qu'il s'agit d'une gestion par un
    amortissement visqueux critique
438     };
439 // récupération des paramètres de contrôles de l'algo
440 ParaAlgoControle& ParaAlgoControle_de_lalgo() {return pa;};
441
442 // récupération de la position lue par l'algo dans le .BI, changé lors d'un restart par exemple
443 // il s'agit de la sauvegarde par l'algo de la position qu'il a lue en restart
444 streampos Debut_increment() const {return debut_increment;};
445
446 // =====VARIABLES PROTEGEES :
447 // 1)----- protégées: spécifiques à un algo -----
448 int mode_debug; // permet de spécifier des options particulières pour débbuger
449 // controle de la sortie des informations: utilisé par les classes dérivées
450 int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les erreurs
    et warning
451
452 EnumTypeCalcul typeCalcul;
453 bool avec_typeDeCalcul; // indique si le calcul doit être effectué ou pas
454 int nb_CasAssemblage; // indique le nombre de cas d'assemblage
455 Tableau<double> paraTypeCalcul; // paramètres particuliers à chaque algorithme
456 // identificateur secondaire optionnel
457 //du type de Calcul renseigne sur le fait de calculer l'erreur,
458 // une relocation, un raffinement, éventuel.
459 list <EnumSousTypeCalcul> const * soustypeDeCalcul;
460 list <bool> const *avec_soustypeDeCalcul; // indique les sous types actifs ou non
461 ParaAlgoControle pa; // parametres de controle de l'algorithme
462
463 // 2) ----- protégées: communs à tous les algos -----
464 double temps_derniere_sauvegarde; // variable stockant le dernier temps de la sauvegarde sur .BI
465 // liste de tous les différents incréments et temps des sauvegardes effectuées dans le .BI
466 List_io <Entier_et_Double> list_incre_temps_sauvegarder;
467 List_io <Entier_et_Double> list_incre_temps_calculer; // tous les incréments calculés
468 double tempsDerniereSortieFilCalcul; // variable stockant le dernier temps de sortie au fil du
    calcul
469 // définition des flots externes éventuels
470 Tableau <ListDeuxString> noms_fichier;
471 Tableau <int > typeFlotExterne;
472 UtilLecture* entreePrinc; // sauvegarde pour éviter les passages de paramètres
473 streampos debut_increment; // sauvegarde de la position du début d'incrément
474 // lors d'un restart
475 Visualisation visualise; // instance sur les utilitaires de visualisation en vrml
476 Visualisation_maple visualise_maple; // instance sur les utilitaires de visualisation en maple
477 Visualisation_geomview visualise_geomview; // instance sur les utilitaires de visualisation en
    // geomview
478 Visualisation_Gid visualise_Gid; // instance sur les utilitaires de visualisation en Gid
479 Visualisation_Gmsh visualise_Gmsh; // instance sur les utilitaires de visualisation en Gmsh
480
481
482 private:
483     Vecteur * toto_masse; // pointeur intermédiaire utilisée dans la méthode Cal_matrice_masse
484
485 protected:
486     // compteur d'incréments de charge: s'avère dans les faits, utilisés par tous les algorithmes
487     // est renseigné par les classes dérivées

```

```

488     int  icharge;
489
490     // --- les énergies et forces généralisées ( utilisées par certains algorithmes )
491     double E_cin_0,E_cin_t,E_cin_tdt,E_int_t,E_int_tdt,E_ext_t,E_ext_tdt,bilan_E; // différentes
énergies et bilan
492     double q_mov_t,q_mov_tdt; // la quantité de mouvement
493     double P_acc_tdt,P_int_tdt,P_ext_tdt,bilan_P_tdt; // différentes puissances et bilan
494     double P_acc_t,P_int_t,P_ext_t,bilan_P_t; // différentes puissances et bilan
495     double E_visco_numerique_t,E_visco_numerique_tdt; // énergie due à la viscosité numérique
496     // == vecteurs
497     Vecteur F_int_t,F_ext_t,F_int_tdt,F_ext_tdt; // forces généralisées int et ext à t et t+dt
498     Vecteur F_totale_tdt; // bilan des forces internes et externes, = F_int_tdt + F_ext_tdt
499     // évite d'avoir à le construire à chaque utilisation: utilisation en post-traitement uniquement
500     Vecteur residu_final; // vecteur intermédiaire de sauvegarde pour la visualisation uniquement
501     Vecteur X_t,X_tdt,delta_X,var_delta_X; // les positions, variations entre t et tdt, et variation
502     // d'une itération à l'autre en implicite, et d'un incrément à l'autre en explicite
503     double delta_t_precedent_a_convergence; // cf. son nom !
504     Vecteur vitesse_t,vitesse_tdt; // vitesses
505     Vecteur acceleration_t,acceleration_tdt ; // accélérations
506
507     double E_bulk; // énergie due à l'utilisation du bulk viscosity
508     double P_bulk; // puissance due à l'utilisation du bulk viscosity
509     // ... les mêmes informations individuelle (ind) par maillage
510     List_io < double>
E_cin_ind_t,E_cin_ind_tdt,E_int_ind_t,E_int_ind_tdt,E_ext_ind_t,E_ext_ind_tdt,bilan_ind_E; //
différentes énergies et bilan
511     List_io < double> q_mov_ind_t,q_mov_ind_tdt; // la quantité de mouvement
512     List_io < double> P_acc_ind_tdt,P_int_ind_tdt,P_ext_ind_tdt,bilan_P_ind_tdt; // différentes
puissances et bilan
513     List_io < double> P_acc_ind_t,P_int_ind_t,P_ext_ind_t,bilan_P_ind_t; // différentes puissances et
bilan
514     List_io < double> E_visco_numerique_ind_t,E_visco_numerique_ind_tdt; // énergie due à la viscosité
numérique
515     List_io < Vecteur> F_int_ind_t,F_ext_ind_t,F_int_ind_tdt,F_ext_ind_tdt; // forces généralisées int
et ext à t et t+dt
516     List_io < double> E_bulk_ind; // énergie due à l'utilisation du bulk viscosity
517     List_io < double> P_bulk_ind; // puissance due à l'utilisation du bulk viscosity
518
519     // liste des variables globales scalaires sous forme d'un arbre pour faciliter la recherche
520     // contient par exemple des pointeurs sur les énergies
521     map < string, const double * , std::less <string> > listeVarGlob;
522     // liste des grandeurs globales vectorielles qui représentent en fait des infos aux noeuds: exe:
F_int_t et F_ext_t
523     List_io < TypeQuelconque > listeVecGlob;
524
525     int  deja_lue_entete_parametre; // une variable interne qui permet de dire si l'algo spécifique a
déjà lue
526     // l'entete des paramètres ou pas : =0 : pas de procédure spécifique de lecture de paramètre
527     // =1, procédure spécifique, mais pas de lecture effective de l'entête par la procédure
spécifique
528     // =2, procédure spécifique et lecture effective de l'entête par la procédure spécifique
529
530     // variables privées internes servant au pilotage de la convergence
531     int  PhaseDeConvergence() const {return phase_de_convergence;}; // pour l'accès en lecture uniquement
532     void Change_PhaseDeConvergence(int phase) {phase_de_convergence=phase;};
533 private :
534     int  phase_de_convergence; // =0 indique si l'on est en phase de convergence
535     // =1 on a convergé soit très bien soit correctement
536     // =2 on a convergé, mais avec une mauvaise convergence
537     // =-1 on a divergé dans l'algo du à un nb d'iter trop grand, =1 on a convergé
538     // =-2, =-3, =-4 indique qu'un arrêt a été demandé par la méthode Convergence
539     // = -5 : indique que l'on prend en compte un jacobien négatif sous forme d'une
divergence
540     // = -6 : idem pour une variation de jacobien trop grande
541     // = -7 : idem pour une non convergence sur la loi de comportement
542     // = -8 : idem mais la non convergence "couve" depuis plusieurs incréments !!
543     // = -9 : divergence due au fait que la résolution du système linéaires est impossible
544     // = -10 : problème due au chargement
545     // = -11 : on a trouvé un nan ou un nombre infini sur la norme des efforts int et/ou ext
546     int  nombre_de_bonnes_convergences; // indique le nombre actuel de bonnes convergences en implicite
547     int  nombre_de_mauvaises_convergences; // indique le nombre actuel de mauvaises convergences
consécutives en implicite
548     int  a_converge; // indique que l'on a convergé
549     int  a_converge_iterMoins1; // garde en mémoire la convergence de l'itération précédente, n'est utile
550     // que si on veut une vérification double (deux fois) de la convergence
551     Tableau <double> max_var_residu; // tableau qui garde en souvenir la suite des maxi pour le
pilotage
552     int  nb_cycle_test_max_var_residu; // utilisé dans Convergence()
553
554     //--- gestion des stockages
555     // -- une classe de stockage de paramètres pour le changement de largeur de bande
556     class ParaStockage {public: Nb_assemb nbass; int demi,total;
557     ParaStockage():nbass(),demi(0),total(0) {} ; };
558     // dans le cas ou on ne fait que changer la largeur de bande en fonction du contact
559     // avec une numérotation qui reste fixe (et donc des pointeurs d'assemblage fixe)
560     // on se sert de tab_para_stockage pour ne pas recalculer la largeur de bande sans contact

```

```

561 // par compte dès que les pointeurs d'assemblage change, il faut remettre à jour
562 // cf. -> Mise_a_jour_Choix_matriciel_contact
563 Tableau < ParaStockage > tab_para_stockage;
564
565 // variables internes de travail
566 Vecteur v_int; // utilisé par CalEnergieAffichage
567
568 // cas de la remontée aux contraintes ou déformation, et calcul d'erreur
569 Nb_assemb* cas_ass_sig; // assemblage pour de tous les variables contraintes définit aux noeuds
570 Assemblage* Ass_sig; // "
571 Nb_assemb* cas_ass_sigll; // assemblage pour des variables contraintes sigmall définit aux noeuds
572 Assemblage* Ass_sigll; // "
573 Nb_assemb* cas_ass_eps; // assemblage pour de tous les variables déformations définit aux noeuds
574 Assemblage* Ass_eps; // "
575 Nb_assemb* cas_ass_epsll; // assemblage pour des variables déformations EPSll définit aux noeuds
576 Assemblage* Ass_epsll; // "
577 Nb_assemb* cas_ass_err; // assemblage pour des variables d'erreurs définit aux noeuds
578 Assemblage* Ass_err; // "
579 Mat_abstraite* mat_glob_sigeps; // matrice utilisée pour la forme variationnelle du pb
580 Tableau <Vecteur>* vglob_sigeps; // tableau de seconds membres globaux
581 Vecteur* vglobal_sigeps; // vecteur global de tous les ddl de sigma ou epsilon
582 int nbddl_sigeps,nbddl_sigepsll; // nombre total de ddl de type sigma ou eps, et nb total uniquement
du premier
583 int nbddl_err; // nombre de ddl d'erreur
584 bool avec_remont_sigma,avec_remont_eps,avec_erreur; // indicateurs intermédiaires utilisés par
CalculRemont et InitRemont
585 // -- volume matière
586 double volume_total_matiere; // volume totale de matière
587 // -- volume déplacé: cas d'élément 2D dans un calcul 3D
588 // un tableau pour avoir un volume par maillage
589 Tableau <Coordonnee> vol_total2D_avec_plan_ref; // volume total entre la surface et : yz, xz,xy
590 // -- cas des énergies
591 EnergieMeca energTotal; // énergie totale du système
592 double energHourglass; // énergie totale liée au contrôle d'hourglass
593 EnergieMeca energFrottement; // énergie liée au frottement de contact
594 double energPenalisation; // énergie liée à la mise en place d'une méthode de pénalisation
595 double energStabiliMembBiel; // énergie liée à la stabilisation des membranes et biels éventuelles
596
597 // ----- debut amortissement cinétique: => spécifique à un algo -----
598 protected:
599 bool amortissement_cinetique; // indique si oui ou non on utilise de l'amortissement cinétique
600 Vecteur X_EcinMax; // le X à l'énergie cinétique maxi (ou moy glissante)
601
602 private:
603 // deux techniques pour appliquer le test, qui conduit à mettre les vitesses à 0
604 // 1) on regarde si l'énergie cinétique| diminue n1 fois (pas forcément consécutives)
605 // ou est inférieure n1 fois fraction * dernier_max_E_cin, si fraction (=
test_fraction_energie) est non nulle
606 // NB: le test ne se déclenche que lorsque l'on a scruté au moins n1 valeurs de l'énergie
cinétique
607 // 2) on regarde si la moyenne glissante de n2 |énergie cinétique| consécutive diminue n1 fois (pas
forcément consécutives)
608 // ou est inférieure n1 fois à fraction * dernier_max_E_cin, si fraction (=
test_fraction_energie) est non nulle,
609 // NB: le test ne se déclenche que lorsque l'on a au moins n2 valeurs consécutives disponibles
de l'énergie cinétique
610 int max_nb_decroit_pourRelaxDyn; // paramètre n1
611 int taille_moyenne_glissante; // paramètre n2
612 double moyenne_glissante, moy_gliss_t; // valeur de la moyenne glissante actuelle et moyennes à t
613 int inter_nb_entre_relax; // indique un nombre mini d'iter avec la précédente relaxation, avant
d'autoriser
614 // une nouvelle relaxation
615 Fonction_nD* fct_nD_inter_nb_entre_relax; // si non NULL: donne une valeur qui remplace
inter_nb_entre_relax
616 string nom_fct_nD_inter_nb_entre_relax; // nom éventuel de la fonction associée
617
618 int nb_depuis_derniere_relax; // variable inter qui permet d'appliquer inter_nb_entre_relax
619
620 Vecteur sauve_E_cin; // sauvegarde des valeurs pour le calcul de la moyenne glissante
621 int indice_Ec1; // indice dans sauve_E_cin, de la première valeur d'énergie cinétique, stockée
622 int nb_Ecin_stocker; // indique le nombre courant de valeur d'énergies cinétiques stockées
623 double test_fraction_energie; // si diff de 0, indique la valeur de "fraction"
624 int compteur_decroit_pourRelaxDyn; // compteur associé à max_nb_decroit_pourRelaxDyn
625 // ==-1 lorsque la relaxation est suspendue
626 double coef_arret_pourRelaxDyn; // on arrête la relaxation lorsque E_cin_tdt < E_cin_t * coef
627 double dernier_pic,pic_E_cin_t; // dernier pic, et pic à l'instant t d'énergie cinétique
enregistrée
628 double max_pic_E_cin; // le max des pic d'énergie cinétique
629 double coef_redemarrage_pourRelaxDyn; // on redémarre la relaxation lorsque
630 // E_cin_tdt > max_pic_E_cin * coef_redemarrage_pourRelaxDyn
631 double max_deltaX_pourRelaxDyn; // le calcul s'arrête lorsque deltaX < max_deltaX_pourRelaxDyn m
fois
632 int nb_max_dX_OK_pourRelaxDyn; // = le m de la ligne précédente
633 int nb_dX_OK_pourRelaxDyn; // le nombre de dx OK courant
634 int nb_deb_testfin_pourRelaxDyn; // le nombre mini de passage dans la relaxation, a partir duquel on
test la fin
635 int nb_deb_test_amort_cinetique; // le nombre mini d'iteration, a partir duquel on démarre

```

```

l'algorithme
636   int compteur_test_pourRelaxDyn; // compteur associé avec nb_deb_testfin_pourRelaxDyn
637   int compteur_pic_energie; // compteur absolu numérotant les pic d'énergie, pas sauvegardé, donc mis
    à 0 à chaque
638                                     // démarrage de calcul
639   //+ cas de l'amortissement local
640   int amortissement_cinetique_au_noeud; // si diff de 0, indique une technique d'amortissement
    individuelle à chaque noeud
641   int nb_deb_test_amort_cinetique_noe; // le nombre mini d'iteration, a partir duquel on démarre
l'algorithme
642
643   Vecteur E_cinNoe_tdt,E_cinNoe_t; // énergie cinétique à chaque noeud, à t et tdt
644
645   int max_nb_decroit_pourRelaxDyn_noe; // idem paramètre n1 sur le globale
646   Tableau <int> compteur_decroit_pourRelaxDyn_noe; // compteur associé à max_nb_decroit_pourRelaxDyn
647                                     // ==-1 lorsque la relaxation est suspendue
648   double coef_arret_pourRelaxDyn_noe; // on arrête la relaxation lorsque eN_cin_tdt < eN_cin_t * coef
649
650   Vecteur dernier_pic_noe; // pour chaque noeuds: dernier pic, et pic à l'instant t d'énergie
    cinétique enregistrée
651   Vecteur pic_E_cint_t_noe; //
652   Vecteur max_pic_E_cin_noe; // pour chaque noeud: le max des pic d'énergie cinétique
653   double coef_redemarrage_pourRelaxDyn_noe; // on redémarre la relaxation lorsque
654                                     // eN_cin_tdt > max_pic_E_cin_noe *
    coef_redemarrage_pourRelaxDyn_noe
655   int compteur_pic_energie_noe; // compteur absolu numérotant les pic d'énergie, pas sauvegardé, donc
    mis à 0 à chaque
656                                     // démarrage de calcul
657
658   Vecteur Puiss_loc_t,Puiss_loc_tdt; // puissances locales à chaque noeuds (comprend les forces
    externes et internes)
659 // ----- fin amortissement cinétique: => spécifique à un algo -----
660
661 // ----- début amortissement visqueux critique: => spécifique à un algo -----
662   int opt_cal_C_critique; // choix pour le type de calcul du quotient de Rayleigh
663   double f_; // le coefficient multiplicateur pour la borne maxi de la fréquence mini
664   double ampli_visco; // un coefficient d'amplification arbitraire de la viscosité
665   int type_calcul_visqu_critique; // choix entre différents calcul de la partie visqueuse
666   Mat_abstraite* C_inter; // matrice de travail utilisé dans la méthode
    CalculEnContinuMatriceViscositeCritique
667 // ----- fin amortissement visqueux critique: => spécifique à un algo -----
668
669 // -- pour les algo dynamiques, convergences vers une solution statique
670   int arret_a_equilibre_statique ;
671       // indique, si diff de 0, que l'on veut contrôler une convergence
672       //         Si = 1: le residu statique est utilise comme critere
673       //         Si = 2: nécessite que les deux critères précédents soient satisfait
674
675 // -- modulation de la précision de convergence via une fonction nD
676   string nom_modulation_precision; // nom éventuel de la fonction de modulation
677   Fonction_nD* modulation_precision; // si non nulle => modulation
678
679 //-- utilisation éventuelle d'une fonction nD pour calculer le type de norme de convergence
680 // fait un peu double emploi avec la modulation de précision, mais est plus souple d'emplois
681 // du fait que l'on peut définir ainsi directement le type de norme qu'on veut utiliser
682   Fonction_nD* fct_norme;
683
684 protected:
685   Vecteur F_int_tdt_prec, F_ext_tdt_prec; // forces généralisées de l'itération précédente
    (éventuellement)
686   // vecteur intermédiaire, utilisé par les classes dérivées pour stocker la puissance totale
    statique: sans accélération et sans visqueux numérique
687   Vecteur * vglob_stat; // dans le cas où on veut une convergence vers la solution statique, sans
    viscosité dynamique
688   inline int Arret_A_Equilibre_Statique() const {return arret_a_equilibre_statique;};
689
690   Vecteur vec_trav; // vecteur de travail, de dim le nb de ddl par exe, utilisé par Pilotage_maxi_X_V
691
692 // récupération de la précision en cours
693   double Precision_equilibre() const
694   {double pa_precision = pa.Precision();
695     if (modulation_precision != NULL) // cas où en plus on fait une modulation de précision
696       pa_precision *= (modulation_precision->Valeur_pour_variables_globales())(1);
697     return pa_precision;
698   };
699
700
701 //----- temps cpu ----- on garde des temps spécifiques pour chaque algo
702 // ainsi en sortie on pourra différencier les temps totaux et les temps partiels
703   Temps_CPU_HZpp tempsInitialisation; // lesTempsCpu(1)
704   Temps_CPU_HZpp tempsMiseAJourAlgo; // lesTempsCpu(2)
705   Temps_CPU_HZpp tempsCalEquilibre; // lesTempsCpu(3)
706   Temps_CPU_HZpp tempsRaidSmEner; // lesTempsCpu(4)
707   Temps_CPU_HZpp tempsSecondMembreEner; // lesTempsCpu(5)
708   Temps_CPU_HZpp tempsResolSystemLineaire; // lesTempsCpu(6)
709   Temps_CPU_HZpp tempsSauvegarde; // lesTempsCpu(7)
710   Temps_CPU_HZpp tempsSortieFilCalcul; // lesTempsCpu(8)

```

```

711     Temps_CPU_HZpp tempsRaidSmEnerContact;          // lesTempsCpu(9)
712     Temps_CPU_HZpp tempsSecondMembreEnergyContact; // lesTempsCpu(10)
713     Temps_CPU_HZpp temps_CL;                       // lesTempsCpu(11)
714     Temps_CPU_HZpp temps_CLL;                     // lesTempsCpu(12)
715     Temps_CPU_HZpp temps_lois_comportement;       // lesTempsCpu(13)
716     Temps_CPU_HZpp temps_metrique_K_SM;          // lesTempsCpu(14)
717     Temps_CPU_HZpp temps_chargement;             // lesTempsCpu(15)
718     Temps_CPU_HZpp temps_rech_contact;           // lesTempsCpu(16)
719
720     Tableau <Coordonnee3> lesTempsCpu; // un tableau intermédiaire qui récupère et globalise les temps
pour les sorties
721
// via listeVarGlob, mais c'est les variables Temps_CPU_HZpp qui
stockent
722
// réellement les temps
723 // la petite méthode qui sert au transfert et qui doit-être appelé avant les sorties
724 void Temps_CPU_HZpp_to_lesTempsCpu(const LesCondLim& lesCondLim, const Charge& charge,const
LesContacts & contact);
725
726 //===== cas particulier des algo combiner =====
727 // un pointeur dans le cas où les temps à sortir ne sont pas ceux stockés
728 // --> utilisé uniquement par l'algo combiner, car dans ce cas il faut globaliser les temps
729 // de tous les sous-algo
730 // par défaut ce pointeur est NULL
731 AlgoriCombine* ptalgotcombi;
732 // et la méthode pour changer
733 void Change_ptalgotcombi (AlgoriCombine* ptal) {ptalgotcombi = ptal;};
734 // idem la méthode de transfert si-dessus mais concerne uniquement les temps internes à l'algo
735 // ajoute au tableau passé en paramètre, les temps de l'algo
736 Tableau <Temps_CPU_HZpp> & Ajout_Temps_CPU_HZpp_to_lesTempsCpu(Tableau <Temps_CPU_HZpp> &
lesTsCpu);
737 //===== fin cas particulier des algo combiner =====
738
739
740 // --- METHODES PRIVEES
741
742 // METHODES PROTEGEES :
743 protected:
744
745 // -- pour les algo dynamiques, convergences vers une solution statique
746 const int& ArretEquilibreStatique()const {return arret_a_equilibre_statique;}; // indique, si
diff de 0, que l'on veut contrôler une convergence vers la solution
747 // initialisation du calcul de la remontée des contraintes aux noeuds
748 // initialisation valable également dans le cas où aucun calcul n'a précédé ce calcul
749 void InitRemontSigma(LesMaillages * lesMail,LesReferences* ,
DiversStockage* ,Charge* ,
LesCondLim* ,LesContacts* ,Resultats* );
750 // initialisation du calcul de la remontée des déformations aux noeuds
751 // initialisation valable également dans le cas où aucun calcul n'a précédé ce calcul
752 void InitRemontEps(LesMaillages * lesMail,LesReferences* ,
DiversStockage* ,Charge* ,
LesCondLim* ,LesContacts* ,Resultats* );
753 // initialisation du calcul d'erreur aux éléments
754 // initialisation valable également dans le cas où aucun calcul n'a précédé ce calcul
755 void InitErreur(LesMaillages * lesMail,LesReferences* lesRef,
DiversStockage* toto,Charge* charge,
LesCondLim* lesCondLim,LesContacts* titi,Resultats* tutu);
756 // calcul de la remontée des contraintes aux noeuds
757 void RemontSigma(LesMaillages * lesMail);
758 // calcul de la remontée des déformations aux noeuds
759 void RemontEps(LesMaillages * lesMail);
760 // calcul de l'erreur et de sa remontée aux noeuds, après un calcul de mécanique.
761 // nécessite d'avoir fait auparavant la remontée des contraintes (initialisation et remontée !!)
762 void RemontErreur(LesMaillages * lesMail);
763 // initialisation d'un calcul de remontée à des variables secondaires ou d'erreur
764 // globalement, cette initialisation dépend des paramètres utilisateurs
765 // ramène true s'il y a une initialisation
766 bool InitRemont(LesMaillages * lesMail,LesReferences* lesRef, DiversStockage* ,Charge* charge,
LesCondLim* lesCondLim,LesContacts* ,Resultats* );
767 // calcul d'une remontée à des variables secondaires ou d'erreur, ceci pour un post traitement de
visualisation
768 // ce calcul dépend des paramètres utilisateurs, il n'est fait qu'à chaque sortie .BI
769 // ramène true s'il y a un calcul effectif
770 bool CalculRemont(LesMaillages * lesMail,OrdreVisu::EnumTypeIncre type_incre,int incre);
771
772 // lecture d'un type externe
773 void LectureUnTypeExterne(UtilLecture* entreePrinc,const int type);
774
775 // lecture des paramètres du calcul généraux pour tous les algos (dans le cas où il y en a)
776 // qui peuvent éventuellement ne pas servir !!
777 virtual void lecture_Parametres(UtilLecture& entreePrinc);
778
779 // création d'un fichier de commande: cas des paramètres spécifiques
780 // par défaut on indique qu'il n'y a pas de paramètres,
781 // cette méthode est appelée par les classes dérivées
782 void Info_com_parametres(UtilLecture& entreePrinc);
783
784 // algorithme classique de line search

```



```

792 // le line seach agit sur une partie de la puissance et non sur la norme du résidu
793 // le line search à pour objectif de minimiser une partie de l'acroissement de la puissance
794 // en jeux, ainsi on peut avoir une minimisation réussi tout en ayant une norme d'équilibre
795 // qui continue à grandir
796 // cependant en général les deux sont lié et l'application du line_search garanti que dans
797 // la direction de descente choisi on minimise norme (mais ce minimum peut cependant être
798 // supérieur à l'ancienne norme!)
799 // la méthode renvoi si oui ou non le line search a été effecué
800 bool Line_search1(Vecteur& sauve_deltadept,double& puis_precedente,
801                 Vecteur& Vres,LesMaillages * lesMail,Vecteur* sol
802                 ,int& compteur,Vecteur& sauve_dept_a_tdt,Charge* charge,Vecteur& vglobex
803                 ,Assemblage& Ass,Vecteur& v_travail,LesCondLim* lesCondLim,Vecteur& vglobal
804                 ,LesReferences* lesRef,Vecteur & vglobin,const Nb_assemb& nb_casAssemb
805                 ,int cas_combi_ddl,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD);
806
807 // algorithmme de line search qui utilise une approximation cubique du résidu ( (R).<ddl>)
808 // en fonction du paramètre lambda de line search ( X+lambda. delta_ddl)
809 // la méthode renvoi si oui ou non le line search a été effecué
810 bool Line_search2(Vecteur& sauve_deltadept,double& puis_precedente,
811                 Vecteur& Vres,LesMaillages * lesMail,Vecteur* sol
812                 ,int& compteur,Vecteur& sauve_dept_a_tdt,Charge* charge,Vecteur& vglobex
813                 ,Assemblage& Ass,Vecteur& v_travail,LesCondLim* lesCondLim,Vecteur& vglobal
814                 ,LesReferences* lesRef,Vecteur & vglobin,const Nb_assemb& nb_casAssemb
815                 ,int cas_combi_ddl,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD);
816
817 // calcul de la raideur et du second membre pour tous les maillages ainsi que des énergies
818 // si pb retour false: indique que le calcul c'est mal passé, il y a une erreur
819 // généré, les résultats: raideur et second membres calculées sont inexploitable
820 bool RaidSmEner(LesMaillages * lesMail,Assemblage& Ass,Vecteur & vglobin
821               ,Mat_abstraite& matglob );
822 // calcul du second membre pour tous les maillages ainsi que des énergies
823 // si pb retour false: indique que le calcul c'est mal passé, il y a une erreur
824 // généré, les résultats: raideur et second membres calculées sont inexploitable
825 bool SecondMembreEnerg(LesMaillages * lesMail,Assemblage& Ass,Vecteur & vglobin);
826
827 // Mise à jour de la raideur et du second membre pour le contact
828 // calcul des énergies spécifiques au contact
829 // si pb retour false: indique que le calcul c'est mal passé, il y a une erreur
830 // généré, les résultats: raideur et second membres calculées sont inexploitable
831 bool RaidSmEnerContact(LesContacts * lesCont,Assemblage& Ass,Vecteur & vglobin
832                       ,Mat_abstraite& matglob );
833
834 // cas du contact: calcul et mise à jour du second membre ainsi que des énergies spécifiques
835 bool SecondMembreEnergContact(LesContacts * lesContacts,Assemblage& Ass,Vecteur & vglobin
836                               ,bool aff_iteration);
837
838 // choix de la (les) matrices de raideur du système linéaire
839 // 1) Si le pointeur est non null, on vérifie les tailles, si c'est différent
840 // la matrice est reconstruite
841 // 2) Si le pointeur est null, il y a création d'une matrice
842 // lesCondLim est utilisé pour modifier la largeur de bande par exemple
843 // en fonction des conditions limites linéaire
844 // lescontacts: si différent de NULL indique qu'il y a du contact
845 // NB: la dimension du tableau dépend du paramétrage lu dans ParaAlgoGlobal
846 Tableau <Mat_abstraite* > Choix_matriciel
847 (int nbddl,Tableau <Mat_abstraite* >& tab_matglob,LesMaillages * lesMail
848 ,LesReferences* lesRef,const Nb_assemb& nb_casAssemb,LesCondLim* lesCondLim
849 , LesContacts*lescontacts = NULL);
850
851 // Mise à jour éventuelle de la taille de la matrice de raideur ou masse du système linéaire
852 // en fonction du contact ou en fonction de "nouvelle_largeur_imposee" dans ce cas, lescontacts
853 // n'est pas utilisé ! donc la méthode peut servir dans ce cas sans le contact
854 //
855 // la matrice *matglob: doit déjà existée
856 // en retour le pointeur désigne toujours la même matrice qu'en entrée
857 // avec les même caractéristique de fonctionnement que l'ancienne (mêm résolution, stockage ...)
858 // NB: la dimension du tableau dépend du paramétrage lu dans ParaAlgoGlobal
859 // 1) si niveau_substitution =0 : -> mise à jour de toutes les matrices
860 // 2) si niveau_substitution = i : -> uniquement mise à jour de la matrices i
861 // par contre la dimension de tab_matglob est toujours mis à jour si c'est nécessaire
862 // si: nouvelle_largeur_imposee : n'est pas nulle,
863 // alors c'est directement cette valeur en noeud qui est imposée sur la matrice
864 // nouvelle_largeur_imposee.un = la largeur totale
865 // nouvelle_largeur_imposee.deux = la demie largeur
866 // nouvelles_largeur_en_ddl.trois = la demie largeur maximale pour la partie éléments finis
867 // uniquement (sans les CLL)
868 Tableau <Mat_abstraite* > Mise_a_jour_Choix_matriciel_contact
869 (Tableau <Mat_abstraite* >& tab_matglob,const Nb_assemb& nb_casAssemb
870 , LesContacts*lescontacts
871 ,int niveau_substitution ,TroisEntiers* nouvelle_largeur_imposee = NULL);
872
873 // choix de la matrice de la matrice de masse globale
874 // ou mise à jour de la matrice masse si elle existe
875 Mat_abstraite* Choix_matrice_masse
876 (int nbddl,Mat_abstraite* matglob,LesMaillages * lesMail,LesReferences* lesRef
877 ,const Nb_assemb& nb_casAssemb, LesContacts* lescontacts,LesCondLim* lesCondLim);
878

```

```

879 // mise à jour éventuelle du type et/ou de la taille de la matrice de masse
880 // retourne un pointeur sur la nouvelle matrice, s'il y a eu une modification
881 // l'ancienne est supprimée
882 // sinon retour d'un pointeur NULL
883 Mat_abstraite* Mise_a_jour_type_et_taille_matrice_masse_en_explicite
884 (int nbddl, Mat_abstraite* matglob, LesMaillages * lesMail, LesReferences* lesRef
885     , const Nb_assemb& nb_casAssemb, LesContacts* lescontacts);
886
887 // calcul la matrice de masse, y compris les masses ponctuelles si elles existent
888 // N_ddl : donne le ddl où doit être mis la masse
889 void Cal_matrice_masse(LesMaillages * lesMail, Assemblage& Ass, Mat_abstraite& matglob
890     , const DiversStockage* diversStockage, LesReferences* lesRef
891     , const Enum_ddl & N_ddl, LesFonctions_nD* lesFonctionsND);
892
893 // uniquement ajout dans la matrice de masse des masses ponctuelles si elles existent
894 // N_ddl : donne le ddl où doit être mis la masse
895 // par exemple, sert pour la relaxation dynamique
896 void Ajout_masses_ponctuelles(LesMaillages * lesMail, Assemblage& Ass, Mat_abstraite& matglob
897     , const DiversStockage* diversStockage, LesReferences* lesRef
898     , const Enum_ddl & N_ddl, LesFonctions_nD* lesFonctionsND);
899
900 // pilotage de la convergence, ceci en fonction du type de pilotage choisit
901 // ramène true si le pas de temps a été modifié, false sinon
902 // arret = true: indique dans le cas d'une non convergence, néanmoins rien n'a changé, il faut donc
arrêter
903 bool Pilotage_du_temps(Charge* charge, bool& arret);
904 // pilotage de la fin des itération en implicite : utilisation conjointe avec: Pilotage_du_temps()
905 // retourne true dans le cas normale
906 bool Pilotage_fin_iteration_implicite(int compteur);
907 // examine s'il y a convergence on pas en fonction des parametres
908 // de controle et du resultat du systeme lineaire
909 // affiche: indique si l'on veut l'affichage ou non des infos
910 // itera : correspond au compteur d'itération dans un cas d'équilibre itératif
911 // en sortie: ramene le maxi du residu et le pointeur d'assemblage correspondant
912 // arrêt: indique qu'il vaut mieux arrêter le calcul
913 bool Convergence(bool affiche, double last_var_ddl_max, Vecteur& residu, double maxPuissExt, double
maxPuissInt
914     , double maxReaction, int itera, bool& arret);
915 // pilotage spécifique de la fin de la relaxation dynamique, en tenant compte des différents
critères demandés
916 // relax_vit_acce :
917 // = 1 : il y a eu relaxation mais le calcul continue
918 // = 0 : pas de relaxation
919 // = -1 : on peut arrêter le calcul car le critère sur delta_ddl est satisfait
920 // compteur: indique le nombre d'itération en cours
921 // arretResidu: =true : le critère sur le résidu est ok
922 // iter_ou_incr : =1: il s'agit d'itération", =0 il s'agit d'incrément
923 // lit arret, puis modifie arret en true ou false = il faut arrêter ou non
924 // si true met à jour en interne certains paramètres de gestion de pilotage pour dire que l'on a
bien convergé
925 bool Pilotage_fin_relaxation_et_ou_residu(const int& relax_vit_acce, const int& iter_ou_incr,
926     const int& compteur, const bool& arretResidu, bool&
arrêt);
927
928 // pilotage à chaque itération:
929 // - sous ou sur relaxation
930 // - limitation de la norme de delta ddl
931 // - indication de précision pour les éléments (utilisée pour certain type de gestion
d'hourglass)
932 // en entrée: le maxi de var ddl juste après résolution
933 // ramène le maxi de var ddl maxDeltaDdl modifié, ainsi que sol modifié éventuellement,
934 void Pilotage_chaque_iteration(Vecteur* sol, double& maxDeltaDdl, const int& itera, LesMaillages *
lesMail);
935 // pilotage pour l'initialisation de Xtdt à chaque début d'incrément, en implicite
936 // avec la même direction qu'au pas précédent
937 // ramène true, si c'est ok pour initialiser Xtdt
938 bool Pilotage_init_Xtdt();
939
940 // pilotage en dynamique implicite, du maxi des déplacements et/ou vitesses
941 // limitation spécifiquement des maxi en déplacements et/ou vitesses
942 // il y a calcul éventuel (pas toujours) de delta_X, mais c'est considéré comme une variable de
travail
943 void Pilotage_maxi_X_V(const Vecteur& X_t, Vecteur& X_tdt, const Vecteur& V_t, Vecteur& V_tdt);
944
945 // calcul une approximation des différences énergies et puissances en jeux
946 // affichage éventuelle des ces énergies et du bilan
947 // V : ddl de vitesse
948 // coef_mass : en fait il faut utiliser 1/coef_mass * mat_mass pour la matrice masse, ceci due à la
spécificité de l'algo
949 // icharge : compteur d'incrément
950 // brestart : booleen qui indique si l'on est en restart ou pas
951 // gamma : vecteur accélération
952 void CalEnergieAffichage(const double& coef_mass, const Vecteur & V, const Mat_abstraite& mat_mass
953     , const Vecteur & delta_ddl, int icharge, bool brestart, const Vecteur & gamma
954     , const Vecteur & forces_vis_num);
955 // idem pour un calcul statique, c'est-à-dire sans énergie cinématique et puissance d'accélération
956 // affichage éventuelle des ces énergies et du bilan

```



```

957 // ichage : compteur d'increment
958 // brestart : booleen qui indique si l'on est en restart ou pas
959 void CalEnergieAffichage(const Vecteur & delta_ddl,int ichage,bool brestart,const Vecteur &
forces_vis_num);

960
961 // affichage eventuelle de la matrice de raideur et du second membre
962 void Affiche_RaidSM(const Vecteur & vglobin,const Mat_abstraite& matglob) const;
963
964 // methode d'application de l'amortissement cinetique
965 // retour un int :
966 // = 1 : il y a eu amortissement mais le calcul continue
967 // = 0 : pas d'amortissement
968 // = -1 : on peut arrêter le calcul car le critère sur delta_ddl est satisfait
969 int AmortissementCinetique(const Vecteur & delta_ddl,const double& coef_mass,const Vecteur& X
970 ,const Mat_abstraite& mat_mass,int ichage,Vecteur& V);
971 // initialisation des paramètres pour l'amortissement cinétique (au cas ou on veut plusieurs
amortissement)
972 void InitialiseAmortissementCinetique();
973
974 // définition de la matrice de viscosité numérique
975 // inita : true-> indique si l'on est dans une phase d'initialisation (creation de la matrice) ou
non
976 // dans ce dernier cas il y a modification eventuelle des valeurs de C
977 // ramène un pointeur sur la matrice visqueuse instancié ou nul si aucune instantiation
978 Mat_abstraite* Cal_mat_visqueux_num_expli(const Mat_abstraite& mat_mass,Mat_abstraite* mat_C_pt
979 ,const Vecteur & delta_X,bool inita, const Vecteur &
vitesse);
980
981 // calcul de l'impact d'une viscosité artificielle sur la raideur et sur le second membre
982 // l'algo travaille avec les vecteurs: delta_X, var_delta_X, et vitesse stockés dans Algori
983 // l'algo modifie mat_glob et calcul les forces visqueuses
984 void Cal_mat_visqueux_num_stat(Mat_abstraite& mat_glob,Vecteur& forces_vis_num);
985
986
987 // mise à jour qui sont gérées par la classe mère algo
988 // a priori que des choses génériques du type gestion des variables privées
989 void MiseAJourAlgoMere(ParaGlob * paraGlob,LesMaillages * lesMail,LesReferences* lesRef
990 ,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD
991 ,VariablesExporter* varExpor
992 ,LesLoisDeComp* lesLoisDeComp,DiversStockage* diversStockage
993 ,Charge* charge,LesCondLim* lesCondLim,LesContacts* lesContacts
994 ,Resultats* resultats);
995
996 // gestion eventuelle d'une renumérotation, qui prend en compte les éléments de contact
997 // premier_calcul : indique s'il s'agit d'un premier calcul on non
998 // nouvelle_situation_contact : indique s'il y a du nouveau sur le contact
999 // 1) si niveau_substitution = 0 : -> mise à jour de toutes les matrices
1000 // 2) si niveau_substitution = i : -> uniquement mise à jour de la matrices i
1001 // par contre la dimension de tab_matglob est toujours mis à jour si c'est nécessaire
1002 // ramène true si la matrice a été changée
1003 bool Gestion_stockage_et_renumérotation_avec_contact(bool premier_calcul
1004 ,LesMaillages * lesMail, bool & nouvelle_situation_contact
1005 ,LesCondLim* lesCondLim,LesReferences* lesRef
1006 ,Tableau <Mat_abstraite* >& tab_matglob,const Nb_assemb& nb_casAssemb
1007 ,LesContacts*lescontacts,int niveau_substitution);
1008
1009 // --- même chose mais sans le contact, par contre prise en compte d'un changement eventuelle
1010 // imposé par exemple par les CLL
1011 // gestion eventuelle d'une renumérotation des pointeurs d'assemblage , qui prend en compte les CLL
1012 // premier_calcul : indique s'il s'agit d'un premier calcul on non
1013 // nouvelle_situation_CLL : indique s'il y a du nouveau sur les CLL
1014 // -> la renumérotation des pointeurs s'effectue que si:
1015 // a) ParaAlgoControleActifs().Optimisation_pointeur_assemblage() est ok
1016 // b) soit c'est un premier calcul, soit il y a du nouveau sur les CLL
1017 //
1018 // 1) si niveau_substitution = 0 : -> mise à jour de toutes les matrices
1019 // 2) si niveau_substitution = i : -> uniquement mise à jour de la matrices i
1020 // par contre la dimension de tab_matglob est toujours mis à jour si c'est nécessaire
1021 // ramène true si la matrice a été changée
1022 // NB: lescontacts est quand même passé en paramètre, car on doit le renseigner au niveau d'un
1023 // tableau d'indice (num de noeuds) lorsque les num ont changés, ceci pour être opérationnel
1024 // par la suite si le contact devient actif
1025 bool Gestion_stockage_et_renumérotation_sans_contact(LesContacts* lescontacts,bool premier_calcul
1026 ,LesMaillages * lesMail, bool & nouvelle_situation_CLL
1027 ,LesCondLim* lesCondLim,LesReferences* lesRef
1028 ,Tableau <Mat_abstraite* >& tab_matglob,const Nb_assemb& nb_casAssemb
1029 ,int niveau_substitution);
1030
1031 private:
1032 // mise à jour de la viscosité critique en continu: spécifique au algo de relaxation dynamique,
appelé par Cal_mat_visqueux_num_expli
1033 void CalculEnContinuMatriceViscositeCritique(const Mat_abstraite& mat_mass,Mat_abstraite&
mat_C_pt
1034 ,const Vecteur & delta_X, const Vecteur
& vitesse);
1035 protected:
1036 // passage des grandeurs gérées par l'algorithme de tdt à t

```

```

1037 // en particulier les énergies et les puissances
1038 void TdtversT();
1039
1040 // mise à jour de delta_X, var_delta_X et passage en global des maxi
1041 void Cal_Transfert_delta_et_var_X(double& max_delta_X, double& max_var_delta_X);
1042
1043 // vérification d'une singularité éventuelle de la matrice de raideur de dim: nbddl
1044 // pour un problème de mécanique, en comparant le nombre de point d'intégration total
1045 // et le nombre totale de degré de liberté
1046 void VerifSingulariteRaideurMeca(int nbddl,const LesMaillages& lesMail) const;
1047
1048 // fonction virtuelle permettant de mettre à jour les infos aux noeuds
1049 // à cause de certains contact (ex: cas_contact = 4)
1050 // Les vecteurs Xtdt et Vtdt sont mis à jour par la méthode
1051 // la vitesse locale du noeud est mis à jour en fonction de la position locale et de l'algo
1052 virtual void Repercussion_algo_sur_cinematique(LesContacts* lesContacts,Vecteur& Xtdt,Vecteur&
Vtdt) {};
1053
1054 // retrouve le numéro de l'incrément sauvegardé trouvé dont le numéro est juste inf ou égale au
1055 // dernier incrément calculé - nb_incr_en_arriere.
1056 // ramène false si on n'a pas trouvé d'incrément a-doc
1057 // en retour ichearge prend la valeur de l'incrément trouvé (si tout c'est bien passé)
1058 bool Controle_retour_sur_un_incrément_enregistre(int nb_incr_en_arriere,int& ichearge);
1059
1060 // méthode interne d'application de l'amortissement cinétique, exploratoire
1061 // cas d'un amortissement local
1062 // retour un int :
1063 // = 1 : il y a eu amortissement mais le calcul continue
1064 // = 0 : pas d'amortissement
1065 // = -1 : on peut arrêter le calcul car le critère sur delta_ddl est satisfait
1066 int AmortissementCinétique_individuel_aux_noeuds(const Vecteur & delta_ddl,const double&
coef_mass,const Vecteur& X
, const Mat_abstraite& mat_mass,int ichearge,Vecteur& V);
1067
1068 // passage aux noeuds éventuellement des grandeurs globales, pour une sortie de post-traitement par
1069 // exemple mais pas seulement
1070 // le principe est que ce passage s'effectue si les conteneurs existent au niveau des noeuds
1071 void Passage_aux_noeuds_grandeurs_globales(LesMaillages * lesMail);
1072
1073 // passage aux noeuds de F_int_t et F_ext_t
1074 void Passage_aux_noeuds_F_int_t_et_F_ext_t(LesMaillages * lesMail);
1075
1076 // passage aux noeuds éventuellement d'une grandeur globale particulière,
1077 // typeGeneriqu : indique le type de grandeur à passer
1078 // seules les grandeurs globales qui n'ont pas été transférées par la méthode
1079 // Passage_aux_noeuds_grandeurs_globales , sont concernées
1080 // Le passage s'effectue si le conteneur existe au niveau des noeuds sinon erreur
1081 void Passage_aux_noeuds_grandeur_globale_particuliere(const TypeQuelconque& typeGeneriqu,
LesMaillages * lesMail);
1082
1083 // passage des grandeurs globales aux noeuds où il y a des variables globales attachées
1084 // nb_casAssemb correspond au cas d'assemblage de X1
1085 void Passage_de_grandeurs_globales_vers_noeuds_pour_variables_globales(LesMaillages *
lesMail,VariablesExporter* varExpor,const Nb_assemb& nb_casAssemb,const LesReferences& lesRef);
1086
1087 // des fonctions inlines pour mettre à jour des grandeurs globales
1088 // -- initialisation du compteur d'itérations
1089 void Transfert_ParaGlob_COMPTEUR_ITERATION_ALGO_GLOBAL(int compteur) const
1090 {void* pt_void =
ParaGlob::param->Mise_a_jour_grandeur_consultable(COMPTEUR_ITERATION_ALGO_GLOBAL);
TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
Grandeur_scalaire_entier& gr
= *((Grandeur_scalaire_entier*) pt_quelc->Grandeur_pointee()); // pour simplifier
*(gr.ConteneurEntier()) = compteur;
};
1091
1092 // --- cas de la norme de convergence
1093 void Transfert_ParaGlob_NORME_CONVERGENCE(double laNorme) const
1094 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(NORME_CONVERGENCE);
TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
Grandeur_scalaire_double& gr
= *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
*(gr.ConteneurDouble()) = laNorme;
};
1095
1096 // --- cas du compteur d'incrément
1097 void Transfert_ParaGlob_COMPTEUR_INCREMENT_CHARGE_ALGO_GLOBAL(int incre) const
1098 {void* pt_void =
ParaGlob::param->Mise_a_jour_grandeur_consultable(COMPTEUR_INCREMENT_CHARGE_ALGO_GLOBAL);
TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
Grandeur_scalaire_entier& gr
= *((Grandeur_scalaire_entier*) pt_quelc->Grandeur_pointee()); // pour simplifier
*(gr.ConteneurEntier()) = incre;
};
1099
1100 // --- cas de l'algorithme global actuellement en service
1101 void Transfert_ParaGlob_ALGO_GLOBAL_ACTUEL(EnumTypeCalcul type_algo) const
1102 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ALGO_GLOBAL_ACTUEL);
TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116

```

```

1117     Grandeur_scalaire_entier& gr
1118         = *((Grandeur_scalaire_entier*) pt_quelc->Grandeur_pointee()); // pour simplifier
1119     *(gr.ConteneurEntier()) = type_algo;
1120 };
1121 // --- cas des énergies internes venants des lois de comportement :
1122 // ENERGIE_ELASTIQUE, ENERGIE_PLASTIQUE, ENERGIE_VISQUEUSE,
1123 void Transfert_ParaGlob_energies_interneLoisComp() const
1124 {{void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_ELASTIQUE);
1125   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1126   Grandeur_scalaire_double& gr
1127       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1128   *(gr.ConteneurDouble()) = energTotal.EnergieElastique();
1129 }};
1130 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_PLASTIQUE);
1131   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1132   Grandeur_scalaire_double& gr
1133       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1134   *(gr.ConteneurDouble()) = energTotal.DissipationPlastique();
1135 }};
1136 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_VISQUEUSE);
1137   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1138   Grandeur_scalaire_double& gr
1139       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1140   *(gr.ConteneurDouble()) = energTotal.DissipationVisqueuse();
1141 }};
1142 };
1143 // --- cas des énergies de contact :
1144 // ENERGIE_PENALISATION, ENERGIE_FROT_ELAST, ENERGIE_FROT_PLAST, ENERGIE_FROT_VISQ
1145 void Transfert_ParaGlob_energies_contact() const
1146 {{void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_PENALISATION);
1147   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1148   Grandeur_scalaire_double& gr
1149       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1150   *(gr.ConteneurDouble()) = energPenalisation;
1151 }};
1152 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_FROT_ELAST);
1153   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1154   Grandeur_scalaire_double& gr
1155       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1156   *(gr.ConteneurDouble()) = energFrottement.EnergieElastique();
1157 }};
1158 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_FROT_PLAST);
1159   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1160   Grandeur_scalaire_double& gr
1161       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1162   *(gr.ConteneurDouble()) = energFrottement.DissipationPlastique();
1163 }};
1164 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_FROT_VISQ);
1165   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1166   Grandeur_scalaire_double& gr
1167       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1168   *(gr.ConteneurDouble()) = energFrottement.DissipationVisqueuse();
1169 }};
1170 };
1171 // --- cas des énergies non physiques, d'origine numérique :
1172 // ENERGIE_BULK_VISCOSITY, ENERGIE_HOURLASS_, stabilisation de membrane ou et biel
1173 void Transfert_ParaGlob_energies_hourglass_bulk_stab() const
1174 {{void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_BULK_VISCOSITY);
1175   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1176   Grandeur_scalaire_double& gr
1177       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1178   *(gr.ConteneurDouble()) = E_bulk;
1179 }};
1180 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_HOURLASS_);
1181   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1182   Grandeur_scalaire_double& gr
1183       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1184   *(gr.ConteneurDouble()) = energHourglass;
1185 }};
1186 {void* pt_void =
1187   ParaGlob::param->Mise_a_jour_grandeur_consultable(ENERGIE_STABILISATION_MEMB_BIEL);
1188   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1189   Grandeur_scalaire_double& gr
1190       = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour simplifier
1191   *(gr.ConteneurDouble()) = energStabiliMembBiel;
1192 }};
1193 };
1194 // --- cas des volumes entre plans :
1195 void Transfert_ParaGlob_volume_entre_plans() const
1196 // on met à jour les grandeurs que si on a fait la demande de calcul
1197 // sinon cela ne sert à rien car "vol_total2D_avec_plan_ref" n'est pas mis à jour par ailleurs
1198 if (!(pa.CalVolTotalEntreSurfaceEtPlansRef()))
1199     return;
1200 switch (ParaGlob::Dimension())
1201 { case 3:
1202   {void* pt_void =

```

```

ParaGlob::param->Mise_a_jour_grandeur_consultable(VOL_TOTAL2D_AVEC_PLAN_XY);
1203     TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1204     Grandeur_scalaire_double& gr
1205         = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour
    simplifier
1206         *(gr.ConteneurDouble()) = vol_total2D_avec_plan_ref(1)(3);
1207     };
1208     case 2:
1209     {void* pt_void =
ParaGlob::param->Mise_a_jour_grandeur_consultable(VOL_TOTAL2D_AVEC_PLAN_YZ);
1210     TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1211     Grandeur_scalaire_double& gr
1212         = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour
    simplifier
1213         *(gr.ConteneurDouble()) = vol_total2D_avec_plan_ref(1)(1);
1214     };
1215     case 1:
1216     {void* pt_void =
ParaGlob::param->Mise_a_jour_grandeur_consultable(VOL_TOTAL2D_AVEC_PLAN_XZ);
1217     TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1218     Grandeur_scalaire_double& gr
1219         = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour
    simplifier
1220         *(gr.ConteneurDouble()) = vol_total2D_avec_plan_ref(1)(2);
1221     };
1222     default: break;
1223 };
1224 // si on a plus d'un maillage on ajoute les autres valeurs
1225 int nbMailMax = vol_total2D_avec_plan_ref.Taille();
1226 if (nbMailMax > 1)
1227     {for (int nbMail=1;nbMail< nbMailMax+1;nbMail++)
1228     {
1229         switch (ParaGlob::Dimension())
1230         { case 3:
1231             {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable
1232                 ("vol_total2D_avec_plan_xy_"+ChangeEntierString(nbMail));
1233             TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1234             Grandeur_scalaire_double& gr
1235                 = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour
    simplifier
1236                 *(gr.ConteneurDouble()) = vol_total2D_avec_plan_ref(nbMail)(3);
1237             };
1238             case 2:
1239             {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable
1240                 ("vol_total2D_avec_plan_yz_"+ChangeEntierString(nbMail));
1241             TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1242             Grandeur_scalaire_double& gr
1243                 = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour
    simplifier
1244                 *(gr.ConteneurDouble()) = vol_total2D_avec_plan_ref(nbMail)(1);
1245             };
1246             case 1:
1247             {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable
1248                 ("vol_total2D_avec_plan_xz_"+ChangeEntierString(nbMail));
1249             TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
1250             Grandeur_scalaire_double& gr
1251                 = *((Grandeur_scalaire_double*) pt_quelc->Grandeur_pointee()); // pour
    simplifier
1252                 *(gr.ConteneurDouble()) = vol_total2D_avec_plan_ref(nbMail)(2);
1253             };
1254             default: break;
1255         };
1256     };
1257 };
1258 };
1259 };
1260 };
1261 };
1262 /// @} // end of group
1263
1264 #include "Charge.h"
1265 #include "Resultats.h"
1266
1267 #endif

```

## 7.2 AlgoriCombine.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.

```

```

9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28 //
29 /*****
30 *      DATE:      06/03/2023
31 *
32 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
33 *
34 *      PROJET:    Herezh++
35 *
36 *
37 *      BUT:      Algorithme combiné, l'objectif est de mettre en oeuvre
38 *               plusieurs algorithmes existants, suivant une stratégie
39 *               que l'utilisateur impose au travers de fonction nD
40 *               particulières.
41 *
42 *      *****
43 *
44 *      VERIFICATION:
45 *
46 *      ! date ! auteur ! but
47 *      -----
48 *      ! ! !
49 *
50 *      *****
51 *      MODIFICATIONS:
52 *      ! date ! auteur ! but
53 *      -----
54 *
55 *      $
56 *****/
57 #ifndef AGORICOMBINER_T
58 #define AGORICOMBINER_T
59
60 #include "Algori.h"
61 #include "Assemblage.h"
62 #include "MatLapack.h"
63
64 /// @addtogroup Les_algorithmes_de_resolutions_globales
65 /// @{
66 ///
67
68 /// BUT: Algorithme combiné, l'objectif est de mettre en oeuvre
69 /// plusieurs algorithmes existants, suivant une stratégie
70 /// que l'utilisateur impose au travers de fonction nD
71 /// particulières.
72
73 class AlgoriCombine : public Algori
74 {
75 public :
76 // CONSTRUCTEURS :
77 AlgoriCombine () ; // par défaut
78
79 // constructeur en fonction du type de calcul
80 // du sous type (pour les erreurs, remaillage etc...)
81 // il y a ici lecture des parametres attaches au type
82 AlgoriCombine (const bool avec_typeDeCal
83               ,const list <EnumSousTypeCalcul>& soustype
84               ,const list <bool>& avec_soustypeDeCal
85               ,UtilLecture& entreePrinc);
86 // constructeur de copie
87 AlgoriCombine (const AlgoriCombine& algo);
88
89 // constructeur de copie à partie d'une instance indifférenciée
90 Algori * New_idem(const Algori* algo) const
91 { // on vérifie qu'il s'agit bien d'une instance
92   if (algo->TypeDeCalcul() != COMBINER)
93     { cout << "\n *** erreur lors de la creation par copie d'un algo COMBINER "
94       << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
95       << " arret !! " << flush;

```

```

96     Sortie(1);
97     }
98     else
99     { AlgoriCombine* inter = (AlgoriCombine*) algo;
100     return ((Algori *) new AlgoriCombine(*inter));
101     };
102 };
103
104 // DESTRUCTEUR :
105 ~AlgoriCombine () ;
106
107 // METHODES PUBLIQUES :
108 //lecture des parametres de controle
109 // peut-être modifié dans le cas d'un algo particulier
110 virtual void Lecture(UtilLecture & entreePrinc,ParaGlob & paraGlob,LesMaillages& lesMail);
111
112 // execution de l'algorithme dans le cas non dynamique
113 void Execution(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbes1D* ,LesFonctions_nD*
114     ,VariablesExporter* varExpor,LesLoisDeComp*
115     ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* );
116 //----- décomposition en 3 du calcul d'équilibre -----
117 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
118 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
calcul
119 //          certaines variables ont-été changés
120
121 // initialisation
122 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbes1D*
123     ,LesFonctions_nD* ,VariablesExporter*
124     ,LesLoisDeComp*
125     ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* );
126
127 // mise à jour
128 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbes1D*
129     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
130     ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* );
131 // calcul de l'équilibre
132 // si tb_combiner est non null -> un tableau de 2 fonctions
133 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
134 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
135 //
136 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
137 // en fonction de la demande de sauvegard,
138 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
139 // qui lui fonctionnent indépendamment
140 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbes1D*
141     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
142     ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats*
143     ,Tableau < Fonction_nD* > * tb_combiner);
144 // dernière passe
145 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbes1D*
146     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
147     ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* );
148
149 // sortie du schemaXML: en fonction de enu
150 void SchemaXML_Algori(ofstream& ,const Enum_IO_XML ) const ;
151
152 // sortie des temps cpu cumulées de tous les algos
153 void AutreSortieTempsCPU(ofstream& sort,const int cas) const;
154
155 // sortie pour info de la liste des énumérés de tous les sous-algo
156 list <EnumTypeCalcul> List_Sous_Algo() const;
157
158 // sortie sur fichier des temps cpu
159 // idem la forme générique de la classe mère
160 // + des infos partielles pour chaque sous-algorithme
161 virtual void Sortie_temps_cpu(const LesCondLim& lesCondLim
162     , const Charge& charge,const LesContacts & contact);
163 // une méthode qui a pour objectif de terminer tous les comptages, utile
164 // dans le cas d'un arrêt impromptu
165 virtual void Arret_du_comptage_CPU();
166
167 private :
168 // variables privées
169 Tableau <Algori *> tab_algo; // pointeur sur les différents algos
170
171 // le type de calcul principal et sous type etc. sont stockés dans paraglob
172 // et des pointeurs définis dans Algori, permettent d'accéder à l'info sans
173 // la dupliquer
174
175 // mais il faut également stocker les infos particulières pour les algo internes
176 // du coup on crée des listes locales qui seront utilisées par pointeurs, via Algori
177 // pour chaque algo interne (sinon les grandeurs ne sont pas pérennes !!)
178 // l'association est faite dans AlgoriCombine::Lecture_algo_interne
179 // au moment de la définition des algos internes
180 // NB: important: les listes sont rangées : la première -> le dernier algo, etc.
181 list <list<EnumSousTypeCalcul> > list_sousTypeCalcul;

```

```

182     list <list<bool> > list_avec_sousCalcul;
183
184     ///-- choix de l'algo en cours ---
185     string nom_choix_algo; // le nom éventuel de la fonction du choix de l'algo
186     Fonction_nD* choix_algo; // la fonction du choix de l'algo si le pointeur est non nul
187     ///-- gestion de la sauvegarde de chaque algo
188     string nom_gestion_sauvegarde; // le nom éventuel de la fonction qui gère la sauvegarde
189     Fonction_nD* gestion_sauvegarde; // si non NULL, gestion de la sauvegarde de chaque algo
190     int nb_dernier_algo_qui_a_fait_une_sauvegarde; //
191     ///-- gestion de la sortie à convergence de chaque algo
192     string nom_gestion_sortie_a_convergence; // le nom éventuel de la fonction qui gère la sortie
193     Fonction_nD* gestion_sortie_a_convergence; // si non NULL, gestion de la sortie de chaque algo
194
195     Tableau < Fonction_nD* > * tb_combiner; // le tableau des 3 fctnD de gestion que l'on passe aux algo
196
197     // deux fonctions par défaut qui servent pour tb_combiner
198     Fonction_nD* cst_0; // une fonction Cst qui retourne 0
199     Fonction_nD* cst_1; // une fonction Cst qui retourne 1
200
201     // METHODES PROTEGEES :
202     // // sauvegarde sur base info
203     // // cas donne le niveau de sauvegarde
204     // // = 0 : initialisation de la sauvegarde -> c'est-à-dire de la sortie base info
205     // // = 1 : on sauvegarde tout
206     // // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
207     // // incre : numero d'incrément auquel on sauvegarde
208     // // éventuellement est définit de manière spécifique pour chaque algorithme
209     // // dans les classes filles
210     // virtual void Ecriture_base_info
211     //         (ofstream& sort,const int cas);
212     // // cas de la lecture spécifique à l'algorithme dans base_info
213     // virtual void Lecture_base_info(ifstream& ent,const int cas);
214
215     // lecture des paramètres du calcul
216     void lecture_Paramètres(UtilLecture& entreePrinc);
217
218     void Creation_fct_cst(); // création de cst_0 et cst_1
219
220     // écriture des paramètres dans la base info
221     // // = 1 : on écrit tout
222     // // = 2 : on écrit uniquement les données variables (supposées comme telles)
223     void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
224     // lecture des paramètres dans la base info
225     // // = 1 : on récupère tout
226     // // = 2 : on récupère uniquement les données variables (supposées comme telles)
227     // choix = true : fonctionnement normal
228     // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
229     // // car la lecture est impossible
230     void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
231     // création d'un fichier de commande: cas des paramètres spécifiques
232     void Info_commande_parametres(UtilLecture& entreePrinc);
233
234     // ----- concernant les algo internes -----
235     // lecture des algorithmes internes
236     void Lecture_algo_interne(const bool avec_typeDeCal
237                             ,const list <EnumSousTypeCalcul>& soustype
238                             ,const list <bool>& avec_soustypeDeCal
239                             ,UtilLecture& entreePrinc);
240
241     //---- gestion des commndes interactives -----
242     // écoute et prise en compte d'une commande interactive
243     // ramène true tant qu'il y a des commandes en cours
244     bool ActionInteractiveAlgo();
245
246 };
247 /// @} // end of group
248
249 #endif

```

## 7.3 Algoinformations.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //

```

```

14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           05/01/2000
31 *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:        Herezh++
35 *
36 *   *****
37 *   BUT:          Calcul d'informations générales types, géométriques
38 *                par exemple, ou de visualisation.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date ! auteur ! but
44 *   -----
45 *   ! ! !
46 *   *****
47 *
48 *   MODIFICATIONS:
49 *   ! date ! auteur ! but
50 *   -----
51 *
52 *   *****/
53 #ifndef ALGOINFORMATION_T
54 #define ALGOINFORMATION_T
55
56
57 #include "Algori.h"
58 #include "MatBand.h"
59 #include "Assemblage.h"
60
61
62 /// @addtogroup Les_algorithmes_de_resolutions_globales
63 /// @{
64 ///
65
66 /// BUT: Calcul d'informations générales types, géométriques
67 /// par exemple, ou de visualisation.
68
69 class AlgoInformations : public Algori
70 {
71 public :
72 // CONSTRUCTEURS :
73 AlgoInformations () ; // par défaut
74
75 // constructeur en fonction du type de calcul
76 // du sous type (pour les erreurs, remaillage etc..)
77 // il y a ici lecture des parametres attaches au type
78 AlgoInformations (const bool avec_typeDeCal
79 ,const list <EnumSousTypeCalcul>& soustype
80 ,const list <bool>& avec_soustypeDeCal
81 ,UtilLecture& entreePrinc);
82 // constructeur de copie
83 AlgoInformations (const AlgoInformations& algo);
84
85 // constructeur de copie à partie d'une instance indifférenciée
86 Algori * New_idem(const Algori* algo) const
87 { // on vérifie qu'il s'agit bien d'une instance
88 if (algo->TypeDeCalcul() != INFORMATIONS)
89 { cout << "\n *** erreur lors de la creation par copie d'un algo INFORMATIONS "
90 << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
91 << " arret !! " << flush;
92 Sortie(1);
93 }
94 else
95 { AlgoInformations* inter = (AlgoInformations*) algo;
96 return ((Algori *) new AlgoInformations(*inter));
97 };
98 };
99

```



```

100 // DESTRUCTEUR :
101 ~AlgoInformations () ;
102
103 // METHODES PUBLIQUES :
104 // execution de l'algorithme
105 void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D* ,LesFonctions_nD*
106 ,VariablesExporter* varExpor ,LesLoisDeComp* ,DiversStockage*
107 ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
108 //----- décomposition en 3 du calcul d'équilibre -----
109 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
110 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
calculs
111 // certaines variables ont-été changés
112
113 // pas opérationnel pour l'instant !
114
115 // initialisation
116 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
117 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
118 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
119 // mise à jour
120 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
121 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
122 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* )
123 {cout << "\n MiseAJourAlgo(.. pas encore operationnelle ";
124 Sortie(1);
125 };
126 // calcul de l'équilibre
127 // si tb_combiner est non null -> un tableau de 2 fonctions
128 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
129 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
130 //
131 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
132 // en fonction de la demande de sauvegard,
133 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
134 // qui lui fonctionne indépendamment
135 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
136 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
137 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
138 ,Tableau < Fonction_nD* > * tb_combiner){};
139 // dernière passe
140 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
141 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
142 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
143
144 // sortie du schemaXML: en fonction de enu
145 void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const {};
146
147
148 private :
149
150 // METHODES PROTEGEES :
151 // --- venant du virtuel ---
152 // écriture des paramètres dans la base info (ici rien)
153 void Ecrit_Base_info_Parametre(UtilLecture& ,const int& ) {};
154 // lecture des paramètres dans la base info (ici rien)
155 // choix = true : fonctionnement normal
156 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
157 // car la lecture est impossible
158 void Lecture_Base_info_Parametre(UtilLecture& ,const int& ,bool ) {};
159 // création d'un fichier de commande: cas des paramètres spécifiques
160 void Info_commande_parametres(UtilLecture& ) {Algori::Info_com_parametres(*entreePrinc)};
161 };
162 /// @} // end of group
163
164 #endif

```

## 7.4 AlgoriRemontErreur.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by

```

```

16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      23/01/97
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 * *****/
37 *   BUT:      Algorithme de calcul de la remontée des contraintes aux
38 *            noeuds, puis de calcul d'erreur, après un calcul de mécanique.
39 *
40 *            *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !           but
45 *   -----
46 *   !       !           !
47 *
48 *   MODIFICATIONS:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 * *****/
54 #ifndef AGORIREMONTERREUR_T
55 #define AGORIREMONTERREUR_T
56
57 #include "Algori.h"
58 #include "MatBand.h"
59 #include "Assemblage.h"
60
61 /// @addtogroup Les_algorithmes_de_resolutions_globales
62 /// @{
63 ///
64
65 ///   BUT:      Algorithme de calcul de la remontée des contraintes aux
66 ///   noeuds, puis de calcul d'erreur, après un calcul de mécanique.
67
68 class AlgoriRemontErreur : public Algori
69 {
70 public :
71     // CONSTRUCTEURS :
72     AlgoriRemontErreur () ; // par défaut
73
74     // constructeur en fonction du type de calcul
75     // il y a ici lecture des parametres attaches au type
76     AlgoriRemontErreur (EnumTypeCalcul type,Utilecture& entreePrinc);
77     // DESTRUCTEUR :
78     ~AlgoriRemontErreur () ;
79
80     // METHODES PUBLIQUES :
81     // execution de l'algorithme dans le cas non dynamique
82     void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID* ,LesFonctions_nD*
83                 ,VariablesExporter* ,LesLoisDeComp*
84                 ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* );
85
86 private :
87     // VARIABLES PROTEGEES :
88
89     // METHODES PROTEGEES :
90
91 };
92 /// @} // end of group
93
94 #endif

```

## 7.5 AlgoUmatAbaqus.h

```

1 // This file is part of the Herezh++ application.

```

```

2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           24/02/2005
31 *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:        Herezh++
35 *
36 *   ****
37 *   BUT:           Algorithmme de calcul permettant l'utilisation d'herezh++
38 *                  comme Umat pour Abaqus
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date ! auteur ! but
44 *   -----
45 *   ! ! !
46 *   $
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date ! auteur ! but
50 *   -----
51 *   $
52 *   *****/
53 #ifndef AGORI_UMAT_ABAQUS_T
54 #define AGORI_UMAT_ABAQUS_T
55
56
57 #include "Algori.h"
58 #include "Assemblage.h"
59
60
61 /// @addtogroup Les_algorithmes_de_resolutions_globales
62 /// @{
63 ///
64
65 /// BUT:   Algorithmme de calcul permettant l'utilisation d'herezh++
66 ///       comme Umat pour Abaqus
67
68 class AlgoUmatAbaqus : public Algori
69 {
70 public :
71     // CONSTRUCTEURS :
72     AlgoUmatAbaqus () ; // par default
73
74     // constructeur en fonction du type de calcul
75     // du sous type (pour les erreurs, remaillage etc...)
76     // il y a ici lecture des parametres attaches au type
77     AlgoUmatAbaqus (const bool avec_typeDeCal
78                   ,const list <EnumSousTypeCalcul>& soustype
79                   ,const list <bool>& avec_soustypeDeCal
80                   ,UtilLecture& entreePrinc);
81     // constructeur de copie
82     AlgoUmatAbaqus (const AlgoUmatAbaqus& algo);
83
84     // constructeur de copie à partie d'une instance indifférenciée
85     Algori * New_idem(const Algori* algo) const
86     { // on vérifie qu'il s'agit bien d'une instance
87     if (algo->TypeDeCalcul() != UMAT_ABAQUS)

```

```

88     { cout << "\n *** erreur lors de la creation par copie d'un algo UMAT_ABAQUS "
89         << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
90         << " arret !! " << flush;
91     }
92     Sortie(1);
93 }
94 else
95 { AlgoUmatAbaqus* inter = (AlgoUmatAbaqus*) algo;
96   return ((Algori *) new AlgoUmatAbaqus(*inter));
97 };
98 };
99 // DESTRUCTEUR :
100 ~AlgoUmatAbaqus () ;
101
102 // METHODES PUBLIQUES :
103 // execution de l'algorithme dans le cas non dynamique
104 void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID* ,LesFonctions_nD*
105               ,VariablesExporter* varExpor,LesLoisDeComp*
106               ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
107
108 //----- décomposition en 3 du calcul d'équilibre -----
109 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
110 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre
111 // deux calculs
112 //          certaines variables ont-été changés
113
114 // pas opérationnel pour l'instant !
115
116 // initialisation
117 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
118                   ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
119                   ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
120
121 // mise à jour
122 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
123                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
124                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* )
125 {cout << "\n MiseAJourAlgo(.. pas encore operationnelle ";
126   Sortie(1);
127 };
128 // calcul de l'équilibre
129 // si tb_combiner est non null -> un tableau de 2 fonctions
130 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
131 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
132 //
133 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
134 // en fonction de la demande de sauvegard,
135 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
136 // qui lui fonctionne indépendamment
137 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
138                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
139                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
140                  ,Tableau < Fonction_nD* > * tb_combiner ){};
141
142 // dernière passe
143 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
144               ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
145               ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
146
147 // sortie du schemaXML: en fonction de enu
148 void SchemaXML_Algori(ofstream& ,const Enum_IO_XML ) const {};
149
150 private :
151 // VARIABLES PROTEGEES :
152
153 // METHODES PROTEGEES :
154
155 // Calcul de l'équilibre de la pièce
156 void Calcul_Umat
157 (ParaGlob * paraGlob,LesMaillages * lesMail,
158  LesReferences* lesRef,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD
159  ,LesLoisDeComp* lesLoisDeComp,
160  DiversStockage* diversStockage,Charge* charge,
161  LesCondLim* lesCondLim,LesContacts* lesContacts,Resultats* resultats);
162
163 // écriture des paramètres dans la base info
164 // = 1 : on écrit tout
165 // = 2 : on écrit uniquement les données variables (supposées comme telles)
166 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
167 // lecture des paramètres dans la base info
168 // = 1 : on récupère tout
169 // = 2 : on récupère uniquement les données variables (supposées comme telles)
170 // choix = true : fonctionnement normal
171 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
172 // défaut
173 //          car la lecture est impossible
174 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
175 // création d'un fichier de commande: cas des paramètres spécifiques
176 void Info_commande_parametres(UtilLecture& entreePrinc);
177

```

```

173 };
174 ///< @} // end of group
175
176 #endif

```

## 7.6 AlgoUtils.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *      DATE:      14/03/2003
31 *
32 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *      PROJET:    Herezh++
35 *
36 *      *****
37 *      BUT:      Utilitaires divers.
38 *                1) transformation de maillage, quadratique incomplet
39 *                en quadratique complet.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *
45 *      ! date ! auteur ! but
46 *      -----
47 *      ! ! !
48 *
49 *      *****
50 *      MODIFICATIONS:
51 *      ! date ! auteur ! but
52 *      -----
53 *
54 *      *****/
55 #ifndef ALGOUTILS_T
56 #define ALGOUTILS_T
57
58
59 #include "Algori.h"
60 #include "MatBand.h"
61 #include "Assemblage.h"
62
63
64 ///< @addtogroup Les_algorithmes_de_resolutions_globales
65 ///< @{
66 ///<
67
68 ///< BUT: Utilitaires divers.
69 ///< ex: transformation de maillage, quadratique incomplet
70 ///< en quadratique complet.
71
72 class AlgoUtils : public Algori
73 {
74 public :
75 // CONSTRUCTEURS :
76 AlgoUtils () ; // par défaut
77
78 // constructeur en fonction du type de calcul

```

```

79 // du sous type (pour les erreurs, remaillage etc...)
80 // il y a ici lecture des parametres attaches au type
81 AlgoUtils (const bool avec_typeDeCal
82           ,const list <EnumSousTypeCalcul>& soustype
83           ,const list <bool>& avec_soustypeDeCal
84           ,UtilLecture& entreePrinc);
85
86 // constructeur de copie
87 AlgoUtils (const AlgoUtils& algo);
88
89 // constructeur de copie à partie d'une instance indifférenciée
90 Algori * New_idem(const Algori* algo) const
91 { // on vérifie qu'il s'agit bien d'une instance
92   if (algo->TypeDeCalcul() != UTILITAIRES)
93     { cout << "\n *** erreur lors de la creation par copie d'un algo UTILITAIRES "
94       << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
95       << " arret !! " << flush;
96       Sortie(1);
97     }
98   else
99     { AlgoUtils* inter = (AlgoUtils*) algo;
100      return ((Algori *) new AlgoUtils(*inter));
101    };
102  };
103
104 // DESTRUCTEUR :
105 ~AlgoUtils () ;
106
107 // METHODES PUBLIQUES :
108 // execution de l'algorithme
109 void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D* ,LesFonctions_nD*
110              ,VariablesExporter* varExpor ,LesLoisDeComp*
111              ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
112
113 //----- décomposition en 3 du calcul d'équilibre -----
114 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
115 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
116 // calculs
117 // certaines variables ont-été changés
118
119 // pas opérationnel pour l'instant !
120
121 // initialisation
122 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
123                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
124                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
125
126 // mise à jour
127 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
128                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
129                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* )
130 { cout << "\n MiseAJourAlgo(.. pas encore operationnelle ";
131   Sortie(1);
132 };
133
134 // calcul de l'équilibre
135 // si tb_combiner est non null -> un tableau de 2 fonctions
136 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
137 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
138 //
139 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
140 // en fonction de la demande de sauvegard,
141 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
142 // qui lui fonctionne indépendamment
143 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
144                 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
145                 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
146                 ,Tableau < Fonction_nD* > * tb_combiner) {};
147
148 // dernière passe
149 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
150               ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
151               ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
152
153 // sortie du schemaXML: en fonction de enu
154 void SchemaXML_Algori(ofstream& ,const Enum_IO_XML ) const {};
155
156 private :
157
158 // METHODES PROTEGEES :
159 // --- venant du virtuel ---
160 // écriture des paramètres dans la base info (ici rien)
161 void Ecrit_Base_info_Parametre(UtilLecture& ,const int& ) {};
162 // lecture des paramètres dans la base info (ici rien)
163 // choix = true : fonctionnement normal
164 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
165 // défaut
166 // car la lecture est impossible
167 void Lecture_Base_info_Parametre(UtilLecture& ,const int& ,bool) {};
168 // création d'un fichier de commande: cas des paramètres spécifiques
169 void Info_commande_parametres(UtilLecture& ) {Algori::Info_com_parametres(*entreePrinc);};

```

```

164 };
165 /// @} // end of group
166
167 #endif

```

## 7.7 Algori\_chung\_lee.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           24/10/2001
31 *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:         Herezh++
35 *
36 *   BUT:            Algorithme de calcul dynamique explicite, pour de la
37 *                   mecanique du solide déformable en coordonnees materielles*
38 *                   entrainees. On ne prend pas en compte les phénomènes de
39 *                   contact. Ici il s'agit de l'algorithme proposé par
40 *                   Chung Lee
41 *
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *   ! date ! auteur ! but
47 *   -----
48 *   ! ! !
49 *
50 *   *****
51 *
52 *   MODIFICATIONS:
53 *   ! date ! auteur ! but
54 *   -----
55 *
56 #ifndef AGORI_CHUNG_LEE_T
57 #define AGORI_CHUNG_LEE_T
58
59
60 #include "Algori.h"
61 #include "Assemblage.h"
62
63
64 /// @addtogroup Les_algorithmes_de_resolutions_globales
65 /// @{
66 ///
67
68 /// BUT: Algorithme de calcul dynamique explicite, pour de la
69 /// mecanique du solide déformable en coordonnees materielles
70 /// entrainees. On ne prend pas en compte les phénomènes de
71 /// contact. Ici il s'agit de l'algorithme proposé par
72 /// Chung Lee
73
74 class Algori_chung_lee : public Algori
75 {
76 public :
77     // CONSTRUCTEURS :

```

```

78     Algori_chung_lee () ; // par default
79
80     // constructeur en fonction du type de calcul
81     // du sous type (pour les erreurs, remailage etc...)
82     // il y a ici lecture des parametres attaches au type
83     Algori_chung_lee (const bool avec_typeDeCal
84         ,const list <EnumSousTypeCalcul>& soustype
85         ,const list <bool>& avec_soustypeDeCal
86         ,UtilLecture& entreePrinc);
87     // constructeur de copie
88     Algori_chung_lee (const Algori_chung_lee& algo);
89
90     // constructeur de copie à partie d'une instance indifférenciée
91     Algori * New_idem(const Algori* algo) const
92     { // on vérifie qu'il s'agit bien d'une instance
93         if (algo->TypeDeCalcul() != DYNA_EXP_CHUNG_LEE)
94             { cout << "\n *** erreur lors de la creation par copie d'un algo DYNA_EXP_CHUNG_LEE "
95                 << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
96                 << " arret !! " << flush;
97                 Sortie(1);
98             }
99         else
100             { Algori_chung_lee* inter = (Algori_chung_lee*) algo;
101               return ((Algori *) new Algori_chung_lee(*inter));
102             };
103         };
104
105     // DESTRUCTEUR :
106     ~Algori_chung_lee () ;
107
108     // METHODES PUBLIQUES :
109     // execution de l'algorithme explicite dans le cas dynamique sans contact
110     void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D* ,LesFonctions_nD*
111         ,VariablesExporter* varExpor,LesLoisDeComp*
112         ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
113
114     //----- décomposition en 3 du calcul d'équilibre -----
115     // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
116     // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
117     // calcul
118     // certaines variables ont-été changés
119
120     // initialisation
121     void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
122         ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
123         ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
124
125     // mise à jour
126     void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
127         ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
128         ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
129
130     // calcul de l'équilibre
131     // si tb_combiner est non null -> un tableau de 2 fonctions
132     // - la première fct dit si on doit valider ou non le calcul à convergence ok,
133     // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
134     // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
135     // en fonction de la demande de sauvegard,
136     // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
137     // qui lui fonctionne indépendamment
138     void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
139         ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
140         ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
141         ,Tableau < Fonction_nD* > * tb_combiner);
142
143     // dernière passe
144     void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
145         ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
146         ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
147
148     // sortie du schemaXML: en fonction de enu
149     void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const {};
150
151     protected :
152     // VARIABLES PROTEGEES :
153     // paramètre de la méthode de Chung lee
154     double* beta_cl;
155
156     // liste de variables de travail déclarées ici pour éviter le passage de paramètre entre les
157     // méthodes internes à la classe
158     double delta_t,deltat2,unsurdeltat,deltatSurDeux;
159     double betachapeau,gammaa,gammachapeau;
160
161     // -----
162     // -- variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
163     // -----
164
165     // ==> pointeurs d'instance et classe particulières

```



```

164   Assemblage * Ass1_, * Ass2_, * Ass3_; // pointeurs d'assemblages
165
166   // === variables scalaires
167   int cas_combi_ddl; // def combinaison des ddl
168   int icas; // idem cas_combi_ddl mais pour lesCondlim
169   bool prepa_avec_remont; // comme son nom l'indique
170   bool brestart; // booleen qui indique si l'on est en restart ou pas
171   OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
172
173   // === vecteurs
174   Vecteur vglobin; // puissance interne : pour ddl accélération
175   Vecteur vglobex; // puissance externe
176   Vecteur vglobaal; // puissance totale
177   Vecteur vcontact; // puissance des forces de contact
178   Vecteur acceleration; // vecteur intermédiaire
179   Vecteur X_Bl,V_Bl,G_Bl; // stockage transitoirement des X V GAMMA <-> CL
180   Vecteur forces_vis_num; // forces visqueuses d'origines numériques
181   // === les listes
182   list <LesCondlim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués
183   // === les tableaux
184   Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
185   Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
186   // === les matrices
187   Mat_abstraite* mat_masse,* mat_masse_sauve; // choix de la matrice de masse
188   Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
189
190
191
192   // METHODES PROTEGEES :
193   void Calcul_Equilibre
194       (ParaGlob * paraGlob,LesMaillages * lesMail,
195        LesReferences* lesRef,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD
196        ,LesLoisDeComp* lesLoisDeComp,
197        DiversStockage* diversStockage,Charge* charge,
198        LesCondlim* lesCondlim,LesContacts* lesContacts,Resultats* resultats);
199
200
201
202   //---- gestion des commndes interactives -----
203   // écoute et prise en compte d'une commande interactive
204   // ramène true tant qu'il y a des commandes en cours
205   bool ActionInteractiveAlgo();
206
207   // lecture des paramètres du calcul
208   void lecture_Parametres(UtilLecture& entreePrinc);
209   // écriture des paramètres dans la base info
210   // = 1 : on écrit tout
211   // = 2 : on écrit uniquement les données variables (supposées comme telles)
212   void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
213   // lecture des paramètres dans la base info
214   // = 1 : on récupère tout
215   // = 2 : on récupère uniquement les données variables (supposées comme telles)
216   // choix = true : fonctionnement normal
217   // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
218   // car la lecture est impossible
219   void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
220   // création d'un fichier de commande: cas des paramètres spécifiques
221   void Info_commande_parametres(UtilLecture& entreePrinc);
222   // gestion et vérification du pas de temps et modif en conséquence si nécessaire
223   // cas = 1: initialisation du pas de temps et vérif / au pas de temps critique
224   // ceci pour le temps t=0
225   // cas = 2: initialisation du pas de temps et vérif / au pas de temps critique
226   // ceci pour le temps t
227   // en entrée: modif_pas_de_temps: indique qu'il y a eu par ailleurs (via Charge->Avance())
228   // une modification du pas de temps depuis le dernier appel
229   // retourne vrai s'il y a une modification du pas de temps, faux sinon
230   bool Gestion_pas_de_temps(bool modif_pas_de_temps,LesMaillages * lesMail,int cas);
231 };
232 /// @} // end of group
233
234 #endif

```

## 7.8 Algori\_relax\_dyna.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)

```

```

11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           3/09/99
31 *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:        Herezh++
35 *
36 *   ****
37 *   BUT:           Algorithmme de relaxation dynamique selon la méthode
38 *                  de Barnes, modifiée par Julien Troufflard puis
39 *                  généralisée.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date ! auteur ! but
46 *   -----
47 *   ! ! !
48 *   *****
49 *
50 *   MODIFICATIONS:
51 *
52 *   ! date ! auteur ! but
53 *   -----
54 *   *****/
54 #ifndef AGORI_RELAX_DYNA_H
55 #define AGORI_RELAX_DYNA_H
56
57
58 #include "Algori.h"
59 #include "Assemblage.h"
60 #include "Ponderation.h"
61
62
63 /// @addtogroup Les_algorithmes_de_resolutions_globales
64 /// @{
65 ///
66
67 /// BUT: Algorithmme de relaxation dynamique selon la méthode
68 /// de Barnes, modifiée par Julien Troufflard puis
69 /// généralisée.
70
71 class AlgoriRelaxDyna : public Algori
72 {
73 public :
74 // CONSTRUCTEURS :
75 AlgoriRelaxDyna () ; // par défaut
76
77 // constructeur en fonction du type de calcul
78 // du sous type (pour les erreurs, remaillage etc...)
79 // il y a ici lecture des parametres attaches au type
80 AlgoriRelaxDyna (const bool avec_typeDeCal
81                 ,const list <EnumSousTypeCalcul>& soustype
82                 ,const list <bool>& avec_soustypeDeCal
83                 ,UtilLecture& entreePrinc);
84 // constructeur de copie
85 AlgoriRelaxDyna (const AlgoriRelaxDyna& algo);
86
87 // constructeur de copie à partie d'une instance indifférenciée
88 Algori * New_idem(const Algori* algo) const
89 { // on vérifie qu'il s'agit bien d'une instance
90   if (algo->TypeDeCalcul() != RELAX_DYNA)
91     { cout << "\n *** erreur lors de la creation par copie d'un algo RELAX_DYNA "
92       << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
93       << " arret !! " << flush;
94     }
95   Sortie(1);
96 }
97 else

```

```

97     { AlgoriRelaxDyna* inter = (AlgoriRelaxDyna*) algo;
98     return ((Algori *) new AlgoriRelaxDyna(*inter));
99     };
100 };
101
102 // DESTRUCTEUR :
103 ~AlgoriRelaxDyna () ;
104
105 // METHODES PUBLIQUES :
106 // execution de l'algorithme explicite dans le cas dynamique sans contact
107 void Execution(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID* , LesFonctions_nD*
108               , VariablesExporter* varExpor, LesLoisDeComp*
109               , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats* );
110
111 //----- décomposition en 3 du calcul d'équilibre -----
112 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
113 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
calcul
114 //          certaines variables ont-été changés
115
116 // initialisation
117 void InitAlgorithme(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID*
118                   , LesFonctions_nD* , VariablesExporter* , LesLoisDeComp*
119                   , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats* );
120
121 // mise à jour
122 void MiseAJourAlgo(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID*
123                  , LesFonctions_nD* , VariablesExporter* , LesLoisDeComp*
124                  , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats* );
125
126 // calcul de l'équilibre
127 // si tb_combiner est non null -> un tableau de 2 fonctions
128 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
129 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
130 //
131 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
132 // en fonction de la demande de sauvegard,
133 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
134 // qui lui fonctionne indépendamment
135 void CalEquilibre(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID*
136                  , LesFonctions_nD* , VariablesExporter* , LesLoisDeComp*
137                  , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats*
138                  , Tableau < Fonction_nD* > * tb_combiner);
139
140 // dernière passe
141 void FinCalcul(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID*
142               , LesFonctions_nD* , VariablesExporter* , LesLoisDeComp*
143               , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats* );
144
145 // sortie du schemaXML: en fonction de enu
146 void SchemaXML_Algori(ofstream& sort, const Enum_IO_XML enu) const;
147
148 protected :
149 // VARIABLES PROTEGEES :
150
151 // indicateur disant s'il faut calculer les conditions limites à chaque itération on non
152 int cL_a_chaque_iteration; // par défaut, non, == uniquement à chaque début d'incrément
153
154 // liste de variables de travail déclarées ici pour éviter le passage de paramètre entre les
155 // méthodes internes à la classe
156 double delta_t;
157
158 int typeCalRelaxation ; // type de calcul de relaxation
159
160 // ---para de contrôle du calcul de la masse
161 double lambda; // le lambda effectif qui peut évoluer pendant le calcul
162 double lambda_initial; // lambda initiale lue
163 // a) contrôle éventuel de lambda via une ou plusieurs grandeurs globales
164 Ponderation_GGlobal* niveauLambda_grandeurGlobale;
165 // b) via éventuellement un ddl étendu
166 Ponderation * niveauLambda_ddlEtendu;
167 // c) via éventuellement le temps
168 Ponderation_temps * niveauLambda_temps;
169 // pilotage automatique éventuel
170 double lambda_min, lambda_max, delta_lambda;
171 bool pilotage_auto_lambda;
172 List_io<int> list_iter_relax; // sauvegarde des iters à suivre
173
174 int casMass_relax; // indique le type de calcul de la matrice masse
175 Tableau <int> option_recalcul_masse; // indique le type de recalcul de la masse
176 // par défaut dim = 1, si dim = 2, le premier est pour le cinétique le second
177 // pour le visqueux
178 Tableau <Fonction_nD* > fct_nD_option_recalcul_masse; // si non NULL: donne une
179 // valeur qui remplace option_recalcul_masse
180 Tableau <string > nom_fct_nD_option_recalcul_masse; // nom éventuel de la fonction associée
181
182 // --- pour typeCalRelaxation = 1
183 // .... pour la première version
184 double alpha,beta; // para de répartition entre K et mu

```

```

183 double gamma; // para multiplicatif du rôle de la trace du tenseur de contrainte
184 double theta; // para multiplicatif du rôle de la contrainte de mises
185
186 // --- pour typeCalRelaxation = 2
187 int ncycle_calcul; int type_calcul_masse;
188 double fac_epsilon; // facteur pour le test du recalcul de la matrice masse
189
190 // -- pour typeCalRelaxation = 4 c-a-d mixte: amortissement cinétique au début puis visqueux à la
191 // fin
192 double proportion_cinetique; // indique la proportion de la précision qui est faite en amortissement
193 // cinétique
194 bool visqueux_activer; // permet d'éviter de revenir en arrière
195 int et_recalcul_masse_a_la_transition; // indique si on recalcule la masse à la transition cinétique
196 // visqueux
197 Ponderation_GGlobal* niveauF_grandeurGlobale;
198 // b) via éventuellement un ddl étendu
199 Ponderation * niveauF_ddlEtendu;
200 // c) via éventuellement le temps
201 Ponderation_temps* niveauF_temps;
202
203 // === parametre pour le contact:
204 int type_activation_contact; // indique quand le contact doit-être activé
205 // 0: par défaut la recherche de nouveaux contacts est activée à la fin de chaque incrément
206 // d'une manière analogue au cas d'un algo implicite classique
207 // 1: la recherche de nouveaux contacts est activée après chaque itération
208 // 2: la recherche de nouveaux contacts est activée après chaque pic d'énergie et à la fin
209 // de chaque incrément. Si l'amortissement n'est pas cinétique, c'est équivalent au cas 0
210
211 // permet de choisir le type de remplacement des masses nulles
212 // = 0 : on ne change rien, on laisse les 0
213 // = 1 : on choisit la valeur moyenne des valeurs de la matrice masse
214 // = 2 : on choisit la valeur maxi des valeurs de la matrice masse
215 int choix_mini_masse_nul;
216
217 // -----
218 // -- variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
219 // -----
220
221 // === pointeurs d'instance et classe particulières
222 Assemblage * Ass1_, * Ass2_, * Ass3_; // pointeurs d'assemblages
223
224 // === variables scalaires
225 int cas_combi_ddl; // def combinaison des ddl
226 int icas; // idem cas_combi_ddl mais pour lesCondlim
227 int compteur; // compteur d'itération
228 bool prepa_avec_remont; // comme son nom l'indique
229 bool brestart; // boolean qui indique si l'on est en restart ou pas
230 OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
231 int compteur_demarrage; // compteur pour les premiers incréments
232
233 // === vecteurs
234 Vecteur vglobin; // puissance interne (sans accélération): pour ddl accélération
235 Vecteur vglobex; // puissance externe (sans accélération)
236 Vecteur vglobaal; // puissance totale qui écrase vglobin
237 Vecteur vcontact; // puissance des forces de contact
238 Vecteur save_X_t; // vecteur intermédiaire de sauvegarde
239 Vecteur X_Bl,V_Bl,G_Bl; // stockage transitoirement des X V GAMMA <-> CL
240 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
241
242 // === les listes
243 list <LesCondlim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués
244 // === les tableaux
245 Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
246 Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
247 // === les matrices
248 Mat_abstraite* mat_masse,* mat_masse_sauve; // choix de la matrice de masse
249 Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
250 Vecteur v_masse,v_masse1; // vecteurs intermédiaires qui servent pour la construction
251 // de la matrice masse
252 Tableau <Coordonnee > sortie_masse_noeud; // pour la sortie
253 // de dimension le nombre de noeud
254 Mat_abstraite* matglob; // choix de la matrice de raideur pour un assemblage à partir de la raideur
255 // réelle
256
257 // -----
258 // -- fin variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
259 // -----
260
261 // ----- pour le calcul de la masse pour la méthode de relaxation dynamique
262 static Ddl_enum_etendu masseRelax_1; // simplement pour le définir une fois pour toute
263 static TypeQuelconque masseRelax_2; // simplement pour le définir une fois pour
264
265 // METHODES PROTEGEES :
266 // lecture des paramètres du calcul
267 void lecture_Parametres(UtilLecture& entreePrinc);
268 // écriture des paramètres dans la base info
269 // = 1 : on écrit tout

```

```

266 // = 2 : on écrit uniquement les données variables (supposées comme telles)
267 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc, const int& cas);
268 // lecture des paramètres dans la base info
269 // = 1 : on récupère tout
270 // = 2 : on récupère uniquement les données variables (supposées comme telles)
271 // choix = true : fonctionnement normal
272 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
273 // car la lecture est impossible
274 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc, const int& cas, bool choix);
275 // création d'un fichier de commande: cas des paramètres spécifiques
276 void Info_commande_parametres(UtilLecture& entreePrinc);
277
278 // gestion et vérification du pas de temps et modif en conséquence si nécessaire
279 // cas = 1: initialisation du pas de temps et vérif / au pas de temps critique
280 // cas = 2: vérif / au pas de temps critique
281 // en entrée: modif_pas_de_temps: indique qu'il y a eu par ailleurs (via Charge->Avance())
282 // une modification du pas de temps depuis le dernier appel
283 // retourne vrai s'il y a une modification du pas de temps, faux sinon
284 bool Gestion_pas_de_temps(bool modif_pas_de_temps, LesMaillages * lesMail, int cas);
285
286 //---- gestion des commndes interactives -----
287 // écoute et prise en compte d'une commande interactive
288 // ramène true tant qu'il y a des commandes en cours
289 bool ActionInteractiveAlgo();
290
291 // initialisation pour le calcul de la matrice masse dans le cas de l'algorithme de relaxation
dynamique
292 void InitCalculMatriceMasse(LesMaillages * lesMail, Mat_abstraite& mat_masse
293                             , Assemblage& Ass, Mat_abstraite& mat_masse_sauve
294                             , LesFonctions_nD* lesFonctionsND);
295
296 // calcul de la matrice masse dans le cas de l'algorithme de relaxation dynamique
297 void CalculMatriceMasse(const int& relax_vit_acce, LesMaillages * lesMail
298                         , Assemblage& Ass, int compteur
299                         , const DiversStockage* diversStockage, LesReferences* lesRef
300                         , const Enum_ddl & N_ddl, LesContacts* lescontacts
301                         , LesFonctions_nD* lesFonctionsND);
302 // calcul de la matrice masse dans le cas de l'algorithme de relaxation dynamique avec optimisation
en continu
303 // de la matrice masse
304 // force_recalcul_masse : un indicateur pour forcer le calcul éventuellement
305 // en retour est mis à false s'il était true en entrée
306 void CalculEnContinuMatriceMasse(const int& relax_vit_acce, LesMaillages * lesMail
307                                  , Assemblage& Ass, int compteur
308                                  , const DiversStockage* diversStockage, LesReferences* lesRef
309                                  , const Enum_ddl & N_ddl, bool premier_calcul
310                                  , LesContacts* lescontacts
311                                  , bool & force_recalcul_masse, LesFonctions_nD* lesFonctionsND);
312
313 // dans le cas d'une relaxation de type 4 (mixte cinétique et visqueuse, on test pour savoir
314 // s'il faut du visqueux ou du cinétique
315 // ramène true si on continue en cinétique
316 // false si on bascule
317 // force_recalcul_masse : un indicateur de retour qui s'applique uniquement si
318 // et_recalcul_masse_a_la_transition = true
319 // et que l'on est juste à la transition, sinon il est toujours false
320 bool Cinetique_ou_visqueux(bool & force_recalcul_masse);
321
322 // calcul du facteur lambda en fonction de ce qui est demandé (automatique ou via une fonction etc.)
323 double Calcul_Lambda();
324
325 // fonction virtuelle permettant de mettre à jour les infos aux noeuds
326 // à cause de certains contact (ex: cas_contact = 4)
327 // Les vecteurs Xtdt et Vtdt sont mis à jour par la méthode
328 // la vitesse locale du noeud est mis à jour en fonction de la position locale et de l'algo
329 virtual void Repercussion_algo_sur_cinematique(LesContacts* lesContacts, Vecteur& Xtdt, Vecteur& Vtdt)
;
330
331 // des fonctions inlines pour mettre à jour des grandeurs globales
332 // -- initialisation du fait que l'on est en amortissement cinétique (=1) ou visqueux (0)
333 void Transfert_ParaGlob_AMOR_CINET_VISQUEUX() const
334 {void* pt_void = ParaGlob::param->Mise_a_jour_grandeur_consultable(AMOR_CINET_VISQUEUX);
335   TypeQuelconque* pt_quelc = (TypeQuelconque*) pt_void;
336   Grandeur_scalaire_entier& gr
337     = *((Grandeur_scalaire_entier*) pt_quelc->Grandeur_pointee()); // pour simplifier
338   if (visqueux_activer)
339     {*(gr.ConteneurEntier()) = 0;}
340   else {*(gr.ConteneurEntier()) = 1;};
341 };
342 };
343 /// @} // end of group
344
345 #endif

```

## 7.9 Algori\_tchamwa.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           3/09/99                               $   *
31 *                                                         *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)   $   *
33 *                                                         *
34 *   PROJET:        Herezh++                               $   *
35 *                                                         *
36 *****/
37 *   BUT:           Algorithme de calcul dynamique explicite, pour de la
38 *                 mecanique du solide déformable en coordonnees materielles*
39 *                 entrainees. On ne prend pas en compte les phénomènes de *
40 *                 contact. Correspond à l'implantation de l'algo *
41 *                 de wielgosz tchamwa.                               $   *
42 *                 *                                             *
43 *   ***** *
44 *
45 *   VERIFICATION: *
46 *
47 *   ! date !   auteur !           but                               !   *
48 *   ----- *
49 *   !           !           !           !                           !   *
50 *   ----- *
51 *   ***** *
52 *   MODIFICATIONS: *
53 *   ! date !   auteur !           but                               !   *
54 *   ----- *
55 *   ----- *
56 *****/
57 #ifndef AGORI_TCHAMWA_H
58 #define AGORI_TCHAMWA_H
59
60
61 #include "Algori.h"
62 #include "Assemblage.h"
63
64
65 /// @addtogroup Les_algorithmes_de_resolutions_globales
66 /// @{
67 ///
68
69 ///   BUT:           Algorithme de calcul dynamique explicite, pour de la
70 ///                 mecanique du solide déformable en coordonnees materielles
71 ///                 entrainees. On ne prend pas en compte les phénomènes de
72 ///                 contact. Correspond à l'implantation de l'algo
73 ///                 de wielgosz tchamwa.
74
75 class AlgoriTchamwa : public Algori
76 {
77 public :
78     // CONSTRUCTEURS :
79     AlgoriTchamwa () ; // par défaut
80
81     // constructeur en fonction du type de calcul
82     // du sous type (pour les erreurs, remaillage etc...)
83     // il y a ici lecture des parametres attaches au type
84     AlgoriTchamwa (const bool avec_typeDeCal
85                 ,const list <EnumSousTypeCalcul>& soustype

```

```

86         ,const list <bool>& avec_soustypeDeCal
87         ,UtilLecture& entreePrinc);
88 // constructeur de copie
89 AlgoriTchamwa (const AlgoriTchamwa& algo);
90
91 // constructeur de copie à partie d'une instance indifférenciée
92 Algori * New_idem(const Algori* algo) const
93 { // on vérifie qu'il s'agit bien d'une instance
94   if (algo->TypeDeCalcul() != DYNA_EXP_TCHAMWA)
95     { cout << "\n *** erreur lors de la creation par copie d'un algo DYNA_EXP_TCHAMWA "
96       << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
97       << " arret !! " << flush;
98       Sortie(1);
99     }
100   else
101     { AlgoriTchamwa* inter = (AlgoriTchamwa*) algo;
102       return ((Algori *) new AlgoriTchamwa(*inter));
103     };
104 };
105
106 // DESTRUCTEUR :
107 ~AlgoriTchamwa () ;
108
109 // METHODES PUBLIQUES :
110 // execution de l'algorithme explicite dans le cas dynamique sans contact
111 void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID* ,LesFonctions_nD*
112               ,VariablesExporter* varExpor,LesLoisDeComp*
113               ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
114
115 //----- décomposition en 3 du calcul d'équilibre -----
116 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
117 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
calcul
118 //          certaines variables ont-été changés
119
120 // initialisation
121 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
122                   ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
123                   ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
124 // mise à jour
125 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
126                   ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
127                   ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
128
129 // calcul de l'équilibre
130 // si tb_combiner est non null -> un tableau de 2 fonctions
131 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
132 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
133 //
134 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
135 //          en fonction de la demande de sauvegard,
136 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
137 // qui lui fonctionne indépendamment
138 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
139                 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
140                 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
141                 ,Tableau < Fonction_nD > * tb_combiner);
142 // dernière passe
143 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
144               ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
145               ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
146
147
148 // sortie du schemaXML: en fonction de enu
149 void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const;
150
151 protected :
152 // VARIABLES PROTEGEES :
153 // paramètre de la méthode de tchamwa Wielgoz
154 double* phi_1;
155
156 // valeurs figées pour les paramètres de l'algorithme
157 const double alphaa; const double gammaa; const double lambdaa;
158 double betaa;
159 // liste de variables de travail déclarées ici pour éviter le passage de paramètre entre les
160 // méthodes internes à la classe
161 double delta_t,delta_t_critique; // pas de temps et critique
162 double deltat2,unsurdeltat,lambdaDeltat,alphaDelta_t,betaDeltat2,gammaDelta_t;
163
164 int type_cal_equilibre; // para pour le choix entre différents type de déclinaison d'algo
165 // Vecteur tabPourPhi; // vecteur modulation pour le phi, appliqué à tous les ddl
166 CourbeID* CGamma_pourVarPhi; // courbe de modulation pour phi, fonction de gamma
167 string nom_courbe_CGamma_pourVarPhi; // nom de la courbe lue au début
168
169 // utilisation d'une pondération avec une moyenne de l'accélération
170 int npas_moyacc, npas_effectue; // nombre de pas de temps où l'on fait la moyenne
171 double valmin,valmax; // mini maxi sur la moyenne

```

```

172 Vecteur moy_acc, moy_acc_en_calcul;
173
174 // exploratoire avec l'utilisation des approximants de pade
175 int degre_num,degre_deno;
176
177 // exploratoire avec type_cal_equilibre=4
178 double type4_inc_deb,type4_inc_deplace;
179
180 // -----
181 // -- variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
182 // -----
183
184 // === pointeurs d'instance et classe particulières
185 Assemblage * Ass1_, * Ass2_, * Ass3_; // pointeurs d'assemblages
186
187 // === variables scalaires
188 int cas_combi_ddl; // def combinaison des ddl
189 int icas; // idem cas_combi_ddl mais pour lesCondlim
190 bool prepa_avec_remont; // comme son nom l'indique
191 bool brestart; // booleen qui indique si l'on est en restart ou pas
192 OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
193
194 // === vecteurs
195 Vecteur vglobin; // puissance interne : pour ddl accélération
196 Vecteur vglobex; // puissance externe
197 Vecteur vglobaal; // puissance totale
198 Vecteur vcontact; // puissance des forces de contact
199 Vecteur X_Bl,V_Bl,G_Bl; // stockage transitoirement des X V GAMMA <-> CL
200 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
201 // === les listes
202 list <LesCondlim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués
203 // === les tableaux
204 Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
205 Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
206 // === les matrices
207 Mat_abstraite* mat_masse,* mat_masse_sauve; // choix de la matrice de masse
208 Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
209
210 // -----
211 // -- fin variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
212 // -----
213
214
215 // METHODES PROTEGEES :
216 // void Calcul_Equilibre // *** algo global historique
217 // (ParaGlob * paraGlob,LesMaillages * lesMail,
218 // LesReferences* lesRef,LesCourbes1D*lesCourbes1D,LesLoisDeComp* lesLoisDeComp,
219 // DiversStockage* diversStockage,Charge* charge,
220 // LesCondlim* lesCondlim,LesContacts* lesContacts,Resultats* resultats);
221 // idem mais version exploratoire
222 void Calcul_Equilibre2(ParaGlob * paraGlob,LesMaillages * lesMail,
223 LesReferences* lesRef,LesCourbes1D* lesCourbes1D,LesFonctions_nD* lesFonctionsnD
224 VariablesExporter* varExpor,LesLoisDeComp* lesLoisDeComp
225 DiversStockage* diversStockage,Charge* charge,LesCondlim* lesCondlim,LesContacts*
lesContacts
226 Resultats* resultats);
227 // idem autre version exploratoire
228 void Calcul_Equilibre4(ParaGlob * paraGlob,LesMaillages * lesMail,
229 LesReferences* lesRef,LesCourbes1D* lesCourbes1D,LesFonctions_nD* lesFonctionsnD
230 VariablesExporter* varExpor,LesLoisDeComp* lesLoisDeComp
231 DiversStockage* diversStockage,Charge* charge,LesCondlim* lesCondlim,LesContacts*
lesContacts
232 Resultats* resultats);
233 // lecture des paramètres du calcul
234 void lecture_Parametres(UtilLecture& entreePrinc);
235 // écriture des paramètres dans la base info
236 // = 1 : on écrit tout
237 // = 2 : on écrit uniquement les données variables (supposées comme telles)
238 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
239 // lecture des paramètres dans la base info
240 // = 1 : on récupère tout
241 // = 2 : on récupère uniquement les données variables (supposées comme telles)
242 // choix = true : fonctionnement normal
243 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
244 // car la lecture est impossible
245 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
246 // création d'un fichier de commande: cas des paramètres spécifiques
247 void Info_commande_parametres(UtilLecture& entreePrinc);
248
249 // gestion et vérification du pas de temps et modif en conséquence si nécessaire
250 // cas = 1: initialisation du pas de temps et vérif / au pas de temps critique
251 // cas = 2: vérif / au pas de temps critique
252 // cas = 3: il y a une demande de changement de pas de temps: nouveau_dt
253 // si nouveau_dt = 0, on ne fait rien
254 // si nouveau_dt = 0, on ne fait rien
255 // en entrée: modif_pas_de_temps: indique qu'il y a eu par ailleurs (via Charge->Avance())

```



```

256 // une modification du pas de temps depuis le dernier appel
257 // retourne vrai s'il y a une modification du pas de temps, faux sinon
258 bool Gestion_pas_de_temps(bool modif_pas_de_temps, LesMaillages * lesMail, int cas, double
nouveau_dt);

259
260 //---- gestion des commndes interactives -----
261 // écoute et prise en compte d'une commande interactive
262 // ramène true tant qu'il y a des commandes en cours
263 bool ActionInteractiveAlgo();
264
265 };
266 /// @} // end of group
267
268 #endif

```

## 7.10 AlgoriDynaExpli.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           3/09/99
31 *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:         Herezh++
35 *
36 *****/
37 *   BUT:           Algorithme de calcul dynamique explicite, pour de la
38 *                  mecanique du solide déformable en coordonnees materielles*
39 *                  entrainees. On ne prend pas en compte les phénomènes de *
40 *                  contact.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *   ! date ! auteur ! but
46 *   -----
47 *   ! ! !
48 *   $
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date ! auteur ! but
52 *   -----
53 *   $
54 *****/
55 #ifndef AGORIDYNAEXPLI_T
56 #define AGORIDYNAEXPLI_T
57
58
59 #include "Algori.h"
60 #include "Assemblage.h"
61
62
63 /// @addtogroup Les_algorithmes_de_resolutions_globales
64 /// @{
65 ///
66
67 /// BUT: Algorithme de calcul dynamique explicite, pour de la

```

```

68 ///          mecanique du solide déformable en coordonnees materielles
69 ///          entrainees. On ne prend pas en compte les phénomènes de
70 ///          contact.
71
72 class AlgoriDynaExpli : public Algori
73 {
74 public :
75     // CONSTRUCTEURS :
76     AlgoriDynaExpli () ; // par défaut
77
78     // constructeur en fonction du type de calcul
79     // du sous type (pour les erreurs, remaillage etc...)
80     // il y a ici lecture des parametres attaches au type
81     AlgoriDynaExpli (const bool avec_typeDeCal
82                     ,const list <EnumSousTypeCalcul>& soustype
83                     ,const list <bool>& avec_soustypeDeCal
84                     ,UtilLecture& entreePrinc);
85     // constructeur de copie
86     AlgoriDynaExpli (const AlgoriDynaExpli& algo);
87
88     // constructeur de copie à partie d'une instance indifférenciée
89     Algori * New_idem(const Algori* algo) const
90     { // on vérifie qu'il s'agit bien d'une instance
91       if (algo->TypeDeCalcul() != DYNA_EXP)
92         { cout << "\n *** erreur lors de la creation par copie d'un algo DYNA_EXP "
93           << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
94           << " arret !! " << flush;
95           Sortie(1);
96         }
97       else
98         { AlgoriDynaExpli* inter = (AlgoriDynaExpli*) algo;
99           return ((Algori *) new AlgoriDynaExpli(*inter));
100         };
101     };
102
103     // DESTRUCTEUR :
104     ~AlgoriDynaExpli () ;
105
106     // METHODES PUBLIQUES :
107     // execution de l'algorithme explicite dans le cas dynamique sans contact
108     void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D* ,LesFonctions_nD*
109                 ,VariablesExporter* varExpor ,LesLoisDeComp*
110                 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
111
112     //----- décomposition en 3 du calcul d'équilibre -----
113     // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
114     // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
115     // calcul
116     //          certaines variables ont-été changés
117
118     // initialisation
119     void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
120                      ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
121                      ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
122
123     // mise à jour
124     void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
125                      ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
126                      ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
127
128     // calcul de l'équilibre
129     // si tb_combiner est non null -> un tableau de 2 fonctions
130     // - la première fct dit si on doit valider ou non le calcul à convergence ok,
131     // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
132     //
133     // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
134     // en fonction de la demande de sauvegard,
135     // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
136     // qui lui fonctionne indépendamment
137     void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
138                    ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
139                    ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
140                    ,Tableau < Fonction_nD* > * tb_combiner);
141
142     // dernière passe
143     void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
144                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
145                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
146
147     // sortie du schemaXML: en fonction de enu
148     void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const ;
149
150 protected :
151     // VARIABLES PROTEGEES :
152     // vecteurs globaux définissant les vitesses intermédiaires
153     Vecteur vitesse_tMoinsUnDemi , vitesse_tPlusUnDemi;
154
155     // liste de variables de travail déclarées ici pour éviter le passage de paramètre entre les

```

```

154 // méthodes internes à la classe
155 double delta_t, unsurdeltat, deltatSurDeux, deltat2, deltat2SurDeux;
156
157 int type_cal_equilibre; // para pour le choix entre Calcul_Equilibre et Calcul_Equilibre2
158
159 // -----
160 // -- variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
161 // -----
162
163 // === pointeurs d'instance et classe particulières
164 Assemblage * Ass1_, * Ass2_, * Ass3_; // pointeurs d'assemblages
165
166 // === variables scalaires
167 int cas_combi_ddl; // def combinaison des ddl
168 int icas; // idem cas_combi_ddl mais pour lesCondlim
169 bool prepa_avec_remont; // comme son nom l'indique
170 bool brestart; // booléen qui indique si l'on est en restart ou pas
171 OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
172 int compteur_demarrage; // compteur pour les premiers incréments
173
174 // === vecteurs
175 Vecteur vglobin; // puissance interne : pour ddl accélération
176 Vecteur vglobex; // puissance externe
177 Vecteur vglobaal; // puissance totale
178 Vecteur vcontact; // puissance des forces de contact
179 Vecteur X_Bl,V_Bl,G_Bl; // stockage transitoirement des X V GAMMA <-> CL
180 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
181
182 // === les listes
183 list <LesCondlim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués
184 // === les tableaux
185 Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
186 Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
187 // === les matrices
188 Mat_abstraite* mat_masse,* mat_masse_sauve; // choix de la matrice de masse
189 Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
190
191 // -----
192 // -- fin variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
193 // -----
194
195 // METHODES PROTEGEES :
196 void Calcul_Equilibre
197     (ParaGlob * paraGlob, LesMaillages * lesMail,
198      LesReferences* lesRef, LesCourbes1D* lesCourbes1D, LesFonctions_nD* lesFonctionsnD
199      , VariablesExporter* varExpor, LesLoisDeComp* lesLoisDeComp,
200      DiversStockage* diversStockage, Charge* charge,
201      LesCondlim* lesCondlim, LesContacts* lesContacts, Resultats* resultats);
202 // nouvelle algo avec un passage à vide au premier passage pour être plus conforme à la dfc
203 classique
204 void Calcul_Equilibre2
205     (ParaGlob * paraGlob, LesMaillages * lesMail,
206      LesReferences* lesRef, LesCourbes1D* lesCourbes1D, LesFonctions_nD* lesFonctionsnD
207      , VariablesExporter* varExpor, LesLoisDeComp* lesLoisDeComp,
208      DiversStockage* diversStockage, Charge* charge,
209      LesCondlim* lesCondlim, LesContacts* lesContacts, Resultats* resultats);
210 // lecture des paramètres du calcul
211 void lecture_Parametres(UtilLecture& entreePrinc);
212 // écriture des paramètres dans la base info
213 // = 1 : on écrit tout
214 // = 2 : on écrit uniquement les données variables (supposées comme telles)
215 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc, const int& cas);
216 // lecture des paramètres dans la base info
217 // = 1 : on récupère tout
218 // = 2 : on récupère uniquement les données variables (supposées comme telles)
219 // choix = true : fonctionnement normal
220 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
221 // défaut
222 // car la lecture est impossible
223 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc, const int& cas, bool choix);
224 // création d'un fichier de commande: cas des paramètres spécifiques
225 void Info_commande_parametres(UtilLecture& entreePrinc);
226
227 // gestion et vérification du pas de temps et modif en conséquence si nécessaire
228 // cas = 1: initialisation du pas de temps et vérif / au pas de temps critique
229 // cas = 2: vérif / au pas de temps critique
230 // en entrée: modif_pas_de_temps: indique qu'il y a eu par ailleurs (via Charge->Avance())
231 // une modification du pas de temps depuis le dernier appel
232 // retourne vrai s'il y a une modification du pas de temps, faux sinon
233 bool Gestion_pas_de_temps(bool modif_pas_de_temps, LesMaillages * lesMail, int cas);
234
235 //---- gestion des commndes interactives -----
236 // écoute et prise en compte d'une commande interactive
237 // ramène true tant qu'il y a des commandes en cours
238 bool ActionInteractiveAlgo();

```

```

239
240 };
241 /// @} // end of group
242
243 #endif

```

## 7.11 AlgoriDynaExpli\_zhai.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           3/09/99
31 *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:         Herezh++
35 *
36 *****/
37 *   BUT:           Algorithme de calcul dynamique explicite, pour de la
38 *                 mecanique du solide déformable en coordonnees materielles*
39 *                 entrainees. On ne prend pas en compte les phénomènes de *
40 *                 contact.
41 *                 La méthode implantée est celle de zhai.
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *
47 *   ! date ! auteur ! but
48 *   -----
49 *   ! ! !
50 *   *****
51 *   MODIFICATIONS:
52 *   ! date ! auteur ! but
53 *   -----
54 *
55 *****/
56 #ifndef AGORIDYNAEXPLI_ZHAI_T
57 #define AGORIDYNAEXPLI_ZHAI_T
58
59
60 #include "Algori.h"
61 #include "Assemblage.h"
62
63 /// @addtogroup Les_algorithmes_de_resolutions_globales
64 /// @{
65 ///
66
67 /// BUT:           Algorithme de calcul dynamique explicite, pour de la
68 ///                 mecanique du solide déformable en coordonnees materielles
69 ///                 entrainees. On ne prend pas en compte les phénomènes de
70 ///                 contact.
71 ///                 La méthode implantée est celle de zhai.
72
73 class AlgoriDynaExpli_zhai : public Algori
74 {
75 public :
76     // CONSTRUCTEURS :

```

```

77   AlgoriDynaExpli_zhai () ; // par default
78
79   // constructeur en fonction du type de calcul
80   // du sous type (pour les erreurs, remaillage etc...)
81   // il y a ici lecture des parametres attaches au type
82   AlgoriDynaExpli_zhai (const bool avec_typeDeCal
83                       ,const list <EnumSousTypeCalcul>& soustype
84                       ,const list <bool>& avec_soustypeDeCal
85                       ,UtilLecture& entreePrinc);
86   // constructeur de copie
87   AlgoriDynaExpli_zhai (const AlgoriDynaExpli_zhai& algo);
88
89   // constructeur de copie à partie d'une instance indifférenciée
90   Algori * New_idem(const Algori* algo) const
91   { // on vérifie qu'il s'agit bien d'une instance
92     if (algo->TypeDeCalcul() != DYNA_EXP_ZHAI)
93       { cout << "\n *** erreur lors de la creation par copie d'un algo DYNA_EXP_ZHAI "
94         << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
95         << " arret !! " << flush;
96         Sortie(1);
97       }
98     else
99       { AlgoriDynaExpli_zhai* inter = (AlgoriDynaExpli_zhai*) algo;
100        return ((Algori *) new AlgoriDynaExpli_zhai(*inter));
101      };
102   };
103
104   // DESTRUCTEUR :
105   ~AlgoriDynaExpli_zhai () ;
106
107   // METHODES PUBLIQUES :
108   // execution de l'algorithme explicite dans le cas dynamique sans contact
109   void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D* ,LesFonctions_nD*
110                ,VariablesExporter* varExpor,LesLoisDeComp*
111                ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
112
113   //----- décomposition en 3 du calcul d'équilibre -----
114   // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
115   // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
116   // calcul
117   // certaines variables ont-été changés
118
119   // initialisation
120   void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
121                     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
122                     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
123
124   // mise à jour
125   void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
126                     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
127                     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
128
129   // calcul de l'équilibre
130   // si tb_combiner est non null -> un tableau de 2 fonctions
131   // - la première fct dit si on doit valider ou non le calcul à convergence ok,
132   // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
133   //
134   // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
135   // en fonction de la demande de sauvegard,
136   // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
137   // qui lui fonctionne indépendamment
138   void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
139                    ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
140                    ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
141                    ,Tableau < Fonction_nD* > * tb_combiner);
142
143   // dernière passe
144   void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
145                 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
146                 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
147
148   // sortie du schemaXML: en fonction de enu
149   void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const {};
150
151 protected :
152   // VARIABLES PROTEGEES :
153   // paramètre de la méthode
154   double* grand_phi;
155   double* phi_minus;
156   double* gamma_pt;
157   double* beta;
158
159   // liste de variables de travail déclarées ici pour éviter le passage de paramètre entre les
160   // méthodes internes à la classe
161   double delta_t,deltat2,unsurdeltat;
162   double undemiplusgrandphi_deltat2,grandphi_deltat2,unpluphi_delta_t,phi_delta_t;
163   double un_moins_gamma_delta_t,gamma_delta_t,undemi_moins_beta_deltat2,beta_deltat2;
164
165   // -----
166   // -- variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --

```

```

163 // -----
164
165 // === pointeurs d'instance et classe particulières
166 Assemblage * Ass1_, * Ass2_, * Ass3_; // pointeurs d'assemblages
167
168 // === variables scalaires
169 int cas_combi_ddl; // def combinaison des ddl
170 int icas; // idem cas_combi_ddl mais pour lesCondlim
171 bool prepa_avec_remont; // comme son nom l'indique
172 bool brestart; // booleen qui indique si l'on est en restart ou pas
173 OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
174
175 // === vecteurs
176 Vecteur vglobin; // puissance interne : pour ddl accélération
177 Vecteur vglobex; // puissance externe
178 Vecteur vglobaal; // puissance totale
179 Vecteur vcontact; // puissance des forces de contact
180 Vecteur acceleration_prec; // vecteur intermédiaire
181 Vecteur X_Bl,V_Bl,G_Bl; // stockage transitoirement des X V GAMMA <-> CL
182 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
183 // === les listes
184 list <LesCondlim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués
185 // === les tableaux
186 Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
187 Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
188 // === les matrices
189 Mat_abstraite* mat_masse,* mat_masse_sauve; // choix de la matrice de masse
190 Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
191
192
193
194 // METHODES PROTEGEES :
195 void Calcul_Equilibre
196     (ParaGlob * paraGlob,LesMaillages * lesMail,
197      LesReferences* lesRef,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD
198      ,VariablesExporter* varExpor,LesLoisDeComp* lesLoisDeComp,
199      DiversStockage* diversStockage,Charge* charge,
200      LesCondlim* lesCondlim,LesContacts* lesContacts,Resultats* resultats);
201
202
203 //---- gestion des commndes interactives -----
204 // écoute et prise en compte d'une commande interactive
205 // ramène true tant qu'il y a des commandes en cours
206 bool ActionInteractiveAlgo();
207
208 // lecture des paramètres du calcul
209 void lecture_Parametres(UtilLecture& entreePrinc);
210 // écriture des paramètres dans la base info
211 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
212 // lecture des paramètres dans la base info
213 // choix = true : fonctionnement normal
214 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
215 // défaut
216 // car la lecture est impossible
217 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
218 // création d'un fichier de commande: cas des paramètres spécifiques
219 void Info_commande_parametres(UtilLecture& entreePrinc);
220 // gestion et vérification du pas de temps et modif en conséquence si nécessaire
221 // cas = 1: initialisation du pas de temps et vérif / au pas de temps critique
222 // ceci pour le temps t=0
223 // cas = 2: initialisation du pas de temps et vérif / au pas de temps critique
224 // ceci pour le temps t
225 // une modification du pas de temps depuis le dernier appel
226 // retourne vrai s'il y a une modification du pas de temps, faux sinon
227 bool Gestion_pas_de_temps(bool modif_pas_de_temps,LesMaillages * lesMail,int cas);
228 };
229 /// @} // end of group
230 #endif

```

## 7.12 AlgoRungeKutta.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //

```

```

14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           8/06/06
31 *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:        Herezh++
35 *
36 *   ****
37 *   BUT:           Algorithmme de calcul dynamique, méthode de Runge Kutta,
38 *                 pour de la mecanique du solide.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date ! auteur ! but
44 *   -----
45 *   ! ! !
46 *   $
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date ! auteur ! but
50 *   -----
51 *   $
52 *****/
53 #ifndef AGORIRUNGEKUTTA_T
54 #define AGORIRUNGEKUTTA_T
55
56
57 #include "Algori.h"
58 #include "Assemblage.h"
59 #include "Algo_edp.h"
60
61 /// @addtogroup Les_algorithmes_de_resolutions_globales
62 /// @{
63 ///
64
65 /// BUT:   Algorithmme de calcul dynamique, méthode de Runge Kutta,
66 /// pour de la mecanique du solide.
67
68 class AlgoRungeKutta : public Algori
69 {
70 public :
71     // CONSTRUCTEURS :
72     AlgoRungeKutta () ; // par default
73
74     // constructeur en fonction du type de calcul
75     // du sous type (pour les erreurs, remaillage etc...)
76     // il y a ici lecture des parametres attaches au type
77     AlgoRungeKutta (const bool avec_typeDeCal
78                   ,const list <EnumSousTypeCalcul>& soustype
79                   ,const list <bool>& avec_soustypeDeCal
80                   ,UtilLecture& entreePrinc);
81     // constructeur de copie
82     AlgoRungeKutta (const AlgoRungeKutta& algo);
83
84     // constructeur de copie à partie d'une instance indifférenciée
85     AlgoRungeKutta * New_idem(const AlgoRungeKutta* algo) const
86     { // on vérifie qu'il s'agit bien d'une instance
87       if (algo->TypeDeCalcul() != DYNA_RUNGE_KUTTA)
88         { cout << "\n *** erreur lors de la creation par copie d'un algo DYNA_RUNGE_KUTTA "
89           << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
90           << " arret !! " << flush;
91           Sortie(1);
92         }
93       else
94         { AlgoRungeKutta* inter = (AlgoRungeKutta*) algo;
95           return ((Algori *) new AlgoRungeKutta(*inter));
96         };
97     };
98
99     // DESTRUCTEUR :

```

```

100   ~AlgoriRungeKutta () ;
101
102   // METHODES PUBLIQUES :
103   // execution de l'algorithme explicite dans le cas dynamique sans contact
104   void Execution(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID* , LesFonctions_nD*
105                , VariablesExporter* varExpor, LesLoisDeComp*
106                , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats* );
107
108   //----- décomposition en 3 du calcul d'équilibre -----
109   // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
110   // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
111   // calcul
112   // certaines variables ont-été changés
113
114   // initialisation
115   void InitAlgorithme(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID*
116                    , LesFonctions_nD* , VariablesExporter* , LesLoisDeComp*
117                    , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats* );
118
119   // mise à jour
120   void MiseAJourAlgo(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID*
121                    , LesFonctions_nD* , VariablesExporter* , LesLoisDeComp*
122                    , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats* );
123
124   // calcul de l'équilibre
125   // si tb_combiner est non null -> un tableau de 2 fonctions
126   // - la première fct dit si on doit valider ou non le calcul à convergence ok,
127   // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
128   //
129   // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
130   // en fonction de la demande de sauvegard,
131   // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
132   // qui lui fonctionne indépendamment
133   void CalEquilibre(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID*
134                   , LesFonctions_nD* , VariablesExporter* , LesLoisDeComp*
135                   , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats*
136                   , Tableau < Fonction_nD* > * tb_combiner);
137
138   // dernière passe
139   void FinCalcul(ParaGlob * , LesMaillages * , LesReferences* , LesCourbesID*
140                , LesFonctions_nD* , VariablesExporter* , LesLoisDeComp*
141                , DiversStockage* , Charge* , LesCondLim* , LesContacts* , Resultats* );
142
143   // sortie du schemaXML: en fonction de enu
144   void SchemaXML_Algori(ofstream& sort, const Enum_IO_XML enu) const ;
145
146   protected :
147   // VARIABLES PROTEGEES :
148
149   // liste de variables de travail déclarées ici pour éviter le passage de paramètre entre les
150   // méthodes internes à la classe
151   // variables modifiées par Modif_transi_pas_de_temps, et Gestion_pas_de_temps
152   double delta_t, unsurdeltat, deltatSurDeux, deltat2;
153
154   // -- variables de transferts internes nécessaires pour : Dyna_point --
155   // == pointeurs d'instance et classe particulières
156   LesMaillages * lesMail_;
157   LesReferences* lesRef_;
158   LesCourbesID* lesCourbesID_;
159   LesFonctions_nD* lesFonctionsnD_;
160   Charge* charge_;
161   LesCondLim* lesCondLim_;
162   LesContacts* lesContacts_;
163   Assemblage * Ass1, * Ass2, * Ass3; // pointeurs d'assemblages
164
165   // == variables scalaires
166   double maxPuissExt; // maxi de la puissance des efforts externes
167   double maxPuissInt; // maxi de la puissance des efforts internes
168   double maxReaction; // maxi des reactions
169   int inReaction; // pointeur d'assemblage pour le maxi de reaction
170   int inSol; // pointeur d'assemblage du maxi de variation de ddl
171   double maxDeltaDdl; // maxi de variation de ddl
172   int cas_combi_ddl; // def combinaison des ddl
173   int icas; // idem cas_combi_ddl mais pour lesCondlim
174   bool erreurSecondMembre; // pour la gestion des erreurs de calcul au second membre
175   bool prepa_avec_remont; // comme son nom l'indique
176   bool brestart; // boolean qui indique si l'on est en restart ou pas
177   OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
178
179   // == vecteurs
180   Vecteur vglobin; // puissance interne : pour ddl accélération
181   Vecteur vglobex; // puissance externe
182   Vecteur vglobaal; // puissance totale
183   Vecteur vcontact; // puissance des forces de contact
184   Vecteur X_Bl, V_Bl, G_Bl; // stockage transitoirement des X V GAMMA <-> CL
185   Vecteur forces_vis_num; // forces visqueuses d'origines numériques
186
187   // == les listes
188   list <LesCondLim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués

```



```

186 // == les tableaux
187 Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
188 Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
189 // == les matrices
190 Mat_abstraite* mat_masse,* mat_masse_sauve; // choix de la matrice de masse
191 Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
192
193 // ... partie relative à une résolution de l'avancement par une intégration de l'équation
différentielle
194 Algo_edp alg_edp;
195 int cas_kutta; // indique le type de runge_kutta que l'on veut utiliser
196 double erreurAbsolue,erreurRelative; // précision absolue et relative que l'on désire sur le
calcul Xtdt et vitessetdt
197 double erreur_maxi_global; // l'erreur obtenue
198 int nbMaxiAppel; // nombre maxi d'appel de la fonction dérivée
199 bool pilotage_un_step; // indique s'il y a pilotage ou pas
200 Vecteur estime_erreur;
201 Vecteur val_fonc_initiale,der_val_fonc_initiale,val_fonc,der_val_fonc;
202 Vecteur val_fonc_final,der_val_fonc_final;
203 double scale_fac; // facteur d'homogénéisation des vecteurs _val_fonc_
204
205 // METHODES PROTEGEES :
206
207 // lecture des paramètres du calcul
208 void lecture_Parametres(UtilLecture& entreePrinc);
209 // écriture des paramètres dans la base info
210 // = 1 : on écrit tout
211 // = 2 : on écrit uniquement les données variables (supposées comme telles)
212 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
213 // lecture des paramètres dans la base info
214 // = 1 : on récupère tout
215 // = 2 : on récupère uniquement les données variables (supposées comme telles)
216 // choix = true : fonctionnement normal
217 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
218 // car la lecture est impossible
219 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
220 // création d'un fichier de commande: cas des paramètres spécifiques
221 void Info_commande_parametres(UtilLecture& entreePrinc);
222
223 // gestion et vérification du pas de temps et modif en conséquence si nécessaire
224 // cas = 1: initialisation du pas de temps et vérif / au pas de temps critique
225 // cas = 2: vérif / au pas de temps critique, et division par nbstep
226 // ceci pour garantir que l'on fait le calcul avec 1 step
227 // en entrée: modif_pas_de_temps: indique qu'il y a eu par ailleurs (via Charge->Avance())
228 // une modification du pas de temps depuis le dernier appel
229 // retourne vrai s'il y a une modification du pas de temps, faux sinon
230 bool Gestion_pas_de_temps(bool modif_pas_de_temps,LesMaillages * lesMail,int cas,int nbstep);
231
232 // modification transitoire du pas de temps et modif en conséquence si nécessaire
233 // utilisée en continu par RK
234 // delta_tau : nouveau pas de temps transitoire imposé
235 void Modif_transi_pas_de_temps(const double & delta_tau);
236
237 //---- gestion des commndes interactives -----
238 // écoute et prise en compte d'une commande interactive
239 // ramène true tant qu'il y a des commandes en cours
240 bool ActionInteractiveAlgo();
241 // ----- pour RK : calcul du vecteur dérivée -----
242 // calcul de l'expression permettant d'obtenir la dérivée temporelle du problème
243 // utilisée dans la résolution de l'équation d'équilibre dynamique par la méthode RK
244 // en entrée:
245 // tau: temps courant
246 // val_fonc: qui contient à la suite X et X_point à tau
247 // en sortie:
248 // der_val_fonc : qui contient à la suite: X_point et gamma
249 // erreur : =0: le calcul est licite, si diff de 0, indique qu'il y a eu une erreur
250 // =1: la norme de sigma est supérieure à la valeur limite de saturation
251 Vecteur& Dyna_point(const double & tau, const Vecteur & val_fonc
, Vecteur & der_val_fonc,int& erreur);
252 // vérification de l'intégrité du résultat calculé
253 // erreur : =0: le calcul est licite, si diff de 0, indique qu'il y a eu une erreur
254 // =1: la norme de sigma est supérieure à la valeur limite de saturation
255 void Verif_integrite_Solution(const double & tau, const Vecteur & val_fonc,int & erreur);
256
257
258
259 };
260 /// @} // end of group
261
262 #endif

```

## 7.13 AlgoriNewmark.h

```

1 // This file is part of the Herezh++ application.
2 //

```

```

3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *      DATE:      29/09/2001
31 *
32 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
33 *
34 *      PROJET:    Herezh++
35 *
36 *
37 *      BUT:      Algorithme de calcul dynamique, pour de la mecanique
38 *                du solide déformable en coordonnees materielles
39 *                entrainees, en utilisant la discrétisation temporelle de
40 *                newmark .
41 *
42 *      *****
43 *
44 *      VERIFICATION:
45 *      ! date ! auteur ! but
46 *      -----
47 *      ! ! !
48 *
49 *      *****
50 *      MODIFICATIONS:
51 *      ! date ! auteur ! but
52 *      -----
53 *
54 *      *****/
55 #ifndef AGORINEWMARK_T
56 #define AGORINEWMARK_T
57
58
59 #include "Algori.h"
60 #include "Assemblage.h"
61
62 /// @addtogroup Les_algorithmes_de_resolutions_globales
63 /// @{
64 ///
65
66 /// BUT: Algorithme de calcul dynamique, pour de la mecanique
67 /// du solide déformable en coordonnees materielles
68 /// entrainees, en utilisant la discrétisation temporelle de
69 /// newmark .
70
71 class AlgoriNewmark : public Algori
72 {
73 public :
74 // CONSTRUCTEURS :
75 AlgoriNewmark () ; // par default
76
77 // constructeur en fonction du type de calcul
78 // du sous type (pour les erreurs, remaillage etc...)
79 // il y a ici lecture des parametres attaches au type
80 AlgoriNewmark (const bool avec_typeDeCal
81 ,const list <EnumSousTypeCalcul>& soustype
82 ,const list <bool>& avec_soustypeDeCal
83 ,UtilLecture& entreePrinc);
84 // constructeur de copie
85 AlgoriNewmark (const AlgoriNewmark& algo);
86
87 // constructeur de copie à partie d'une instance indifférenciée
88 Algori * New_idem(const Algori* algo) const

```

```

89     // on vérifie qu'il s'agit bien d'une instance
90     if (algo->TypeDeCalcul() != DYNA_IMP)
91     { cout << "\n *** erreur lors de la creation par copie d'un algo DYNA_IMP "
92       << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
93       << " arret !! " << flush;
94       Sortie(1);
95     }
96     else
97     { AlgoriNewmark* inter = (AlgoriNewmark*) algo;
98       return ((Algori *) new AlgoriNewmark(*inter));
99     };
100   };
101
102   // DESTRUCTEUR :
103   ~AlgoriNewmark () ;
104
105   // METHODES PUBLIQUES :
106   // execution de l'algorithme dans le cas non dynamique
107   void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID* ,LesFonctions_nD*
108     ,VariablesExporter* varExpor ,LesLoisDeComp*
109     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
110
111   //----- décomposition en 3 du calcul d'équilibre -----
112   // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
113   // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
114   // calcul
115   // certaines variables ont-été changés
116
117   // initialisation
118   void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
119     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
120     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
121
122   // mise à jour
123   void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
124     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
125     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
126
127   // calcul de l'équilibre
128   // si tb_combiner est non null -> un tableau de 2 fonctions
129   // - la première fct dit si on doit valider ou non le calcul à convergence ok,
130   // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
131   //
132   // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
133   // en fonction de la demande de sauvegard,
134   // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
135   // qui lui fonctionne indépendamment
136   void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
137     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
138     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
139     ,Tableau < Fonction_nD* > * tb_combiner);
140
141   // dernière passe
142   void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
143     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
144     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
145
146   // sortie du schemaXML: en fonction de enu
147   void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const {};
148
149 private :
150   // VARIABLES PROTEGEES :
151   // paramètre de newmark
152   double* beta_newmark;
153   double* gamma_newmark;
154   double * hht; // paramètre de hilbert hught taylor
155   bool stabilite; // indique si le calcul est inconditionnellement stable ou pas
156
157   // -- cas où on veut faire du masse scaling avec éventuellement du pilotage
158   bool avec_masse_scaling;
159   bool et_pilotage_masse_scaling_par_maxRatioForce_inertieSurStatique;
160   bool ou_pilotage_masse_scaling_par_maxRatioForce_inertieSurStatique;
161   double ratioForce_inertieSurStatique;
162   bool et_pilotage_masse_scaling_par_maxRatioDiag_masseSurRaideur;
163   bool ou_pilotage_masse_scaling_par_maxRatioDiag_masseSurRaideur;
164   double ratioDiag_masseSurRaideur;
165
166   // indicateur disant s'il faut calculer les conditions limites à chaque itération on non
167   int cL_a_chaque_iteration; // par défaut, non, == uniquement à chaque début d'incrément
168
169   // paramètres intermédiaires de calcul
170   // relatif au temps
171   double delta_t; // pas de temps
172   double delta_t_2,unSurBetaDeltaTcarre,betaDelta_t_2;
173   // def de coeffs simplificateurs
174   double betaMoinsGammaSurBeta,unSurGammaDelta,unMoinGamma;
175   double unMoinGammaSurgamma,zero5deltatcarreUnMoinDeuxBeta,zero5deltatcarreDeuxBeta;
176   double deltatUnMoinGamma,deltatGamma,unsurbetadeltat,unmoinsdeuxbetasurdeuxbeta;
177   double unSurBetaDeltaTcarre_o;
178   // cas de l'amortissement numérique

```

```

175 double nuAphaPrime,coef_masse,nuBetaPrime,coef_raideur;
176 double nuAphaPrimeunsurcoef_masse,unsurcoef_masse;
177
178 // VARIABLES PROTEGEES :
179 // -----
180 // -- variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
181 // -----
182
183 // === pointeurs d'instance et classe particulières
184 Assemblage * Ass1_, * Ass2_, * Ass3_; // pointeurs d'assemblages
185
186 // === variables scalaires
187 int cas_combi_ddl; // def combinaison des ddl
188 int icas; // idem cas_combi_ddl mais pour lesCondlim
189 bool prepa_avec_remont; // comme son nom l'indique
190 bool brestart; // booleen qui indique si l'on est en restart ou pas
191 OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
192
193 // === vecteurs
194 Vecteur vglobin; // puissance interne : pour ddl accélération
195 Vecteur vglobex; // puissance externe
196 Vecteur vcontact; // puissance des forces de contact
197 Vecteur vglobaal; // puissance totale qui ecrase vglobin
198 Vecteur vglobal_n,vglobal_n_inter; // pour HHT
199 Vecteur delta_prec_X; // les positions
200 Vecteur inter_tdt ; // accélérations et un vecteur intermédiaire de travail
201 Vecteur X_Bl,V_Bl,G_Bl; // stockage transitoirement des X V GAMMA <-> CL
202 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
203
204 // === les matrices
205 Mat_abstraite* matglob; // choix de la matrice de raideur
206 // --- cas où l'on a des matrices secondaires pour switcher si la première matrice
207 // ne fonctionne pas bien, par défaut matglob, matsauve et Tab_matmoysauve sont les premiers
208 // éléments des tableaux
209 // choix des matrices de raideur de substitution éventuelles : par défaut matglob = tab_mato(1)
210 Tableau < Mat_abstraite*> tab_mato;
211
212 void (Assemblage::* assemblMat) // un pointeur de fonction d'assemblage
213 (Mat_abstraite & matglob,const Mat_abstraite & matloc,
214 const DdlElement& tab_ddl,const Tableau<Noeud *>&tab_noeud);
215 // dans le cas où l'on fait du line search on dimensionne des vecteurs globaux supplémentaires
216 Vecteur * sauve_deltadept,*sauve_dept_a_tdt,*Vres,*v_travail;
217
218 // === les listes
219 list <LesCondlim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués
220 // === les tableaux
221 Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
222 Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
223 // === les matrices
224 Mat_abstraite* mat_masse; // choix de la matrice de masse
225 Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
226
227 // -----
228 // -- fin variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
229 // -----
230
231 // METHODES PROTEGEES :
232
233 // lecture des paramètres du calcul
234 void lecture_Parametres(UtilLecture& entreePrinc);
235 // écriture des paramètres dans la base info
236 // = 1 : on écrit tout
237 // = 2 : on écrit uniquement les données variables (supposées comme telles)
238 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
239 // lecture des paramètres dans la base info
240 // = 1 : on récupère tout
241 // = 2 : on récupère uniquement les données variables (supposées comme telles)
242 // choix = true : fonctionnement normal
243 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
244 // défaut
245 // car la lecture est impossible
246 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
247 // création d'un fichier de commande: cas des paramètres spécifiques
248 void Info_commande_parametres(UtilLecture& entreePrinc);
249
250 // gestion et vérification du pas de temps et modif en conséquence si nécessaire
251 // cas = 1: initialisation du pas de temps et de l'amortissement numérique si nécessaire
252 // ceci pour le temps t=0
253 // cas = 2: initialisation du pas de temps
254 // ceci pour le temps t
255 void Gestion_pas_de_temps(LesMaillages * lesMail,int cas);
256
257 // pilotage éventuel du masse scaling
258 void Pilotage_masse_scaling(const double& moy_diag_masse,const double& moy_diag_K
259 ,const int& sauve_indic_convergence);
260
261 //---- gestion des commndes interactives -----

```

```

261 // écoute et prise en compte d'une commande interactive
262 // ramène true tant qu'il y a des commandes en cours
263 bool ActionInteractiveAlgo();
264
265 };
266 /// @} // end of group
267
268 #endif

```

## 7.14 AlgoriMixte.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      12/09/2010
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 * *****/
37 *   BUT:      Algorithme de calcul mixte: statique - pseudo-dynamique
38 *             explicite, pour de la mecanique du solide déformable en
39 *             coordonnees materielles entrainees.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 * *****/
54 #ifndef AGORISTATEXPLI_T
55 #define AGORISTATEXPLI_T
56
57
58 #include "Algori.h"
59 #include "Assemblage.h"
60
61 /// @addtogroup Les_algorithmes_de_resolutions_globales
62 /// @{
63 ///
64
65 ///   BUT:      Algorithme de calcul mixte: statique - pseudo-dynamique
66 ///             explicite, pour de la mecanique du solide déformable en
67 ///             coordonnees materielles entrainees.
68
69 class AlgoristatExpli : public Algori
70 {
71 public :
72     // CONSTRUCTEURS :
73     AlgoristatExpli () ; // par default

```

```

74
75 // constructeur en fonction du type de calcul
76 // du sous type (pour les erreurs, remaillage etc...)
77 // il y a ici lecture des parametres attaches au type
78 AlgoristatExpli (const bool avec_typeDeCal
79                 ,const list <EnumSousTypeCalcul>& soustype
80                 ,const list <bool>& avec_soustypeDeCal
81                 ,UtilLecture& entreePrinc);
82 // constructeur de copie
83 AlgoristatExpli (const AlgoristatExpli& algo);
84
85 // constructeur de copie à partie d'une instance indifférenciée
86 Algori * New_idem(const Algori* algo) const
87 { // on vérifie qu'il s'agit bien d'une instance
88   if (algo->TypeDeCalcul() != STAT_DYNA_EXP)
89     { cout << "\n *** erreur lors de la creation par copie d'un algo STAT_DYNA_EXP "
90       << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
91       << " arret !! " << flush;
92     }
93   }
94   else
95     { AlgoristatExpli* inter = (AlgoristatExpli*) algo;
96     return ((Algori *) new AlgoristatExpli(*inter));
97     };
98 };
99
100 // DESTRUCTEUR :
101 ~AlgoristatExpli () ;
102
103 // METHODES PUBLIQUES :
104 // execution de l'algorithme explicite dans le cas dynamique sans contact
105 void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID* ,LesFonctions_nD*
106               ,VariablesExporter* varExpor,LesLoisDeComp*
107               ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
108
109 //----- décomposition en 3 du calcul d'équilibre -----
110 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
111 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre
112 // deux calculs
113 // certaines variables ont-été changés
114
115 // pas opérationnel pour l'instant !
116
117 // initialisation
118 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
119                   ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
120                   ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
121
122 // mise à jour
123 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
124                   ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
125                   ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* )
126 { cout << "\n MiseAJourAlgo(.. pas encore operationnelle ";
127   Sortie(1);
128 };
129
130 // calcul de l'équilibre
131 // si tb_combiner est non null -> un tableau de 2 fonctions
132 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
133 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
134 //
135 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
136 // en fonction de la demande de sauvegard,
137 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
138 // qui lui fonctionne indépendamment
139 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
140                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
141                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
142                  ,Tableau < Fonction_nD* > * tb_combiner) {};
143
144 // dernière passe
145 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
146               ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
147               ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
148
149 // sortie du schemaXML: en fonction de enu
150 //**** inexploitable pour l'instant
151 void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const {};
152
153 protected :
154 // VARIABLES PROTEGEES :
155 //---- les deux instances de classe : statique et dynamique
156 Algori * algoStat;
157 Algori * algoExp;
158
159 // vecteurs globaux définissant les vitesses intermédiaires
160 Vecteur vitesse_tMoinsUnDemi,vitesse_tPlusUnDemi;
161
162 // liste de variables de travail déclarées ici pour éviter le passage de paramètre entre les
163 // méthodes internes à la classe
164 double delta_t,unsurdeltat,deltatSurDeux,deltat2,deltat2SurDeux;

```

```

160
161 int type_cal_equilibre; // para pour le choix entre Calcul_Equilibre et Calcul_Equilibre2
162
163 // -----
164 // -- variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
165 // -----
166
167 // === pointeurs d'instance et classe particulières
168 Assemblage * Ass1_, * Ass2_, * Ass3_; // pointeurs d'assemblages
169
170 // === variables scalaires
171 int cas_combi_ddl; // def combinaison des ddl
172 int icas; // idem cas_combi_ddl mais pour lesCondlim
173 bool prepa_avec_remont; // comme son nom l'indique
174 bool brestart; // booleen qui indique si l'on est en restart ou pas
175 OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
176
177 // === vecteurs
178 Vecteur vglobin; // puissance interne : pour ddl accélération
179 Vecteur vglobex; // puissance externe
180 Vecteur vglobaal; // puissance totale
181 Vecteur vcontact; // puissance des forces de contact
182 Vecteur X_Bl,V_Bl,G_Bl; // stockage transitoirement des X V GAMMA <-> CL
183 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
184
185 // === les listes
186 list <LesCondlim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués
187 // === les tableaux
188 Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
189 Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
190 // === les matrices
191 Mat_abstraite* mat_masse,* mat_masse_sauve; // choix de la matrice de masse
192 Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
193
194 // -----
195 // -- fin variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
196 // -----
197
198
199 // METHODES PROTEGEES :
200 void Calcul_Equilibre
201     (ParaGlob * paraGlob,LesMaillages * lesMail,
202     LesReferences* lesRef,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD
203     ,VariablesExporter* varExpor
204     ,LesLoisDeComp* lesLoisDeComp,
205     DiversStockage* diversStockage,Charge* charge,
206     LesCondlim* lesCondlim,LesContacts* lesContacts,Resultats* resultats);
207 // nouvelle algo avec un passage à vide au premier passage pour être plus conforme à la dfc
classique
208 //**** inexploitable pour l'instant
209 void Calcul_Equilibre2
210     (ParaGlob * paraGlob,LesMaillages * lesMail,
211     LesReferences* lesRef,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD
212     ,VariablesExporter* varExpor
213     ,LesLoisDeComp* lesLoisDeComp,
214     DiversStockage* diversStockage,Charge* charge,
215     LesCondlim* lesCondlim,LesContacts* lesContacts,Resultats* resultats) {};
216 // lecture des paramètres du calcul
217 //**** inexploitable pour l'instant
218 void lecture_Parametres(UtilLecture& entreePrinc){};
219 // écriture des paramètres dans la base info
220 // = 1 : on écrit tout
221 // = 2 : on écrit uniquement les données variables (supposées comme telles)
222 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
223 // lecture des paramètres dans la base info
224 // = 1 : on récupère tout
225 // = 2 : on récupère uniquement les données variables (supposées comme telles)
226 // choix = true : fonctionnement normal
227 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
228 // car la lecture est impossible
229 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
230 // création d'un fichier de commande: cas des paramètres spécifiques
231 //**** inexploitable pour l'instant
232 void Info_commande_parametres(UtilLecture& entreePrinc){};
233
234 // gestion et vérification du pas de temps et modif en conséquence si nécessaire
235 // cas = 1: initialisation du pas de temps et vérif / au pas de temps critique
236 // cas = 2: vérif / au pas de temps critique
237 void Gestion_pas_de_temps(LesMaillages * lesMail,int cas);
238
239 //---- gestion des commndes interactives -----
240 // écoute et prise en compte d'une commande interactive
241 // ramène true tant qu'il y a des commandes en cours
242 //**** inexploitable pour l'instant
243 bool ActionInteractiveAlgo(){};
244

```

```

245
246
247
248
249
250
251 };
252 /// @} // end of group
253
254 #endif

```

## 7.15 AlgoriFlambLineaire.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           23/01/97
31 *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:        Herezh++
35 *
36 *   *****
37 *   BUT:           Algorithme de calcul pour le flambement linéaire, le
38 *                  calcul préliminaire est non dynamique, pour de la mecanique
39 *                  en coordonnees materielles entrainees.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date ! auteur ! but
46 *   -----
47 *   ! ! !
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date ! auteur ! but
51 *   -----
52 *
53 *   *****/
54 #ifndef AGORIFLAMBLINEAIRE_T
55 #define AGORIFLAMBLINEAIRE_T
56
57
58 #include "Algori.h"
59 #include "MatBand.h"
60 #include "Assemblage.h"
61
62 /// @addtogroup Les_algorithmes_de_resolutions_globales
63 /// @{
64 ///
65
66 /// BUT: Algorithme de calcul pour le flambement linéaire, le
67 /// calcul préliminaire est non dynamique, pour de la mecanique
68 /// en coordonnees materielles entrainees.
69
70 class AlgoriFlambLineaire : public Algori
71 {

```



```

72 public :
73 // CONSTRUCTEURS :
74 AlgoriFlambLineaire () ; // par default
75
76 // constructeur en fonction du type de calcul
77 // du sous type (pour les erreurs, remaillage etc...)
78 // il y a ici lecture des parametres attaches au type
79 AlgoriFlambLineaire (const bool avec_typeDeCal
80 ,const list <EnumSousTypeCalcul>& soustype
81 ,const list <bool>& avec_soustypeDeCal
82 ,UtilLecture& entreePrinc);
83 // constructeur de copie
84 AlgoriFlambLineaire (const AlgoriFlambLineaire& algo);
85
86 // constructeur de copie à partie d'une instance indifférenciée
87 Algori * New_idem(const Algori* algo) const
88 { // on vérifie qu'il s'agit bien d'une instance
89   if (algo->TypeDeCalcul() != FLAMB_LINEAIRE)
90     { cout << "\n *** erreur lors de la creation par copie d'un algo FLAMB_LINEAIRE "
91       << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
92       << " arret !! " << flush;
93     }
94   }
95   else
96     { AlgoriFlambLineaire* inter = (AlgoriFlambLineaire*) algo;
97     return ((Algori *) new AlgoriFlambLineaire(*inter));
98     };
99 };
100
101 // DESTRUCTEUR :
102 ~AlgoriFlambLineaire () ;
103
104 // METHODES PUBLIQUES :
105 // execution de l'algorithme dans le cas non dynamique
106 void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID* ,LesFonctions_nD*
107 ,VariablesExporter* varExpor,LesLoisDeComp*
108 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
109
110 //----- décomposition en 3 du calcul d'équilibre -----
111 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
112 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre
113 // deux calculs
114 // certaines variables ont-été changés
115
116 // pas opérationnel pour l'instant !
117
118 // initialisation
119 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
120 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
121 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
122
123 // mise à jour
124 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
125 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
126 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* )
127 {cout << "\n MiseAJourAlgo(.. pas encore operationnelle ";
128   Sortie(1);
129 };
130
131 // calcul de l'équilibre
132 // si tb_combiner est non null -> un tableau de 2 fonctions
133 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
134 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
135 //
136 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
137 // en fonction de la demande de sauvegard,
138 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
139 // qui lui fonctionne indépendamment
140 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
141 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
142 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
143 ,Tableau < Fonction_nD* > * tb_combiner){};
144
145 // dernière passe
146 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
147 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
148 ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* ) {};
149
150 // sortie du schemaXML: en fonction de enu
151 void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const {};
152
153 private :
154 // == vecteurs
155 Vecteur vglobin; // puissance interne : pour ddl accélération
156 Vecteur vglobex; // puissance externe
157 Vecteur vcontact; // puissance des forces de contact
158 Vecteur vglobaal; // puissance totale qui écrase vglobin
159 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
160 // VARIABLES PROTEGEES :
161 // Concernant les paramètres spécifiques à l'algorithme :
162 // Le premier paramètre correspond au type de méthode employée pour

```

```

158 // la recherche de valeur propre : 1 pour la méthode QR, 2 pour les itérations inverses.
159 // Le second paramètre correspond au nombre de valeur propre et vecteur propre
160 // désiré.
161 double * methode,* nbValPropre;
162
163 // === les matrices
164 Mat_abstraite* matglob; // choix de la matrice de raideur
165 // choix des matrices de raideur de substitution éventuelles : par défaut matglob = tab_mato(1)
166 Tableau < Mat_abstraite> tab_mato;
167
168 // METHODES PROTEGEES :
169 // lecture des paramètres du calcul
170 void lecture_Paramètres(UtilLecture& entreePrinc);
171 // écriture des paramètres dans la base info
172 // = 1 : on écrit tout
173 // = 2 : on écrit uniquement les données variables (supposées comme telles)
174 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas) ;
175 // lecture des paramètres dans la base info
176 // = 1 : on récupère tout
177 // = 2 : on récupère uniquement les données variables (supposées comme telles)
178 // choix = true : fonctionnement normal
179 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
180 // car la lecture est impossible
181 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix) ;
182 // création d'un fichier de commande: cas des paramètres spécifiques
183 void Info_commande_parametres(UtilLecture& entreePrinc);
184 };
185 /// @} // end of group
186
187 #endif

```

## 7.16 AlgoriNonDyna.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *      DATE:      23/01/97
31 *
32 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *      PROJET:    Herezh++
35 *
36 *
37 *      BUT:      Algorithme de calcul non dynamique, pour de la mecanique
38 *                du solide déformable en coordonnees materielles
39 *                entraînées.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date !   auteur !           but
45 *      -----
46 *      !           !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but

```

```

51 * ----- *
52 *                                     $ *
53 *****/
54 #ifndef AGORINONDYNA_T
55 #define AGORINONDYNA_T
56
57
58 #include "Algori.h"
59 #include "Assemblage.h"
60 #include "MatLapack.h"
61
62 /// @addtogroup Les_algorithmes_de_resolutions_globales
63 /// @{
64 ///
65
66 ///     BUT:   Algorithme de calcul non dynamique, pour de la mecanique
67 ///           du solide déformable en coordonnees materielles
68 ///           entrainees.
69
70 class AlgoriNonDyna : public Algori
71 {
72 public :
73     // CONSTRUCTEURS :
74     AlgoriNonDyna () ; // par default
75
76     // constructeur en fonction du type de calcul
77     // du sous type (pour les erreurs, remaillage etc...)
78     // il y a ici lecture des parametres attaches au type
79     AlgoriNonDyna (const bool avec_typeDeCal
80                  ,const list <EnumSousTypeCalcul>& soustype
81                  ,const list <bool>& avec_soustypeDeCal
82                  ,UtilLecture& entreePrinc);
83     // constructeur de copie
84     AlgoriNonDyna (const AlgoriNonDyna& algo);
85
86     // constructeur de copie à partie d'une instance indifférenciée
87     Algori * New_idem(const Algori* algo) const
88     { // on vérifie qu'il s'agit bien d'une instance
89     if (algo->TypeDeCalcul() != NON_DYNA)
90     { cout << "\n *** erreur lors de la creation par copie d'un algo NON_DYNA "
91         << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
92         << " arret !! " << flush;
93         Sortie(1);
94     }
95     else
96     { AlgoriNonDyna* inter = (AlgoriNonDyna*) algo;
97       return ((Algori *) new AlgoriNonDyna(*inter));
98     };
99 };
100
101 // DESTRUCTEUR :
102 ~AlgoriNonDyna () ;
103
104 // METHODES PUBLIQUES :
105 // execution de l'algorithme dans le cas non dynamique
106 void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D* ,LesFonctions_nD*
107              ,VariablesExporter* varExpor ,LesLoisDeComp*
108              ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
109
110 //----- décomposition en 3 du calcul d'équilibre -----
111 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
112 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
calcul
113 //          certaines variables ont-été changés
114
115 // initialisation
116 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
117                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
118                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
119
120 // mise à jour
121 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
122                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
123                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
124
125 // calcul de l'équilibre
126 // si tb_combiner est non null -> un tableau de 2 fonctions
127 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
128 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
129 //
130 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
131 // en fonction de la demande de sauvegard,
132 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
133 // qui lui fonctionne indépendamment
134 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbes1D*
135                  ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
136                  ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
137                  ,Tableau < Fonction_nD* > * tb_combiner);
138
139 // dernière passe

```

```

137 void FinCalcul(ParaGlob * ,LesMaillages *,LesReferences*,LesCourbesID*
138               ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
139               ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* );
140
141 // sortie du schemaXML: en fonction de enu
142 void SchemaXML_Algori(ofstream& ,const Enum_IO_XML ) const ;
143
144 private :
145
146 // VARIABLES PROTEGEES :
147 // pilotage d'un newton modifié éventuel
148 int deb_newton_modifie; // NB iter de debut d'utilisation de NM
149 int fin_newton_modifie; // NB iter de fin de la méthode et retour N classique
150 int nb_iter_NM; // NB d'iter en NM
151 // pilotage de l'utilisation d'une matrice moyenne
152 int deb_raideur_moyenne; // NB iter de debut
153 int fin_raideur_moyenne; // NB iter de fin de la méthode
154 int nb_raideur_moyenne; // NB de matrice pour la moyenne
155
156 // indicateur disant s'il faut calculer les conditions limites à chaque itération on non
157 int cL_a_chaque_iteration; // par défaut, non, == uniquement à chaque début d'incrément
158
159 // -----
160 // -- variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
161 // -----
162
163 // === pointeurs d'instance et classe particulières
164 Assemblage * Ass_; // pointeur d'assemblage
165
166 // === variables scalaires
167 int cas_combi_ddl; // def combinaison des ddl
168 int icas; // idem cas_combi_ddl mais pour lesCondlim
169 int compteur; // compteur d'itération
170
171 bool prepa_avec_remont; // comme son nom l'indique
172 bool brestart; // booleen qui indique si l'on est en restart ou pas
173 OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
174
175 // === vecteurs
176 Vecteur vglobin; // puissance interne : pour ddl accélération
177 Vecteur vglobex; // puissance externe
178 Vecteur vcontact; // puissance des forces de contact
179 Vecteur vglobaal; // puissance totale qui écrase vglobin
180 Vecteur delta_prec_X; // les positions
181 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
182
183 // === les matrices
184 Mat_abstraite* matglob; // choix de la matrice de raideur
185 Mat_abstraite* matsauve; // sauvegarde pour le newton modifié
186 Tableau <Mat_abstraite*>* tab_matmoysauve; // sauvegarde pour les moyennes de raideur
187
188 // --- cas où l'on a des matrices secondaires pour switcher si la première matrice
189 // ne fonctionne pas bien, par défaut matglob, matsauve et Tab_matmoysauve sont les premiers
190 // éléments des tableaux
191 // choix des matrices de raideur de substitution éventuelles : par défaut matglob = tab_mato(1)
192 Tableau < Mat_abstraite*> tab_mato;
193 // matrices de substitution pour le newton modifié: par défaut matsauve = tab_matsauve(1)
194 Tableau < Mat_abstraite*> tab_matsauve;
195 // tab de matrices de substitution pour le niveau de substitution (i)
196 // c-a-d que l'on boucle sur j pour la moyenne avec (*tab_tab_matmoysauve(j))(i)
197 // -> j est le premier indice !!
198 Tableau < Tableau <Mat_abstraite*> * > tab_tab_matmoysauve; // sauvegarde pour les moyennes de
raideur
199
200 void (Assemblage::* assembMat) // un pointeur de fonction d'assemblage
201     (Mat_abstraite & matglob,const Mat_abstraite & matloc,
202      const DdlElement& tab_ddl,const Tableau<Noeud *>&tab_noeud);
203 // dans le cas où l'on fait du line search on dimensionne des vecteurs globaux supplémentaires
204 Vecteur * sauve_deltadept,*sauve_dept_a_tdt,*vres,*v_travail;
205
206 // -----
207 // -- fin variables de transferts internes entre: InitAlgorithme, CalEquilibre, FinCalcul --
208 // -----
209
210 // --- accélération de convergence -----
211 Tableau <Vecteur> Vi,Yi; // les vecteurs projection et accroissement d'une itération à l'autre
212 Vecteur coef_ai; // coefficients de la série
213 MatLapack mat_ai; // matrice pour le calcul des coef_ai
214 Vecteur Sm_ai; // second membre de mat_ai
215 Vecteur X_extrapol; // la position extrapolée
216 Vecteur S_0; // la solution initiale
217 int nb_vec_cycle; // nombre de vecteur maxi pour le cycle
218 int cas_acceleration_convergence; // classe les différents cas:
219 // = 1: les vecteurs de projection sont les positions Xtdt
220 // = 2: les vecteurs de projection sont les résidus
221 bool acceleration_convergence; // indique si oui ou non on veut une accélération de convergence
222 Vecteur sauve_sol; // sauvegarde de la solution de la résolution de l'équilibre

```

```

223
224 // METHODES PROTEGEES :
225
226 // lecture des paramètres du calcul
227 void lecture_Paramètres(UtilLecture& entreePrinc);
228 // écriture des paramètres dans la base info
229 // = 1 : on écrit tout
230 // = 2 : on écrit uniquement les données variables (supposées comme telles)
231 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc, const int& cas);
232 // lecture des paramètres dans la base info
233 // = 1 : on récupère tout
234 // = 2 : on récupère uniquement les données variables (supposées comme telles)
235 // choix = true : fonctionnement normal
236 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
237 // car la lecture est impossible
238 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc, const int& cas, bool choix);
239 // création d'un fichier de commande: cas des paramètres spécifiques
240 void Info_commande_parametres(UtilLecture& entreePrinc);
241
242
243 // Calcul de l'équilibre de la pièce avec pilotage de longueur d'arc
244 void Calcul_Equilibre_longueur_arc
245     (ParaGlob * paraGlob, LesMaillages * lesMail,
246      LesReferences* lesRef, LesCourbesID* lesCourbesID, LesFonctions_nD* lesFonctionsnD
247      , VariablesExporter* varExpor, LesLoisDeComp* lesLoisDeComp,
248      DiversStockage* diversStockage, Charge* charge,
249      LesCondLim* lesCondLim, LesContacts* lesContacts, Resultats* resultats);
250
251 // accélération de convergence, suivant une méthode MMPE (modified minimal polynomial
extrapolation)
252 // accélération de convergence, suivant une méthode MMPE (modified minimal polynomial
extrapolation)
253 // compteur : compteur d'itération
254 // retour = 0: l'accélération n'est pas active (aucun résultat, aucune action, aucune sauvegarde)
255 //          = 1: phase de préparation: sauvegarde d'info, mais aucun résultat
256 //          = 2: calcul de l'accélération mais ... pas d'accélération constatée
257 //          = 3: calcul de l'accélération et l'accélération est effective
258 // dans le cas différent de 0 et 1, le programme a priori, change le stockage des conditions
259 // limites, en particulier CL linéaire, mais avec des hypothèses, pour les sorties de réactions
260 // etc, il vaut mieux refaire une boucle normale avec toutes les initialisation ad hoc
261 // affiche: indique si l'on veut l'affichage ou non des infos
262 int AccelerationConvergence(bool affiche, const Vecteur& vres, LesMaillages * lesMail
263     , const int& compteur, const Vecteur& X_t, const Vecteur& delta_X, Charge* charge
264     , Vecteur& vglobex, Assemblage& Ass, const Vecteur& delta_delta_X
265     , LesCondLim* lesCondLim, Vecteur& vglobal
266     , LesReferences* lesRef, Vecteur & vglobin, const Nb_assemb& nb_casAssemb
267     , int cas_combi_ddl, LesCourbesID* lesCourbesID, LesFonctions_nD* lesFonctionsnD);
268
269 //---- gestion des commndes interactives -----
270 // écoute et prise en compte d'une commande interactive
271 // ramène true tant qu'il y a des commandes en cours
272 bool ActionInteractiveAlgo();
273
274 };
275 /// @} // end of group
276
277 #endif

```

## 7.17 ImpliNonDynaCont.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //

```

```

24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      23/01/97
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *****/
37 *   BUT:      Algorithme de calcul implicite non dynamique avec contact
38 *            , pour de la mecanique en coordonnees materielles entrainees.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date !   auteur !           but
44 *   -----
45 *   !         !           !
46 *
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date !   auteur !           but
50 *   -----
51 *
52 *****/
53 #ifndef IMPLINONDYNACONT_T
54 #define IMPLINONDYNACONT_T
55
56
57 #include "Algori.h"
58 #include "MatBand.h"
59 #include "Assemblage.h"
60
61 /// @addtogroup Les_algorithmes_de_resolutions_globales
62 /// @{
63 ///
64
65 ///   BUT:      Algorithme de calcul implicite non dynamique avec contact
66 ///   , pour de la mecanique en coordonnees materielles entrainees.
67
68 class ImpliNonDynaCont : public Algori
69 {
70 public :
71   // CONSTRUCTEURS :
72   ImpliNonDynaCont () ; // par default
73
74   // constructeur en fonction du type de calcul
75   // il y a ici lecture des parametres attaches au type
76   ImpliNonDynaCont (const bool avec_typeDeCal
77                   ,const list <EnumSousTypeCalcul>& soustype
78                   ,const list <bool>& avec_soustypeDeCal
79                   ,UtilLecture& entreePrinc);
80   // constructeur de copie
81   ImpliNonDynaCont (const ImpliNonDynaCont& algo);
82
83   // constructeur de copie à partie d'une instance indifférenciée
84   Algori * New_idem(const Algori* algo) const
85   { // on vérifie qu'il s'agit bien d'une instance
86     if (algo->TypeDeCalcul() != NON_DYNA_CONT)
87       { cout << "\n *** erreur lors de la creation par copie d'un algo NON_DYNA_CONT "
88         << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
89         << " arret !! " << flush;
90       }
91     Sortie(1);
92   }
93   else
94   { ImpliNonDynaCont* inter = (ImpliNonDynaCont*) algo;
95     return ((Algori *) new ImpliNonDynaCont(*inter));
96   }
97
98   // DESTRUCTEUR :
99   ~ImpliNonDynaCont () ;
100
101   // METHODES PUBLIQUES :
102   // execution de l'algorithme dans le cas non dynamique avec prise en
103   // compte du contact
104   void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID * ,LesFonctions_nd*
105                ,VariablesExporter* varExpor ,LesLoisDeComp*
106                ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
107
108   //----- décomposition en 3 du calcul d'équilibre -----
109   // a priori   : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,

```

```

110 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre
// deux calculs
111 //          certaines variables ont-été changés
112
113 // pas opérationnel pour l'instant !
114
115 // initialisation
116 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbesID*
117 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
118 ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* ) {};
119
120 // mise à jour
121 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbesID*
122 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
123 ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* )
124 {cout << "\n MiseAJourAlgo(.. pas encore operationnelle ";
125   Sortie(1);
126 };
127 // calcul de l'équilibre
128 // si tb_combiner est non null -> un tableau de 2 fonctions
129 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
130 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
131 //
132 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
133 // en fonction de la demande de sauvegard,
134 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
135 // qui lui fonctionne indépendamment
136 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbesID*
137 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
138 ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats*
139 ,Tableau < Fonction_nD* > * tb_combiner) {};
140 // dernière passe
141 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences*,LesCourbesID*
142 ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
143 ,DiversStockage*,Charge*,LesCondLim*,LesContacts*,Resultats* ) {};
144 // sortie du schemaXML: en fonction de enu
145 void SchemaXML_Algori(ofstream& ,const Enum_IO_XML ) const {};
146
147 protected:
148 // === vecteurs
149 Vecteur vglobin; // puissance interne : pour ddl accélération
150 Vecteur vglobex; // puissance externe
151 Vecteur vcontact; // puissance des forces de contact
152 Vecteur vglobaal; // puissance totale qui ecrase vglobin
153 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
154
155 private :
156 // METHODES PROTEGEES :
157 // écriture des paramètres dans la base info (ici rien)
158 void Ecrit_Base_info_Parametre(UtilLecture& ,const int& ) {};
159 // lecture des paramètres dans la base info (ici rien)
160 // choix = true : fonctionnement normal
161 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
162 // défaut
163 //          car la lecture est impossible
164 void Lecture_Base_info_Parametre(UtilLecture& ,const int& ,bool ) {};
165 // création d'un fichier de commande: cas des paramètres spécifiques
166 void Info_commande_parametres(UtilLecture& entreePrinc) {Algori::Info_com_parametres(entreePrinc)};
167
168 // Calcul de l'équilibre de la pièce
169 void Calcul_Equilibre
170 (ParaGlob * paraGlob,LesMaillages * lesMail,
171 LesReferences* lesRef,LesCourbesID* lesCourbesID,LesFonctions_nD* lesFonctionsnD
172 ,VariablesExporter* varExpor
173 ,LesLoisDeComp* lesLoisDeComp,
174 DiversStockage* diversStockage,Charge* charge,
175 LesCondLim* lesCondLim,LesContacts* lesContacts,Resultats* resultats);
176 };
177 /// @} // end of group
178 #endif

```

## 7.18 AlgoBonelli.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)

```

```

12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           28/07/2006
32 *
33 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:        Herezh++
36 *
37 *   *****
38 *   BUT:           Algorithmme de calcul dynamique explicite, pour de la
39 *                 mecanique du solide déformable en coordonnees materielles*
40 *                 entrainees.
41 *                 Correspond à l'implantation de l'algo de Bonelli et Bursi.*
42 *                 Le modèle est de type galerkin discontinu en tempsP1-P1, *
43 *                 pour la discrétisation. L'algo est de type prédiction
44 *                 suivi de 1 ou plusieurs cycle de correction.
45 *                 Il est explicite, dans le sens où on n'utilise pas de
46 *                 comportement tangent du matériau et que l'on contrôle
47 *                 exactement le nombre d'itération du processus.
48 *
49 *
50 *   *****
51 *   VERIFICATION:
52 *
53 *   ! date ! auteur ! but
54 *   -----
55 *   ! ! !
56 *   *****
57 *   MODIFICATIONS:
58 *   ! date ! auteur ! but
59 *   -----
60 *
61 *   *****/
62
63 #ifndef AGORI_BONELLI_H
64 #define AGORI_BONELLI_H
65
66
67 #include "Algori.h"
68 #include "Assemblage.h"
69 #include "GeomSeg.h"
70
71 /// @addtogroup Les_algorithmes_de_resolutions_globales
72 /// @{
73 ///
74
75 /// BUT:   Algorithmme de calcul dynamique explicite, pour de la
76 ///       mecanique du solide déformable en coordonnees materielles
77 ///       entrainees.
78 ///       Correspond à l'implantation de l'algo de Bonelli et Bursi.
79 ///       Le modèle est de type galerkin discontinu en tempsP1-P1,
80 ///       pour la discrétisation. L'algo est de type prédiction
81 ///       suivi de 1 ou plusieurs cycle de correction.
82 ///       Il est explicite, dans le sens où on n'utilise pas de
83 ///       comportement tangent du matériau et que l'on contrôle
84 ///       exactement le nombre d'itération du processus.
85
86 class AlgoBonelli : public Algori
87 {
88 public :
89     // CONSTRUCTEURS :
90     AlgoBonelli () ; // par default
91
92     // constructeur en fonction du type de calcul
93     // du sous type (pour les erreurs, remaillage etc...)
94     // il y a ici lecture des parametres attaches au type
95     AlgoBonelli (const bool avec_typeDeCal
96                 ,const list <EnumSousTypeCalcul>& soustype
97                 ,const list <bool>& avec_soustypeDeCal

```



```

98         ,UtilLecture& entreePrinc);
99 // constructeur de copie
100 AlgoBonelli (const AlgoBonelli& algo);
101
102 // constructeur de copie à partie d'une instance indifférenciée
103 Algori * New_idem(const Algori* algo) const
104 { // on vérifie qu'il s'agit bien d'une instance
105     if (algo->TypeDeCalcul() != DYNA_EXP_BONELLI)
106     { cout << "\n *** erreur lors de la creation par copie d'un algo DYNA_EXP_BONELLI "
107         << " l'algo passe en parametre est en fait : " << Nom_TypeCalcul(algo->TypeDeCalcul())
108         << " arret !! " << flush;
109         Sortie(1);
110     }
111     else
112     { AlgoBonelli* inter = (AlgoBonelli*) algo;
113         return ((Algori *) new AlgoBonelli(*inter));
114     };
115 };
116
117 // DESTRUCTEUR :
118 ~AlgoBonelli () ;
119
120 // METHODES PUBLIQUES :
121 // execution de l'algorithme explicite dans le cas dynamique sans contact
122 void Execution(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID* ,LesFonctions_nD*
123     ,VariablesExporter* varExpor,LesLoisDeComp*
124     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
125
126 //----- décomposition en 3 du calcul d'équilibre -----
127 // a priori : InitAlgorithme et FinCalcul ne s'appellent qu'une fois,
128 // par contre : CalEquilibre peut s'appeler plusieurs fois, le résultat sera différent si entre deux
129 // calcul
130 // certaines variables ont-été changés
131
132 // initialisation
133 void InitAlgorithme(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
134     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
135     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
136
137 // mise à jour
138 void MiseAJourAlgo(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
139     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
140     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
141
142 // calcul de l'équilibre
143 // si tb_combiner est non null -> un tableau de 2 fonctions
144 // - la première fct dit si on doit valider ou non le calcul à convergence ok,
145 // - la seconde dit si on doit sortir de la boucle ou non à convergence ok
146 //
147 // si la validation est effectuée, la sauvegarde pour le post-traitement est également effectuée
148 // en fonction de la demande de sauvegard,
149 // sinon pas de sauvegarde pour le post-traitement à moins que l'on a demandé un mode debug
150 // qui lui fonctionne indépendamment
151 void CalEquilibre(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
152     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
153     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats*
154     ,Tableau < Fonction_nD* > * tb_combiner);
155
156 // dernière passe
157 void FinCalcul(ParaGlob * ,LesMaillages * ,LesReferences* ,LesCourbesID*
158     ,LesFonctions_nD* ,VariablesExporter* ,LesLoisDeComp*
159     ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts* ,Resultats* );
160
161 // sortie du schemaXML: en fonction de enu
162 void SchemaXML_Algori(ofstream& sort,const Enum_IO_XML enu) const ;
163
164 protected :
165 // VARIABLES PROTEGEES :
166 // paramètres de l'algorithme
167 double a_a,b_b;
168 int k_max;
169 double omega_b; // omega de bifurcation < omega critique
170 double rho_b; // rayon spectral à la bifurcation
171 GeomSeg* pt_seg_temps; // un élément segment pour l'intégration temporelle avec gauss
172 int nb_pt_int_t; // le nombre de points d'intégration temporelle
173
174 // liste de variables de travail déclarées ici pour éviter le passage de paramètre entre les
175 // méthodes internes à la classe
176 double delta_t; // pas de temps instantané
177 double deltat2,unsurdeltat,deltatSurDeux; // variable intermédiaire temps instantané
178 double delta_t_total; // pas de temps total
179 double deltat2_total,unsurdeltat_total,deltatSurDeux_total; // variable intermédiaire temps totales
180
181 // pour clarifier la lecture des différentes phases pour la mise en place des CL sur ddl
182 enum enuTypePhase {PREDICTION_bonelli, INTEGRATION_bonelli
183     , CORRECTION_bonelli, FIN_DELTA_T_bonelli };
184
185 // ----- variables de transferts internes -----
186 // == pointeurs d'instance et classe particulières

```

```

184 LesMaillages * lesMail_;
185 LesReferences* lesRef_;
186 LesCourbes1D* lesCourbes1D_;
187 LesFonctions_nD* lesFonctionsnD_ ;
188 Charge* charge_;
189 LesCondLim* lesCondLim_;
190 LesContacts* lesContacts_;
191 Assemblage * Ass1, * Ass2, * Ass3; // pointeurs d'assemblages
192
193 // === variables scalaires
194 double maxPuissExt; // maxi de la puissance des efforts externes
195 double maxPuissInt; // maxi de la puissance des efforts internes
196 double maxReaction; // maxi des reactions
197 int inReaction; // pointeur d'assemblage pour le maxi de reaction
198 int inSol; // pointeur d'assemblage du maxi de variation de ddl
199 double maxDeltaDdl; // maxi de variation de ddl
200 int cas_combi_ddl; // def combinaison des ddl
201 int icas; // idem cas_combi_ddl mais pour lesCondlim
202 bool erreurSecondMembre; // pour la gestion des erreurs de calcul au second membre
203 bool prepa_avec_remont; // comme son nom l'indique
204 bool brestart; // booleen qui indique si l'on est en restart ou pas
205 OrdreVisu::EnumTypeIncre type_incre; // pour la visualisation au fil du calcul
206
207 // === vecteurs
208 Vecteur q_0,p_0; // position et quantité de mouvement à ti^{ - }
209 Vecteur q_1,p_1; // position et quantité de mouvement à ti^{ + }
210 Vecteur q_2,p_2; // position et quantité de mouvement à ti+1^{ - }
211
212 Vecteur pPoint_i; // quantité de mouvement à ti^{ - }
213 Vecteur P_i_1,P_i_2,P_q1_k,P_q2_k; // inter
214 Vecteur r_1_k,r_2_k; // les résidus
215 Vecteur vec_travail,vec_travail2; // vecteurs de travail
216 Vecteur dernier_phi; // fonction d'interpolation temporelle au dernier pt d'integ en temps
217
218
219 Vecteur vglobin; // puissance interne : pour ddl accélération
220 Vecteur vglobex; // puissance externe
221 Vecteur vglobaal; // puissance totale qui écrase vglobin
222 Vecteur vcontact; // puissance des forces de contact
223 Vecteur vitesse_tplus; // vitesses
224 Vecteur X_Bl,V_Bl,G_Bl; // stockage transitoirement des X V GAMMA <-> CL
225 Vecteur forces_vis_num; // forces visqueuses d'origines numériques
226
227 // === les listes
228 list <LesCondLim::Gene_asso> li_gene_asso; // tableaux d'indices généraux des ddl bloqués
229 // === les tableaux
230 Tableau <Nb_assemb> t_assemb; // tableau globalisant les numéros d'assemblage de X V gamma
231 Tableau <Enum_ddl> tenuXVG; // les enum des inconnues
232 // === les matrices
233 Mat_abstraite* mat_masse,* mat_masse_sauve; // choix de la matrice de masse
234 Mat_abstraite* mat_C_pt; // matrice visqueuse numérique
235
236
237 // METHODES PROTEGEES :
238
239 // lecture des paramètres du calcul
240 void lecture_Parametres(UtilLecture& entreePrinc);
241 // écriture des paramètres dans la base info
242 // = 1 : on écrit tout
243 // = 2 : on écrit uniquement les données variables (supposées comme telles)
244 void Ecrit_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas);
245 // lecture des paramètres dans la base info
246 // = 1 : on récupère tout
247 // = 2 : on récupère uniquement les données variables (supposées comme telles)
248 // choix = true : fonctionnement normal
249 // choix = false : la méthode ne doit pas lire mais initialiser les données à leurs valeurs par
défaut
250 // car la lecture est impossible
251 void Lecture_Base_info_Parametre(UtilLecture& entreePrinc,const int& cas,bool choix);
252 // création d'un fichier de commande: cas des paramètres spécifiques
253 void Info_commande_parametres(UtilLecture& entreePrinc);
254
255 // gestion et vérification du pas de temps et modif en conséquence si nécessaire
256 // cas = 0: premier passage à blanc, delta t = 0
257 // cas = 1: initialisation du pas de temps et vérif / au pas de temps critique
258 // ceci pour le temps t=0
259 // cas = 2: initialisation du pas de temps et vérif / au pas de temps critique
260 // ceci pour le temps t. et on divise par nbstep
261 // ceci pour garantir que l'on fait le calcul avec 1 step
262 void Gestion_pas_de_temps(LesMaillages * lesMail,int cas,int nbstep);
263
264 // modification transitoire du pas de temps et modif en conséquence si nécessaire
265 // utilisée pour les différentes intégrales temporelles
266 // delta_tau : nouveau pas de temps transitoire imposé
267 void Modif_transi_pas_de_temps(double delta_tau);
268
269 // calcul des ddl avec prise en comptes des conditions limites suivant une technique

```

```

270 // s'appuyant sur une différence finie décentrée à droite
271 // utilise les vecteurs globaux : q_2, q_1, q_0
272 // --> mise à jour des ddl au niveau globaux sur les X, V, gamma puis au niveau des noeuds
273 // --> ou mise à jour des composantes adoc des vecteurs globaux q_2, q_1, q_0, selon "phasage"
274 // phii : interpolation en fonction du temps, ne sert que pour phasage = INTEGRATION_bonelli
275 // phasage =
276 // PREDICTION_bonelli: phase de prédiction, on n'utilise "pas" l'interpolation en temps (phii)
277 // --> calcul de q_1 et q_2 avec les conditions limites
278 // INTEGRATION_bonelli: phase d'intégration, "on utilise l'interpolation en temps"
279 // --> calcul de X_tdt et vitesse_tdt
280 // CORRECTION_bonelli: phase de correction, on n'utilise "pas" l'interpolation en temps (phii)
281 // --> calcul de q_1 et q_2 avec les conditions limites
282 // FIN_DELTA_T_bonelli: phase final, on n'utilise "pas" l'interpolation en temps (phii)
283 // --> calcul de X_tdt et vitesse_tdt pour le temps final
284 void AvanceDDL_avec_CL(const Vecteur & phii,enuTypePhase phasage);
285
286 // calcul des différences énergie en jeux dans le cas Bonelli
287 // affichage éventuelle des ces énergies et du bilan
288 // V : ddl de vitesse
289 // coef_mass : en fait il faut utiliser 1/coef_mass * mat_mass pour la matrice masse, ceci due à la
spécificité de l'algo
290 // les différentes énergie et efforts généralisées sont des variables internes à l'algo
291 // en sortie: énergie cinétique : E_cin_tdt, énergie interne : E_int_tdt, énergie externe =
E_ext_tdt, bilan : bilan_E
292 // idem avec les puissances
293 // énergie visqueuse due aux forces visqueuses numériques
294 // icharge : compteur d'increment
295 void CalEnergieAffichageBonelli(const double& coef_mass,int icharge);
296
297 //---- gestion des commndes interactives -----
298 // écoute et prise en compte d'une commande interactive
299 // ramène true tant qu'il y a des commandes en cours
300 bool ActionInteractiveAlgo();
301
302 };
303 /// @} // end of group
304
305 #endif

```

## 7.19 Hypo\_ortho3D\_entrainee.h

```

1 // FICHER : Hypo_ortho3D_entrainee.h
2 // CLASSE : Hypo_ortho3D_entrainee
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPIUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33
34 /*****
35 * DATE: 18/09/2019 *
36 * * * $ *
37 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
38 * * * $ *
39 * PROJET: Herezh++ *
40 * * * $ *
41 *****/
42 * BUT: *
43 * La classe Hypo_ortho3D_entrainee permet de calculer la contrainte *
44 * et ses variations pour une loi hypo-élastique orthotrope entraînée *

```

```

45 *   elastique en 3D. Le comportement est non linéaire, a priori correct*
46 *   quelque soit l'intensité des déformations.                               *
47 *   La loi nécessite la définition des coefficients classiques                 *
48 *   d'orthotropie et d'un repère particulier d'anisotropie. Le             *
49 *   repère est ensuite entraîné par la matière.                             *
50 *                                                                                   $ *
51 *   ***** *
52 *
53 *   VERIFICATION:
54 *
55 *   ! date !   auteur !           but
56 *   -----
57 *   !           !           !
58 *                                                                                   $ *
59 *   ***** *
60 *   MODIFICATIONS:
61 *   ! date !   auteur !           but
62 *   -----
63 *                                                                                   $ *
64 *****/
65
66
67 #ifndef HYPO_ORTHO3D_ENTRAINEE_H
68 #define HYPO_ORTHO3D_ENTRAINEE_H
69 #include "Base3D3.h"
70
71
72 #include "Loi_comp_abstraite.h"
73
74
75 /// @addtogroup Les_lois_anisotropes
76 /// @{
77 ///
78
79
80 class Hypo_ortho3D_entrainee : public Loi_comp_abstraite
81 {
82
83
84     public :
85
86         // CONSTRUCTEURS :
87
88         // Constructeur par défaut
89         Hypo_ortho3D_entrainee ();
90
91         // Constructeur fonction de tous les paramètres constants de la loi
92         Hypo_ortho3D_entrainee(const double& EE1,const double& EE2,const double& EE3
93             ,const double& nunu12,const double& nunu13,const double& nunu23
94             ,const double& GG12,const double& GG13,const double& GG23
95             ,const string& nom_rep);
96
97         // Constructeur de copie
98         Hypo_ortho3D_entrainee (const Hypo_ortho3D_entrainee& loi) ;
99
100        // DESTRUCTEUR :
101        ~Hypo_ortho3D_entrainee ();
102
103        // initialise les donnees particulieres a l'elements
104        // de matiere traite ( c-a-dire au pt calcule)
105        // Il y a creation d'une instance de SaveResul particuliere
106        // a la loi concernee
107        // la SaveResul classe est remplie par les instances heritantes
108        // le pointeur de SaveResul est sauvegarde au niveau de l'element
109        // c'a-d que les info particulieres au point considere sont stocke
110        // au niveau de l'element et non de la loi.
111        class SaveResulHypo_ortho3D_entrainee: public SaveResul
112        { public :
113            SaveResulHypo_ortho3D_entrainee(const int type_transport=0); // constructeur par défaut :
114            SaveResulHypo_ortho3D_entrainee(const SaveResulHypo_ortho3D_entrainee& sav); // de copie
115            virtual ~SaveResulHypo_ortho3D_entrainee(); // destructeur
116            // définition d'une nouvelle instance identique
117            // appelle du constructeur via new
118            SaveResul * Nevez_SaveResul() const{return (new SaveResulHypo_ortho3D_entrainee(*this));};
119            // affectation
120            virtual SaveResul & operator = ( const SaveResul & a);
121            //===== lecture écriture dans base info =====
122            // cas donne le niveau de la récupération
123            // = 1 : on récupère tout
124            // = 2 : on récupère uniquement les données variables (supposées comme telles)
125            void Lecture_base_info (ifstream& ent,const int cas);
126            // cas donne le niveau de sauvegarde
127            // = 1 : on sauvegarde tout
128            // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
129            void Ecriture_base_info(ofstream& sort,const int cas);
130
131            // mise à jour des informations transitoires

```

```

132     void TdtversT();
133     void TversTdt();
134
135     // affichage à l'écran des infos
136     void Affiche() const;
137
138     //changement de base de toutes les grandeurs internes tensorielles stockées
139     // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gp
140     // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
141     // gpH(i) = gamma(i,j) * gH(j)
142     virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
143
144     // procedure permettant de completer éventuellement les données particulières
145     // de la loi stockées
146     // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
147     // completer est appelé apres sa creation avec les donnees du bloc transmis
148     // peut etre appeler plusieurs fois
149     SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
150         ,const Loi_comp_abstraite* loi);
151
152     // ---- méthodes spécifiques
153     // initialise les informations de travail concernant le pas de temps en cours
154     void Init_debut_calcul();
155
156     //-----
157     // données
158     //-----
159     // - partie repère d'orthotropie
160     // 1) le repère lui-même
161     // soit O_B est non nul et O_H est NULL
162     // soit l'inverse,
163     // c'est celui qui est non nul, qui indique le type de convection
164     BaseB * O_B; // définit éventuellement les coordonnées covariante
165                 // convectées donc fixes, de O, dans gi_H_tdt
166     BaseH * O_H; // définit éventuellement les coordonnées contravariante
167                 // convectées donc fixes, de O, dans gi_B_tdt
168     // 2) le repère obtenu par convection -> O', exprimé
169     BaseH Op_H,Op_H_t; // définit les coordonnées contravariante
170                       // de la base transportée
171
172     // 2) les tenseurs intermédiaires
173     TenseurHH* eps_loc_HH; // def local dans le repère O'_a
174     TenseurHH* sig_loc_HH; // contrainte local dans le repère O'_a
175     TenseurHH* delta_eps_loc_HH; // delta def local dans le repère O'_a
176     TenseurHH* delta_sig_loc_HH; // delta contrainte local dans le repère O'_a
177
178     // les 9 paramètres de la loi dans l'ordre suivant
179     // double E1,E2,E3,nul2,nul3,nu23,G12,G13,G23; // paramètres de la loi
180     Vecteur *para_loi;
181
182 };
183
184 SaveResul * New_et_Initialise()
185 { SaveResulHypo_ortho3D_entrainee * pt = new SaveResulHypo_ortho3D_entrainee();
186   return pt;
187 };
188
189 friend class SaveResulHypo_ortho3D_entrainee;
190
191     // Lecture des donnees de la classe sur fichier
192     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
193         ,LesFonctions_nD& lesFonctionsnD);
194
195     // affichage de la loi
196     void Affiche() const ;
197     // test si la loi est complete
198     // = 1 tout est ok, =0 loi incomplete
199     int TestComplet();
200
201 // calcul d'un module d'young équivalent à la loi, ceci pour un
202 // chargement nul
203 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
204 // récupération d'un module de compressibilité équivalent à la loi pour un chargement non nul
205 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
206 // saveResul);
207 // récupération de la variation relative d'épaisseur calculée: h/h0
208 // cette variation n'est utile que pour des lois en contraintes planes
209 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
210 // - pour les lois 2D def planes: retour de 0
211 // les infos nécessaires à la récupération , sont stockées dans saveResul
212 // qui est le conteneur spécifique au point où a été calculé la loi
213 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
214
215 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
216 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hypo_ortho3D_entrainee(*this)); };
217

```

```

218 //----- lecture écriture de restart -----
219 // cas donne le niveau de la récupération
220 // = 1 : on récupère tout
221 // = 2 : on récupère uniquement les données variables (supposées comme telles)
222 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbesID&
    lesCourbesID
223                                     ,LesFonctions_nD& lesFonctionsnD);
224 // cas donne le niveau de sauvegarde
225 // = 1 : on sauvegarde tout
226 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
227 void Ecriture_base_info_loi(ofstream& sort,const int cas);
228
229 // affichage et definition interactive des commandes particulières à chaque lois
230 void Info_commande_LoisDeComp(UtilLecture& lec);
231
232 // récupération des grandeurs particulière (hors ddl )
233 // correspondant à liTQ
234 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
235 virtual void Grandeur_particuliere
236     (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
    ;
237 // récupération de la liste de tous les grandeurs particulières
238 // ces grandeurs sont ajoutées à la liste passées en paramètres
239 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
240 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
241
242 // récupe du nom de repère
243 const string& NomRepere() const {return nom_repere;};
244
245 // récupe du type de transport
246 const int& Type_transport() const {return type_transport;};
247
248 protected :
249
250 // donnees protegees
251 // - coef de la loi
252 double E1,E2,E3,nul2,nul3,nu23,G12,G13,G23; // paramètres de la loi
253 Tableau<Fonction_nD* > fct_para; // fonction nD éventuelle d'évolution des paramètres
254 bool null_fct_para; // indicateur pour le cas particulier où il n'y a aucune fct_para
255 int type_derive; // type de dérivée objective utilisée pour sigma
256 string nom_repere; // le nom du repère d'anisotropie associé
257 int cas_calcul; // indique le choix entre différents types de calcul possible
258 // = 0 : calcul normal
259 // = 1 : calcul seulement déviatorique (la partie sphérique est mise à zéro)
260 // = 2 : calcul seulement sphérique (la partie déviatorique est mise à zéro)
261 double ratio_inf_module_compressibilite; // indique le ratio mini / au module initial
262 Mat_pleine inv_loi; // matrice pour inverser la relation eps_jj = mat * sig_ii
263 // type de transport
264 int type_transport; // = 0 : par défaut: transport de type contravariant
265 // = 1 : transport de type covariant
266
267 BaseB Op_B; // coordonnée de la base de travail : correspond à la base O'_i actuelle exprimée dans
    g^j
268 // correspond au transport de type covariant
269
270 BaseB d_Op_B; // variation du repère: c'est une grandeur de travail
271 BaseB pO_B; // les vecteurs O_i non normés: grandeur de travail
272 BaseH d_Op_H; // idem
273 BaseH pO_H; // les vecteurs O_i non normés: grandeur de travail
274
275 BaseH alpha_H; // coordonnées locales de O_a: O_a = alpha_a^{.i} * g_i
276
277 Mat_pleine beta; // matrice de passage de g_i à O'_i: O'_i = beta_i^{.j} * g_j
278 Mat_pleine gamma; // matrice de passage de g^i à O'^i: O'^i = gamma^i_{.j} * g^j
279 Mat_pleine beta_transpose;
280 Mat_pleine gamma_transpose;
281
282 Mat_pleine beta_inv; // l'inverse
283
284 // ----- controle de la sortie des informations
285 // -> maintenant définit dans LoiAbstraiteGeneral
286 // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes,
287 // // pour les erreurs et des warnings
288
289 int sortie_post; // permet de stocker et ensuite d'accéder en post-traitement à certaines données
290 // = 0 par défaut,
291 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
292
293 // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
294 // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonormee
295 Tenseur3HHHH I_x_I_HHHH,I_xbarre_I_HHHH,I_x_eps_HHHH,Ixbarre_eps_HHHH;
296
297 // codage des METHODES VIRTUELLES protegees:
298 // calcul des contraintes a t+dt
299 // calcul des contraintes
300 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DDLElement & tab_ddl
301 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_

```

```

302     ,TenseurBB & delta_epsBB_
303     ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
304     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
305     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
306     ,const Met_abstraite::Expli_t_tdt& ex);
307
308     // calcul des contraintes et de ses variations a t+dt
309 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
310     ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
311     ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
312     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
313     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
314     ,Tableau <TenseurBB *>& d_gijBB_tdt
315     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
316     ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
317     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
318     ,const Met_abstraite::Impli& ex);
319
320     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
321     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
322     // le tenseur de déformation et son incrément sont également en orthonormees
323     // si = false: les bases transmises sont utilisées
324     // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
325 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
326     ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
327     ,TenseurHH& sigHH,TenseurHHH& d_sigma_deps
328     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
329     ,const Met_abstraite::Umat_cont& ex) ;
330
331     // fonction surchargée dans les classes dérivée si besoin est
332 virtual void CalculGrandeurTravail(const PtIntegMecaInterne&
333     ,const Deformation & ,Enum_dure,const ThermoDonnee&
334     ,const Met_abstraite::Impli* ex_impli
335     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
336     ,const Met_abstraite::Umat_cont* ex_umat
337     ,const List_io<Ddl_etendu>* exclure_dd_etend
338     ,const List_io<const TypeQuelconque *>* exclure_Q) {};
339 };
340 /// @} // end of group
341
342
343 #endif
344
345
346

```

## 7.20 Loi\_ortho2D\_C\_entrainee.h

```

1 // FICHER : Loi_ortho2D_C_entrainee.h
2 // CLASSE : Loi_ortho2D_C_entrainee
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33
34 /*****

```

```

35 *      DATE:          12/01/2018
36 *
37 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *      PROJET:        Herezh++
40 *
41 *      $
42 * *****
43 *      BUT:
44 *      La classe Loi_ortho2D_C_entrainee permet de calculer la contrainte et
45 *      et ses variations pour une loi orthotrope entraînée élastique
46 *      en 3D. Il s'agit d'un comportement linéaire, a priori correct
47 *      pour des déformations modérées (typiquement < à 5%).
48 *      La loi nécessite la définition des coefficients classiques
49 *      d'orthotropie et d'un repère particulier d'anisotropie. Le
50 *      repère est ensuite entraîné par la matière.
51 *      $
52 * *****
53 *      VERIFICATION:
54 *
55 *      ! date !      auteur !      but
56 *      -----
57 *      !      !      !
58 *      $
59 * *****
60 *      MODIFICATIONS:
61 *      ! date !      auteur !      but
62 *      -----
63 *      $
64 * *****/
65
66
67 #ifndef LOI_ORTHO_2D_C_ENTRAINEE_H
68 #define LOI_ORTHO_2D_C_ENTRAINEE_H
69 #include "Base3D3.h"
70
71
72 #include "Loi_comp_abstraite.h"
73
74
75 /// @addtogroup Les_lois_anisotropes
76 /// @{
77 ///
78
79
80 class Loi_ortho2D_C_entrainee : public Loi_comp_abstraite
81 {
82
83
84 public :
85
86     // CONSTRUCTEURS :
87
88     // Constructeur par défaut
89     Loi_ortho2D_C_entrainee ();
90
91     // Constructeur fonction de tous les paramètres constants de la loi
92     Loi_ortho2D_C_entrainee(const double& EE1,const double& EE2,const double& EE3
93         ,const double& nunu12,const double& nunu13,const double& nunu23
94         ,const double& GG12
95         ,const string& nom_rep);
96
97     // Constructeur de copie
98     Loi_ortho2D_C_entrainee (const Loi_ortho2D_C_entrainee& loi) ;
99
100    // DESTRUCTEUR :
101    ~Loi_ortho2D_C_entrainee ();
102
103    // initialise les donnees particulieres a l'elements
104    // de matiere traite ( c-a-dire au pt calcule)
105    // Il y a creation d'une instance de SaveResul particuliere
106    // a la loi concernee
107    // la SaveResul classe est remplie par les instances heritantes
108    // le pointeur de SaveResul est sauvegarde au niveau de l'element
109    // c'a-d que les info particulieres au point considere sont stocke
110    // au niveau de l'element et non de la loi.
111    class SaveResulLoi_ortho2D_C_entrainee: public SaveResul
112    { public :
113        SaveResulLoi_ortho2D_C_entrainee(const int type_transport=0); // constructeur par défaut :
114        SaveResulLoi_ortho2D_C_entrainee(const SaveResulLoi_ortho2D_C_entrainee& sav); // de copie
115        virtual ~SaveResulLoi_ortho2D_C_entrainee(); // destructeur
116        // définition d'une nouvelle instance identique
117        // appelle du constructeur via new
118        SaveResul * Nevez_SaveResul() const{return (new SaveResulLoi_ortho2D_C_entrainee(*this));};
119        // affectation
120        virtual SaveResul & operator = ( const SaveResul & a);
121        //===== lecture écriture dans base info =====

```



```

122         // cas donne le niveau de la récupération
123         // = 1 : on récupère tout
124         // = 2 : on récupère uniquement les données variables (supposées comme telles)
125 void Lecture_base_info (ifstream& ent,const int cas);
126         // cas donne le niveau de sauvegarde
127         // = 1 : on sauvegarde tout
128         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
129 void Ecriture_base_info(ofstream& sort,const int cas);
130
131 // mise à jour des informations transitoires
132 void TdtversT();
133 void TversTdt();
134
135 // affichage à l'écran des infos
136 void Affiche() const;
137
138 //changement de base de toutes les grandeurs internes tensorielles stockées
139 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
140 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
141 // gpH(i) = gamma(i,j) * gH(j)
142 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
143
144 // procedure permettant de completer éventuellement les données particulières
145 // de la loi stockées
146 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
147 // completer est appelé apres sa creation avec les donnees du bloc transmis
148 // peut etre appeler plusieurs fois
149 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
150                               ,const Loi_comp_abstraite* loi);
151
152 // ---- méthodes spécifiques
153 // initialise les informations de travail concernant le pas de temps en cours
154 void Init_debut_calcul();
155
156 //-----
157 // données
158 //-----
159 // - partie repère d'orthotropie
160 // 1) le repère lui-même
161 // soit O_B est non nul et O_H est NULL
162 // soit l'inverse,
163 // c'est celui qui est non nul, qui indique le type de convection
164 BaseB * O_B; // définit éventuellement les coordonnées covariante
165             // convectées donc fixes, de 0, dans gi_H_tdt
166 BaseH * O_H; // définit éventuellement les coordonnées contravariante
167             // convectées donc fixes, de 0, dans gi_B_tdt
168 // 2) le repère obtenu par convection -> O', exprimé
169 BaseH Op_H,Op_H_t; // définit les coordonnées contravariante
170                 // de la base transportée
171
172 // 2) les tenseurs intermédiaires
173 TenseurHH* eps_loc_HH; // def local dans le repère O'_a
174 TenseurHH* sig_loc_HH; // contrainte local dans le repère O'_a
175
176 // les 7 paramètres de la loi dans l'ordre suivant
177 // double E1,E2,E3,nul2,nul3,nu23,G12; // paramètres de la loi
178 Vecteur *para_loi;
179
180 // cas spécifique à l'état de contrainte plane
181 double eps33,eps33_t; // déformation d'épaisseur
182
183 };
184
185 SaveResul * New_et_Initialise()
186 { SaveResulLoi_ortho2D_C_entrainee * pt = new SaveResulLoi_ortho2D_C_entrainee();
187   return pt;
188 };
189
190 friend class SaveResulLoi_ortho2D_C_entrainee;
191
192 // Lecture des donnees de la classe sur fichier
193 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
194                                   ,LesFonctions_nD& lesFonctionsnD);
195
196 // affichage de la loi
197 void Affiche() const ;
198 // test si la loi est complete
199 // = 1 tout est ok, =0 loi incomplete
200 int TestComplet();
201
202 // calcul d'un module d'young équivalent à la loi, ceci pour un
203 // chargement nul
204 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
205 // récupération d'un module de compressibilité équivalent à la loi pour un chargement non nul
206 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
207 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
208     saveResul);
209

```

```

208 // récupération de la variation relative d'épaisseur calculée: h/h0
209 // cette variation n'est utile que pour des lois en contraintes planes
210 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
211 // - pour les lois 2D def planes: retour de 0
212 // les infos nécessaires à la récupération , sont stockées dans saveResul
213 // qui est le conteneur spécifique au point où a été calculé la loi
214 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
215
216 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
217 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_ortho2D_C_entrainee(*this)); };
218
219 //----- lecture écriture de restart -----
220 // cas donne le niveau de la récupération
221 // = 1 : on récupère tout
222 // = 2 : on récupère uniquement les données variables (supposées comme telles)
223 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
224                                     ,LesFonctions_nD& lesFonctionsnD);
225
226 // cas donne le niveau de sauvegarde
227 // = 1 : on sauvegarde tout
228 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
229 void Ecriture_base_info_loi(ofstream& sort,const int cas);
230
231 // affichage et definition interactive des commandes particulières à chaque lois
232 void Info_commande_LoisDeComp(UtilLecture& lec);
233
234 // ----- methode propre a une loi en contraintes planes -----
235 // récupération de la dernière déformation d'épaisseur calculée: cette déformaion n'est utile que pour
des lois en contraintes planes ou doublement planes
236 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
237 // - pour les lois 2D def planes: retour de 0
238 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
239 // qui est le conteneur spécifique au point où a été calculé la loi
240 virtual double Eps33BH(SaveResul * saveResul) const
241 {SaveResulLoi_ortho2D_C_entrainee & save_resul = *((SaveResulLoi_ortho2D_C_entrainee*) saveResul);
242   return save_resul.eps33;
243 };
244
245 // indique si la loi est en contraintes planes en s'appuyant sur un comportement 3D
246 virtual bool Contraintes_planes_de_3D() const {return true;};
247
248 // récupération des grandeurs particulière (hors ddl )
249 // correspondant à liTQ
250 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
251 virtual void Grandeur_particuliere
252 (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
253 ;
254 // récupération de la liste de tous les grandeurs particulières
255 // ces grandeurs sont ajoutées à la liste passées en paramètres
256 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
257 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
258
259 // récupe du nom de repère
260 const string& NomRepere() const {return nom_repere;};
261
262 // récupe du type de transport
263 const int& Type_transport() const {return type_transport;};
264
265 protected :
266
267 // donnees protegees
268 // - coef de la loi
269 double E1,E2,E3,nul2,nul3,nu23,G12; // paramètres de la loi
270 Tableau <Fonction_nD* > fct_para; // fonction nD éventuelle d'évolution des paramètres
271 bool null_fct_para; // indicateur pour le cas particulier où il n'y a aucune fct_para
272 string nom_repere; // le nom du repère d'anisotropie associé
273 int cas_calcul; // indique le choix entre différents types de calcul possible
274 // = 0 : calcul normal
275 // = 1 : calcul seulement déviatorique (la partie sphérique est mise à zéro)
276 // = 2 : calcul seulement sphérique (la partie déviatorique est mise à zéro)
277 int verification_convexite; // a priori on vérifie la convexité, mais on peut supprimer la
vérification
278 Mat_pleine inv_loi; // matrice pour inverser la relation eps_jj = mat * sig_ii
279 // type de transport
280 int type_transport; // = 0 : par défaut: transport de type contravariant
281 // = 1 : transport de type covariant
282
283 BaseB Op_B; // coordonnée de la base de travail : correspond à la base O'_i actuelle exprimée dans
g^j
284 // correspond au transport de type covariant
285
286 BaseB d_Op_B; // variation du repère: c'est une grandeur de travail
287 BaseB pO_B; // les vecteurs O_i non normés: grandeur de travail
288 BaseH d_Op_H; // idem
289 BaseH pO_H; // les vecteurs O_i non normés: grandeur de travail

```

```

290 BaseH alpha_H; // coordonnées locales de O_a:   O_a = alpha_a^{.i} * g_i
291
292 Mat_pleine beta; // matrice de passage de g_i à O'_i: O'_i = beta_i^{.j} * g_j
293 Mat_pleine gamma; // matrice de passage de g^i à O'^i: O'^i = gamma^{i.^j} * g^j
294 Mat_pleine beta_transpose;
295 Mat_pleine gamma_transpose;
296
297 Mat_pleine beta_inv; // l'inverse
298
299 // ----- controle de la sortie des informations
300 // -> maintenant définit dans LoiAbstraiteGeneral
301 // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes,
302 //                          // pour les erreurs et des warnings
303
304 int sortie_post; // permet de stocker et ensuite d'accéder en post-traitement à certaines données
305 // = 0 par défaut,
306 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
307
308 // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
309 // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonormee
310 Tenseur3HHHH I_x_I_HHHH, I_xbarre_I_HHHH, I_x_eps_HHHH, Ixbarre_eps_HHHH;
311
312 // codage des METHODES VIRTUELLES protegees:
313 // calcul des contraintes a t+dt
314 // calcul des contraintes
315 void Calcul_SigmaHH (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
316 , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB & giB, BaseH & gi_H, TenseurBB & epsBB_
317 , TenseurBB & delta_epsBB_
318 , TenseurBB & gijBB_, TenseurHH & gijHH_, Tableau <TenseurBB *> & d_gijBB_
319 , double& jacobien_0, double& jacobien, TenseurHH & sigHH
320 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
321 module_cisaillement
322 , const Met_abstraite::Expli_t_tdt & ex);
323
324 // calcul des contraintes et de ses variations a t+dt
325 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
326 , BaseB & giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
327 , BaseB & giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH & giH_tdt, Tableau <BaseH> & d_giH_tdt
328 , TenseurBB & epsBB_tdt, Tableau <TenseurBB *> & d_epsBB
329 , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
330 , Tableau <TenseurBB *> & d_gijBB_tdt
331 , Tableau <TenseurHH *> & d_gijHH_tdt, double& jacobien_0, double& jacobien
332 , Vecteur & d_jacobien_tdt, TenseurHH & sigHH, Tableau <TenseurHH *> & d_sigHH
333 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
334 module_cisaillement
335 , const Met_abstraite::Impli & ex);
336
337 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
338 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
339 // le tenseur de déformation et son incrémentsont également en orthonormees
340 // si = false: les bases transmises sont utilisées
341 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
342 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
343 , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
344 , TenseurHH & sigHH, TenseurHHHH & d_sigma_deps
345 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
346 module_cisaillement
347 , const Met_abstraite::Umat_cont & ex) ;
348
349 //vérification de la convexité du potentiel
350 bool Verif_convexite();
351
352 // fonction surchargée dans les classes dérivée si besoin est
353 virtual void CalculGrandeurTravail(const PtIntegMecaInterne&
354 , const Deformation & , Enum_dure, const ThermoDonnee&
355 , const Met_abstraite::Impli* ex_impli
356 , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
357 , const Met_abstraite::Umat_cont* ex_umat
358 , const List_io<Ddl_etendu>* exclure_dd_etend
359 , const List_io<const TypeQuelconque *>* exclure_Q) {};
360 };
361 /// @} // end of group
362
363 #endif
364

```

## 7.21 Loi\_ortho3D\_entrainee.h

```

1 // FICHER : Loi_ortho_elas3D.h
2 // CLASSE : Loi_ortho_elas3D
3
4

```

```

5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33
34 /*****
35 *   DATE:      12/01/2018
36 *
37 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:    Herezh++
40 *
41 *   *****
42 *   BUT:
43 *   La classe Loi_ortho_elas3D permet de calculer la contrainte et
44 *   et ses variations pour une loi orthotrope entraînée élastique
45 *   en 3D. Il s'agit d'un comportement linéaire, a priori correct
46 *   pour des déformations modérées (typiquement < à 5%).
47 *   La loi nécessite la définition des coefficients classiques
48 *   d'orthotropie et d'un repère particulier d'anisotropie. Le
49 *   repère est ensuite entraîné par la matière.
50 *
51 *   *****
52 *
53 *   VERIFICATION:
54 *
55 *   ! date !   auteur !           but
56 *   -----
57 *   !           !           !
58 *
59 *   *****
60 *   MODIFICATIONS:
61 *   ! date !   auteur !           but
62 *   -----
63 *
64 *   *****/
65
66
67 #ifndef LOI_ORTHO_ELAS3D_H
68 #define LOI_ORTHO_ELAS3D_H
69 #include "Base3D3.h"
70
71
72 #include "Loi_comp_abstraite.h"
73
74
75 /// @addtogroup Les_lois_anisotropes
76 /// @{
77 ///
78
79
80 class Loi_ortho_elas3D : public Loi_comp_abstraite
81 {
82
83     public :
84
85         // CONSTRUCTEURS :
86
87         // Constructeur par défaut
88         Loi_ortho_elas3D ();
89
90
91     // Constructeur fonction de tous les paramètres constants de la loi

```

```

92  Loi_ortho_elas3D(const double& EE1,const double& EE2,const double& EE3
93                ,const double& nunu12,const double& nunu13,const double& nunu23
94                ,const double& GG12,const double& GG13,const double& GG23
95                ,const string& nom_rep);
96
97      // Constructeur de copie
98      Loi_ortho_elas3D (const Loi_ortho_elas3D& loi) ;
99
100     // DESTRUCTEUR :
101     ~Loi_ortho_elas3D ();
102
103     // initialise les donnees particulieres a l'elements
104     // de matiere traite ( c-a-dire au pt calcule)
105     // Il y a creation d'une instance de SaveResul particuliere
106     // a la loi concernee
107     // la SaveResul classe est remplie par les instances heritantes
108     // le pointeur de SaveResul est sauvegarde au niveau de l'element
109     // c'a-d que les info particulieres au point considere sont stocke
110     // au niveau de l'element et non de la loi.
111     class SaveResulLoi_ortho_elas3D: public SaveResul
112     { public :
113         SaveResulLoi_ortho_elas3D(const int type_transport=0); // constructeur par défaut :
114         SaveResulLoi_ortho_elas3D(const SaveResulLoi_ortho_elas3D& sav); // de copie
115         virtual ~SaveResulLoi_ortho_elas3D(); // destructeur
116         // definition d'une nouvelle instance identique
117         // appelle du constructeur via new
118         SaveResul * Nevez_SaveResul() const{return (new SaveResulLoi_ortho_elas3D(*this));};
119         // affectation
120         virtual SaveResul & operator = ( const SaveResul & a);
121         //===== lecture écriture dans base info =====
122         // cas donne le niveau de la récupération
123         // = 1 : on récupère tout
124         // = 2 : on récupère uniquement les données variables (supposées comme telles)
125         void Lecture_base_info (ifstream& ent,const int cas);
126         // cas donne le niveau de sauvegarde
127         // = 1 : on sauvegarde tout
128         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
129         void Ecriture_base_info(ofstream& sort,const int cas);
130
131         // mise à jour des informations transitoires
132         void TdtversT();
133         void TversTdt();
134
135         // affichage à l'écran des infos
136         void Affiche() const;
137
138         //changement de base de toutes les grandeurs internes tensorielles stockées
139         // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gp
140         // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
141         // gpH(i) = gamma(i,j) * gH(j)
142         virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
143
144         // procedure permettant de completer éventuellement les données particulières
145         // de la loi stockées
146         // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
147         // completer est appelé apres sa creation avec les donnees du bloc transmis
148         // peut etre appeler plusieurs fois
149         SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
150                                     ,const Loi_comp_abstraite* loi);
151
152         // ---- méthodes spécifiques
153         // initialise les informations de travail concernant le pas de temps en cours
154         void Init_debut_calcul();
155
156         //-----
157         // données
158         //-----
159         // - partie repère d'orthotropie
160         // 1) le repère lui-même
161         // soit O_B est non nul et O_H est NULL
162         // soit l'inverse,
163         // c'est celui qui est non nul, qui indique le type de convection
164         BaseB * O_B; // définit éventuellement les coordonnées covariante
165                   // convectées donc fixes, de O, dans gi_H_tdt
166         BaseH * O_H; // définit éventuellement les coordonnées contravariante
167                   // convectées donc fixes, de O, dans gi_B_tdt
168         // 2) le repère obtenu par convection -> O', exprimé
169         BaseH Op_H,Op_H_t; // définit les coordonnées contravariante
170                          // de la base transportée
171
172         // 2) les tenseurs intermédiaires
173         TenseurHH* eps_loc_HH; // def local dans le repère O'_a
174         TenseurHH* sig_loc_HH; // contrainte local dans le repère O'_a
175
176         // les 9 paramètres de la loi dans l'ordre suivant
177         // double E1,E2,E3,nul2,nul3,nu23,G12,G13,G23; // paramètres de la loi
178         Vecteur *para_loi;

```

```

179
180 };
181
182 SaveResul * New_et_Initialise()
183 { SaveResulLoi_ortho_elas3D * pt = new SaveResulLoi_ortho_elas3D();
184   return pt;
185 };
186
187 friend class SaveResulLoi_ortho_elas3D;
188
189 // Lecture des donnees de la classe sur fichier
190 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
191                                   ,LesFonctions_nD& lesFonctionsnD);
192 // affichage de la loi
193 void Affiche() const ;
194 // test si la loi est complete
195 // = 1 tout est ok, =0 loi incomplete
196 int TestComplet();
197
198 // calcul d'un module d'young equivalent à la loi, ceci pour un
199 // chargement nul
200 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
201 // récupération d'un module de compressibilité equivalent à la loi pour un chargement non nul
202 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
203 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
204   saveResul);
205 // récupération de la variation relative d'épaisseur calculée: h/h0
206 // cette variation n'est utile que pour des lois en contraintes planes
207 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
208 // - pour les lois 2D def planes: retour de 0
209 // les infos nécessaires à la récupération , sont stockées dans saveResul
210 // qui est le conteneur spécifique au point où a été calculé la loi
211 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
212
213 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
214 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_ortho_elas3D(*this)); };
215
216 //----- lecture écriture de restart -----
217 // cas donne le niveau de la récupération
218 // = 1 : on récupère tout
219 // = 2 : on récupère uniquement les données variables (supposées comme telles)
220 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
221   lesCourbes1D
222                                   ,LesFonctions_nD& lesFonctionsnD);
223 // cas donne le niveau de sauvegarde
224 // = 1 : on sauvegarde tout
225 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
226 void Ecriture_base_info_loi(ofstream& sort,const int cas);
227 // affichage et definition interactive des commandes particulières à chaque lois
228 void Info_commande_LoisDeComp(UtilLecture& lec);
229
230 // récupération des grandeurs particulière (hors ddl )
231 // correspondant à liTQ
232 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
233 virtual void Grandeur_particuliere
234   (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
235 ;
236 // récupération de la liste de tous les grandeurs particulières
237 // ces grandeurs sont ajoutées à la liste passées en paramètres
238 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
239 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
240
241 // récupe du nom de repère
242 const string& NomRepere() const {return nom_repere;};
243
244 // récupe du type de transport
245 const int& Type_transport() const {return type_transport;};
246
247 protected :
248 // donnees protegees
249 // - coef de la loi
250 double E1,E2,E3,nul2,nul3,nu23,G12,G13,G23; // paramètres de la loi
251 Tableau <Fonction_nD* > fct_para; // fonction nD éventuelle d'évolution des paramètres
252 bool null_fct_para; // indicateur pour le cas particulier où il n'y a aucune fct_para
253 string nom_repere; // le nom du repère d'anisotropie associé
254 int cas_calcul; // indique le choix entre différents types de calcul possible
255 // = 0 : calcul normal
256 // = 1 : calcul seulement déviatorique (la partie sphérique est mise à zéro)
257 // = 2 : calcul seulement sphérique (la partie déviatorique est mise à zéro)
258 double ratio_inf_module_compressibilite; // indique le ratio mini / au module initial
259 int verification_convexite; // a priori on vérifie la convexité, mais on peut supprimer la
260 // vérification
261 Mat_pleine inv_loi; // matrice pour inverser la relation eps_jj = mat * sig_ii
262 // type de transport

```

```

262 int type_transport; // = 0 : par défaut: transport de type contravariant
263 // = 1 : transport de type covariant
264
265 BaseB Op_B; // coordonnée de la base de travail : correspond à la base O'_i actuelle exprimée dans
    g^j
266 // correspond au transport de type covariant
267
268 BaseB d_Op_B; // variation du repère: c'est une grandeur de travail
269 BaseB pO_B; // les vecteurs O_i non normés: grandeur de travail
270 BaseH d_Op_H; // idem
271 BaseH pO_H; // les vecteurs O_i non normés: grandeur de travail
272
273 BaseH alpha_H; // coordonnées locales de O_a: O_a = alpha_a^{.i} * g_i
274
275 Mat_pleine beta; // matrice de passage de g_i à O'_i: O'_i = beta_i^{.j} * g_j
276 Mat_pleine gamma; // matrice de passage de g^i à O'^i: O'^i = gamma^i_{.j} * g^j
277 Mat_pleine beta_transpose;
278 Mat_pleine gamma_transpose;
279
280 Mat_pleine beta_inv; // l'inverse
281
282 // ---- controle de la sortie des informations
283 // -> maintenant définit dans LoiAbstraiteGeneral
284 // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes,
285 // // pour les erreurs et des warnings
286
287 int sortie_post; // permet de stocker et ensuite d'accéder en post-traitement à certaines données
288 // = 0 par défaut,
289 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
290
291 // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
292 // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonormee
293 Tenseur3HHHH I_x_I_HHHH, I_xbarre_I_HHHH, I_x_eps_HHHH, Ixbarre_eps_HHHH;
294
295 // codage des METHODES VIRTUELLES protegees:
296 // calcul des contraintes a t+dt
297 // calcul des contraintes
298 void Calcul_SigmaHH (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
299 , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB& giB, BaseH& gi_H, TenseurBB & epsBB_
300 , TenseurBB & delta_epsBB_
301 , TenseurBB & gijBB_, TenseurHH & gijHH_, Tableau <TenseurBB *>& d_gijBB_
302 , double& jacobien_0, double& jacobien, TenseurHH & sigHH
303 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
304 , const Met_abstraite::Expli_t_tdt& ex);
305
306 // calcul des contraintes et de ses variations a t+dt
307 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
308 , BaseB& giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
309 , BaseB& giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH& giH_tdt, Tableau <BaseH> & d_giH_tdt
310 , TenseurBB & epsBB_tdt, Tableau <TenseurBB *>& d_epsBB
311 , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
312 , Tableau <TenseurBB *>& d_gijBB_tdt
313 , Tableau <TenseurHH *>& d_gijHH_tdt, double& jacobien_0, double& jacobien
314 , Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *>& d_sigHH
315 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
316 , const Met_abstraite::Impli& ex);
317
318 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
319 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
320 // le tenseur de déformation et son incrémentsont également en orthonormees
321 // si = false: les bases transmises sont utilisées
322 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
323 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
324 , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
325 , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps
326 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
327 , const Met_abstraite::Umat_cont& ex) ;
328
329 //vérification de la convexité du potentiel
330 bool Verif_convexite();
331
332 // fonction surchargée dans les classes dérivée si besoin est
333 virtual void CalculGrandeurTravail(const PtIntegMecaInterne&
334 , const Deformation & , Enum_dure, const ThermoDonnee&
335 , const Met_abstraite::Impli* ex_impli
336 , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
337 , const Met_abstraite::Umat_cont* ex_umat
338 , const List_io<Ddl_etendu>* exclure_dd_etend
339 , const List_io<const TypeQuelconque *>* exclure_Q ) {};
340 };
341 /// @} // end of group
342
343
344 #endif

```

345  
346  
347

## 7.22 Projection\_anisotrope\_3D.h

```

1 // FICHER : Projection_anisotrope_3D.h
2 // CLASSE : Projection_anisotrope_3D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33
34 /*****
35 *   DATE:           11/06/2019
36 *
37 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:         Herezh++
40 *
41 * *****/
42 *   BUT:
43 *   La classe Projection_anisotrope_3D permet de calculer la contrainte
44 *   et ses variations pour une loi initialement isotrope 3D puis
45 *   ensuite projetée de manière anisotrope en 3D dans l'espace réel.
46 *   Plusieurs projections sont envisagées.
47 *   1) La première implantée concerne une homotétie des composantes de
48 *   contraintes par rapport à un repère particulier orthonormé ce qui
49 *   conduit initialement à une loi orthotrope. L'opérateur d'homotétie
50 *   est représenté par un tenseur du quatrième ordre, diagonal dans
51 *   son repère initiale de définition. Ce repère est donc principal.
52 *   Ensuite ce tenseur est convecté matériellement pour prendre en
53 *   compte les déplacements solides et les variations d'angles entre
54 *   les directions principales du tenseur du 4ième ordre.
55 *
56 *   *****
57 *
58 *   VERIFICATION:
59 *
60 *   ! date ! auteur ! but
61 *   -----
62 *   ! ! !
63 *
64 *   *****
65 *   MODIFICATIONS:
66 *   ! date ! auteur ! but
67 *   -----
68 *
69 * *****/
70
71 /** @defgroup Les_lois_anisotropes
72 *
73 *   BUT: groupe des lois anisotropes
74 *
75 *
76 *   \author Gérard Rio
77 *   \version 1.0
78 *   \date 11/06/2019
79 *   \brief Définition des lois anisotropes

```



```

80 *
81 */
82
83
84 #ifndef PROJECTION_ANISOTROPE_3D_H
85 #define PROJECTION_ANISOTROPE_3D_H
86 #include "Base3D3.h"
87 #include "Coordonnee2.h"
88 #include "Ponderation.h"
89
90
91 #include "Loi_comp_abstraite.h"
92 #include "Enum_proj_aniso.h"
93 #include "TenseurQ3gene.h"
94
95 /// @addtogroup Les_lois_anisotropes
96 /// @
97 ///
98
99
100 class Projection_anisotrope_3D : public Loi_comp_abstraite
101 {
102
103
104     public :
105
106         // CONSTRUCTEURS :
107
108         // Constructeur par défaut
109         Projection_anisotrope_3D ();
110
111         // Constructeur de copie
112         Projection_anisotrope_3D (const Projection_anisotrope_3D& loi) ;
113
114         // DESTRUCTEUR :
115         ~Projection_anisotrope_3D ();
116
117         // initialise les donnees particulieres a l'elements
118         // de matiere traite ( c-a-dire au pt calcule)
119         // Il y a creation d'une instance de SaveResul particuliere
120         // a la loi concernee
121         // la SaveResul classe est remplie par les instances heritantes
122         // le pointeur de SaveResul est sauvegarde au niveau de l'element
123         // c'a-d que les info particulieres au point considere sont stocke
124         // au niveau de l'element et non de la loi.
125         class SaveResulProjection_anisotrope_3D: public SaveResul
126         { public :
127             SaveResulProjection_anisotrope_3D(); // constructeur par défaut (a ne pas utiliser)
128             // le constructeur courant
129             SaveResulProjection_anisotrope_3D(SaveResul* l_SaveResul, TenseurHH* l_siHH
130                 , TenseurHH* l_siHH_t
131                 , EnergieMeca l_energ_, EnergieMeca l_energ_t_
132                 , bool avec_ponderation);
133             SaveResulProjection_anisotrope_3D(const SaveResulProjection_anisotrope_3D& sav); // de copie
134             virtual ~SaveResulProjection_anisotrope_3D(); // destructeur
135             // définition d'une nouvelle instance identique
136             // appelle du constructeur via new
137             SaveResul * Nevez_SaveResul() const{return (new SaveResulProjection_anisotrope_3D(*this));};
138             // affectation
139             virtual SaveResul & operator = ( const SaveResul & a);
140             //===== lecture écriture dans base info =====
141             // cas donne le niveau de la récupération
142             // = 1 : on récupère tout
143             // = 2 : on récupère uniquement les données variables (supposées comme telles)
144             void Lecture_base_info (ifstream& ent, const int cas);
145             // cas donne le niveau de sauvegarde
146             // = 1 : on sauvegarde tout
147             // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
148             void Ecriture_base_info(ofstream& sort, const int cas);
149
150             // mise à jour des informations transitoires
151             void TdtversT();
152             void TversTdt();
153
154             // affichage à l'écran des infos
155             void Affiche() const;
156
157             //changement de base de toutes les grandeurs internes tensorielles stockées
158             // beta(i, j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
159             // gpB(i) = beta(i, j) * gB(j), i indice de ligne, j indice de colonne
160             // gpH(i) = gamma(i, j) * gH(j)
161             virtual void ChBase_des_grandeurs(const Mat_pleine& beta, const Mat_pleine& gamma);
162
163             // procedure permettant de completer éventuellement les données particulières
164             // de la loi stockées
165             // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
166             // completer est appelé apres sa creation avec les donnees du bloc transmis

```

```

167 // peut etre appeler plusieurs fois
168 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
169                               ,const Loi_comp_abstraite* loi);
170
171 // ---- méthodes spécifiques
172 // initialise les informations de travail concernant le pas de temps en cours
173 void Init_debut_calcul();
174
175 //-----
176 // données
177 //-----
178 // - partie repère d'orthotropie
179 // 1) le repère lui-même
180 // soit O_B est non nul et O_H est NULL
181 // soit l'inverse,
182 // c'est celui qui est non nul, qui indique le type de convection
183 BaseB * O_B; // définit éventuellement les coordonnées covariante
184 // convectées donc fixes, de O, dans gi_H_tdt
185 BaseH * O_H; // définit éventuellement les coordonnées contravariante
186 // convectées donc fixes, de O, dans gi_B_tdt
187 // 2) le repère obtenu par convection -> O', exprimé
188 BaseH Op_H,Op_H_t; // définit les coordonnées contravariante
189 // de la base transportée
190
191 // 2) les tenseurs intermédiaires
192 TenseurHH* eps_loc_HH; // def local dans le repère O'_a
193 TenseurHH* sig_loc_HH; // contrainte local dans le repère O'_a
194
195 // les 6 paramètres de la loi dans l'ordre suivant
196 // double A1,A2,A3,B12,B13,B23; // paramètres de la loi
197 Vecteur *para_loi;
198
199 // données protégées
200 // les données protégées de la loi interne
201 SaveResul* SaveResul_interne;
202 // les contraintes initiales particulières de la loi
203 TenseurHH* l_sigoHH,* l_sigoHH_t; // valeur courante, et valeur sauvegardée au pas précédent
204 // énergies pour la loi interne
205 EnergieMeca l_energ,l_energ_t; // valeur courante, et valeur sauvegardée au pas précédent
206 // fonctions de pondération
207 double f_ponder,f_ponder_t; // le résultat des fonctions de pondérations
208 // = 1 si pas de pondération
209
210 };
211
212 SaveResul * New_et_Initialise();
213
214 friend class SaveResulProjection_anisotrope_3D;
215
216 // Lecture des donnees de la classe sur fichier
217 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
218                                   ,LesFonctions_nD& lesFonctionsnD);
219
220 // affichage de la loi
221 void Affiche() const ;
222 // test si la loi est complete
223 // = 1 tout est ok, =0 loi incomplete
224 int TestCompleet();
225
226 // calcul d'un module d'young équivalent à la loi, ceci pour un
227 // chargement nul
228 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
229 // récupération d'un module de compressibilité équivalent à la loi pour un chargement non nul
230 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
231 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
232 saveResul);
233
234 // récupération de la variation relative d'épaisseur calculée: h/h0
235 // cette variation n'est utile que pour des lois en contraintes planes
236 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
237 // - pour les lois 2D def planes: retour de 0
238 // les infos nécessaires à la récupération , sont stockées dans saveResul
239 // qui est le conteneur spécifique au point où a été calculé la loi
240 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
241
242 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
243 // exemple: mise en service des ddl de température aux noeuds
244 virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud,bool dilatation,LesPtIntegMecaInterne&
245 lesPtMecaInt);
246 // récupération des grandeurs particulière (hors ddl )
247 // correspondant à liTQ
248 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
249 virtual void Grandeur_particuliere
250 (bool absolue,List_io<TypeQuelconque>& liTQ,Loi_comp_abstraite::SaveResul * saveDon,list<int>&)
251 const;
252 // récupération de la liste de tous les grandeurs particulières
253 // ces grandeurs sont ajoutées à la liste passées en paramètres
254 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière

```

```

251 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& liTQ) const;
252
253 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
254 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Projection_anisotrope_3D(*this)); };
255
256 // indique le type Enum_comp_3D_CP_DP_1D correspondant à une loi de comportement
257 // la fonction est simple dans le cas d'une loi basique, par contre dans le cas
258 // d'une loi combinée, la réponse dépend des lois internes donc c'est redéfini
259 // dans les classes dérivées
260 virtual Enum_comp_3D_CP_DP_1D Comportement_3D_CP_DP_1D();
261
262 //----- lecture écriture de restart -----
263 // cas donne le niveau de la récupération
264 // = 1 : on récupère tout
265 // = 2 : on récupère uniquement les données variables (supposées comme telles)
266 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
267                                     ,LesFonctions_nD& lesFonctionsnD);
268 // cas donne le niveau de sauvegarde
269 // = 1 : on sauvegarde tout
270 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
271 void Ecriture_base_info_loi(ofstream& sort,const int cas);
272
273 // affichage et definition interactive des commandes particulières à chaque lois
274 void Info_commande_LoisDeComp(UtilLecture& lec);
275
276 // récupère du nom de repère
277 const string& NomRepere() const {return nom_repere;};
278
279 // récupère du type de transport
280 const int& Type_transport() const {return type_transport;};
281
282 protected :
283
284 Enum_proj_aniso type_projection; // le type de projection de la loi
285
286 // donnees protegees
287 //----- premier type de projection: PROJ_ORTHO -----
288 // - coef de la loi
289 double A1,A2,A3,B12,B13,B23; // paramètres de la loi
290 Tableau2 <double*> hij; // idem sous forme de tableau
291 Tableau <Fonction_nD* > fct_para; // fonction nD éventuelle d'évolution des paramètres
292 bool B11_fct_para; // indicateur pour le cas particulier où il n'y a aucune fct_para
293 string nom_repere; // le nom du repère d'anisotropie associé
294 int cas_calcul; // indique le choix entre différents types de calcul possible
295 // = 0 : calcul normal
296 // = 1 : calcul seulement déviatorique (la partie sphérique est mise à zéro)
297 // = 2 : calcul seulement sphérique (la partie déviatorique est mise à zéro)
298 double ratio_inf_module_compressibilite; // indique le ratio mini / au module initial
299 Mat_pleine inv_loi; // matrice pour inverser la relation eps_jj = mat * sig_ii
300 // type de transport
301 int type_transport; // = 0 : par défaut: transport de type contravariant
302 // = 1 : transport de type covariant
303
304 BaseB Op_B; // coordonnée de la base de travail : correspond à la base O'_i actuelle exprimée dans
g^j
305 // correspond au transport de type covariant
306
307 BaseB d_Op_B; // variation du repère: c'est une grandeur de travail
308 BaseB pO_B; // les vecteurs O_i non normés: grandeur de travail
309 BaseH d_Op_H; // idem
310 BaseH pO_H; // les vecteurs O_i non normés: grandeur de travail
311
312 BaseH alpha_H; // coordonnées locales de O_a: O_a = alpha_a^{.i} * g_i
313
314 Mat_pleine beta; // matrice de passage de g_i à O'_i: O'_i = beta_i^{.j} * g_i
315 Mat_pleine gamma; // matrice de passage de g^i à O'^i: O'^i = gamma^i_{.j} * g^j
316 Mat_pleine beta_transpose;
317 Mat_pleine gamma_transpose;
318
319 Mat_pleine beta_inv; // l'inverse
320
321 // ----- controle de la sortie des informations
322 // -> maintenant définit dans LoiAbstraiteGeneral
323 // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes,
324 // pour les erreurs et des warnings
325
326 int sortie_post; // permet de stocker et ensuite d'accéder en post-traitement à certaines données
327 // = 0 par défaut,
328 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
329
330 // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
331 // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonormee
332 Tenseur3HHHH I_x_I_HHHH,I_xbarre_I_HHHH,I_x_eps_HHHH,Ixbarre_eps_HHHH;
333 TenseurQ3geneHHHH dsig_ijkl_HHHH; // tenseur de travail pour Calcul_dsigma_deps
334
335

```

```

336 // cas d'un point d'intégration locale (méthode CalculGrandeurTravail par exemple)
337 PtIntegMecaInterne ptintmeca;
338
339 //----- lois internes -----
340 // un type énuméré pour faciliter la lecture
341 enum Enumcompletudecalcul { CONTRAINTE_ET_TANGENT =0, CONTRAINTE_UNIQUEMENT, TANGENT_UNIQUEMENT};
342 // donnees protegees
343 Loi_comp_abstraite * loi_interne; // la loi constitutive de l'espace de référence
344 Enumcompletudecalcul completude_calcul; // pour savoir si on utilise tout ou une partie
345
346 //-- partie optionnelle
347
348 // une fonction nD via un objet: Ponderation_TypeQuelconque
349 // ici un pointeur nulle indique qu'il n'y a pas de fct
350 // les grandeurs quelconque sont celles de la loi, elles doivent donc être renseignées
351 Ponderation_TypeQuelconque* ponderation_nD_quelconque;
352
353 // ---- tableau de travail
354 Tableau <TenseurHH *> d_sigRef_HH;
355 // tenseur du 4ième orde de travail
356 TenseurHHHH* d_sigma_deps_inter;
357 Tenseur3HHHH d_sig_deps_3D_HHHH;
358 Tenseur3HHBB H_HHBB; // le tenseur H dans la base de travail
359
360
361 // codage des METHODES VIRTUELLES protegees:
362 // calcul des contraintes a t+dt
363 // calcul des contraintes
364 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
365 ,TenseurBB & giBB_t,TenseurHH & giHH_t,BaseB & giB,BaseH & gi_H, TenseurBB & epsBB_
366 ,TenseurBB & delta_epsBB_
367 ,TenseurBB & giBB_,TenseurHH & giHH_,Tableau <TenseurBB *>& d_giBB_
368 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
369 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
370 module_cisaillement
371 ,const Met_abstraite::Expli_t_tdt& ex);
372
373 // calcul des contraintes et de ses variations a t+dt
374 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
375 ,BaseB & giB_t,TenseurBB & giBB_t,TenseurHH & giHH_t
376 ,BaseB & giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH & giH_tdt,Tableau <BaseH> & d_giH_tdt
377 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
378 ,TenseurBB & delta_epsBB,TenseurBB & giBB_tdt,TenseurHH & giHH_tdt
379 ,Tableau <TenseurBB *>& d_giBB_tdt
380 ,Tableau <TenseurHH *>& d_giHH_tdt,double& jacobien_0,double& jacobien
381 ,Vecteur& d_jacobien_tdt,TenseurHH & sigHH,Tableau <TenseurHH *>& d_sigHH
382 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
383 module_cisaillement
384 ,const Met_abstraite::Impli& ex);
385
386 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
387 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
388 // le tenseur de déformation et son incrémentsont également en orthonormees
389 // si = false: les bases transmises sont utilisées
390 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
391 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
392 ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
393 ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
394 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
395 module_cisaillement
396 ,const Met_abstraite::Umat_cont& ex) ;
397
398 // fonction surchargée dans les classes dérivée si besoin est
399 virtual void CalculGrandeurTravail(const PtIntegMecaInterne&
400 ,const Deformation & ,Enum_dure,const ThermoDonnee&
401 ,const Met_abstraite::Impli* ex_impli
402 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
403 ,const Met_abstraite::Umat_cont* ex_umat
404 ,const List_io<Ddl_etendu>* exclure_dd_etend
405 ,const List_io<const TypeQuelconque *>* exclure_Q
406 );
407
408 // fonction interne utilisée par les classes dérivées de Loi_comp_abstraite
409 // pour répercuter les modifications de la température
410 // ici utiliser pour modifier la température des lois élémentaires
411 // l'Enum_dure: indique quel est la température courante : 0 t ou tdt
412 void RepercuteChangeTemperature (Enum_dure temps);
413
414 // vérification et préparation de l'accès aux grandeurs locales
415 void Verif_et_preparation_acces_grandeurs_locale();
416 };
417 /// @} // end of group
418
419 #endif

```

420

## 7.23 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/comportement/ComLoi\_comp\_abstraite.h

définition de fonctions templates utilisées par les classes dérivées de [Loi\\_comp\\_abstraite](#)

### Fonctions

— `template<class TensHH , class TensBB , class TensBH >`  
`void invariants_1 (TensBB &epsBB, TensHH &gijHH, TensBH &IdGBH, double &jacobien_0, double &jacobien, double &V, double &Qeps)`

#### 7.23.1 Description détaillée

définition de fonctions templates utilisées par les classes dérivées de [Loi\\_comp\\_abstraite](#)

## 7.24 ComLoi\_comp\_abstraite.h

[Aller à la documentation de ce fichier.](#)

```

1
2  /*! \file ComLoi_comp_abstraite.h
3  \brief définition de fonctions templates utilisées par les classes dérivées de Loi_comp_abstraite
4  */
5  //-----
6  // définition de fonctions templates utilisées par
7  // les classes dérivées de Loi_comp_abstraite
8  //-----
9
10
11 // This file is part of the Herezh++ application.
12 //
13 // The finite element software Herezh++ is dedicated to the field
14 // of mechanics for large transformations of solid structures.
15 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
16 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
17 //
18 // Herezh++ is distributed under GPL 3 license ou ultérieure.
19 //
20 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
21 // AUTHOR : Gérard Rio
22 // E-MAIL : gerardrio56@free.fr
23 //
24 // This program is free software: you can redistribute it and/or modify
25 // it under the terms of the GNU General Public License as published by
26 // the Free Software Foundation, either version 3 of the License,
27 // or (at your option) any later version.
28 //
29 // This program is distributed in the hope that it will be useful,
30 // but WITHOUT ANY WARRANTY; without even the implied warranty
31 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
32 // See the GNU General Public License for more details.
33 //
34 // You should have received a copy of the GNU General Public License
35 // along with this program. If not, see <https://www.gnu.org/licenses/>.
36 //
37 // For more information, please consult: <https://herezh.irdl.fr/>.
38
39
40
41 // Version 1) du calcul des premiers invariants du tenseurs de déformations
42 // epsBB : le tenseur de déformation, gijHH : le tenseur metrique en locale
43 // IdGBH : tenseur metrique en mixte, jacobien_0 : initial, V : variation de volume
44 // Qeps : une forme du deuxieme invariant
45 template<class TensHH, class TensBB, class TensBH>
46 inline void invariants_1(TensBB & epsBB, TensHH & gijHH, TensBH & IdGBH,
47 double & jacobien_0, double& jacobien, double & V, double& Qeps)
48 { TensBH epsBH = epsBB * gijHH; // deformation en mixte
49 double trace_eps = epsBH.Trace(); // trace de la déformation
50 TensBH eps_barreBH = epsBH - (trace_eps/3.) * IdGBH;
51 Qeps = sqrt(2.*eps_barreBH.II()); // le deuxième invariant
52 V = sqrt(jacobien/jacobien_0);
53 };
54 // Version 1) du calcul des premiers invariants du tenseurs de déformations
55 // ainsi que des variations par rapport aux degrés de liberté

```

```
56 /* void Loi_comp_abstraite::invariants_l_et_var()
57     {}*/
```

## 7.25 CompFrotAbstraite.h

```
1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           04/05/2006
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 *   *****
39 *   BUT:   Interface pour les lois de frottement dans
40 *           le cas de contact.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *   ! date !   auteur !           but
46 *   -----
47 *   !           !           !
48 *
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date !   auteur !           but
52 *   -----
53 *
54 *   *****/
55 // FICHIER : CompFrotAbstraite.h
56 // CLASSE : CompFrotAbstraite
57
58 #ifndef COMP_FROT_ABSTRAITE_H
59 #define COMP_FROT_ABSTRAITE_H
60
61 #include "Enum_comp.h"
62 #include "Tableau_T.h"
63 #include "Tenseur.h"
64 #include "LoiAbstraiteGeneral.h"
65 #include "TypeQuelconque.h"
66 #include "EnergieMeca.h"
67
68 class CompFrotAbstraite : public LoiAbstraiteGeneral
69 {
70     public :
71         // CONSTRUCTEURS :
72
73         // Constructeur par défaut
74         CompFrotAbstraite () ;
75
76         // Constructeur utile si l'identificateur du nom de la loi
77         // de comportement et la dimension sont connus
78         CompFrotAbstraite (Enum_comp id_compor,Enum_categorie_loi_comp categorie_comp,int dimension);
79 }
```

```

80 // Constructeur utile si l'identificateur du nom de la loi
81 // de comportement et la dimension sont connus
82 CompFrotAbstraite (char* nom,Enum_categorie_loi_comp categorie_comp,int dimension);
83
84 // Constructeur de copie
85 CompFrotAbstraite (const CompFrotAbstraite & a );
86
87 // DESTRUCTEUR VIRTUEL :
88
89 virtual ~CompFrotAbstraite ();
90
91
92 // 2) METHODES VIRTUELLES public:
93
94 // initialise les donnees particulieres a l'elements
95 // de matiere traite ( c-a-dire au pt calcule)
96 // Il y a creation d'une instance de SaveResul particuliere
97 // a la loi concernee
98 // la SaveResul classe est remplie par les instances heritantes
99 // le pointeur de SaveResul est sauvegarde au niveau de l'element de contact
100 // c-a-d que les info particulieres au point considere sont stocke
101 // au niveau de l'element et non de la loi.
102 class SaveResul
103 { public :
104 // definition d'une nouvelle instance identique
105 // appelle du constructeur via new
106 virtual SaveResul * Nevez_SaveResul() const =0;
107 // affectation
108 virtual SaveResul & operator = ( const SaveResul & a) = 0;
109 //===== lecture écriture dans base info =====
110 // cas donne le niveau de la récupération
111 // = 1 : on récupère tout
112 // = 2 : on récupère uniquement les données variables (supposées comme telles)
113 virtual void Lecture_base_info (ifstream& ent,const int cas) = 0;
114 // cas donne le niveau de sauvegarde
115 // = 1 : on sauvegarde tout
116 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
117 virtual void Ecriture_base_info(ofstream& sort,const int cas) = 0;
118
119 // mise à jour des informations transitoires en définitif s'il y a convergence
120 // par exemple (pour la plasticité par exemple)
121 virtual void TdtversT() {};
122 virtual void TversTdt() {};
123
124 //changement de base de toutes les grandeurs internes tensorielles stockées
125 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
126 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
127 // gpH(i) = gamma(i,j) * gH(j)
128 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) = 0;
129
130 };
131
132 virtual SaveResul * New_et_Initialise() { return NULL;};
133
134 // affichage des donnees particulieres a l'elements
135 // de matiere traite ( c-a-dire au pt calcule)
136 virtual void AfficheDataSpecif(ofstream& ,SaveResul * ) const {};
137
138 // schema de calcul explicite à tdt
139 // calcul des efforts de frottement à un instant t+deltat
140 // les indices t se rapporte au pas précédent, sans indice au temps actuel
141 // vit_T : vitesse, force_normale: force normale (à la cible) agissant sur le noeud projectile
142 // force_tangente: force tangente (à la cible) agissant sur le noeud projectile
143 // normale: la normale de contact normée
144 // energie_frottement: l'énergie échangée: élas=la totalité, viqueux et plas= uniquement pendant dt
145 // delta_t : le pas de temps
146 // F_frot: force de frottement calculé,
147 // retour glisse: indique si oui ou non le noeud glisse
148 virtual bool Cal_explicit_contact_tdt(const Coordonnee& vit_T, const Coordonnee& normale
149 ,const Coordonnee& force_normale,const Coordonnee& force_tangente
150 ,EnergieMeca& energie_frottement,const double delta_t
151 ,Coordonnee& F_frot,CompFrotAbstraite::SaveResul * = NULL)
152 {return Calcul_Frottement (vit_T,normale,force_normale,force_tangente
153 ,energie_frottement,delta_t,F_frot);};
154
155 // schema implicit
156 // calcul des efforts de frottement à un instant t+deltat
157 // et ses variations par rapport aux ddl de vitesse
158 // vit_T : vitesse, force_normale: force normale (à la cible) agissant sur le noeud projectile
159 // force_tangente: force tangente (à la cible) agissant sur le noeud projectile
160 // normale: la normale de contact normée
161 // energie_frottement: l'énergie échangée: élas=la totalité, viqueux et plas= uniquement pendant dt
162 // delta_t : le pas de temps
163 // F_frot: force de frottement calculé
164 // d_F_frot_vit: variation de la force de frottement par rapport aux coordonnées de vitesse
165 // retour glisse: indique si oui ou non le noeud glisse
166 virtual bool Cal_implicit(const Coordonnee& vit_T, const Coordonnee& normale

```

```

167         ,const Coordonnee& force_normale,const Coordonnee& force_tangente
168         ,EnergieMeca& energie_frottement,const double delta_t
169         ,Coordonnee& F_frot,Tableau <Coordonnee>& d_F_frot_vit
170         ,CompFrotAbstraite::SaveResul * = NULL)
171     {return Calcul_DFrottement_tdt(vit_T, normale, force_normale, force_tangente
172     , energie_frottement, delta_t, F_frot, d_F_frot_vit);};
173
174     // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
175     // exemple: mise en service de ddl particulier
176     // pour l'instant rien
177     virtual void Activation_donnees() {};
178
179     // modification de l'indicateur de comportement tangent
180     void Modif_comp_tangent_simplifie(bool modif)
181     { comp_tangent_simplifie = modif;};
182
183     // test pour connaître l'état du comportement : simplifié ou non
184     bool Test_loi_simplifie()
185     { return comp_tangent_simplifie;};
186
187     // récupération des grandeurs particulière (hors ddl )
188     // correspondant à liTQ
189     // la liste d'entiers correspond à un décalage éventuel
190     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
191     virtual void Grandeur_particuliere(bool absolue, List_io<TypeQuelconque>& , CompFrotAbstraite::SaveResul
192     *, list<int>& )
193     {};
194     // récupération de la liste de tous les grandeurs particulières
195     // ces grandeurs sont ajoutées à la liste passées en paramètres
196     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
197     virtual void ListeGrandeurs_particulieres(bool absolue, List_io<TypeQuelconque>& ) {};
198
199     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
200     virtual CompFrotAbstraite* Nouvelle_loi_identique() const = 0;
201
202     protected :
203
204     // affichage et definition interactive des commandes particulières à la classe CompFrotAbstraite
205     void Info_commande_don LoisDeComp(UtilLecture& ) const {};
206     //----- lecture écriture de restart spécifique aux données de la classe -----
207     // cas donne le niveau de la récupération
208     // = 1 : on récupère tout
209     // = 2 : on récupère uniquement les données variables (supposées comme telles)
210     void Lecture_don_base_info(ifstream& ,const int , LesReferences& , LesCourbesID& , LesFonctions_nD& ) {};
211     // cas donne le niveau de sauvegarde
212     // = 1 : on sauvegarde tout
213     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
214     void Ecriture_don_base_info(ofstream& ,const int ) const {};
215     // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
216     // exemple: mise en service de de certain ddl aux noeuds
217     // méthode appelée par Activation_donnees principal, ou des classes dérivées
218     // ce qui permet de surcharger ces dernières: actuellement rien !!
219     void Activ_donnees();
220
221     // VARIABLES PROTEGEES :
222     // pointeur de travail utilise par les classes derivantes
223     SaveResul * saveResul;
224     // indic pour définir si oui ou non on utilise un comportement tangent simplifié
225     bool comp_tangent_simplifie;
226     // ----- variables gérées en I/O par les classes dérivées -----
227     protected :
228
229     // 3) METHODES VIRTUELLES PURES protegees:
230     // calcul des efforts de frottement à un instant t+deltat
231     // les indices t se rapporte au pas précédent, sans indice au temps actuel
232     // vit_T : vitesse, force_normale: force normale (à la cible) agissant sur le noeud projectile
233     // force_tangente: force tangente (à la cible) agissant sur le noeud projectile
234     // normale: la normale de contact normée
235     // energie_frottement: l'énergie échangée: élas=la totalité, viqueux et plas= uniquement le pas de
236     // temps
237     // delta_t : le pas de temps
238     // F_frot: force de frottement calculé,
239     // retour glisse: indique si oui ou non le noeud glisse
240     virtual bool Calcul_Frottement(const Coordonnee& vit_T, const Coordonnee& normale
241     ,const Coordonnee& force_normale,const Coordonnee& force_tangente
242     ,EnergieMeca& energie_frottement,const double delta_t
243     ,Coordonnee& F_frot) = 0;
244
245     // calcul des efforts de frottement à un instant t+deltat
246     // et ses variations par rapport aux ddl de vitesse
247     // vit_T : vitesse, force_normale: force normale (à la cible) agissant sur le noeud projectile
248     // force_tangente: force tangente (à la cible) agissant sur le noeud projectile
249     // energie_frottement: l'énergie échangée: élas=la totalité, viqueux et plas= uniquement le pas de
250     // temps
251     // delta_t : le pas de temps
252     // F_frot: force de frottement calculé,

```



```

251 // d_F_frot_vit: variation de la force de frottement par rapport aux coordonnées de vitesse
252 // retour glisse: indique si oui ou non le noeud glisse
253 virtual bool Calcul_DFrottement_tdt
254     (const Coordonnee& vit_T, const Coordonnee& normale
255     ,const Coordonnee& force_normale,const Coordonnee& force_tangente
256     ,EnergieMeca& energie_frottement,const double delta_t
257     ,Coordonnee& F_frot,Tableau <Coordonnee>& d_F_frot_vit) = 0;
258
259 };
260
261
262 #endif

```

## 7.26 CompThermoPhysiqueAbstraite.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      13/04/2004
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++
37 *
38 *****/
39 *   BUT:  Interface pour les lois thermo_physiques.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date ! auteur ! but
45 *   -----
46 *   ! ! !
47 *   $
48 *   *****
49 *
50 *   MODIFICATIONS:
51 *   ! date ! auteur ! but
52 *   -----
53 *   $
54 *****/
55 // FICHER : CompThermoPhysiqueAbstraite.h
56 // CLASSE : CompThermoPhysiqueAbstraite
57 /// pour les comportements thermophysiques
58
59
60 #ifndef COMPTHERMOPHYSIQUEABSTRAITE_H
61 #define COMPTHERMOPHYSIQUEABSTRAITE_H
62
63
64 #include "Enum_comp.h"
65 #include "Tableau_T.h"
66 #include "Tenseur.h"
67 #include "Deformation.h"
68 #include "LoiAbstraiteGeneral.h"
69 #include "ThermoDonnee.h"

```

```

70 #include "TypeQuelconque.h"
71 #include "PtIntegMecaInterne.h"
72 #include "LesPtIntegThermiInterne.h"
73 #include "EnergieThermi.h"
74 #include "Temps_CPU_HZpp.h"
75 #include "Bloc.h"
76
77
78 class CompThermoPhysiqueAbstraite : public LoiAbstraiteGeneral
79 {
80
81     public :
82
83         // CONSTRUCTEURS :
84
85         // Constructeur par défaut
86         CompThermoPhysiqueAbstraite () ;
87
88         // Constructeur utile si l'identificateur du nom de la loi
89         // de comportement et la dimension sont connus
90         CompThermoPhysiqueAbstraite (Enum_comp id_compor,Enum_categorie_loi_comp categorie_comp,int
dimension);
91
92         // Constructeur utile si l'identificateur du nom de la loi
93         // de comportement et la dimension sont connus
94         CompThermoPhysiqueAbstraite (char* nom,Enum_categorie_loi_comp categorie_comp,int dimension);
95
96         // Constructeur de copie
97         CompThermoPhysiqueAbstraite (const CompThermoPhysiqueAbstraite & a ) ;
98
99         // DESTRUCTEUR VIRTUEL :
100
101         virtual ~CompThermoPhysiqueAbstraite ();
102
103
104 // 2) METHODES VIRTUELLES public:
105
106 // classe de stockage de paramètres de travail, utiliser par les classes dérivées
107 class StockParaInt // on globalise, pour ne pas stocker lors que l'on ne veut pas postraiter
108 {public:
109     // constructeur par défaut
110     StockParaInt() : pression(0.),temperature(0.) {};
111     // constructeur fonction des paramètres
112     StockParaInt(double press, double temper) :
113     pression(press),temperature(temper) {};
114     // constructeur de copie
115     StockParaInt(const StockParaInt& a) :
116     pression(a.pression),temperature(a.temperature) {};
117
118     double pression;
119     double temperature;
120 };
121
122 // initialise les donnees particulieres a l'elements
123 // de matiere traite ( c-a-dire au pt calcule)
124 // Il y a creation d'une instance de SaveResul particuliere
125 // a la loi concernee
126 // la SaveResul classe est remplie par les instances heritantes
127 // le pointeur de SaveResul est sauvegarde au niveau de l'element
128 // c'a-d que les info particulieres au point considere sont stocke
129 // au niveau de l'element et non de la loi.
130 class SaveResul
131 { public :
132     // destructeur virtuelle car d'une classe virtuelle
133     virtual ~SaveResul() {};
134     // définition d'une nouvelle instance identique
135     // appelle du constructeur via new
136     virtual SaveResul * Nevez_SaveResul() const =0;
137 // affectation
138 virtual SaveResul & operator = ( const SaveResul &) = 0;
139 //===== lecture écriture dans base info =====
140 // cas donne le niveau de la récupération
141 // = 1 : on récupère tout
142 // = 2 : on récupère uniquement les données variables (supposées comme telles)
143 virtual void Lecture_base_info (ifstream& ent,const int cas) = 0;
144 // cas donne le niveau de sauvegarde
145 // = 1 : on sauvegarde tout
146 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
147 virtual void Ecriture_base_info(ofstream& sort,const int cas) = 0;
148
149 // mise à jour des informations transitoires en définitif s'il y a convergence
150 // par exemple (pour la plasticité par exemple)
151 virtual void TdtversT() = 0;
152 virtual void TversTdt() = 0;
153
154 //changement de base de toutes les grandeurs internes tensorielles stockées
155 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gpB

```

```

156 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
157 // gpH(i) = gamma(i,j) * gH(j)
158 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) = 0;
159
160 // on stocke la température initiale (les autres sont dans les ptinthermi)
161 double temperature_0;
162 };
163
164 virtual SaveResul * New_et_Initialise() { return NULL;};
165
166 // affichage des donnees particulieres a l'elements
167 // de matiere traite ( c-a-dire au pt calcule)
168 virtual void AfficheDataSpecif(ofstream& ,SaveResul * ) const {};
169
170 // ramène les données thermiques définies au point
171 // P: la pression à l'énuméré temps, et P_t la pression au temps t
172 virtual void Cal_donnees_thermiques(const double& P_t, CompThermoPhysiqueAbstraite::SaveResul * saveTP
173 ,const Deformation & def,const double& P,Enum_dure temps,ThermoDonnee&
donneeThermique) = 0;
174 // définition du type de calcul de déformation sur une instance de déformation passée en
paramètre
175 void Def_type_deformation(Deformation & def);
176
177 // calcul de toutes les grandeurs associées à la température (mais pas le flux), qui sont stockées
178 // dans le point d'intégration : ptIntegThermi
179 // et de la variation du gradient thermique / au ddl
180 void Cal_cinematique_thermique(bool premier_calcul,PtIntegThermiInterne& ptIntegThermi
181 ,Tableau <CoordonneeB >& d_gradTB
182 ,Deformation & def,const Met_abstraite::Impli& ex);
183
184
185 // schema de calcul explicite à t
186 // virtual const Met_abstraite::Expli& Cal_explicit_t
187 // (Loi_comp_abstraite::SaveResul * saveDon
188 // ,Deformation & def, DdlElement & tab_ddl
189 // ,PtIntegMecaInterne& ptintmeca,Tableau <TenseurBB *> & d_epsBB,double& Jacobien
190 // ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite* loiTP
191 // ,bool dilatation,EnergieMeca & energ,const EnergieMeca & energ_t,bool premier_calcul
192 // );
193 //
194 // // schema de calcul explicite à tdt
195 // virtual const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt
196 // (Loi_comp_abstraite::SaveResul * saveDon
197 // ,Deformation & def, DdlElement & tab_ddl
198 // ,PtIntegMecaInterne& ptintmeca,Tableau <TenseurBB *> & d_epsBB,double& Jacobien
199 // ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite* loiTP
200 // ,bool dilatation,EnergieMeca & energ,const EnergieMeca & energ_t,bool premier_calcul
201 // );
202 //
203 // // schema implicite
204 // // P et P_t : pression actuelle et pression à t
205 virtual const Met_abstraite::Impli& Cal_implicit
206 (const double & P_t,CompThermoPhysiqueAbstraite::SaveResul * saveDon
207 , const double & P,Deformation & def,DdlElement & tab_ddl
208 ,PtIntegThermiInterne& ptIntegThermi, Tableau <CoordonneeB >& d_gradTB
209 ,Tableau <CoordonneeH >& d_fluxH,const ParaAlgoControle & pa
210 ,CompThermoPhysiqueAbstraite* loiTP
211 ,bool dilatation,EnergieThermi & energ,const EnergieThermi & energ_t,bool
premier_calcul
212 );
213 //
214 // // schema pour le flambage linéaire
215 // virtual void Cal_flamb_lin
216 // (Loi_comp_abstraite::SaveResul * saveDon,Deformation & def
217 // ,DdlElement & tab_ddl
218 // ,PtIntegMecaInterne& ptintmeca, Tableau <TenseurBB *>& d_epsBB_tdt,double& jacobien
219 // ,Vecteur & d_jacobien_tdt,Tableau <TenseurHH *>& d_sigHH,const ParaAlgoControle & pa
220 // ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite*
loiTP
221 // ,bool dilatation,EnergieMeca & energ,const EnergieMeca & energ_t,bool
premier_calcul
222 // );
223 //
224 // // schema pour le calcul de la loi de comportement dans le cas de l'umat
225 // virtual void ComportementUmat
226 // (Loi_comp_abstraite::SaveResul * saveDon
227 // ,Deformation & def,PtIntegMecaInterne& ptintmeca
228 // ,ParaAlgoControle & pa
229 // ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite*
loiTP
230 // ,bool dilatation,UmatAbaqus& umatAbaqusqus,bool premier_calcul
231 // );
232 //
233 // // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
234 // // exemple: mise en service des ddl de température aux noeuds
235 virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud) {Activ_donnees(tabnoeud);};
236

```

```

237         // modification de l'indicateur de comportement tangent
238         void Modif_comp_tangent_simplifie(bool modif)
239         { comp_tangent_simplifie = modif;};
240
241 // test pour connaître l'état du comportement : simplifié ou non
242 bool Test_loi_simplifie()
243     { return comp_tangent_simplifie;};
244
245 // récupération des grandeurs particulière (hors ddl )
246 // correspondant à liTQ
247 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
248 virtual void Grandeur_particuliere(bool absolue,List_io<TypeQuelconque>&
    ,CompThermoPhysiqueAbstraite::SaveResul * ,list<int>&)
249     {};
250 // récupération de la liste de tous les grandeurs particulières
251 // ces grandeurs sont ajoutées à la liste passées en paramètres
252 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
253 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) {};
254
255 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
256 virtual CompThermoPhysiqueAbstraite* Nouvelle_loi_identique() const = 0;
257
258     protected :
259
260 // 3) METHODES VIRTUELLES PURES protegees:
261 // calcul des contraintes à un instant t+deltat
262 // les indices t se rapporte au pas précédent, sans indice au temps actuel
263 // virtual void Calcul_SigmaHH
264 //     (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
265 //     ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB & giB,BaseH & gi_H,TenseurBB & epsBB
266 //     ,TenseurBB & delta_epsBB,TenseurBB & gijBB,TenseurHH & gijHH,Tableau <TenseurBB *>& d_gijBB
267 //     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
268 //     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
269 //     ,const Met_abstraite::Expli_t_tdt& ex) = 0;
270
271 // calcul du flux et ses variations par rapport aux ddl a t+dt: stockage dans ptIntegThermi et dans
    d_flux
272 // calcul également des paramètres thermiques dTP ainsi que des énergies mises en jeu
273 // calcul des énergies thermiques
274 // en entrée: température, gradient de temp, et grandeurs associées, métrique
275 virtual void Calcul_DfluxH_tdt
276     (const double & P_t,PtIntegThermiInterne& ptIntegThermi, const double & P,DdlElement & tab_ddl
277     ,const Deformation & def // prévue pour servir pour l'interpolation
278     , Tableau <CoordonneeB >& d_gradTB,Tableau <CoordonneeH >& d_flux,ThermoDonnee& dTP
279     ,EnergieThermi & energ,const EnergieThermi & energ_t,const Met_abstraite::Impli& ex) = 0;
280
281 // // calcul des contraintes et ses variations par rapport aux déformations a t+dt
282 // // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormée
283 // // le tenseur de déformation et son incrémentsont également en orthonormee
284 // // si = false: les bases transmises sont utilisées
285 // // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
286 // virtual void Calcul_dsigma_deps
287 //     (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
288 //     ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
289 //     ,TenseurHH& sigHH,TenseurHHH& d_sigma_deps
290 //     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
291 //     ,const Met_abstraite::Umat_cont& ex) ; // = 0;
292
293 // // affichage et definition interactive des commandes particulières à la classe
    CompThermoPhysiqueAbstraite
294 void Info_commande_don LoisDeComp(UtilLecture& ) const {};
295 //---- lecture écriture de restart spécifique aux données de la classe ----
296 // cas donne le niveau de la récupération
297 // = 1 : on récupère tout
298 // = 2 : on récupère uniquement les données variables (supposées comme telles)
299 void Lecture_don_base_info(ifstream& ,const int ,LesReferences& ,LesCourbesID& ,LesFonctions_nD&
    );
300 // cas donne le niveau de sauvegarde
301 // = 1 : on sauvegarde tout
302 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
303 void Ecriture_don_base_info(ofstream& ,const int ) const {};
304
305 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
306 // exemple: mise en service des ddl de température aux noeuds
307 // méthode appelée par Activation_donnees principal, ou des classes dérivées
308 // ce qui permet de surcharger ces dernières
309 void Activ_donnees(Tableau<Noeud *>& tabnoeud);
310 // calcul de grandeurs de travail aux points d'intégration via la def
311 // fonction surchargée dans les classes dérivée si besoin est
312 virtual void CalculGrandeurTravail(const PtIntegThermiInterne& ,const Deformation & ,Enum_dure) {};
313
314
315 // VARIABLES PROTEGEES :
316 // pointeur de travail utilise par les classes derivantes
317 SaveResul * saveResul;

```

```

318 // indic pour définir si oui ou non on utilise un comportement tangent simplifié
319 bool comp_tangent_simplifie;
320 // ----- variables gérées en I/O par les classes dérivées -----
321 // et variables de travail
322 // indique si oui ou non la loi dépend de la température
323 bool thermo_dependant; // paramètre lue par les classes dérivées
324 double temperature; // variable valide que si l'on est thermo_dependant
325 // utilisée par les classes dérivées
326
327 Temps_CPU_HZpp temps_loi; // spécifique à ce type de loi: cumule tous les appels
328
329 // du calcul de la contrainte
330
331 };
332
333
334 #endif

```

## 7.27 EnergieMeca.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           14/03/2005
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 *   *****
39 *   BUT: Conteneur pour stocker les differentes energies générées
40 *   par la mécanique.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *   ! date ! auteur ! but
46 *   -----
47 *   ! ! !
48 *   $
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date ! auteur ! but
52 *   -----
53 *   $
54 *   *****/
55 #ifndef ENERGIE_MECA_H
56 #define ENERGIE_MECA_H
57
58 #include <iostream>
59 #include <fstream>
60 #include <stdlib.h>
61 #include "Sortie.h"
62 /** @defgroup Les_conteneurs_energies
63 *
64 *   BUT: groupe des conteneurs pour les différentes énergies

```

```

65 *
66 *
67 * \author   Gérard Rio
68 * \version  1.0
69 * \date     14/03/2005
70 * \brief    groupe des conteneurs pour les différentes énergies
71 *
72 */
73
74 /// @addtogroup Les_conteneurs_energies
75 /// @{
76 ///
77
78
79 class EnergieMeca
80 {
81     // surcharge de l'operator de lecture avec le type
82     friend istream & operator » (istream &, EnergieMeca &);
83     // surcharge de l'operator d'écriture
84     friend ostream & operator « (ostream &, const EnergieMeca &);
85
86 public :
87     // CONSTRUCTEURS :
88     // par défaut
89     EnergieMeca() :
90         energie_elastique(0.),dissipation_plastique(0.),dissipation_visqueuse(0.)
91     {};
92     // constructeur fonction des trois énergies
93     EnergieMeca(const double& e_elastique, const double& d_plastique,
94                const double& d_visqueuse) :
95         energie_elastique(e_elastique),dissipation_plastique(d_plastique)
96         ,dissipation_visqueuse(d_visqueuse)
97     {};
98
99     // de copie
100    EnergieMeca(const EnergieMeca& a) :
101        energie_elastique(a.energie_elastique),dissipation_plastique(a.dissipation_plastique)
102        ,dissipation_visqueuse(a.dissipation_visqueuse)
103    {};
104
105    // DESTRUCTEUR :
106
107    // METHODES PUBLIQUES :
108    // affichage à l'écran
109    void Affiche() const
110    {cout << "\n EnergieMeca: energie_elastique=" << energie_elastique
111     << " dissipation_plastique=" << dissipation_plastique
112     << " dissipation_visqueuse=" << dissipation_visqueuse << " ";
113     };
114
115    // opérateurs
116    EnergieMeca operator + ( const EnergieMeca & a) const
117    {EnergieMeca ret(*this);
118     ret += a;
119     return ret;
120    };
121    EnergieMeca operator - ( const EnergieMeca & a) const
122    {EnergieMeca ret(*this);
123     ret -= a;
124     return ret;
125    };
126    void operator += ( const EnergieMeca & a)
127    {energie_elastique += a.energie_elastique;
128     dissipation_plastique += a.dissipation_plastique;
129     dissipation_visqueuse += a.dissipation_visqueuse;
130    };
131    void operator -= ( const EnergieMeca & a)
132    {energie_elastique -= a.energie_elastique;
133     dissipation_plastique -= a.dissipation_plastique;
134     dissipation_visqueuse -= a.dissipation_visqueuse;
135    };
136    void operator *= ( const double & x)
137    {energie_elastique *= x;dissipation_plastique *= x;dissipation_visqueuse *= x;};
138    void operator /= ( const double & x)
139    {energie_elastique /= x;dissipation_plastique /= x;dissipation_visqueuse /= x;};
140
141    EnergieMeca operator * ( const double & x) const
142    { EnergieMeca re(*this); re *=x; return re;};
143
144    EnergieMeca operator / ( const double & x) const
145    { EnergieMeca re(*this); re /=x; return re;};
146
147    EnergieMeca & operator = ( const EnergieMeca & a)
148    {energie_elastique = a.energie_elastique;
149     dissipation_plastique = a.dissipation_plastique;
150     dissipation_visqueuse = a.dissipation_visqueuse;
151     return *this;

```

```

152     };
153
154     // initialisation à une valeur vale
155     void Initia(const double& vale)
156     {energie_elastique = dissipation_plastique = dissipation_visqueuse =vale;
157     };
158
159     // initialisation différenciée
160     // élastique à 0 et les autres aux valeurs passées en paramètre
161     void Initialisation_differenciee(const EnergieMeca & energlob)
162     {energie_elastique = 0.;
163     dissipation_plastique = energlob.dissipation_plastique;
164     dissipation_visqueuse = energlob.dissipation_visqueuse;
165     };
166
167     // mise à jour différenciée
168     // l'énergie élastique de this = celle d'ener * coef
169     // les deux autres énergies sont incrémentées par le delta valeur
170     void Ajout_differenciee(const EnergieMeca & ener, const EnergieMeca & ener_t,const double& coef)
171     {energie_elastique += coef * ener.energie_elastique;
172     dissipation_plastique += coef * (ener.dissipation_plastique-ener_t.dissipation_plastique);
173     dissipation_visqueuse += coef * (ener.dissipation_visqueuse-ener_t.dissipation_visqueuse);
174     };
175
176     // retourne l'énergie élastique
177     const double& EnergieElastique() const {return energie_elastique;};
178     // retourne la dissipation plastique
179     const double& DissipationPlastique() const {return dissipation_plastique;};
180     // retourne la dissipation visqueuse
181     const double & DissipationVisqueuse() const {return dissipation_visqueuse;};
182
183     // changement de l'énergie élastique
184     void ChangeEnergieElastique(double en) { energie_elastique = en;};
185     // changement de la dissipation plastique
186     void ChangeDissipationPlastique(const double& en) {dissipation_plastique=en;};
187     void ChangeDissipationVisqueuse(const double& en) {dissipation_visqueuse=en;};
188
189     private :
190     // VARIABLES PROTEGEES :
191     double energie_elastique; // énergie élastique
192     double dissipation_plastique ; // dissipation plastique
193     double dissipation_visqueuse; // dissipation visqueuse
194
195     // METHODES PROTEGEES :
196
197 };
198 /// @} // end of group
199
200 #endif

```

## 7.28 EnergieThermi.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      06/03/2023
33 *

```

```

34 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)           *
35 *                                                     $   *
36 *   PROJET:      Herezh++                                     *
37 *                                                     $   *
38 * *****
39 *   BUT: Conteneur pour stocker les differentes energies générées *
40 *   par la thermique.                                         *
41 *                                                     $   *
42 *   ////////////////////////////////////////////////// *
43 *   *
44 *   VERIFICATION:                                           *
45 *   ! date !   auteur !           but                       !   *
46 *   -----
47 *   !           !           !                               !   *
48 *   $
49 *   ////////////////////////////////////////////////// *
50 *   MODIFICATIONS:                                           *
51 *   ! date !   auteur !           but                       !   *
52 *   -----
53 *   $
54 * *****/
55 #ifndef ENERGIE_THERMI_H
56 #define ENERGIE_THERMI_H
57
58 #include <iostream>
59 #include <fstream>
60 #include <stdlib.h>
61 #include "Sortie.h"
62
63 /// @addtogroup Les_conteneurs_energies
64 /// @{
65 ///
66
67
68 class EnergieThermi
69 {
70     // surcharge de l'operator de lecture avec le type
71     friend istream & operator » (istream &, EnergieThermi &);
72     // surcharge de l'operator d'écriture
73     friend ostream & operator « (ostream &, const EnergieThermi &);
74
75 public :
76     // CONSTRUCTEURS :
77     // par défaut
78     EnergieThermi() :
79         energie_elastique(0.),dissipation_plastique(0.),dissipation_visqueuse(0.)
80     {};
81     // constructeur fonction des trois énergies
82     EnergieThermi(const double& e_elastique, const double& d_plastique,
83                 const double& d_visqueuse) :
84         energie_elastique(e_elastique),dissipation_plastique(d_plastique)
85         ,dissipation_visqueuse(d_visqueuse)
86     {};
87
88     // de copie
89     EnergieThermi(const EnergieThermi& a) :
90         energie_elastique(a.energie_elastique),dissipation_plastique(a.dissipation_plastique)
91         ,dissipation_visqueuse(a.dissipation_visqueuse)
92     {};
93
94     // DESTRUCTEUR :
95
96     // METHODES PUBLIQUES :
97     // affichage à l'écran
98     void Affiche() const
99     {cout << "\n EnergieThermi: energie_elastique=" << energie_elastique
100      << " dissipation_plastique=" << dissipation_plastique
101      << " dissipation_visqueuse=" << dissipation_visqueuse << " ";
102     };
103
104     // opérateurs
105     EnergieThermi operator + ( const EnergieThermi & a) const
106     {EnergieThermi ret(*this);
107      ret += a;
108      return ret;
109     };
110     EnergieThermi operator - ( const EnergieThermi & a) const
111     {EnergieThermi ret(*this);
112      ret -= a;
113      return ret;
114     };
115     void operator += ( const EnergieThermi & a)
116     {energie_elastique += a.energie_elastique;
117      dissipation_plastique += a.dissipation_plastique;
118      dissipation_visqueuse += a.dissipation_visqueuse;
119     };

```



```

120 void operator -= ( const EnergieThermi & a)
121 {energie_elastique -= a.energie_elastique;
122  dissipation_plastique -= a.dissipation_plastique;
123  dissipation_visqueuse -= a.dissipation_visqueuse;
124  };
125 void operator *= ( const double & x)
126 {energie_elastique *= x;dissipation_plastique *= x;dissipation_visqueuse *= x;};
127 void operator /= ( const double & x)
128 {energie_elastique /= x;dissipation_plastique /= x;dissipation_visqueuse /= x;};
129
130 EnergieThermi operator * ( const double & x) const
131 { EnergieThermi re(*this); re *=x; return re;};
132
133 EnergieThermi operator / ( const double & x) const
134 { EnergieThermi re(*this); re /=x; return re;};
135
136 EnergieThermi & operator = ( const EnergieThermi & a)
137 {energie_elastique = a.energie_elastique;
138  dissipation_plastique = a.dissipation_plastique;
139  dissipation_visqueuse = a.dissipation_visqueuse;
140  return *this;
141  };
142
143 // initialisation à une valeur vale
144 void Inita(const double& vale)
145 {energie_elastique = dissipation_plastique = dissipation_visqueuse =vale;
146  };
147
148 // retourne l'énergie élastique
149 const double& EnergieElastique() const {return energie_elastique;};
150 // retourne la dissipation plastique
151 const double& DissipationPlastique() const {return dissipation_plastique;};
152 // retourne la dissipation visqueuse
153 const double & DissipationVisqueuse() const {return dissipation_visqueuse;};
154
155 // changement de l'énergie élastique
156 void ChangeEnergieElastique(double en) { energie_elastique = en;};
157 // changement de la dissipation plastique
158 void ChangeDissipationPlastique(const double& en) {dissipation_plastique=en;};
159 void ChangeDissipationVisqueuse(const double& en) {dissipation_visqueuse=en;};
160
161 private :
162 // VARIABLES PROTEGEES :
163 double energie_elastique; // énergie élastique
164 double dissipation_plastique ; // dissipation plastique
165 double dissipation_visqueuse; // dissipation visqueuse
166
167 // METHODES PROTEGEES :
168
169 };
170 /// @} // end of group
171
172 #endif

```

## 7.29 ExceptionsLoiComp.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29

```

```

30 /*****
31 *   DATE:      16/03/2006
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:  Définir des classes d'exception pour la gestion d'erreur
39 *         concernant les lois de comportements.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !       but
45 *   -----
46 *   !       !       !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !       but
51 *   -----
52 *
53 *****/
54 #ifndef EXCEPTIONSLOICOMP_H
55 #define EXCEPTIONSLOICOMP_H
56
57
58 /// @addtogroup Groupe_concernant_le_chargement
59 /// @{
60 ///
61
62 /// cas d'une erreur survenue à cause d'une non convergence pour la résolution
63 /// d'une loi de comportement incrémentale
64
65 class ErrNonConvergence_loiDeComportement
66     // =0 cas courant, pas d'information particulière
67     // =1 cas où l'erreur est sévère et ne pourra pas être corrigé en refaisant un calcul avec un
68     // pas de temps plus petit. Il faut refaire le calcul en se positionnant plusieurs pas de temps
69     // auparavant (utilisé par l'hystérésis par exemple)
70 { public :
71     int cas;
72     ErrNonConvergence_loiDeComportement () : cas(0) {} ; // par défaut
73     ErrNonConvergence_loiDeComportement (int ca) : cas(ca) {} ; // pb
74 };
75 /// @} // end of group
76
77
78 #endif

```

## 7.30 CompFrotCoulomb.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      04/05/2006

```

```

33 *
34 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)      $   *
35 *
36 *   PROJET:      Herezh++                                $   *
37 *
38 * *****
39 *   BUT:   Lois de frottement de coulomb généralisé ou non $   *
40 *
41 *   ////////////////////////////////////////////////// *
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but                       !   *
45 *   -----
46 *   !           !           !                               !   *
47 *   $
48 *   ////////////////////////////////////////////////// *
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but                       !   *
51 *   -----
52 *   $
53 * *****/
54 // FICHER : CompFrotCoulomb.h
55 // CLASSE : CompFrotCoulomb
56
57 #ifndef COMP_FROT_COULOMB_H
58 #define COMP_FROT_COULOMB_H
59
60 #include "CompFrotAbstraite.h"
61
62 class CompFrotCoulomb : public CompFrotAbstraite
63 {
64 public :
65     // CONSTRUCTEURS :
66
67     // Constructeur par défaut
68     CompFrotCoulomb () ;
69
70     // Constructeur utile si l'identificateur du nom de la loi
71     // de comportement et la dimension sont connus
72     CompFrotCoulomb (Enum_comp id_compor,Enum_categorie_loi_comp categorie_comp,int dimension);
73
74     // Constructeur utile si l'identificateur du nom de la loi
75     // de comportement et la dimension sont connus
76     CompFrotCoulomb (char* nom,Enum_categorie_loi_comp categorie_comp,int dimension);
77
78     // Constructeur de copie
79     CompFrotCoulomb (const CompFrotCoulomb & a );
80
81     // DESTRUCTEUR VIRTUEL :
82
83     ~CompFrotCoulomb ();
84
85     // Lecture des donnees de la classe sur fichier
86     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD&
lesFonctionsnD);
87     // affichage de la loi
88     void Affiche() const ;
89     // test si la loi est complete
90     // = 1 tout est ok, =0 loi incomplete
91     int TestCompleet();
92
93 // 2) METHODES public:
94
95
96     //----- lecture écriture de restart -----
97     // cas donne le niveau de la récupération
98     // = 1 : on récupère tout
99     // = 2 : on récupère uniquement les données variables (supposées comme telles)
100    void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
101                                ,LesFonctions_nD& lesFonctionsnD);
102
103     // cas donne le niveau de sauvegarde
104     // = 1 : on sauvegarde tout
105     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
106     void Ecriture_base_info_loi(ofstream& sort,const int cas);
107
108     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
109     CompFrotAbstraite* Nouvelle_loi_identique() const { return (new CompFrotCoulomb(*this)); };
110
111     // affichage et definition interactive des commandes particulières à chaque lois
112     void Info_commande_LoisDeComp(UtilLecture& lec);
113
114     //----- dérivées de virtuelle pures -----
115
116 protected :

```

```

117
118 // affichage et definition interactive des commandes particulières à la classe CompFrotCoulomb
119 void Info_commande_don_LoisDeComp(UtilLecture& ) const {};
120 //----- lecture écriture de restart spécifique aux données de la classe -----
121 // cas donne le niveau de la récupération
122 // = 1 : on récupère tout
123 // = 2 : on récupère uniquement les données variables (supposées comme telles)
124 void Lecture_don_base_info(istream& ,const int ,LesReferences& ,LesCourbesID& ) ;
125 // cas donne le niveau de sauvegarde
126 // = 1 : on sauvegarde tout
127 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
128 void Ecriture_don_base_info(ofstream& ,const int ) const ;
129
130
131 // VARIABLES PROTEGEES :
132 double mu_statique; // coefficient de coulomb statique
133 double* mu_cine; // coefficient de coulomb cinématique (peut ne pas exister)
134 double* x_c; // coefficient qui permet de contrôler le passage de mu_statique à mu_cine
135 // en fonction de la vitesse
136 int regularisation; // = 0 : pas de régularisation
137 // = 1: régularisation quadratique; = 2: régularisation en tangent hyperbolique
138 // = 3: régularisation par morceau de droite
139 // = 4: régularisation par une fonction donnée par l'utilisateur
140 CourbeID* fonc_regul; // courbe éventuelle de régularisation (=4) fonction de la vitesse
141
142 double epsilon; // paramètre de contrôle de l'ampleur de la régularisation
143
144 // 3) METHODES VIRTUELLES PURES protegees:
145 // calcul des efforts de frottement à un instant t+deltat
146 // les indices t se rapporte au pas précédent, sans indice au temps actuel
147 // vit_T : vitesse, force_normale: force normale (à la cible) agissant sur le noeud projectile
148 // force_tangente: force tangente (à la cible) agissant sur le noeud projectile
149 // normale: la normale de contact normée
150 // energie_frottement: l'énergie échangée: élas=la totalité, viqueux et plas= uniquement le pas de
151 // temps
152 // delta_t : le pas de temps
153 // F_frot: force de frottement calculé,
154 // retour glisse: indique si oui ou non le noeud glisse
155 bool Calcul_Frottement(const Coordonnee& vit_T, const Coordonnee& normale
156 // ,const Coordonnee& force_normale,const Coordonnee& force_tangente
157 // ,EnergieMeca& energie_frottement,const double delta_t
158 // ,Coordonnee& F_frot) ;
159
160 // calcul des efforts de frottement à un instant t+deltat
161 // et ses variations par rapport aux ddl de vitesse
162 // vit_T : vitesse, force_normale: force normale (à la cible) agissant sur le noeud projectile
163 // force_tangente: force tangente (à la cible) agissant sur le noeud projectile
164 // energie_frottement: l'énergie échangée: élas=la totalité, viqueux et plas= uniquement le pas de
165 // temps
166 // delta_t : le pas de temps
167 // F_frot: force de frottement calculé,
168 // d_F_frot_vit: variation de la force de frottement par rapport aux coordonnées de vitesse
169 // retour glisse: indique si oui ou non le noeud glisse
170 bool Calcul_DFrottement_tdt
171 // (const Coordonnee& vit_T, const Coordonnee& normale
172 // ,const Coordonnee& force_normale,const Coordonnee& force_tangente
173 // ,EnergieMeca& energie_frottement,const double delta_t
174 // ,Coordonnee& F_frot,Tableau <Coordonnee>& d_F_frot_vit) ;
175
176 };
177 #endif

```

## 7.31 Hart\_Smith3D.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.

```

```

20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      6/12/2007
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++
37 *
38 *
39 *   BUT:      La classe Hart_Smith permet de calculer la contrainte
40 *            et ses derivees pour une loi isotrope hyper elastique
41 *            de type Hart Smith en 3D. Ici on considere
42 *            la variation de volume.
43 *            Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
44 *
45 *            *****
46 *   VERIFICATION:
47 *
48 *   ! date !   auteur !           but
49 *   -----
50 *   !       !           !
51 *
52 *            *****
53 *   MODIFICATIONS:
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *            *****/
58 #ifndef HART_SMITH_3D_H
59 #define HART_SMITH_3D_H
60
61
62
63
64 #include "Loi_comp_abstraite.h"
65 #include "Courbe1D.h"
66 #include "MathUtil.h"
67 #include "Hyper_W_gene_3D.h"
68 #include "CourbePolyLineaire1D.h"
69
70
71 /// @addtogroup Les_lois_hyperelastiques
72 /// @{
73 ///
74
75
76 class Hart_Smith3D : public Hyper_W_gene_3D
77 {
78
79     public :
80
81
82         // CONSTRUCTEURS :
83
84         // Constructeur par default
85         Hart_Smith3D ();
86
87
88         // Constructeur de copie
89         Hart_Smith3D (const Hart_Smith3D& loi) ;
90
91         // DESTRUCTEUR :
92
93         ~Hart_Smith3D ();
94
95         // Lecture des donnees de la classe sur fichier
96         void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD&
97         lesFonctionsnD);
98         // affichage de la loi
99         void Affiche() const ;
100        // test si la loi est complete
101        // = 1 tout est ok, =0 loi incomplete
102        int TestComplet();
103
104        //----- lecture écriture de restart -----
105        // cas donne le niveau de la récupération

```

```

106 // = 1 : on récupère tout
107 // = 2 : on récupère uniquement les données variables (supposées comme telles)
108 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
109                                     ,LesFonctions_nD& lesFonctionsnD);
110 // cas donne le niveau de sauvegarde
111 // = 1 : on sauvegarde tout
112 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
113 void Ecriture_base_info_loi(ofstream& sort,const int cas);
114
115 // calcul d'un module d'young équivalent à la loi,
116 // c'est sans doute complètement débile mais c'est pour pouvoir avancer !!
117 // Annexe B, formule B12
118 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul *) {return 6.* C1;};
119
120 // récupération de la variation relative d'épaisseur calculée: h/h0
121 // cette variation n'est utile que pour des lois en contraintes planes
122 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
123 // - pour les lois 2D def planes: retour de 0
124 // les infos nécessaires à la récupération , sont stockées dans saveResul
125 // qui est le conteneur spécifique au point où a été calculé la loi
126 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
127
128 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
129 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hart_Smith3D(*this)); };
130
131 // affichage et definition interactive des commandes particulières à chaque lois
132 void Info_commande_LoisDeComp(UtilLecture& lec);
133
134 protected :
135 // donnée de la loi
136 double C1,C2,C3,K; // 4 coeffs
137 CourbelD* C1_temperature; // courbe éventuelle d'évolution de C10 en fonction de la température
138 CourbelD* C2_temperature; // courbe éventuelle d'évolution de C10 en fonction de la température
139 CourbelD* C3_temperature; // courbe éventuelle d'évolution de C01 en fonction de la température
140 CourbelD* K_temperature; // courbe éventuelle d'évolution de K en fonction de la température
141 int type_pot_vol; // indique le type de potentiel volumique, par défaut 2
142 bool avec_courbure; // indique s'il y a un potentiel de courbure ou non
143 double a_courbure; // para a utilisé que dans le cas avec courbure
144 double r_courbure; // para r utilisé que dans le cas avec courbure
145 CourbelD* a_temperature; // courbe éventuelle d'évolution de a en fonction de la température
146 CourbelD* r_temperature; // courbe éventuelle d'évolution de r en fonction de la température
147
148 // définition de la fonction énergie
149 CourbePolylineaire1D int_J1; // intégrale de la partie relative à J1
150
151
152 double W_d,W_v; // le potentiel: partie déviatorique, partie sphérique
153 Vecteur W_r; // dérivées premières du potentiel par rapport aux J_r
154 double W_d_J1,W_d_J2; // dérivées premières du potentiel déviatoire par rapport aux J_1 et J_2
155 double W_d_J1_2,W_d_J1_J2,W_d_J2_2; // dérivées secondes
156 double W_v_J3,W_v_J3J3; // dérivées premières et seconde du potentiel volumique / J3
157 Tableau2 <double> W_rs; // dérivées secondes du potentiel par rapport aux J_r
158 // cas éventuel de potentiel additionnel de raidissement: variables intermédiaires de passage
159 double W_c,W_c_J1,W_c_J3,W_c_J1_2,W_c_J3_2,W_c_J1_J3; // potentiel et dérivées 1 et 2
160
161
162 // codage des METHODES VIRTUELLES protegees:
163 // calcul des contraintes a t+dt
164 void Calcul_SigmaHH (TenseurHH& sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl,
165 TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H,TenseurBB & epsBB_,
166 TenseurBB & delta_epsBB,
167 TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_,
168 double& jacobien_0,double& jacobien,TenseurHH & sigHH
169 ,EnergieMeca & energ,const EnergieMeca & energ_t,double&
module_compressibilite,double& module_cisaillement
170 ,const Met_abstraite::Expli_t_tdt& ex);
171
172 // calcul des variations des contraintes a t+dt
173 void Calcul_DsigmaHH_tdt (TenseurHH& sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
174 ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t,
175 BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt,
176 TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB,
177 TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt,
178 Tableau <TenseurBB *>& d_gijBB_tdt,
179 Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien,
180 Vecteur& d_jacobien_tdt,TenseurHH & sigHH,Tableau <TenseurHH *>& d_sigHH
181 ,EnergieMeca & energ,const EnergieMeca & energ_t,double&
module_compressibilite,double& module_cisaillement
182 ,const Met_abstraite::Impli& ex);
183
184 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
185 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
186 // le tenseur de déformation et son incrémentsont également en orthonormees
187 // si = false: les bases transmises sont utilisées
188 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
189 virtual void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB

```

```

190         ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
191         ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
192         ,EnergieMeca & energ,const EnergieMeca & energ_t,double&
module_compressibilite,double& module_cisaillement
193         ,const Met_abstraite::Umat_cont& ex) ;
194
195 // calcul de grandeurs de travail aux points d'intégration via la def
196 // fonction surchargée dans les classes dérivée si besoin est
197 virtual void CalculGrandeurTravail
198     (const PtIntegMecaInterne&
199     ,const Deformation & ,Enum_dure,const ThermoDonnee&
200     ,const Met_abstraite::Impli* ex_impli
201     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
202     ,const Met_abstraite::Umat_cont* ex_umat
203     ,const List_io<Ddl_etendu>* exclure_dd_etend
204     ,const List_io<const TypeQuelconque *>* exclure_Q) {};
205
206 private:
207 // calcul du potentiel et de ses dérivées premières / aux invariants J_r
208 void Potentiel_et_var(double & module_compressibilite);
209 // calcul du potentiel et de ses dérivées premières et secondes / aux invariants J_r
210 void Potentiel_et_var2(double & module_compressibilite);
211 // définition de la courbe représentant l'évolution de l'énergie en fonction de J1
212 void Calcul_courbeevolW_J1();
213 // calcul de la dérivée numérique de la contrainte
214 void Cal_dsigma_deps_num (const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
215     ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
216     ,const double& jacobien_0,const double& jacobien
217     ,Tenseur3HHHH& dSigdepsHHHH);
218 // calcul de la contrainte avec le minimum de variable de passage, utilisé pour le numérique
219 void Cal_sigma_pour_num(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
220     ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
221     ,const double& jacobien_0,const double& jacobien,TenseurHH & sigHH_);
222
223 // idem avec la variation
224 void Cal_sigmaEtDer_pour_num(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
225     ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
226     ,const double& jacobien_0,const double& jacobien
227     ,TenseurHH & sigHH_,Tenseur3HHHH& dSigdepsHHHH);
228 /// @} // end of group
229
230
231 #endif
232
233
234

```

## 7.32 Hyper1.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      1/10/98
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *

```

```

36 *   PROJET:      Herezh++                               *
37 *                                                     $ *
38 *****
39 *   BUT: Concernant le potentiel hyperélastique nb 1, sans phase. *
40 *                                                     $ *
41 *   ////////////////////////////////////////////////// *
42 *   *
43 *   VERIFICATION:                                       *
44 *   ! date !   auteur !           but                 ! *
45 *   -----
46 *   !           !           !                         ! *
47 *                                                     $ *
48 *   ////////////////////////////////////////////////// *
49 *   MODIFICATIONS:                                       *
50 *   ! date !   auteur !           but                 ! *
51 *   -----
52 *                                                     $ *
53 *****/
54 #ifndef HYPER1_H
55 #define HYPER1_H
56
57 #include "Hyper3D.h"
58
59
60 /// @addtogroup Les_lois_hyperelastiques
61 /// @{
62 ///
63
64
65 class Hyper1 :
66 public Hyper3D
67 {
68 public :
69     // VARIABLES PUBLIQUES :
70
71 // CONSTRUCTEURS :
72 // Constructeur par défaut
73 Hyper1 ();
74 // Constructeur de copie
75 Hyper1 (const Hyper1& loi) ;
76 // DESTRUCTEUR :
77 ~Hyper1 ()
78     {};
79
80 // Lecture des donnees de la classe sur fichier
81 void LectureDonneesParticulieres (Utillecture * );
82
83 // affichage de la loi
84 void Affiche();
85 // test si la loi est complete
86 // = 1 tout est ok, =0 loi incomplete
87 int TestComplet();
88
89
90
91 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
92 // méthodes virtuelles de HyperD
93 // pour les alpha on utilise des formules particulieres plus simples que ceux de la thèse à D. Favier
94
95 // METHODES PROTEGEES :
96 // calcul des coefficients alpha dans le cas sans la phase
97 inline void Alpha(double& E,double& EV,double& EQeps,
98     double & Ieps,double & V,double& Qeps,
99     double & alpha_0,double & alpha_1,double & alpha_2);
100
101
102 // calcul des coefficients alpha dans le cas sans la phase et de leurs variations
103 inline void Alpha_var(double& E,Tableau <double>& dE,double& EV,Tableau <double>& dEV,
104     double& EQeps,Tableau<double> & dEQeps,
105     double& EVV, Tableau<double> & dEVV, double& EQQ, Tableau<double> & dEQQ,
106     double& EVQ, Tableau<double> & dEVQ,
107     double & Ieps,Tableau<double> & dIeps,
108     double & V,Tableau<double> & dV,double& Qeps,Tableau<double> & dQeps,
109     double & alpha_0,Tableau<double> & dalpha_0,
110     double & alpha_1,Tableau<double> & dalpha_1,
111     double & alpha_2,Tableau<double> & dalpha_2);
112
113
114 // calcul du potentiel et de ses dérivées non compris la phase
115 void Potentiel(double & Ieps,double & V,double& Qeps,
116     double& E,double& EV,double& EQeps) ;
117 // calcul du potentiel et de ses dérivées avec la phase
118 void PotentielPhase(double & Ieps,double & V,double& Qeps,
119     double& cos3phi,double & sin3phi,
120     double& E,double& EV,double& EQeps,double& EPhi) ;
121 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux ddl

```



```

122 void Potentiel_et_var(double & Ieps,Tableau<double> & dIeps,
123     double & V,Tableau<double> & dV,double& Qeps,
124     Tableau<double> & dQeps,
125     double& E,Tableau <double>& dE,double& EV,Tableau <double>& dEV,
126     double& EQeps,Tableau <double>& dEQeps,double& EVV,Tableau <double>& dEVV,
127     double& EQQ,Tableau <double>& dEQQ,double& EVQ,Tableau <double>& dEVQ ) ;
128 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux ddl
129 void PotentielPhase_et_var(double & Ieps,Tableau<double> & dIeps,
130     double & V,Tableau<double> & dV,double& Qeps,
131     Tableau<double> & dQeps,double& cos3phi,Tableau<double> & dcos3phi,
132     double & sin3phi,Tableau<double> & dsin3phi,
133     double& E,Tableau <double>& dE,double& EV,Tableau <double>& dEV,
134     double& EQeps,Tableau <double>& dEQeps,
135     double& EPhi,Tableau <double>& dEPhi,double& EVV,Tableau <double>& dEVV,
136     double& EQQ,Tableau <double>& dEQQ,double& EVQ,Tableau <double>& dEVQ ,
137     double& Ehiphi,Tableau <double>& dEhiphi,double& EQphi,Tableau <double>& dEQphi,
138     double& EVphi,Tableau <double>& dEVphi ) ;
139
140 // VARIABLES PROTEGEES :
141 double K; // module de dilatation
142 double Qor; // seuil
143 double mur; // pente à l'origine
144 double mu_inf; // pente à l'infini
145
146 // CONSTRUCTEURS :
147
148 // DESTRUCTEUR :
149
150 // METHODES PROTEGEES :
151
152 };
153 /// @} // end of group
154
155 #endif

```

## 7.33 Hyper10.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           1/10/98
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 *****/
39 *   BUT: Concernant le potentiel hyperélastique nb 10 .
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date ! auteur ! but
46 *   -----
47 *

```

```

48 *      *
49 *      MODIFICATIONS:
50 *      ! date ! auteur ! but !
51 *      -----
52 *      *
53 *      *
54 #ifndef HYPER10_H
55 #define HYPER10_H
56
57 #include "Hyper3D.h"
58
59
60 /// @addtogroup Les_lois_hyperelastiques
61 /// @{
62 ///
63
64
65 class Hyper10 :
66 public Hyper3D
67 {
68 public :
69 // VARIABLES PUBLIQUES :
70
71 // CONSTRUCTEURS :
72 // Constructeur par défaut
73 Hyper10 ();
74 // Constructeur de copie
75 Hyper10 (const Hyper10& loi) ;
76 // DESTRUCTEUR :
77 ~Hyper10 ()
78 {};
79
80 // Lecture des donnees de la classe sur fichier
81 void LectureDonneesParticulieres (Utillecture * );
82
83 // affichage de la loi
84 void Affiche();
85 // test si la loi est complete
86 // = 1 tout est ok, =0 loi incomplete
87 int TestCompleter();
88
89
90
91 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
92 // méthodes virtuelles de HyperD
93
94
95 // calcul du potentiel et de ses dérivées non compris la phase
96 void Potentiel(double & Ieps,double & V,double& Qeps,
97 double& E,double& EV,double& EQeps) ;
98 // calcul du potentiel et de ses dérivées avec la phase
99 void PotentielPhase(double & Ieps,double & V,double& Qeps,
100 double& cos3phi,double & sin3phi,
101 double& E,double& EV,double& EQeps,double& EPhi) ;
102 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux ddl
103 void Potentiel_et_var(double & Ieps,Tableau<double> & dIeps,
104 double & V,Tableau<double> & dV,double& Qeps,
105 Tableau<double> & dQeps,
106 double& E,Tableau <double>& dE,double& EV,Tableau <double>& dEV,
107 double& EQeps,Tableau <double>& dEQeps,double& EVV,Tableau <double>& dEVV,
108 double& EQQ,Tableau <double>& dEQQ,double& EVQ,Tableau <double>& dEVQ ) ;
109 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux ddl
110 void PotentielPhase_et_var(double & Ieps,Tableau<double> & dIeps,
111 double & V,Tableau<double> & dV,double& Qeps,
112 Tableau<double> & dQeps,double& cos3phi,Tableau<double> & dcos3phi,
113 double & sin3phi,Tableau<double> & dsin3phi,
114 double& E,Tableau <double>& dE,double& EV,Tableau <double>& dEV,
115 double& EQeps,Tableau <double>& dEQeps,
116 double& EPhi,Tableau <double>& dEPhi,double& EVV,Tableau <double>& dEVV,
117 double& EQQ,Tableau <double>& dEQQ,double& EVQ,Tableau <double>& dEVQ ,
118 double& Ehiphi,Tableau <double>& dEhiphi,double& EQphi,Tableau <double>& dEQphi,
119 double& EVphi,Tableau <double>& dEVphi );
120
121 // VARIABLES PROTEGEES :
122 double K; // module de dilatation
123 double Qor; // seuil
124 double mur; // pente à l'origine
125 double mu_inf; // pente à l'infini
126
127 // CONSTRUCTEURS :
128
129 // DESTRUCTEUR :
130
131 // METHODES PROTEGEES :
132
133 };
134 /// @} // end of group

```

```
135
136 #endif
```

## 7.34 Hyper3D.h

```
1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           1/10/98
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 *   ****
39 *   BUT: Classe générale concernant les hyperélastiques grenoblois.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !           !
47 *   *****
48 *
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *****/
54 #ifndef HYPER3D_H
55 #define HYPER3D_H
56
57 #include "HyperD.h"
58
59 // #include < >
60 // #include " "
61
62 /// @addtogroup Les_lois_hyperelastiques
63 /// @{
64 ///
65
66
67 class Hyper3D :
68 //++ public HyperD<class Tenseur3HH,class Tenseur3BB,class Tenseur3BH,class Tenseur3HB>
69 public HyperD
70 {
71 public :
72     // VARIABLES PUBLIQUES :
73
74     // CONSTRUCTEURS :
75     // Constructeur par défaut
76     Hyper3D ();
77     // Constructeur utile si l'identificateur du nom de la loi
78     // de comportement et le paramètre phase sont connus
79     Hyper3D (Enum_comp id_compor,Enum_categorie_loi_comp categorie_comp, bool avec_ph) ;
```

```

80 // Constructeur utile si l'identificateur du nom de la loi
81 // de comportement et le paramètre phase sont connus
82 Hyper3D (char* nom, Enum_categorie_loi_comp categorie_comp, bool avec_ph) ;
83 // Constructeur de copie
84 Hyper3D (const Hyper3D& loi) ;
85 // DESTRUCTEUR :
86 ~Hyper3D ()
87     {};
88
89 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
90 // méthodes virtuelles de HyperD
91
92
93 // Calcul des invariants et, de epsBH,
94 // retour de IdGBH qui pointe sur le bon tenseur
95 //++ TensBH * Invariants (TenseurBB& epsBB_t, TenseurBB& gijBB_t,
96 TenseurBH * Invariants (const TenseurBB& epsBB_t, const TenseurBB& gijBB_t,
97     const TenseurHH & gijHH_t, const double& jacobien_0, const double& jacobien_t,
98 //++
99     Invariant & invariant, TensBH & epsBH);
100 // calcul des invariants et de leurs variations, de epsBH,
101 // et de sa variation, puis retour de IdGBH qui pointe sur le bon tenseur
102 //++ TensBH * Invariants_et_var (TenseurBB& epsBB_tdt,
103 TenseurBH * Invariants_et_var (const TenseurBB& epsBB_tdt,
104     const TenseurBB& gijBB_tdt, const Tableau <TenseurBB *>& d_gijBB_tdt,
105     const TenseurHH & gijHH_tdt, const Tableau <TenseurHH *>& d_gijHH_tdt,
106     const double& jacobien_0, const double& jacobien_tdt, const Vecteur& d_jacobien_tdt,
107     InvariantVarDdl & invariantVarDdl,
108 //++
109     TensBH & epsBH_tdt, Tableau<TensBH> & depsBH_tdt);
110
111 // calcul des invariants et de leurs variations par rapport aux déformations
112 // et de sa variation, puis retour de IdGBH qui pointe sur le bon tenseur
113 TenseurBH * Invariants_et_varEps (const TenseurBB& epsBB_tdt,
114     const TenseurBB& gijBB_tdt, const TenseurHH & gijHH_tdt,
115     const double& jacobien_0, const double& jacobien_tdt,
116     InvariantVarEps & invariantVarEps, TenseurBH & epsBH_tdt);
117
118 // calcul du potentiel et de ses dérivées non compris la phase
119 PotensansPhaseSansVar Potentiel
120     (const Invariant & invariant, const double& jacobien0);
121 // calcul du potentiel et de ses dérivées avec la phase
122 PotensansPhaseSansVar PotentielPhase
123     (const Invariant & invariant, const double& jacobien0);
124 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux invariants
125 PotensansPhaseAvecVar Potentiel_et_var
126     (const Invariant & invariant, const double& jacobien0);
127 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux invariants
128 PotensansPhaseAvecVar PotentielPhase_et_var
129     (const Invariant & invariant, const double& jacobien0);
130 // calcul du potentiel sans phase et dérivées avec ses variations par rapport à epsBB
131 PotensansPhaseAvecVar Potentiel_et_varEps
132     (const Invariant & invariant, const double& jacobien0);
133
134
135 protected :
136
137 // classe permettant le passage protégé de grandeur
138 // Invariants Qeps et Cosphi seulement
139 class Invariant0QepsCosphi
140 { public :
141     Invariant0QepsCosphi() : Qeps(0.), cos3phi(0.) {};
142     Invariant0QepsCosphi(const double& Qe, const double& cos3p) : Qeps(Qe), cos3phi(cos3p) {};
143     double Qeps, cos3phi;
144 };
145 // Invariants Qeps ainsi que sa variation première
146 class InvariantQeps
147 { public :
148     InvariantQeps() : Qeps(0.), dQepsdbIIB(0.) {};
149     double Qeps;
150     double dQepsdbIIB ; // variation de Qeps par rapport à bIIB
151 };
152 // Invariants Qeps et Cosphi ainsi que leur variations premières
153 class InvariantQepsCosphi
154 { public :
155     InvariantQepsCosphi() : Qeps(0.), cos3phi(0.), dQepsdbIIB(0.)
156         , dcos3phidV(0.), dcos3phidIeps(0.), dcos3phidbIIB(0.) {};
157     double Qeps, cos3phi;
158     double dQepsdbIIB ; // variation de Qeps par rapport à bIIB
159     double dcos3phidV; // variation de Qeps par rapport à V
160     double dcos3phidIeps; // variation de cos3phi par rapport à Ieps
161     double dcos3phidbIIB; // variation de cos3phi par rapport à bIIB
162 };
163 // Invariants Qeps ainsi que ses variations premières et secondes
164 class Invariant2Qeps
165 { public :
166     Invariant2Qeps() : Qeps(0.), dQepsdbIIB(0.), dQepsdbIIB2(0.) {};

```

```

167     Invariant2Qeps(const double & Q, const double & dQdbIIb, const double & dQdbIIb2):
168         Qeps(Q), dQepsdbIIb(dQdbIIb), dQepsdbIIb2(dQdbIIb2)    {};
169     double Qeps;
170     double dQepsdbIIb ; // variation de Qeps par rapport à bIIb
171     double dQepsdbIIb2 ; // variation seconde de Qeps par rapport à bIIb
172     };
173 // Invariants Qeps et Cosphi ainsi que leur variations premières
174 // et secondes
175 class Invariant2QepsCosphi
176 { public :
177     Invariant2QepsCosphi() : Qeps(0.), cos3phi(0.), dQepsdbIIb(0.)
178         , dcos3phidV(0.), dcos3phidIeps(0.), dcos3phidbIIb(0.), dQepsdbIIb2(0.)
179         , dcos3phidV2(0.), dcos3phidIeps2(0.), dcos3phidbIIb2(0.), dcos3phidVdbIIb(0.)
180         , dcos3phidIepsdbIIb(0.) {};
181     double Qeps, cos3phi;
182     double dQepsdbIIb ; // variation de Qeps par rapport à bIIb
183     double dcos3phidV; // variation de cos3phi par rapport à V
184     double dcos3phidIeps; // variation de cos3phi par rapport à Ieps
185     double dcos3phidbIIb; // variation de cos3phi par rapport à Ieps
186     double dQepsdbIIb2 ; // variation seconde de Qeps par rapport à bIIb
187     double dcos3phidV2; // variation seconde de cos3phi par rapport à V
188     double dcos3phidIeps2; // variation seconde de cos3phi par rapport à Ieps
189     double dcos3phidbIIb2; // variation seconde de cos3phi par rapport à Ieps
190     double dcos3phidVdbIIb; // variation seconde de cos3phi par rapport à V et bIIb
191     double dcos3phidIepsdbIIb; // variation seconde de cos3phi par rapport à Ieps et bIIb
192     };
193 // class pour le potentiel et seulement sa variation suivant V
194 class PoGrenoble_V
195 { public :
196     PoGrenoble_V() : E(0.), EV(0.), Ks(0) {};
197     double E; // valeur du potentiel
198     double EV; // variation du potentiel par rapport à V
199     double Ks; // module de compressibilité sécant par rapport à log(V)
200     };
201 // class pour le potentiel et seulement les variations suivant V et Q
202 class PoGrenobleSansPhaseSansVar
203 { public :
204     PoGrenobleSansPhaseSansVar() : E(0.), EV(0.), EQ(0.), Ks(0) {};
205     double E; // valeur du potentiel
206     double EV; // variation du potentiel par rapport à V
207     double EQ; // variation du potentiel par rapport à Q
208     double Ks; // module de compressibilité sécant par rapport à log(V)
209     };
210 // class pour le potentiel et seulement ses variations suivant V
211 class PoGrenoble_VV
212 { public :
213     PoGrenoble_VV() : E(0.), EV(0.), EVV(0.), Ks(0) {};
214     double E; // valeur du potentiel
215     double EV; // variation du potentiel par rapport à V
216     double EVV; // variation seconde par rapport à V
217     double Ks; // module de compressibilité sécant par rapport à log(V)
218     };
219 class PoGrenobleSansPhaseAvecVar
220 { public :
221     PoGrenobleSansPhaseAvecVar() : E(0.), EV(0.), EQ(0.), EVV(0.), EQV(0.), EQQ(0.), Ks(0) {};
222     double E; // valeur du potentiel
223     double EV; // variation du potentiel par rapport à V
224     double EQ; // variation du potentiel par rapport à Q
225     double EVV; // variation seconde par rapport à V
226     double EQV; // variation seconde par rapport à Q et V
227     double EQQ; // variation seconde par rapport à Q
228     double Ks; // module de compressibilité sécant par rapport à log(V)
229     };
230 class PoGrenobleAvecPhaseSansVar
231 { public :
232     PoGrenobleAvecPhaseSansVar() : E(0.), EV(0.), EQ(0.), Ecos3phi(0.), Ks(0) {};
233     double E; // valeur du potentiel
234     double EV; // variation du potentiel par rapport à V
235     double EQ; // variation du potentiel par rapport à Q
236     double Ecos3phi; // variation du potentiel par rapport à cos3phi
237     double Ks; // module de compressibilité sécant par rapport à log(V)
238     };
239 class PoGrenobleAvecPhaseAvecVar
240 { public :
241     PoGrenobleAvecPhaseAvecVar() : E(0.), EV(0.), EQ(0.), Ecos3phi(0.)
242         , EVV(0.), EQV(0.), EQQ(0.), EVcos3phi(0.), EQcos3phi(0.), Ecos3phi2(0.), Ks(0) {};
243     double E; // valeur du potentiel
244     double EV; // variation du potentiel par rapport à V
245     double EQ; // variation du potentiel par rapport à Q
246     double Ecos3phi; // variation du potentiel par rapport à cos3phi
247     double EVV; // variation seconde par rapport à V
248     double EQV; // variation seconde par rapport à Q et V
249     double EQQ; // variation seconde par rapport à Q
250     double EVcos3phi; // variation seconde par rapport à V cos3phi
251     double EQcos3phi; // variation seconde par rapport à Q cos3phi
252     double Ecos3phi2; // variation seconde par rapport à cos3phi
253     double Ks; // module de compressibilité sécant par rapport à log(V)

```

```

254     };
255
256
257
258     // METHODES PROTEGEES :
259
260
261     // calcul de l'invariant Qeps seul valable même si Qeps est nul
262     double Invariant0Qeps(const Invariant & invariant);
263     // calcul des invariants Qeps,cos3phi seuls (Qeps doit être non nul)
264     Invariant0QepsCosphi Invariant0Specif(const Invariant & invariant);
265     // calcul des invariants Qeps,cos3phi en fonction de V, bIIb Ieps
266     // on utilise pas l'invariant bIIIb,
267     Invariant0QepsCosphi InvariantDe_V_bIIb_Ieps(const Invariant & invariant);
268     // calcul de l'invariants Qeps ainsi que sa variation par rapport
269     // à bIIb (Qeps doit être non nul)
270     InvariantQeps InvariantSpecifSansPhase(const Invariant & invariant);
271     // calcul des invariants Qeps,cos3phi, ainsi que leur variation par rapport
272     // aux trois invariants Ieps, V et bIIb (Qeps doit être non nul)
273     InvariantQepsCosphi InvariantSpecif(const Invariant & invariant);
274     // calcul de l'invariants Qeps ainsi que ses variations première et seconde
275     // par rapport à bIIb (Qeps doit être non nul)
276     Invariant2Qeps Invariant2SpecifSansPhase(const Invariant & invariantVarDdl);
277     // calcul des invariants Qeps,cos3phi, ainsi que leur variation première et seconde
278     // par rapport aux trois invariants Ieps, V et bIIb (Qeps doit être non nul)
279     Invariant2QepsCosphi Invariant2Specif(const Invariant & invariantVarDdl);
280
281     /// les fonctions potentielles sont défini dans les classes dérivées
282
283     // calcul du potentiel tout seul sans la phase car Qeps est nul
284     // ou très proche de 0
285     virtual double PoGrenoble
286         (const double & Qeps,const Invariant & invariant) = 0;
287     // calcul du potentiel tout seul avec la phase donc dans le cas où Qeps est non nul
288     virtual double PoGrenoble
289         (const Invariant0QepsCosphi & inv,const Invariant & invariant) = 0;
290     // calcul du potentiel tout seul sans la phase car Qeps est nul
291     // ou très proche de 0, et de sa variation suivant V uniquement
292     virtual PoGrenoble_V PoGrenoble_et_V
293         (const double & Qeps,const Invariant & invariant) = 0;
294     // calcul du potentiel et de sa variation suivant V uniquement
295     virtual PoGrenoble_V PoGrenoble_et_V
296         (const Invariant0QepsCosphi & inv,const Invariant & invariant) = 0;
297     // calcul du potentiel tout seul sans la phase car Qeps est nul
298     // ou très proche de 0, et de ses variations première et seconde suivant V uniquement
299     virtual PoGrenoble_VV PoGrenoble_et_VV
300         (const double & Qeps,const Invariant & invariant) = 0;
301     // calcul du potentiel et de sa variation première et seconde suivant V uniquement
302     virtual PoGrenoble_VV PoGrenoble_et_VV
303         (const Invariant0QepsCosphi & inv,const Invariant & invariant) = 0;
304     // calcul du potentiel et de ses dérivées non compris la phase
305     virtual PoGrenobleSansPhaseSansVar PoGrenoble
306         (const InvariantQeps & inv,const Invariant & invariant) = 0;
307     // calcul du potentiel et de ses dérivées avec la phase
308     virtual PoGrenobleAvecPhaseSansVar PoGrenoblePhase
309         (const InvariantQepsCosphi & inv,const Invariant & invariant) = 0;
310     // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux invariants
311     virtual PoGrenobleSansPhaseAvecVar PoGrenoble_et_var
312         (const Invariant2Qeps & inv,const Invariant & invariantVarDdl) = 0;
313     // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux invariants
314     virtual PoGrenobleAvecPhaseAvecVar PoGrenoblePhase_et_var
315         (const Invariant2QepsCosphi & inv,const Invariant & invariantVarDdl) = 0;
316
317     ///===== fonctions pour la vérification et la mise au point =====
318
319     // vérif des dérivées du potentiels par rapport aux invariants, ceci par différences finies
320     // cas sans la phase
321     void Verif_Potentiel_et_var(const InvariantVarDdl& inv,const double& jacobien0
322         ,const PotenSansPhaseAvecVar& potret );
323     // vérif des dérivées du potentiels par rapport aux invariants, ceci par différences finies
324     // cas avec la phase
325     void Verif_Potentiel_et_var(const InvariantVarDdl& inv,const double& jacobien0
326         ,const PotenAvecPhaseAvecVar& potret );
327     static int indic_Verif_Potentiel_et_var; // indicateur utilisé par Verif_Potentiel_et_var
328
329     static double limite_inf_Qeps; // limite en dessous de laquelle on néglige la phase
330     static double limite_inf_bIIb; // limite en dessous de laquelle on fait une ope spécial
331 };
332 /// @} // end of group
333
334 #endif

```

## 7.35 Hyper3D\_save.h

```

1 /*****
2 *      UNIVERSITE DE BRETAGNE SUD  --- I.U.P/I.U.T. DE LORIENT  *
3 *****/
4 *      LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX  *
5 *      Tel 97.80.80.60  *
6 *      Centre de Genie Industriel  56520 GUIDEL-PLAGES  *
7 *****/
8 *      DATE:      1/10/98  *
9 *      $  *
10 *      AUTEUR:    G RIO  *
11 *      $  *
12 *      PROJET:    Herezh++  *
13 *      $  *
14 *****/
15 *      BUT: Classe générale concernant les hyperélastiques grenoblois.  *
16 *      $  *
17 *      *****  *
18 *      VERIFICATION:  *
19 *      *  *
20 *      ! date ! auteur ! but !  *
21 *      -----  *
22 *      ! ! ! !  *
23 *      $  *
24 *      *****  *
25 *      MODIFICATIONS:  *
26 *      ! date ! auteur ! but !  *
27 *      -----  *
28 *      $  *
29 *****/
30 #ifndef HYPER3D_H
31 #define HYPER3D_H
32
33 #include "HyperD.h"
34
35 // #include < >
36 // #include " "
37
38 class Hyper3D :
39 public HyperD<class Tenseur3HH,class Tenseur3BB,class Tenseur3BH,class Tenseur3HB>
40 {
41 public :
42 // VARIABLES PUBLIQUES :
43
44 // CONSTRUCTEURS :
45 // Constructeur par défaut
46 Hyper3D ();
47 // Constructeur utile si l'identificateur du nom de la loi
48 // de comportement et le paramètre phase sont connus
49 Hyper3D (Enum_comp id_compor, bool avec_ph) ;
50 // Constructeur utile si l'identificateur du nom de la loi
51 // de comportement et le paramètre phase sont connus
52 Hyper3D (char* nom,bool avec_ph) ;
53 // Constructeur de copie
54 Hyper3D (const Hyper3D& loi) ;
55 // DESTRUCTEUR :
56 ~Hyper3D ()
57 {};
58
59
60
61 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
62 // méthodes virtuelles de HyperD
63
64 // Calcul des deux premiers invariants et de la trace, de epsBH,
65 // retour de IdGBH qui pointe sur le bon tenseur
66 Tenseur3BH* Invariants (TenseurBB& epsBB_t,TenseurBB& gijBB_t,
67 TenseurHH & gijHH_t, double& jacobien_0,double& jacobien_t,
68 double & Ieps, double & V,double& Qeps,Tenseur3BH & epsBH) ;
69 // Calcul des trois invariants et de la trace, de epsBH,
70 // retour de IdGBH qui pointe sur le bon tenseur
71 Tenseur3BH* InvariantsPhase (TenseurBB& epsBB_t,TenseurBB& gijBB_t,
72 TenseurHH & gijHH_t, double& jacobien_0,double& jacobien_t,
73 double & Ieps,double & V,double& Qeps,double& cos3phi,double & sin3phi,
74 Tenseur3BH & epsBH) ;
75 // calcul des deux premiers invariants + trace et de leurs variations, de epsBH,
76 // et de sa variation, puis retour de IdGBH qui pointe sur le bon tenseur
77 Tenseur3BH* Invariants_et_var (TenseurBB& epsBB_tdt,
78 TenseurBB& gijBB_tdt,Tableau <TenseurBB *>& d_gijBB_tdt,
79 TenseurHH & gijHH_tdt,Tableau <TenseurHH *>& d_gijHH_tdt,
80 double& jacobien_0,double& jacobien_tdt,Vecteur& d_jacobien_tdt,
81 double & Ieps,Tableau<double> & dIeps,double & V,Tableau<double> & dV,
82 double& Qeps,Tableau<double> & dQeps,
83 Tenseur3BH & epsBH,Tableau<Tenseur3BH> & depsBH) ;
84 // calcul des trois invariants et de leurs variations, de epsBH,

```

```

85 // et de sa variation, puis retour de IdGBH qui pointe sur le bon tenseur
86 Tenseur3BH* InvariantsPhase_et_var
87 (TenseurBB& epsBB_tdt,
88  TenseurBB& gijBB_tdt,Tableau <TenseurBB *>& d_gijBB_tdt,
89  TenseurHH & gijHH_tdt,Tableau <TenseurHH *>& d_gijHH_tdt,
90  double& jacobien_0,double& jacobien_tdt,Vecteur& d_jacobien_tdt,
91  double & Ieps,Tableau<double> & dIeps,
92  double & V,Tableau<double> & dV,double& Qeps,Tableau<double> & dQeps,
93  double& cos3phi,Tableau<double> & dcos3phi,
94  double & sin3phi,Tableau<double> & dsin3phi,
95  Tenseur3BH & epsBH,Tableau<Tenseur3BH> & depsBH) ;
96
97
98 // VARIABLES PROTEGEES :
99
100 // CONSTRUCTEURS :
101
102 // DESTRUCTEUR :
103
104 // METHODES PROTEGEES :
105
106 };
107
108 #endif

```

## 7.36 Hyper3DN.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      1/10/98
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++
37 *
38 *   *****
39 *   BUT: Classe générale concernant les hyperélastiques utilisant
40 *   comme invariants : V,Ieps,bIib.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !           but
47 *   -----
48 *   !           !           !
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date !   auteur !           but
52 *   -----
53 *
54 *   *****
55 #ifndef HYPER3DN_H
56 #define HYPER3DN_H
57

```



```

58 #include "HyperDN.h"
59 #include "TypeConsTens.h"
60
61
62 /// @addtogroup Les_lois_hyperelastiques
63 /// @{
64 ///
65
66 class Hyper3DN :
67 public HyperDN<class Tenseur3HH,class Tenseur3BB,class Tenseur3BH,class Tenseur3HB,
68 class Tenseur_ns3HH,class Tenseur_ns3BB>
69 {
70 public :
71 // VARIABLES PUBLIQUES :
72
73 // CONSTRUCTEURS :
74 // Constructeur par défaut
75 Hyper3DN ();
76 // Constructeur utile si l'identificateur du nom de la loi
77 // de comportement est connu
78 Hyper3DN (Enum_comp id_compor,Enum_categorie_loi_comp categorie_comp) ;
79 // Constructeur utile si l'identificateur du nom de la loi
80 // de comportement est connu
81 Hyper3DN (char* nom,Enum_categorie_loi_comp categorie_comp) ;
82 // Constructeur de copie
83 Hyper3DN (const Hyper3DN& loi) ;
84 // DESTRUCTEUR :
85 ~Hyper3DN ()
86 {};
87
88
89
90 // METHODES internes spécifiques à l'hyperélasticité isotrope décollant de
91 // méthodes virtuelles de HyperDN
92
93 // Calcul des trois invariants et de epsBH,
94 // retour de IdGBH qui pointe sur le bon tenseur
95 Tenseur3BH* Invariants (TenseurBB& epsBB_t,TenseurBB& gijBB_t,
96 TenseurHH & gijHH_t, double& jacobien_0,double& jacobien_t,
97 double & Ieps, double & V,double& bIIb,Tenseur3BH & epsBH) ;
98 // calcul des trois invariants et de leurs variations, de epsBH,
99 // et de sa variation, puis retour de IdGBH qui pointe sur le bon tenseur
100 Tenseur3BH* Invariants_et_var (TenseurBB& epsBB_tdt,
101 TenseurBB& gijBB_tdt,Tableau <TenseurBB *>& d_gijBB_tdt,
102 TenseurHH & gijHH_tdt,Tableau <TenseurHH *>& d_gijHH_tdt,
103 double& jacobien_0,double& jacobien_tdt,Vecteur& d_jacobien_tdt,
104 double & Ieps,Tableau<double> & dIeps,double & V,Tableau<double> & dV,
105 double& bIIb,Tableau<double> & dbIIb,
106 Tenseur3BH & epsBH,Tableau<Tenseur3BH> & depsBH) ;
107
108 // récupération de la variation relative d'épaisseur calculée: h/h0
109 // cette variation n'est utile que pour des lois en contraintes planes
110 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
111 // - pour les lois 2D def planes: retour de 0
112 // les infos nécessaires à la récupération , sont stockées dans saveResul
113 // qui est le conteneur spécifique au point où a été calculé la loi
114 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
115
116
117 // VARIABLES PROTEGEES :
118
119 // CONSTRUCTEURS :
120
121 // DESTRUCTEUR :
122
123 // METHODES PROTEGEES :
124
125 };
126 /// @} // end of group
127
128 #endif

```

## 7.37 Hyper\_externe\_W.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //

```

```

12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30 //
31 /*****
32 *   DATE:           06/03/2023
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 * *****/
39 *   BUT:   La classe Hyper_externe_W permet de calculer la contrainte*
40 *          et ses derivees pour une loi isotrope hyper élastique *
41 *          de type extension de la loi de Mooney Rivlin en 3D *
42 *          à l'aide de l'utilisation d'une fonction externe nD. *
43 *
44 *          ***** *
45 *
46 *   VERIFICATION:
47 *
48 *   ! date !   auteur !           but
49 *   -----
50 *   !           !           !
51 *
52 *          ***** *
53 *   MODIFICATIONS:
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *          *****/
58 #ifndef HYPER_EXTERNE_W_H
59 #define HYPER_EXTERNE_W_H
60
61
62
63
64 #include "Loi_comp_abstraite.h"
65 #include "CourbelD.h"
66 #include "MathUtil.h"
67 #include "Hyper_W_gene_3D.h"
68
69
70 /// @addtogroup Les_lois_hyperelastiques
71 /// @{
72 ///
73
74
75 class Hyper_externe_W : public Hyper_W_gene_3D
76 {
77
78     public :
79
80
81         // CONSTRUCTEURS :
82
83         // Constructeur par défaut
84         Hyper_externe_W ();
85
86
87         // Constructeur de copie
88         Hyper_externe_W (const Hyper_externe_W& loi) ;
89
90         // DESTRUCTEUR :
91
92         ~Hyper_externe_W ();
93
94
95 // initialise les donnees particulieres a l'elements
96 // de matiere traite ( c-a-dire au pt calcule)
97 // Il y a creation d'une instance de SaveResul particuliere
98 // a la loi concerne

```

```

99 // la SaveResul classe est remplie par les instances heritantes
100 // le pointeur de SaveResul est sauvegarde au niveau de l'element
101 // c'a-d que les info particulieres au point considere sont stocke
102 // au niveau de l'element et non de la loi.
103 class SaveResulHyper_externe_W: public SaveResulHyper_W_gene_3D
104 { public :
105
106     SaveResulHyper_externe_W(); // constructeur par defaut
107     SaveResulHyper_externe_W(int sortie_post); // avec init ou pas
108     SaveResulHyper_externe_W(const SaveResulHyper_externe_W& sav); // de copie
109     virtual ~SaveResulHyper_externe_W() ; // destructeur
110
111     // definition d'une nouvelle instance identique
112     // appelle du constructeur via new
113     SaveResul * Nevez_SaveResul() const{return (new SaveResulHyper_externe_W(*this));};
114
115     // affectation
116     virtual SaveResul & operator = ( const SaveResul & a);
117
118     //===== lecture écriture dans base info =====
119     // cas donne le niveau de la récupération
120     // = 1 : on récupère tout
121     // = 2 : on récupère uniquement les données variables (supposées comme telles)
122     void Lecture_base_info (ifstream& ent,const int cas);
123
124     // cas donne le niveau de sauvegarde
125     // = 1 : on sauvegarde tout
126     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
127     void Ecriture_base_info(ofstream& sort,const int cas);
128
129     // mise à jour des informations transitoires
130     void TdtversT();
131     void TversTdt();
132
133     // affichage à l'écran des infos
134     void Affiche() const;
135
136     //changement de base de toutes les grandeurs internes tensorielles stockées
137     // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gpB
138     // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
139     // gpH(i) = gamma(i,j) * gH(j)
140     virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) {};
141
142     // procedure permettant de completer éventuellement les données particulières
143     // de la loi stockées
144     // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
145     // completer est appelé apres sa creation avec les donnees du bloc transmis
146     // peut etre appeler plusieurs fois
147     virtual SaveResul* Complete_SaveResul(const BlocGen & bloc
148         , const Tableau <Coordonnee>& tab_coor
149         ,const Loi_comp_abstraite* loi) {return NULL;};
150
151
152     //-----
153     // données
154     //-----
155     // le potentiel total est stocké dans la classe mère: SaveResulHyper_W_gene_3D
156     double module_compressibilite,module_compressibilite_t;
157     // stockage éventuel des valeurs de la fct nD
158     Vecteur * W_poten,* W_poten_t;
159     // stockage éventuel des invariants
160     Vecteur * inv_B,* inv_B_t;
161     Vecteur * inv_J,* inv_J_t;
162
163 };
164
165 // def d'une instance de données spécifiques, et initialisation
166 SaveResul * New_et_Initialise()
167 { return (new SaveResulHyper_externe_W(this->sortie_post));};
168
169 // affichage des donnees particulieres a l'elements
170 // de matiere traite ( c-a-dire au pt calcule)
171 void AfficheDataSpecif(ofstream& ,SaveResul * a) const
172 { SaveResulHyper_externe_W& sav = *((SaveResulHyper_externe_W*) a);
173   sav.Affiche();
174 };
175
176     // Lecture des donnees de la classe sur fichier
177     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
178         ,LesFonctions_nD& lesFonctionsnD);
179
180     // affichage de la loi
181     void Affiche() const ;
182     // test si la loi est complete
183     // = 1 tout est ok, =0 loi incomplete
184     int TestComplet();
185
186     //----- lecture écriture de restart -----

```

```

186 // cas donne le niveau de la récupération
187 // = 1 : on récupère tout
188 // = 2 : on récupère uniquement les données variables (supposées comme telles)
189 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
190                                     ,LesFonctions_nD& lesFonctionsnD);
191 // cas donne le niveau de sauvegarde
192 // = 1 : on sauvegarde tout
193 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
194 void Ecriture_base_info_loi(ofstream& sort,const int cas);
195
196 // récupération des grandeurs particulière (hors ddl )
197 // correspondant à liTQ
198 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
199 void Grandeur_particuliere
200     (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal)
const;
201 // récupération de la liste de tous les grandeurs particulières
202 // ces grandeurs sont ajoutées à la liste passées en paramètres
203 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
204 void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
205
206 // calcul d'un module d'young équivalent à la loi, ceci pour un
207 // chargement nul
208 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
209
210 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
211 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
212 // >> en fait ici il s'agit du dernier module tangent calculé !!
213 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
saveResul) ;
214
215 // récupération de la variation relative d'épaisseur calculée: h/h0
216 // cette variation n'est utile que pour des lois en contraintes planes
217 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
218 // - pour les lois 2D def planes: retour de 0
219 // les infos nécessaires à la récupération , sont stockées dans saveResul
220 // qui est le conteneur spécifique au point où a été calculé la loi
221 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
222
223
224 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
225 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hyper_externe_W(*this)); };
226
227 // affichage et definition interactive des commandes particulières à chaque lois
228 void Info_commande_LoisDeComp(UtilLecture& lec);
229
230 // calcul de grandeurs de travail aux points d'intégration via la def et autres
231 // ici éventuellement permet de récupérer la compressibilité
232 virtual void CalculGrandeurTravail
233     (const PtIntegMecaInterne& ,const Deformation &
234     ,Enum_dure,const ThermoDonnee&
235     ,const Met_abstraite::Impli* ex_impli
236     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
237     ,const Met_abstraite::Umat_cont* ex_umat
238     ,const List_io<Ddl_etendu>* exclure_dd_etend
239     ,const List_io<const TypeQuelconque *>* exclure_Q
240     ) {};
241
242
243 protected :
244     // données de la loi
245     // la fonction potentielle
246     Fonction_nD* W_potentiel;
247
248     double W_d,W_v; // le potentiel: partie déviatorique, partie sphérique
249     double W_d_J1,W_d_J2; // dérivées premières du potentiel déviatoire par rapport aux J_1 et J_2
250     double W_d_J1_2,W_d_J1_J2,W_d_J2_2; // dérivées secondes
251     double W_v_J3,W_v_J3J3; // dérivées premières et seconde du potentiel volumique / J3
252
253     // variables de travail
254     List_io<TypeQuelconque> invar; // contient INVAR_B,INVAR_J
255     List_io<Ddl_enum_etendu> exclure_ddenum; // pour exclure le calcul des invariants
256     // qui est en fait calculé localement
257
258
259     // --- codage des METHODES VIRTUELLES protegees:
260 // calcul des contraintes a t+dt
261 // calcul des contraintes
262 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
263     ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
264     ,TenseurBB & delta_epsBB_
265     ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
266     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
267     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
268     ,const Met_abstraite::Expli_t_tdt& ex);

```

```

269
270 // calcul des contraintes et de ses variations a t+dt
271 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
272 , BaseB& giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
273 , BaseB& giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH& giH_tdt, Tableau <BaseH> & d_giH_tdt
274 , TenseurBB & epsBB_tdt, Tableau <TenseurBB *>& d_epsBB
275 , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
276 , Tableau <TenseurBB *>& d_gijBB_tdt
277 , Tableau <TenseurHH *>& d_gijHH_tdt, double& jacobien_0, double& jacobien
278 , Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *>& d_sigHH
279 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
module_cisaillement
280 , const Met_abstraite::Impli& ex);
281
282 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
283 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
284 // le tenseur de déformation et son incrémentsont également en orthonormees
285 // si = false: les bases transmises sont utilisées
286 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
287 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
288 , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
289 , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps
290 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
module_cisaillement
291 , const Met_abstraite::Umat_cont& ex) ; // = 0;
292
293 private:
294
295 // calcul du potentiel et de ses dérivées premières et secondes / aux invariants J_r
296 void Potentiel_et_var2(const Met_abstraite::Impli* ex_impli
297 , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
298 , const Met_abstraite::Umat_cont* ex_umat);
299 // calcul de la dérivée numérique de la contrainte
300 void Cal_dsigma_deps_num (const TenseurBB & gijBB_0, const TenseurHH & gijHH_0
301 , const TenseurBB & gijBB_tdt, const TenseurHH & gijHH_tdt
302 , const double& jacobien_0, const double& jacobien
303 , Tenseur3HHHH& dSigdepsHHHH
304 , const Met_abstraite::Impli& ex );
305 // calcul de la contrainte avec le minimum de variable de passage, utilisé pour le numérique
306 void Cal_sigma_pour_num(const TenseurBB & gijBB_0, const TenseurHH & gijHH_0
307 , const TenseurBB & gijBB_tdt, const TenseurHH & gijHH_tdt
308 , const double& jacobien_0, const double& jacobien, TenseurHH & sigHH_
309 , const Met_abstraite::Impli& ex);
310 // idem avec la variation
311 void Cal_sigmaEtDer_pour_num(const TenseurBB & gijBB_0, const TenseurHH & gijHH_0
312 , const TenseurBB & gijBB_tdt, const TenseurHH & gijHH_tdt
313 , const double& jacobien_0, const double& jacobien
314 , TenseurHH & sigHH_, Tenseur3HHHH& dSigdepsHHHH
315 , const Met_abstraite::Impli& ex);
316
317
318 };
319 /// @} // end of group
320
321
322 #endif
323
324
325

```

## 7.38 Hyper\_w1.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

```

```

24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      1/10/98
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++
37 *
38 * *****/
39 *   BUT: Concernant le potentiel hyperélastique nb 10 .
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 * *****/
54 #ifndef HYPER10_H
55 #define HYPER10_H
56
57 #include "Hyper3D.h"
58
59
60 /// @addtogroup Les_lois_hyperelastiques
61 /// @{
62 ///
63
64
65 class Hyper10 :
66 public Hyper3D
67 {
68 public :
69     // VARIABLES PUBLIQUES :
70
71     // CONSTRUCTEURS :
72     // Constructeur par défaut
73     Hyper10 ();
74     // Constructeur de copie
75     Hyper10 (const Hyper10& loi) ;
76     // DESTRUCTEUR :
77     ~Hyper10 ()
78     {};
79
80 // Lecture des donnees de la classe sur fichier
81 void LectureDonneesParticulieres (Utillecture * );
82
83 // affichage de la loi
84 void Affiche();
85 // test si la loi est complete
86 // = 1 tout est ok, =0 loi incomplete
87 int TestComplet();
88
89
90
91 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
92 // méthodes virtuelles de HyperD
93
94
95 // calcul du potentiel et de ses dérivées non compris la phase
96 void Potentiel(double & Ieps,double & V,double& Qeps,
97               double& E,double& EV,double& EQeps) ;
98 // calcul du potentiel et de ses dérivées avec la phase
99 void PotentielPhase(double & Ieps,double & V,double& Qeps,
100                    double& cos3phi,double & sin3phi,
101                    double& E,double& EV,double& EQeps,double& EPhi) ;
102 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux ddl
103 void Potentiel_et_var(double & Ieps,Tableau<double> & dIeps,
104                      double & V,Tableau<double> & dV,double& Qeps,
105                      Tableau<double> & dQeps,
106                      double& E,Tableau <double>& dE,double& EV,Tableau <double>& dEV,
107                      double& EQeps,Tableau <double>& dEQeps,double& EVV,Tableau <double>& dEVV,
108                      double& EQQ,Tableau <double>& dEQQ,double& EVQ,Tableau <double>& dEVQ ) ;
109 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux ddl

```

```

110 void PotentielPhase_et_var(double & Ieps,Tableau<double> & dIeps,
111     double & V,Tableau<double> & dV,double& Qeps,
112     Tableau<double> & dQeps,double& cos3phi,Tableau<double> & dcos3phi,
113     double & sin3phi,Tableau<double> & dsin3phi,
114     double& E,Tableau <double>& dE,double& EV,Tableau <double>& dEV,
115     double& EQeps,Tableau <double>& dEQeps,
116     double& EPhi,Tableau <double>& dEPhi,double& EVV,Tableau <double>& dEVV,
117     double& EQQ,Tableau <double>& dEQQ,double& EVQ,Tableau <double>& dEVQ ,
118     double& Ephiphi,Tableau <double>& dEphiphi,double& EQphi,Tableau <double>& dEQphi,
119     double& EVphi,Tableau <double>& dEVphi );
120
121 // VARIABLES PROTEGEES :
122 double K; // module de dilatation
123 double Qor; // seuil
124 double mur; // pente à l'origine
125 double mu_inf; // pente à l'infini
126
127 // CONSTRUCTEURS :
128
129 // DESTRUCTEUR :
130
131 // METHODES PROTEGEES :
132
133 };
134 /// @} // end of group
135
136 #endif

```

## 7.39 Hyper\_W\_gene\_3D.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:      19/6/2005
33 *
34 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:     Herezh++
37 *
38 *
39 *      BUT:        La classe Hyper_W_gene_3D permet de calculer
40 *                 les grandeurs générales nécessaires pour exploiter une
41 *                 loi hyper-élastique utilisant un potentiel W défini
42 *                 a partir des élongations.
43 *      Il s'agit d'une classe de travail.
44 *
45 *      *****
46 *
47 *      VERIFICATION:
48 *      ! date !   auteur !           but
49 *      -----
50 *      !           !           !           !
51 *
52 *      *****
53 *      MODIFICATIONS:
54 *      ! date !   auteur !           but

```

```

55 * ----- *
56 *                                     $ *
57 *****/
58 #ifndef HYPER_W_GENE_3D_H
59 #define HYPER_W_GENE_3D_H
60
61
62 #include "LesReferences.h"
63 #include "LesCourbes1D.h"
64 #include "Tenseur.h"
65 #include "Tenseur3.h"
66 #include "TenseurQ-3.h"
67 #include "TenseurQ3gene.h"
68 #include "Loi_comp_abstraite.h"
69
70
71 /// @addtogroup Les_lois_hyperelastiques
72 /// @{
73 ///
74
75 class Hyper_W_gene_3D : public Loi_comp_abstraite
76 {
77
78     public :
79
80         // CONSTRUCTEURS :
81
82         // Constructeur par défaut
83         Hyper_W_gene_3D ();
84
85         // Constructeur utile si l'identificateur du nom de la loi
86         // de comportement et la dimension sont connus
87         // vit_def indique si oui ou non la loi utilise la vitesse de déformation
88         Hyper_W_gene_3D (Enum_comp id_compor,Enum_categorie_loi_comp categorie_comp
89             ,int dimension,bool vit_def = false);
90
91         // Constructeur de copie
92         Hyper_W_gene_3D (const Hyper_W_gene_3D& loi) ;
93
94         // DESTRUCTEUR :
95         virtual ~Hyper_W_gene_3D ();
96
97
98 // classe permettant le stockage de grandeurs de post-traitement
99 class Invariantpost3D
100 { public :
101     Invariantpost3D() : potentiel(0.) {}; // constructeur par défaut
102     Invariantpost3D(const double& potent)
103         : potentiel(potent) {};
104     Invariantpost3D(const Invariantpost3D & a)
105         : potentiel(a.potentiel) {};
106     // Surcharge de l'opérateur = : réalise l'affectation
107     Invariantpost3D& operator= (const Invariantpost3D& a)
108     {potentiel=a.potentiel; return *this;};
109     // surcharge de l'opérateur de lecture
110     friend istream & operator » (istream & ent, Invariantpost3D & a)
111     { string toto;
112       ent » toto » a.potentiel; return ent;
113     };
114     // surcharge de l'opérateur d'écriture
115     friend ostream & operator « (ostream & sort , const Invariantpost3D & a)
116     { sort « " potent= "« a.potentiel « " "; return sort;
117     };
118     // data
119     double potentiel;
120 };
121
122 class SaveResulHyper_W_gene_3D: public Loi_comp_abstraite::SaveResul//HyperD
123 { public :
124     // constructeur
125     SaveResulHyper_W_gene_3D(); // par défaut
126     SaveResulHyper_W_gene_3D(int sortie_post); // avec init ou pas
127     SaveResulHyper_W_gene_3D(const SaveResulHyper_W_gene_3D& sav); // de copie
128     virtual ~SaveResulHyper_W_gene_3D(); // destructeur
129
130     // définition d'une nouvelle instance identique
131     // appelle du constructeur via new
132     SaveResul * Nevez_SaveResul() const {return (new SaveResulHyper_W_gene_3D(*this));};
133     // affectation
134     virtual SaveResul & operator = ( const SaveResul & a);
135     //===== lecture écriture dans base info =====
136     // cas donne le niveau de la récupération
137     // = 1 : on récupère tout

```



```

142 // = 2 : on récupère uniquement les données variables (supposées comme telles)
143 virtual void Lecture_base_info( ifstream& ,const int ) ;
144 // cas donne le niveau de sauvegarde
145 // = 1 : on sauvegarde tout
146 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
147 virtual void Ecriture_base_info(ofstream& ,const int ) ;
148
149 // mise à jour des informations transitoires
150 void TdtversT() { if (invP != NULL) { (*invP_t) = (*invP);};};
151 void TversTdt() { if (invP != NULL) { (*invP) = (*invP_t);};};
152
153 // affichage à l'écran des infos
154 void Affiche() const ;
155
156 //changement de base de toutes les grandeurs internes tensorielles stockées
157 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
158 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
159 // ici il ne s'agit que de grandeurs scalaires donc rien n'a faire
160 // gpH(i) = gamma(i,j) * gH(j)
161 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma){};
162
163 // procedure permettant de completer éventuellement les données particulières
164 // de la loi stockées
165 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
166 // completer est appelé apres sa creation avec les donnees du bloc transmis
167 // peut etre appeler plusieurs fois
168 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
169                               ,const Loi_comp_abstraite* loi){};
170
171
172 // des grandeurs qui sont éventuellement créent si on le demande
173 Hyper_W_gene_3D::Invariantpost3D * invP, * invP_t;
174 };
175
176 SaveResul * New_et_Initialise() { return (new SaveResulHyper_W_gene_3D(this->sortie_post));};
177
178 // récupération des grandeurs particulière (hors ddl )
179 // correspondant à liTQ
180 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
181 virtual void Grandeur_particuliere
182 (bool absolue,List_io<TypeQuelconque>& liTQ,Loi_comp_abstraite::SaveResul * saveDon,list<int>&
183  decal) const;
184 // récupération de la liste de tous les grandeurs particulières
185 // ces grandeurs sont ajoutées à la liste passées en paramètres
186 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
187 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& liTQ) const;
188
189
190 // calcul des invariants et de leurs variations premières
191 void Invariants_et_var1(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
192                        ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
193                        ,const double& jacobien_0,const double& jacobien)
194 {InvariantsEtVar1(gijBB_0,gijHH_0,gijBB_tdt,gijHH_tdt,jacobien_0,jacobien);
195 // vérification numérique de la dérivée
196 // Calcul_derivee_numerique(gijBB_0,gijHH_0,gijBB_tdt,gijHH_tdt,jacobien_0,jacobien);
197 };
198 // calcul des invariants et de leurs variations premières et secondes
199 void Invariants_et_var2(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
200                        ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
201                        ,const double& jacobien_0,const double& jacobien)
202 {InvariantsEtVar2(gijBB_0,gijHH_0,gijBB_tdt,gijHH_tdt,jacobien_0,jacobien);
203 // Calcul_derivee_numerique2(gijBB_0,gijHH_0,gijBB_tdt,gijHH_tdt,jacobien_0,jacobien);
204 };
205
206 // --- acces en lectures aux différentes grandeurs calculées
207 // par Invariants_et_var1 et Invariants_et_var2
208 const double& I__B() const {return I_B;};
209 const double& I__BB() const {return I_BB;};
210 // const double& I__BBB() const {return I_BBB;};
211 const double& II__B() const {return II_B;};
212 const Vecteur& JJ_r() const {return J_r;};
213 // const Tenseur3HH& D_I_B_epsBB_HH() const {return d_I_B_epsBB_HH;};
214 // const Tenseur3HH& D_II_B_epsBB_HH() const {return d_II_B_epsBB_HH;};
215 // const Tenseur3HH& D_III_B_epsBB_HH() const {return d_III_B_epsBB_HH;};
216 const Tableau <Tenseur3HH>& D_J_r_epsBB_HH() const {return d_J_r_epsBB_HH;};
217 // --- acces en lectures aux différentes grandeurs calculées
218 // uniquement par Invariants_et_var2
219 // const Tenseur3HHHH& D_I_B_eps2BB_HHHH () const {return d_I_B_eps2BB_HHHH;};
220 // const Tenseur3HHHH& D_II_B_eps2BB_HHHH () const {return d_II_B_eps2BB_HHHH;};
221 // const Tenseur3HHHH& D_III_B_eps2BB_HHHH () const {return d_III_B_eps2BB_HHHH;};
222 const Tenseur3HHHH& D_J_1_eps2BB_HHHH () const {return d_J_1_eps2BB_HHHH;};
223 const Tenseur3HHHH& D_J_2_eps2BB_HHHH () const {return d_J_2_eps2BB_HHHH;};
224 const Tenseur3HHHH& D_J_3_eps2BB_HHHH () const {return d_J_3_eps2BB_HHHH;};
225
226
227 protected :

```

```

228 // données
229 int sortie_post; // permet de stocker et ensuite d'accéder en post-traitement à certaines données
230 // = 0 par défaut,
231 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
232 // lecture dans les classes dérivées !!
233
234 double I_B,I_BB;//,I_BBB; // les 3 invariants primaires de B c'est-à-dire les trois traces
235 // I_B, I_{B.B}, I_{B.B.B}, en fait on ne se sert pas de I_BBB
236 double II_B,III_B; // les 3 invariants en I de B: I_1=I_B, I_2=II_B,I_3=III_B
237 Vecteur J_r ; // les 3 invariants en J
238 // variation des I II III par rapport aux composantes covariantes de la déformation
239 Tenseur3HH d_I_B_epsBB_HH,d_II_B_epsBB_HH,d_III_B_epsBB_HH;
240 // variation des J_r par rapport aux composantes covariantes de la déformation
241 Tableau <Tenseur3HH> d_J_r_epsBB_HH;
242 // variation seconde des I_r par rapport aux composantes covariantes de la déformation
243 // Tenseur3HHHH d_I_B_eps2BB_HHHH,d_II_B_eps2BB_HHHH,d_III_B_eps2BB_HHHH;
244 // variation seconde des J_r par rapport aux composantes covariantes de la déformation
245 Tenseur3HHHH d_J_1_eps2BB_HHHH,d_J_2_eps2BB_HHHH,d_J_3_eps2BB_HHHH;
246 // autres variables utiles pour la loi de comportement
247 double V; // variation relative de volume
248 Tenseur3HH BB_HH; // B . B
249 Tenseur3HB B_HB; // B en mixte
250 Tenseur3HB BB_HB; // B . B en mixte
251
252 // variables tensorielles du second ordre intermédiaires
253 Tenseur3HHHH IxI_HHHH,IxbarreI_HHHH,IxB_HHHH,BxI_HHHH;
254
255
256 // grandeurs de travail
257 double J3_puiss_untiers // J3 puissance 1/3
258 ,unSurJ3_puissuntiers // 1/(J3^{1/3})
259 ,unSurJ3_puissuntiers2 // (1/(J3^{1/3}))^2
260 ,unSurJ3_puissuntiers4; // (1/(J3^{1/3}))^4
261
262
263
264 // -----méthodes internes
265
266 // affichage et definition interactive des commandes particulières ici = une partie générique
267 // pour les lois, utilisée par les classes dérivées
268 void Info_commande LoisDeComp_hyper3D(UtilLecture& lec);
269
270 // calcul des invariants et de leurs variations premières
271 void InvariantsEtVar1(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
272 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
273 ,const double& jacobien_0,const double& jacobien);
274
275 // calcul des invariants et de leurs variations premières et secondes
276 void InvariantsEtVar2(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
277 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
278 ,const double& jacobien_0,const double& jacobien);
279 // vérification de la dérivée avec une dérivée numérique
280 // calcul des dérivées numériques premières
281 void Calcul_derivee_numerique(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
282 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
283 ,const double& jacobien_0,const double& jacobien);
284 // calcul des dérivées numériques premières et secondes
285 void Calcul_derivee_numerique2(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
286 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
287 ,const double& jacobien_0,const double& jacobien);
288 // --- pour le debug ---
289 // calcul des invariants et de leurs variations premières en numérique
290 void Invariants_et_var1_deb(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
291 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
292 ,const double& jacobien_0,const double& jacobien)
293 {InvariantsEtVar1(gijBB_0,gijHH_0,gijBB_tdt,gijHH_tdt,jacobien_0,jacobien);
294 // vérification numérique de la dérivée
295 Calcul_derivee_numerique(gijBB_0,gijHH_0,gijBB_tdt,gijHH_tdt,jacobien_0,jacobien);
296 };
297 // calcul des invariants et de leurs variations premières et secondes
298 void Invariants_et_var2_deb(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
299 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
300 ,const double& jacobien_0,const double& jacobien)
301 {InvariantsEtVar2(gijBB_0,gijHH_0,gijBB_tdt,gijHH_tdt,jacobien_0,jacobien);
302 Calcul_derivee_numerique2(gijBB_0,gijHH_0,gijBB_tdt,gijHH_tdt,jacobien_0,jacobien);
303 };
304
305 };
306 /// @} // end of group
307
308
309 #endif
310
311
312

```

## 7.40 HyperD.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           1/10/98
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 *
39 *   BUT: Classe générale des potentiels hyperélastiques isotropes,
40 *   tels que définis par Denis Favier.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !           but
47 *   -----
48 *   !           !           !
49 *
50 *   *****
51 *   MODIFICATIONS:
52 *
53 *   ! date !   auteur !           but
54 *   -----
55 *
56 *   *****/
57
58 #ifndef HYPERD_H
59 #define HYPERD_H
60
61 #include "Loi_comp_abstraite.h"
62 #include "Tenseur.h"
63 #include "Tenseur3.h"
64 #include "TenseurQ-3.h"
65 #include "TenseurQ3gene.h"
66 #include "TypeConsTens.h"
67
68 /** @defgroup Les_lois_hyperelastiques
69 *
70 *   BUT: groupe des lois hyperelastiques
71 *
72 *   \author Gérard Rio
73 *   \version 1.0
74 *   \date 11/06/2019
75 *   \brief groupe des lois hyperelastiques
76 */
77
78 /// @addtogroup Les_lois_hyperelastiques
79 /// @{
80
81
82 //template <class TensHH,class TensBB,class TensBH,class TensHB>
83 class HyperD : public Loi_comp_abstraite
84 {

```

```

85 public :
86     // CONSTRUCTEURS :
87
88     HyperD (); // Constructeur par défaut
89     // Constructeur utile si l'identificateur du nom de la loi
90     // de comportement, le paramètre phase sont connus
91     HyperD (Enum_comp id_compor, Enum_categorie_loi_comp categorie_comp, int dimension, bool avec_ph) ;
92     // Constructeur utile si l'identificateur du nom de la loi
93     // de comportement, le paramètre phase sont connus
94     HyperD (char* nom, Enum_categorie_loi_comp categorie_comp, int dimension, bool avec_ph) ;
95     // DESTRUCTEUR :
96     ~HyperD ();
97     // constructeur de copie
98     HyperD (const HyperD & a) ;
99
100
101
102 // classe permettant le stockage de grandeurs de post-traitement
103 class Invariantpost3D
104 { public :
105     Invariantpost3D() : V(0.), Qeps(0.), cos3phi(0.), potentiel(0.) {}; // constructeur par défaut
106     Invariantpost3D(const double& V1, const double& Qe, const double& cos3p, const double& potent)
107     : V(V1), Qeps(Qe), cos3phi(cos3p), potentiel(potent) {};
108     Invariantpost3D(const Invariantpost3D & a)
109     : V(a.V), Qeps(a.Qeps), cos3phi(a.cos3phi), potentiel(a.potentiel) {};
110     // Surcharge de l'operateur = : realise l'affectation
111     Invariantpost3D& operator= (const Invariantpost3D& a)
112     {V=a.V; Qeps=a.Qeps; cos3phi=a.cos3phi; potentiel=a.potentiel; return (*this);};
113     // surcharge de l'operateur de lecture
114     friend istream & operator » (istream & ent, Invariantpost3D & a)
115     { string toto;
116     ent » toto » a.V » toto » a.Qeps » toto » a.cos3phi
117     » toto » a.potentiel; return ent;
118     };
119     // surcharge de l'operateur d'écriture
120     friend ostream & operator « (ostream & sort, const Invariantpost3D & a)
121     { sort « " V= " « a.V « " Qeps= " « a.Qeps « " cos3phi= " « a.cos3phi
122     « " potent= " « a.potentiel « " "; return sort;
123     };
124     // data
125     double V, Qeps, cos3phi, potentiel;
126 };
127
128 class SaveResulHyperD : public Loi_comp_abstraite::SaveResul//HyperD
129 { public :
130     // constructeur
131     SaveResulHyperD(); // par défaut
132     SaveResulHyperD(int sortie_post); // avec init ou pas
133     SaveResulHyperD(const SaveResulHyperD& sav); // de copie
134     virtual ~SaveResulHyperD(); // destructeur
135
136     // définition d'une nouvelle instance identique
137     // appelle du constructeur via new
138     SaveResul * Nevez_SaveResul() const {return (new SaveResulHyperD(*this));};
139     // affectation
140     virtual SaveResul & operator = (const SaveResul & a);
141     //===== lecture écriture dans base info =====
142     // cas donne le niveau de la récupération
143     // = 1 : on récupère tout
144     // = 2 : on récupère uniquement les données variables (supposées comme telles)
145     virtual void Lecture_base_info (ifstream&, const int );
146     // cas donne le niveau de sauvegarde
147     // = 1 : on sauvegarde tout
148     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
149     virtual void Ecriture_base_info (ofstream&, const int );
150
151     // mise à jour des informations transitoires
152     void TdtversT(); ;
153     void TversTdt(); ;
154
155     // affichage à l'écran des infos
156     void Affiche() const ;
157
158     //changement de base de toutes les grandeurs internes tensorielles stockées
159     // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
160     // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
161     // ici il ne s'agit que de grandeurs scalaires donc rien n'a faire
162     // gpH(i) = gamma(i,j) * gH(j)
163     virtual void ChBase_des_grandeurs (const Mat_pleine& beta, const Mat_pleine& gamma){};
164
165     // procedure permettant de completer éventuellement les données particulières
166     // de la loi stockées
167     // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
168     // completer est appelé apres sa creation avec les donnees du bloc transmis
169     // peut etre appeler plusieurs fois
170     SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
171     ,const Loi_comp_abstraite* loi) {};

```

```

172
173 // des grandeurs qui sont éventuellement créées si on le demande
174 HyperD::Invariantpost3D * invP, * invP_t;
175
176 // --- gestion d'une map de grandeurs quelconques éventuelles ---
177
178 // une map de grandeurs quelconques particulière qui peut servir aux classes appelantes
179 // il s'agit ici d'une map interne qui a priori ne doit servir qu'aux class loi de comportement
180 // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
181 // -> n'est pas sauvegardé, car a priori il s'agit de grandeurs redondantes
182 map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque > >
map_type_quelconque;
183
184 // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
185 const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque > >*
Map_type_quelconque()
186 const {return &map_type_quelconque;};
187 private:
188 void Mise_a_jour_map_type_quelconque();
189
190 // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ---
191 };
192
193 // création d'une nouvelle instance de SaveResul
194 SaveResul * New_et_Initialise();
195
196 // récupération des grandeurs particulière (hors ddl )
197 // correspondant à liTQ
198 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
199 virtual void Grandeur_particuliere
200 (bool absolue, List_io<TypeQuelconque>& liTQ, Loi_comp_abstraite::SaveResul * saveDon, list<int>&
decal) const;
201 // récupération de la liste de tous les grandeurs particulières
202 // ces grandeurs sont ajoutées à la liste passées en paramètres
203 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
204 virtual void ListeGrandeurs_particulieres(bool absolue, List_io<TypeQuelconque>& liTQ) const;
205
206 // 1) definition d'une classe dérivée de SaveResul pour stocker le
207 // jacobien initial
208
209 //class SaveResulHyperD : public Loi_comp_abstraite::SaveResul
210 // { public :
211 // // constructeur
212 // SaveResulHyperD() : jacobien_0(0) {};
213 // SaveResulHyperD(const double & jacob) : jacobien_0(jacob) {};
214 //
215 // // définition d'une nouvelle instance identique
216 // // appelle du constructeur via new
217 // SaveResul * Nevez_SaveResul() const {return (new SaveResulHyperD(*this));};
218 // //===== lecture écriture dans base info =====
219 // // cas donne le niveau de la récupération
220 // // = 1 : on récupère tout
221 // // = 2 : on récupère uniquement les données variables (supposées comme telles)
222 // virtual void Lecture_base_info (ifstream& ,const int ) {};
223 // // cas donne le niveau de sauvegarde
224 // // = 1 : on sauvegarde tout
225 // // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
226 // virtual void Ecriture_base_info(ofstream& ,const int ) {};
227 //
228 // // affichage à l'écran des infos
229 // void Affiche() const { cout << "\n SaveResulHyperD: jacobien_0= " << jacobien_0 << " "; };
230 //
231 // double jacobien_0;
232 //
233 // };
234 //
235 // SaveResul * New_et_Initialise() { return (new SaveResulHyperD(0.));};
236
237 // 2) METHODES public découlant de méthodes virtuelles :
238
239 // affichage des donnees particulieres a l'elements
240 // de matiere traite ( c-a-dire au pt calcule)
241 void AfficheDataSpecif(ofstream& ,SaveResul * ) const {};
242
243 // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
244 // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
245 virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >&
listEnuQuelc);
246
247 // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
248 // passée en paramètre, dans le save result: ces conteneurs doivent être valides
249 // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
250 virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul);
251
252
253
-----

```

```

254 // on définit des classes conteneurs pour le passage sécurisé d'information au niveau des potentiels etc
255 ..
256 //-----
257 public :
258
259 class PotensSansPhaseSansVar
260 { public :
261     PotensSansPhaseSansVar() : E(0.),EV(0.),EbIIB(0.),Ks(0) {};
262     PotensSansPhaseSansVar(const PotensSansPhaseSansVar& a) : E(a.E),EV(a.EV),EbIIB(a.EbIIB),Ks(a.Ks)
263     {};
264     PotensSansPhaseSansVar& operator = (const PotensSansPhaseSansVar& a)
265     {E=a.E;EV=a.EV;EbIIB=a.EbIIB;Ks=a.Ks;return *this;};
266     double E; // valeur du potentiel
267     double EV; // variation du potentiel par rapport à V
268     double EbIIB; // variation du potentiel par rapport à bIIB
269     double Ks; // module de compressibilité sécant par rapport à log(V)
270 };
271
272 class PotensSansPhaseAvecVar : public PotensSansPhaseSansVar
273 { public :
274     PotensSansPhaseAvecVar() : PotensSansPhaseSansVar(),EVV(0.),EbIIBV(0.),EbIIB2(0.) {};
275     PotensSansPhaseAvecVar(const PotensSansPhaseAvecVar& a) :
276     PotensSansPhaseSansVar(a),EVV(a.EVV),EbIIBV(a.EbIIBV),EbIIB2(a.EbIIB2) {};
277     PotensSansPhaseAvecVar& operator = (const PotensSansPhaseAvecVar& a)
278     {this->PotensSansPhaseSansVar::operator = (a);
279     EVV=a.EVV;EbIIBV=a.EbIIBV;EbIIB2=a.EbIIB2;return *this;};
280     double EVV; // variation seconde par rapport à V
281     double EbIIBV; // variation seconde par rapport à bIIB et V
282     double EbIIB2; // variation seconde par rapport à bIIB
283 };
284
285 class Invariant
286 { public :
287     Invariant() : Ieps(0.),V(0.),bIIB(0.),bIIIb(0.) {};
288     Invariant(const double& I_n,const double& V_n,const double& bIIB_n,const double& bIIIb_n) :
289     Ieps(I_n),V(V_n),bIIB(bIIB_n),bIIIb(bIIIb_n) {};
290     Invariant(const Invariant& a) :
291     Ieps(a.Ieps),V(a.V),bIIB(a.bIIB),bIIIb(a.bIIIb) {};
292     Invariant& operator = (const Invariant& a)
293     {Ieps=a.Ieps;V=a.V;bIIB=a.bIIB;bIIIb=a.bIIIb;return *this; }
294     double Ieps; // trace du tenseur de déformation
295     double V; // variation relative de volume
296     double bIIB; // second invariant barre du déviateur de déformation
297     double bIIIb; // troisième invariant barre du déviateur de déformation
298 };
299
300 class InvariantVarDdl : public Invariant
301 { public :
302     InvariantVarDdl(int nddl = 0) :
303     Invariant(),dIeps(nddl),dV(nddl),dbIIB(nddl),dbIIIb(nddl) {};
304     InvariantVarDdl(const InvariantVarDdl& a) :
305     Invariant(a),dIeps(a.dIeps),dV(a.dV),dbIIB(a.dbIIB),dbIIIb(a.dbIIIb) {};
306     InvariantVarDdl& operator = (const InvariantVarDdl& a)
307     {this->Invariant::operator = (a);
308     dIeps=a.dIeps;dV=a.dV;dbIIB=a.dbIIB;dbIIIb=a.dbIIIb;return *this; }
309     // données
310     Tableau<double> dIeps; // variation de Ieps par rapport au ddl
311     Tableau<double> dV; // variation de V par rapport au ddl
312     Tableau<double> dbIIB; // variation de bIIB par rapport au ddl
313     Tableau<double> dbIIIb; // variation de bIIIb par rapport au ddl
314 };
315
316 class InvariantVarEps : public Invariant // utilisable uniquement en 3D (ce qui est a priori les cas
317 voulu)
318 { public :
319     InvariantVarEps() :
320     Invariant(),dIeps_deps_HH(),dV_deps_HH(),dbIIB_deps_HH(),dbIIIb_deps_HH() {};
321     InvariantVarEps(const InvariantVarEps& a) :
322     Invariant(a),dIeps_deps_HH(a.dIeps_deps_HH),
323     dV_deps_HH(a.dV_deps_HH),dbIIB_deps_HH(a.dbIIB_deps_HH),dbIIIb_deps_HH(a.dbIIIb_deps_HH) {};
324     InvariantVarEps& operator = (const InvariantVarEps& a)
325     {this->Invariant::operator = (a);dIeps_deps_HH=a.dIeps_deps_HH;
326     dV_deps_HH=a.dV_deps_HH;dbIIB_deps_HH=a.dbIIB_deps_HH;dbIIIb_deps_HH=a.dbIIIb_deps_HH;
327     return *this; }
328     // données
329     Tenseur3HH dIeps_deps_HH; // variation de Ieps par rapport à epsBB
330     Tenseur3HH dV_deps_HH; // variation de V par rapport à epsBB
331     Tenseur3HH dbIIB_deps_HH; // variation de bIIB par rapport à epsBB
332     Tenseur3HH dbIIIb_deps_HH; // variation de bIIIb par rapport à epsBB
333 };
334
335 class PotenAvecPhaseSansVar
336 { public :
337     PotenAvecPhaseSansVar() : E(0.),EV(0.),EbIIB(0.),EIeps(0.),Ks(0) {};
338     PotenAvecPhaseSansVar(const PotenAvecPhaseSansVar& a) :
339     E(a.E),EV(a.EV),EbIIB(a.EbIIB),EIeps(a.EIeps),Ks(a.Ks) {};
340     PotenAvecPhaseSansVar& operator = (const PotenAvecPhaseSansVar& a)

```

```

335     {E=a.E;EV=a.EV;EbIib=a.EbIib;EIeps=a.EIeps;Ks=a.Ks;return *this; }
336 double E; // valeur du potentiel
337 double EV; // variation du potentiel par rapport à V
338 double EbIib; // variation du potentiel par rapport à bIib
339 double EIeps; // variation du potentiel par rapport à Ieps
340 double Ks; // module de compressibilité sécant par rapport à log(V)
341 };
342 class PotenAvecPhaseAvecVar : public PotenAvecPhaseSansVar
343 { public :
344     PotenAvecPhaseAvecVar() : PotenAvecPhaseSansVar()
345     ,EVV(0.),EbIib2(0.),EIeps2(0.),EbIibV(0.),EIepsV(0.),EbIibIeps(0.),Ks(0) {};
346     PotenAvecPhaseAvecVar (const PotenAvecPhaseAvecVar& a) :
347     PotenAvecPhaseSansVar(a)
348     ,EVV(a.EVV),EbIib2(a.EbIib2),EIeps2(a.EIeps2),EbIibV(a.EbIibV)
349     ,EIepsV(a.EIepsV),EbIibIeps(a.EbIibIeps),Ks(a.Ks) {};
350     PotenAvecPhaseAvecVar& operator = (const PotenAvecPhaseAvecVar& a)
351     {this->PotenAvecPhaseSansVar::operator = (a);
352     EVV=a.EVV;EbIib2=a.EbIib2;EIeps2=a.EIeps2;EbIibV=a.EbIibV;
353     EIepsV=a.EIepsV;EbIibIeps=a.EbIibIeps;Ks=a.Ks;return *this; }
354
355     double EVV; // variation seconde par rapport à V
356     double EbIib2; // variation seconde par rapport à bIib
357     double EIeps2; // variation seconde par rapport à Ieps
358     double EbIibV; // variation seconde par rapport à bIib et V
359     double EIepsV; //variation seconde par rapport à Ieps et V
360     double EbIibIeps; //variation seconde par rapport à bIib et Ieps
361     double Ks; // module de compressibilité sécant par rapport à log(V)
362 };
363
364 class A_i
365 { public :
366     A_i() : a_0(0.),a_1(0.),a_2(0.) {};
367     A_i (const A_i& a) :
368     a_0(a.a_0),a_1(a.a_1),a_2(a.a_2) {};
369     A_i& operator = (const A_i& a)
370     {a_0=a.a_0;a_1=a.a_1;a_2=a.a_2;return *this; }
371     double a_0,a_1,a_2; // coeff ai pour le calcul de la contrainte
372 };
373 class A_iAvecVarDdl : public A_i
374 { public :
375     A_iAvecVarDdl (int nddl = 0) :
376     A_i(),da_0(nddl),da_1(nddl),da_2(nddl) {};
377     A_iAvecVarDdl (const A_iAvecVarDdl& a) :
378     A_i(a),da_0(a.da_0),da_1(a.da_1),da_2(a.da_2) {};
379     A_iAvecVarDdl& operator = (const A_iAvecVarDdl& a)
380     {this->A_i::operator = (a);
381     da_0=a.da_0;da_1=a.da_1;da_2=a.da_2;return *this;};
382     // données
383     Tableau<double> da_0; // variation de a_0 par rapport au ddl
384     Tableau<double> da_1; // variation de a_1 par rapport au ddl
385     Tableau<double> da_2; // variation de a_2 par rapport au ddl
386 };
387 class A_iAvecVarEps : public A_i
388 { public :
389     A_iAvecVarEps () :
390     A_i(),da_0_deps_HH(),da_1_deps_HH(),da_2_deps_HH() {};
391     A_iAvecVarEps (const A_iAvecVarEps& a) :
392     A_i(a),da_0_deps_HH(a.da_0_deps_HH)
393     ,da_1_deps_HH(a.da_1_deps_HH),da_2_deps_HH(a.da_2_deps_HH) {};
394     A_iAvecVarEps& operator = (const A_iAvecVarEps& a)
395     {this->A_i::operator = (a);da_0_deps_HH=a.da_0_deps_HH;
396     da_1_deps_HH=a.da_1_deps_HH;da_2_deps_HH=a.da_2_deps_HH;return *this;};
397     // données
398     Tenseur3HH da_0_deps_HH; // variation de a_0 par rapport à epsBB
399     Tenseur3HH da_1_deps_HH; // variation de a_1 par rapport à epsBB
400     Tenseur3HH da_2_deps_HH; // variation de a_2 par rapport à epsBB
401 };
402
403
404 protected :
405
406 // 3) METHODES protegees decoulant de virtuelles pures:
407 // calcul des contraintes a t+dt
408 // calcul des contraintes
409 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
410 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
411 ,TenseurBB & delta_epsBB_
412 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
413 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
414 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
415 module_cisaillement
416 ,const Met_abstraite::Expli_t_tdt& ex);
417
418 // calcul des contraintes et de ses variations a t+dt
419 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
420 ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
421 ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt

```

```

421     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
422     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
423     ,Tableau <TenseurBB *>& d_gijBB_tdt
424     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
425     ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
426     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
427     ,const Met_abstraite::Impli& ex);
428
429     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
430     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
431     // le tenseur de déformation et son incrémentsont également en orthonormees
432     // si = false: les bases transmises sont utilisées
433     // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
434 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
435     ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
436     ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
437     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
438     ,const Met_abstraite::Umat_cont& ex) ; // = 0;
439
440 // calcul de grandeurs de travail aux points d'intégration via la def
441 // fonction surchargée dans les classes dérivée si besoin est
442 virtual void CalculGrandeurTravail
443     (const PtIntegMecaInterne& ,const Deformation &
444     ,Enum_dure,const ThermoDonnee&
445     ,const Met_abstraite::Impli* ex_impli
446     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
447     ,const Met_abstraite::Umat_cont* ex_umat
448     ,const List_io<Ddl_etendu>* exclure_dd_etend
449     ,const List_io<const TypeQuelconque *>* exclure_Q
450     ) {};
451
452 // 4) METHODES internes spécifiques à l'hyperélasticité isotrope
453
454 // par exemple, regarder dans hyper10 pour le calcul des invariants
455
456 // Calcul des invariants et, de epsBH,
457 // retour de IdGBH qui pointe sur le bon tenseur
458 //++ virtual TensBH * Invariants (TenseurBB& epsBB_t,TenseurBB& gijBB_t,
459 virtual TenseurBH * Invariants (const TenseurBB& epsBB_t,const TenseurBB& gijBB_t,
460     const TenseurHH & gijHH_t,const double& jacobien_0,const double& jacobien_t,
461 //++ Invariant & invariant,TensBH & epsBH) = 0;
462     Invariant & invariant,TenseurBH & epsBH) = 0;
463 // calcul des invariants et de leurs variations par rapport au ddl, de epsBH,
464 // et de sa variation, puis retour de IdGBH qui pointe sur le bon tenseur
465 //++ virtual TensBH * Invariants_et_var (TenseurBB& epsBB_tdt,
466 virtual TenseurBH * Invariants_et_var (const TenseurBB& epsBB_tdt,
467     const TenseurBB& gijBB_tdt,const Tableau <TenseurBB *>& d_gijBB_tdt,
468     const TenseurHH & gijHH_tdt,const Tableau <TenseurHH *>& d_gijHH_tdt,
469     const double& jacobien_0,const double& jacobien_tdt,const Vecteur& d_jacobien_tdt,
470     InvariantVarDdl & invariantVarDdl,
471 //++ TensBH & epsBH_tdt,Tableau<TensBH> & depsBH_tdt) = 0;
472     TenseurBH & epsBH_tdt,Tableau<TenseurBH*> & depsBH_tdt) = 0;
473
474 // calcul des invariants et de leurs variations par rapport aux déformations
475 virtual TenseurBH * Invariants_et_varEps (const TenseurBB& epsBB_tdt,
476     const TenseurBB& gijBB_tdt,const TenseurHH & gijHH_tdt,
477     const double& jacobien_0,const double& jacobien_tdt,
478     InvariantVarEps & invariantVarEps,TenseurBH & epsBH_tdt) = 0;
479
480
481 // calcul du potentiel et de ses dérivées non compris la phase
482 virtual PotenSansPhaseSansVar Potentiel
483     (const Invariant & invariant,const double& jacobien0) = 0;
484 // calcul du potentiel et de ses dérivées avec la phase
485 virtual PotenAvecPhaseSansVar PotentielPhase
486     (const Invariant& invariant,const double& jacobien0) = 0;
487 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux invariants
488 virtual PotenSansPhaseAvecVar Potentiel_et_var
489     (const Invariant& invariant,const double& jacobien0) = 0;
490 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux invariants
491 virtual PotenAvecPhaseAvecVar PotentielPhase_et_var
492     (const Invariant& invariant,const double& jacobien0) = 0;
493
494
495 protected :
496     // VARIABLES PROTEGEES :
497     bool avec_phase; // vrai quand on travail avec la phase
498     // ---utilisé par les potentiels Orgeas
499 // Invariant invariant_t; //invariants à t donc au début du pas
500 bool avec_regularisation; // si oui, on régularise les termes qui tendent vers 0
501 double fact_regularisation; // spécifie le facteur de régularisation
502
503 int sortie_post; // permet de stocker et ensuite d'accéder en post-traitement à certaines données
504 // = 0 par défaut,
505 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie

```



```

506         // lecture dans les classes dérivées !!
507
508         // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonormee
509         Tenseur3HHHH I_xbarre_I_HHHH;
510
511         // variables intermédiaire qui sont renseignées par
512         // HyperD::Calcul_SigmaHH ou HyperD::Calcul_DsigmaHH_tdt ou HyperD::Calcul_dsigma_deps
513         // et qui sont utilisées ensuite par les potentiels spécifiques pour les fct nD
514         const Met_abstraite::Impli* ex_impli_hyper;
515         const Met_abstraite::Expli_t_tdt* ex_expli_tdt_hyper;
516         const Met_abstraite::Umat_cont* ex_umat_hyper;
517
518
519         // double scale_eps; // paramètre de scale d'eps, pour limiter les divisions par 0
520         // la définition est faite dans les classes dérivées ainsi que la sauvegarde, lecture etc...
521
522         // METHODES PROTEGEES :
523
524         // affichage et definition interactive des commandes particulières ici = une partie générique
525         // pour les lois hyper 3D, utilisée par les classes dérivées
526         void Info_commande_LoisDeComp_hyper3D(UtilLecture& lec);
527
528         // calcul des coefficients alpha dans le cas sans la phase
529         inline void AAA_i (const PotenSansPhaseSansVar& potenSansPhaseSansVar, const Invariant& invariant
530             , const double& jaco, A_i& a_i);
531
532         // calcul des coefficients alpha dans le cas avec la phase
533         inline void AAA_iPhase(const PotenAvecPhaseSansVar& potenAvecPhaseSansVar, const Invariant& invariant
534             , const double& jaco, A_i& a_i);
535
536         // calcul des coefficients alpha dans le cas sans la phase et de leurs variations / ddl
537         inline void AAA_i_var(const PotenSansPhaseAvecVar& potenSansPhaseAvecVar, const double& jaco0
538             , const InvariantVarDdl & invariantVarDdl
539             , const double& jaco, const Vecteur& d_jacobien_tdt, A_iAvecVarDdl&
540             a_iAvecVarDdl);
541
542         // calcul des coefficients alpha dans le cas avec la phase et de leurs variations / ddl
543         inline void AAA_iPhase_var(const PotenAvecPhaseAvecVar& potenAvecPhaseAvecVar, const double& jaco0
544             , const InvariantVarDdl & invariantVarDdl
545             , const double& jaco, const Vecteur& d_jacobien_tdt, A_iAvecVarDdl&
546             a_iAvecVarDdl);
547
548         // calcul des coefficients alpha dans le cas sans la phase et de leurs variations / eps
549         inline void AAA_i_varEps(const PotenSansPhaseAvecVar& potenSansPhaseAvecVar, const double& jaco0
550             , const InvariantVarEps & invariantVarEps
551             , const double& jaco, A_iAvecVarEps& a_iAvecVarEps);
552
553         // calcul des coefficients alpha dans le cas avec la phase et de leurs variations / eps
554         inline void AAA_iPhase_varEps (const PotenAvecPhaseAvecVar& potenAvecPhaseAvecVar, const double&
555             jaco0
556             , const InvariantVarEps & invariantVarEps
557             , const double& jaco, A_iAvecVarEps& a_iAvecVarEps);
558
559     };
560     /// @} // end of group
561
562     ///++#include "HyperD.cc"
563 #endif

```

## 7.41 HyperDN.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty

```

```

23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      1/10/98
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++
37 *
38 *****/
39 *   BUT: Classe générale des potentiels hyperélastiques isotropes,
40 *   définis à partir des invariants :
41 *   - V : variation relative de volume,
42 *   - Ieps : trace du tenseur de déformation d'Almansi,
43 *   - bIIB : deuxième invariant barre, du déviateur des
44 *   déformations.
45 *
46 *   *****
47 *
48 *   VERIFICATION:
49 *   ! date !   auteur !           but
50 *   -----
51 *   !           !           !
52 *   *****
53 *
54 *   MODIFICATIONS:
55 *   ! date !   auteur !           but
56 *   -----
57 *
58 *****/
59 #ifndef HYPERDN_H
60 #define HYPERDN_H
61
62 #include "Loi_comp_abstraite.h"
63
64
65
66 /// @addtogroup Les_lois_hyperelastiques
67 /// @{
68 ///
69
70 template <class TensHH,class TensBB,class TensBH,class TensHB,
71           class Tens_nHH,class Tens_nBB>
72 class HyperDN : public Loi_comp_abstraite
73 {
74 public :
75     // CONSTRUCTEURS :
76
77     HyperDN (); // Constructeur par défaut
78     // Constructeur utile si l'identificateur du nom de la loi
79     // de comportement et la dimension sont connus
80     HyperDN (Enum_comp id_compor,Enum_categorie_loi_comp categorie_comp,int dimension) ;
81     // Constructeur utile si l'identificateur du nom de la loi
82     // de comportement et la dimension sont connus
83     HyperDN (char* nom,Enum_categorie_loi_comp categorie_comp,int dimension) ;
84     // DESTRUCTEUR :
85     ~HyperDN ();
86     // constructeur de copie
87     HyperDN (const HyperDN & a) ;
88
89     // 1) definition d'une classe dérivée de SaveResul pour stocker le
90     // jacobien initial
91
92     class SaveResulHyperDN : public Loi_comp_abstraite::SaveResul
93     { public :
94         // constructeur
95         SaveResulHyperDN(const double & jacob) : jacobien_0(jacob) {};
96         // destructeur
97         ~SaveResulHyperDN() {};
98
99         // définition d'une nouvelle instance identique
100        // appelle du constructeur via new
101        SaveResul * Nevez_SaveResul() const {return (new SaveResulHyperDN(*this));};
102        // affectation
103        virtual SaveResul & operator = ( const SaveResul & a)
104        { SaveResulHyperDN& sav = *((SaveResulHyperDN*) &a);
105          jacobien_0=sav.jacobien_0;
106          return *this;
107        };
108        //===== lecture écriture dans base info =====

```

```

109         // cas donne le niveau de la récupération
110         // = 1 : on récupère tout
111         // = 2 : on récupère uniquement les données variables (supposées comme telles)
112     virtual void Lecture_base_info (ifstream& ent,const int cas) {};
113         // cas donne le niveau de sauvegarde
114         // = 1 : on sauvegarde tout
115         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
116     virtual void Ecriture_base_info(ofstream& sort,const int cas) {};
117
118     // affichage à l'écran des infos
119     void Affiche() const { cout << "\n SaveResulHyperDN: jacobien_0= " << jacobien_0 << " "};
120
121     //changement de base de toutes les grandeurs internes tensorielles stockées
122     // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
123     // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
124     // ici il ne s'agit que de grandeurs scalaires donc rien n'a faire
125     // gpH(i) = gamma(i,j) * gH(j)
126     virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma){};
127
128     // procedure permettant de completer éventuellement les données particulières
129     // de la loi stockées
130     // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
131     // completer est appelé apres sa creation avec les donnees du bloc transmis
132     // peut etre appeler plusieurs fois
133     SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
134         ,const Loi_comp_abstraite* loi) {return NULL;};
135
136
137     double jacobien_0;
138 };
139
140 SaveResul * New_et_Initialise() { return (new SaveResulHyperDN(0.));};
141
142 // 2) METHODES public découlant de méthodes virtuelles :
143
144 // affichage des donnees particulieres a l'elements
145 // de matiere traite ( c-a-dire au pt calcule)
146 void AfficheDataSpecif(ofstream& ,SaveResul * ) const {};
147
148
149 protected :
150
151 // 3) METHODES protegees découlant de virtuelles pures:
152 // calcul des contraintes a t+dt
153 // calcul des contraintes
154 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
155     ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
156     ,TenseurBB & delta_epsBB_
157     ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
158     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
159     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
160     module_cisaillement
161     ,const Met_abstraite::Expli_t_tdt& ex);
162
163 // calcul des contraintes et de ses variations a t+dt
164 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
165     ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
166     ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
167     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
168     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
169     ,Tableau <TenseurBB *>& d_gijBB_tdt
170     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
171     ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
172     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
173     module_cisaillement
174     ,const Met_abstraite::Impli& ex);
175
176 // fonction surchargée dans les classes dérivée si besoin est
177 virtual void CalculGrandeurTravail
178     (const PtIntegMecaInterne& ,const Deformation &
179     ,Enum_dure,const ThermoDonnee&
180     ,const Met_abstraite::Impli* ex_impli
181     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
182     ,const Met_abstraite::Umat_cont* ex_umat
183     ,const List_io<Ddl_etendu>* exclure_dd_etend
184     ,const List_io<const TypeQuelconque *>* exclure_Q
185     ) {};
186
187 // 4) METHODES internes spécifiques à l'hyperélasticité isotrope
188 // par exemple, regarder dans hyper10N pour le calcul des invariants
189
190 // Calcul des trois invariants ,de epsBH,
191 // retour de IdGBH qui pointe sur le bon tenseur
192 virtual TensBH * Invariants (TenseurBB& epsBB_t,TenseurBB& gijBB_t,
193     TenseurHH & gijHH_t, double& jacobien_0,double& jacobien_t,

```

```

194         double & Ieps,double & V,double& bIib,TensBH & epsBH) = 0;
195 // calcul des trois invariants et de leurs variations, de epsBH,
196 // et de sa variation, puis retour de IdGBH qui pointe sur le bon tenseur
197 virtual TensBH * Invariants_et_var
198 (TenseurBB& epsBB_tdt,
199  TenseurBB& gijBB_tdt,Tableau <TenseurBB *>& d_gijBB_tdt,
200  TenseurHH & gijHH_tdt,Tableau <TenseurHH *>& d_gijHH_tdt,
201  double& jacobien_0,double& jacobien_tdt,Vecteur& d_jacobien_tdt,
202  double & Ieps,Tableau<double> & dIeps,
203  double & V,Tableau<double> & dV,double& bIib,Tableau<double> & dbIib,
204  TensBH & epsBH,Tableau<TensBH> & depsBH) = 0;
205
206 // calcul du potentiel et de ses dérivées premières
207 virtual void Potentiel(double& jacobien_0, double& Ieps,double& V,double& bIib,
208                      double& E,double& EV,double& EbIib,double& EIeps) = 0;
209 // calcul du potentiel et de ses dérivées premières et secondes
210 virtual void Potentiel_et_var(double& jacobien_0,double& Ieps,double& V,double& bIib,
211                             double& E,double& EV,double& EbIib,double& EIeps ,
212                             double& EVV,double& EbIib2,double& EIeps2,
213                             double& EVbIib,double& EVIeps,double& EbIibIeps ) = 0;
214
215     protected :
216         // VARIABLES PROTEGEES :
217
218         // METHODES PROTEGEES :
219         // calcul des coefficients alpha
220         inline void Alpha (double& E,double& EV,double& EbIib,double& EIeps,
221                          double& jacobien_0,double& Ieps,double& V,double& bIib,
222                          double& alpha_0,double& alpha_1,double& alpha_2);
223
224
225         // calcul des coefficients alpha et de leurs variations par rapport aux degrés de libertés
226         inline void Alpha_var (double& E,double& EV,double& EbIib,double& EIeps,
227                               double& EVV,double& EbIib2,double& EIeps2,
228                               double& EVbIib,double& EVIeps,double& EbIibIeps,
229                               double& jacobien_0,double& Ieps,Tableau<double> & dIeps,double& V,Tableau<double> & dV,
230                               double& bIib,Tableau<double> & dbIib,
231                               double& alpha_0,Tableau<double> & dalpha_0,
232                               double& alpha_1,Tableau<double> & dalpha_1,
233                               double& alpha_2,Tableau<double> & dalpha_2);
234
235     };
236 /// @} // end of group
237
238 #include "HyperDN.cc"
239
240
241 #endif

```

## 7.42 IsoHyper3DFavier3.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *    DATE:      1/10/98
33 *
34 *    AUTEUR:    G RIO    (mailto:gerardrio56@free.fr)
35 *

```

```

35 *                                     $ *
36 *   PROJET:      Herezh++                                     $ *
37 *                                     $ *
38 *****
39 *   BUT: Concernant le potentiel hyperélastique proposé par denis *
40 *           favier, pour modéliser le comportement superélastique et *
41 *           ferroélastique des AMF.                                     $ *
42 *                                     $ *
43 *   ////////////////////////////////////////////////// *
44 *                                     *
45 *   VERIFICATION:                                             *
46 *   ! date !   auteur !           but                       ! *
47 *   -----                                             *
48 *   !           !           !                               ! *
49 *                                     $ *
50 *   ////////////////////////////////////////////////// *
51 *   MODIFICATIONS:                                           *
52 *   ! date !   auteur !           but                       ! *
53 *   -----                                             *
54 *                                     $ *
55 *****/
56 #ifndef ISOHYPER3DFAVIER3_H
57 #define ISOHYPER3DFAVIER3_H
58
59 #include "Hyper3D.h"
60
61
62 /// @addtogroup Les_lois_hyperelastiques
63 /// @{
64 ///
65
66
67 class IsoHyper3DFavier3 :
68 public Hyper3D
69 {
70 public :
71     // VARIABLES PUBLIQUES :
72
73     // CONSTRUCTEURS :
74     // Constructeur par défaut
75     IsoHyper3DFavier3 ();
76     // Constructeur de copie
77     IsoHyper3DFavier3 (const IsoHyper3DFavier3& loi) ;
78     // DESTRUCTEUR :
79     ~IsoHyper3DFavier3 ()
80     {};
81
82 // Lecture des donnees de la classe sur fichier
83 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
84     ,LesFonctions_nD& lesFonctionsnD);
85
86 // affichage de la loi
87 void Affiche() const ;
88 // test si la loi est complete
89 // = 1 tout est ok, =0 loi incomplete
90 int TestCompleter();
91
92
93 //----- lecture écriture de restart -----
94 // cas donne le niveau de la récupération
95 // = 1 : on récupère tout
96 // = 2 : on récupère uniquement les données variables (supposées comme telles)
97 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
98     ,LesFonctions_nD& lesFonctionsnD);
99 // cas donne le niveau de sauvegarde
100 // = 1 : on sauvegarde tout
101 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
102 void Ecriture_base_info_loi(ofstream& sort,const int cas);
103
104 // calcul d'un module d'young équivalent à la loi, ceci pour un
105 // chargement nul
106 double Module_young_equivalent(Enum_dure ,const Deformation & ,SaveResul * )
107     { return (9.*K*(mur+mu_inf)/((mur+mu_inf)+3.*K));
108     };
109 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
110 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
111 double Module_compressibilite_equivalent(Enum_dure ,const Deformation & def)
112     { return K/3.;};
113
114 // récupération de la variation relative d'épaisseur calculée: h/h0
115 // cette variation n'est utile que pour des lois en contraintes planes
116 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
117 // - pour les lois 2D def planes: retour de 0
118 // les infos nécessaires à la récupération , sont stockées dans saveResul
119 // qui est le conteneur spécifique au point où a été calculé la loi
120 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};

```

```

121
122 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
123 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new IsoHyper3DFavier3(*this)); };
124
125 // affichage et definition interactive des commandes particulières à chaque lois
126 void Info_commande_LoisDeComp(UtilLecture& lec);
127
128 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
129 // méthodes virtuelles de Hyper3D
130
131
132 // calcul du potentiel tout seul sans la phase car Qeps est nul
133 // ou très proche de 0
134 double PoGrenoble
135     (const double & Qeps,const Invariant & invariant);
136 // calcul du potentiel tout seul avec la phase donc dans le cas où Qeps est non nul
137 double PoGrenoble
138     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
139 // calcul du potentiel tout seul sans la phase car Qeps est nul
140 // ou très proche de 0, et de sa variation suivant V uniquement
141 PoGrenoble_V PoGrenoble_et_V
142     (const double & Qeps,const Invariant & invariant);
143 // calcul du potentiel et de sa variation suivant V uniquement
144 PoGrenoble_V PoGrenoble_et_V
145     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
146 // calcul du potentiel tout seul sans la phase car Qeps est nul
147 // ou très proche de 0, et de ses variations première et seconde suivant V uniquement
148 PoGrenoble_VV PoGrenoble_et_VV
149     (const double & Qeps,const Invariant & invariant);
150 // calcul du potentiel et de sa variation première et seconde suivant V uniquement
151 PoGrenoble_VV PoGrenoble_et_VV
152     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
153 // calcul du potentiel et de ses dérivées non compris la phase
154 PoGrenobleSansPhaseSansVar PoGrenoble
155     (const InvariantQeps & inv,const Invariant & invariant);
156 // calcul du potentiel et de ses dérivées avec la phase
157 PoGrenobleAvecPhaseSansVar PoGrenoblePhase
158     (const InvariantQepsCosphi& inv,const Invariant & invariant);
159 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux invariants
160 PoGrenobleSansPhaseAvecVar PoGrenoble_et_var
161     (const Invariant2Qeps& inv,const Invariant & invariant);
162 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux invariants
163 PoGrenobleAvecPhaseAvecVar PoGrenoblePhase_et_var
164     (const Invariant2QepsCosphi& inv,const Invariant & invariant);
165
166
167 protected :
168
169 // VARIABLES PROTEGEES :
170 double K; // module de dilatation
171 double Qor; // seuil
172 double mur; // pente à l'origine
173 double mu_inf; // pente à l'infini
174 double nQor,gammaQor; // paramètres de phase pour Qor
175 double n_mu_inf,gamma_mu_inf; // paramètre de phase pour mu_inf
176
177 static double limite_co2; // limite à partir de laquelle on considère que cosh(co2) = infini
178
179 //===== fonctions pour la vérification et la mise au point =====
180
181 // vérif des dérivées du potentiels par rapport aux invariants, ceci par différences finies
182 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv
183     ,const PoGrenobleSansPhaseAvecVar& potret );
184 // idem mais avec la phase
185 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv,const double& cos3phi
186     ,const PoGrenobleAvecPhaseAvecVar& potret );
187
188 static int indic_Verif_PoGrenoble_et_var; // indicateur utilisé par Verif_Potentiel_et_var
189
190 };
191 /// @} // end of group
192
193 #endif

```

## 7.43 IsoHyper3DOrgeas1.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.

```

```

9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30 //
31 /*****
32 *   DATE:           1/10/98
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 * *****/
39 *   BUT: Concernant le potentiel hyperélastique proposé par laurent
40 *         orgeas, pour modéliser le comportement superélastique et
41 *         ferroélastique des AMF.
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *   ! date !   auteur !           but
47 *   -----
48 *   !           !           !
49 *
50 *   *****
51 *   MODIFICATIONS:
52 *   ! date !   auteur !           but
53 *   -----
54 *
55 *****/
56 #ifndef ISOHYPER3DORGEAS1_H
57 #define ISOHYPER3DORGEAS1_H
58
59 #include "Hyper3D.h"
60
61
62
63 /// @addtogroup Les_lois_hyperelastiques
64 /// @{
65 ///
66
67 class IsoHyper3DOrgeas1 : public Hyper3D
68 {
69 public :
70     // VARIABLES PUBLIQUES :
71
72     // CONSTRUCTEURS :
73     // Constructeur par défaut
74     IsoHyper3DOrgeas1 ();
75     // Constructeur de copie
76     IsoHyper3DOrgeas1 (const IsoHyper3DOrgeas1& loi) ;
77     // DESTRUCTEUR :
78     ~IsoHyper3DOrgeas1 ();
79
80     // Lecture des donnees de la classe sur fichier
81     void LectureDonneesParticulieres
82         (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD& lesFonctionsnD);
83
84     // affichage de la loi
85     void Affiche() const ;
86     // test si la loi est complete
87     // = 1 tout est ok, =0 loi incomplete
88     int TestComplet();
89
90
91     //----- lecture écriture de restart -----
92     // cas donne le niveau de la récupération
93     // = 1 : on récupère tout
94     // = 2 : on récupère uniquement les données variables (supposées comme telles)

```

```

95 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbesID& lesCourbesID
96                                     ,LesFonctions_nD& lesFonctionsnD);
97 // cas donne le niveau de sauvegarde
98 // = 1 : on sauvegarde tout
99 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
100 void Ecriture_base_info_loi(ofstream& sort,const int cas);
101
102 // calcul d'un module d'young équivalent à la loi, ceci pour un
103 // chargement nul
104 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * );
105 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
106 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
107 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def);
108
109 // récupération de la variation relative d'épaisseur calculée: h/h0
110 // cette variation n'est utile que pour des lois en contraintes planes
111 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
112 // - pour les lois 2D def planes: retour de 0
113 // les infos nécessaires à la récupération , sont stockées dans saveResul
114 // qui est le conteneur spécifique au point où a été calculé la loi
115 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
116
117 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
118 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new IsoHyper3DOrgeas1(*this)); };
119
120 // affichage et definition interactive des commandes particulières à chaque lois
121 void Info_commande_LoisDeComp(UtilLecture& lec);
122
123 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
124 // méthodes virtuelles de Hyper3D
125
126
127 // calcul du potentiel tout seul sans la phase car Qeps est nul
128 // ou très proche de 0
129 double PoGrenoble
130     (const double & Qeps,const Invariant & invariant);
131 // calcul du potentiel tout seul avec la phase donc dans le cas où Qeps est non nul
132 double PoGrenoble
133     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
134 // calcul du potentiel tout seul sans la phase car Qeps est nul
135 // ou très proche de 0, et de sa variation suivant V uniquement
136 PoGrenoble_V PoGrenoble_et_V
137     (const double & Qeps,const Invariant & invariant);
138 // calcul du potentiel et de sa variation suivant V uniquement
139 PoGrenoble_V PoGrenoble_et_V
140     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
141 // calcul du potentiel tout seul sans la phase car Qeps est nul
142 // ou très proche de 0, et de ses variations première et seconde suivant V uniquement
143 PoGrenoble_VV PoGrenoble_et_VV
144     (const double & Qeps,const Invariant & invariant);
145 // calcul du potentiel et de sa variation première et seconde suivant V uniquement
146 PoGrenoble_VV PoGrenoble_et_VV
147     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
148 // calcul du potentiel et de ses dérivées non compris la phase
149 PoGrenobleSansPhaseSansVar PoGrenoble
150     (const InvariantQeps & inv,const Invariant & invariant);
151 // calcul du potentiel et de ses dérivées avec la phase
152 PoGrenobleAvecPhaseSansVar PoGrenoblePhase
153     (const InvariantQepsCosphi& inv,const Invariant & invariant);
154 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux invariants
155 PoGrenobleSansPhaseAvecVar PoGrenoble_et_var
156     (const Invariant2Qeps& inv,const Invariant & invariant);
157 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux invariants
158 PoGrenobleAvecPhaseAvecVar PoGrenoblePhase_et_var
159     (const Invariant2QepsCosphi& inv,const Invariant & invariant);
160
161
162 protected :
163
164 // VARIABLES PROTEGEES :
165 double K; // module de dilatation
166 double Q0s;
167 double mu01,mu02,mu03; // les 3 pentes
168 double alpha1,alpha3; // paramètres réglants les courbures
169 double Q0e;
170
171 //variables utilisées dans le cas d'une dépendance à la phase
172 double nQs,gammaQs; // pour Qs
173 double nQe,gammaQe; // pour Qe
174 double nMu1,gammaMu1; // pour mu1
175 double nMu2,gammaMu2; // pour mu2
176 double nMu3,gammaMu3; // pour mu3
177 // variables intermédiaires fixes pour optimiser le calcul
178 double alpha1_2;
179 double alpha3_2;
180 // double mu1_2;
181 // cas d'une thermo-dépendance de Q0s

```



```

182 int    cas_Q0s_thermo_dependant; // =0 pas de thermo-dépendances
183     // =1 : thermo-dépendance du type de celle donnée par Laurent dans sa thèse
184     //     page 154
185     // =2 : thermo-dépendance via une fonction de charge classique
186 double T0r,Gr,Qrmax,ar; // coeff pour la dépendance pour Qs: cf page 54 thèse Laurent
187 int    cas_Q0e_thermo_dependant; // =0 pas de thermo-dépendances
188     // =1 : thermo-dépendance fonction de la pente mu3+mu2 -> Qe(T) = Qe(T0rQe) +
189     //     (Qs(T)-Qs(T0rQe))/(h1*3(mu3+mu2))
190     // =2 : thermo-dépendance via une fonction de charge classique
191 double T0rQe,Qe_T0rQe,h1,Qs_T0rQe; // coeff dans le cas cas_Q0e_thermo_dependant = 1
192 double Tc,Ts; // grandeurs intermédiaires qui sont fixes
193 CourbelD* Q0s_temperature; // courbe éventuelle d'évolution de Q0s
194 CourbelD* Q0e_temperature; // courbe éventuelle d'évolution de Q0e
195
196
197
198 ///===== fonctions pour la vérification et la mise au point =====
199
200     // vérif des dérivées du potentiels par rapport aux invariants, ceci par différences finies
201 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv
202     ,const PoGrenobleSansPhaseAvecVar& potret );
203     // idem mais avec la phase
204 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv,const double& cos3phi
205     ,const PoGrenobleAvecPhaseAvecVar& potret );
206     // vérif des dérivées du potentiels par rapport à l'invariants V, ceci par différences finies
207 void Verif_PoGrenoble_et_var_VV(const double & Qeps,const Invariant & inv
208     ,const PoGrenoble_VV& potret );
209     // utilitaire de vérif pour comparaison avec des calculs effectués avec un programme
210     // de calcul formel: par exemple maxima
211 void CompaFormel();
212
213 static int indic_Verif_PoGrenobleorgeas1_et_var; // indicateur utilisé par Verif_Potentiel_et_var
214
215 static int indic_Verif_PoGrenobleorgeas1_et_var_VV; // indicateur utilisé par
216     Verif_Potentiel_et_var_VV
217 static int indic_Verif_CompFormel; // indicateur utilisé par CompaFormel
218
219 };
220 /// @} // end of group
221
222 #endif

```

## 7.44 IsoHyper3DOrgeas2.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *    DATE:      1/10/98
33 *
34 *    AUTEUR:    G RIO    (mailto:gerardrio56@free.fr)
35 *
36 *    PROJET:    Herezh++
37 *
38 *
39 *    BUT:       Concernant le potentiel hyperélastique proposé par laurent
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```

```

41 *      ferroélastique des AMF. *
42 *      Idem que IsoHyper3DOrgeas1 mais avec la possibilité *
43 *      d'utiliser des lois fonction de la phase quelconque. *
44 *      on a ici toujours la phase, sinon il faut utiliser le potentiel *
45 *      IsoHyper3DOrgeas1. *
46 *      * * * * * $ *
47 *      ***** *
48 *      VERIFICATION: *
49 *      * * * * * *
50 *      ! date ! auteur ! but ! *
51 *      ----- *
52 *      ! ! ! ! *
53 *      * * * * * $ *
54 *      ***** *
55 *      MODIFICATIONS: *
56 *      ! date ! auteur ! but ! *
57 *      ----- *
58 *      * * * * * $ *
59 *****/
60 #ifndef ISOHYPER3DORGEAS2_H
61 #define ISOHYPER3DORGEAS2_H
62
63 #include "Hyper3D.h"
64
65
66 /// @addtogroup Les_lois_hyperelastiques
67 /// @{
68 ///
69
70
71 class IsoHyper3DOrgeas2 :
72 public Hyper3D
73 {
74 public :
75 // VARIABLES PUBLIQUES :
76
77 // CONSTRUCTEURS :
78 // Constructeur par défaut
79 IsoHyper3DOrgeas2 ();
80 // Constructeur de copie
81 IsoHyper3DOrgeas2 (const IsoHyper3DOrgeas2& loi) ;
82 // DESTRUCTEUR :
83 ~IsoHyper3DOrgeas2 ();
84
85 // Lecture des donnees de la classe sur fichier
86 void LectureDonneesParticulieres
87 (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD& lesFonctionsnD);
88
89 // affichage de la loi
90 void Affiche() const ;
91 // test si la loi est complete
92 // = 1 tout est ok, =0 loi incomplete
93 int TestComplet();
94
95
96 //----- lecture écriture de restart -----
97 // cas donne le niveau de la récupération
98 // = 1 : on récupère tout
99 // = 2 : on récupère uniquement les données variables (supposées comme telles)
100 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
,LesFonctions_nD& lesFonctionsnD);
101
102 // cas donne le niveau de sauvegarde
103 // = 1 : on sauvegarde tout
104 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
105 void Ecriture_base_info_loi(ofstream& sort,const int cas);
106
107 // calcul d'un module d'young équivalent à la loi, ceci pour un
108 // chargement nul
109 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
110 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
111 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
112 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
saveResul);
113
114 // récupération de la variation relative d'épaisseur calculée: h/h0
115 // cette variation n'est utile que pour des lois en contraintes planes
116 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
117 // - pour les lois 2D def planes: retour de 0
118 // les infos nécessaires à la récupération , sont stockées dans saveResul
119 // qui est le conteneur spécifique au point où a été calculé la loi
120 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
121
122 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
123 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new IsoHyper3DOrgeas2(*this)); };
124

```

```

125 // affichage et definition interactive des commandes particulières à chaque lois
126 void Info_commande_LoisDeComp(UtilLecture& lec);
127
128 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
129 // méthodes virtuelles de Hyper3D
130
131
132 // calcul du potentiel tout seul sans la phase car Qeps est nul
133 // ou très proche de 0
134 double PoGrenoble
135     (const double & Qeps,const Invariant & invariant);
136 // calcul du potentiel tout seul avec la phase donc dans le cas où Qeps est non nul
137 double PoGrenoble
138     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
139 // calcul du potentiel tout seul sans la phase car Qeps est nul
140 // ou très proche de 0, et de sa variation suivant V uniquement
141 PoGrenoble_V PoGrenoble_et_V
142     (const double & Qeps,const Invariant & invariant);
143 // calcul du potentiel et de sa variation suivant V uniquement
144 PoGrenoble_V PoGrenoble_et_V
145     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
146 // calcul du potentiel tout seul sans la phase car Qeps est nul
147 // ou très proche de 0, et de ses variations première et seconde suivant V uniquement
148 PoGrenoble_VV PoGrenoble_et_VV
149     (const double & Qeps,const Invariant & invariant);
150 // calcul du potentiel et de sa variation première et seconde suivant V uniquement
151 PoGrenoble_VV PoGrenoble_et_VV
152     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
153 // calcul du potentiel et de ses dérivées non compris la phase
154 PoGrenobleSansPhaseSansVar PoGrenoble
155     (const InvariantQeps & inv,const Invariant & invariant);
156 // calcul du potentiel et de ses dérivées avec la phase
157 PoGrenobleAvecPhaseSansVar PoGrenoblePhase
158     (const InvariantQepsCosphi& inv,const Invariant & invariant);
159 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux invariants
160 PoGrenobleSansPhaseAvecVar PoGrenoble_et_var
161     (const Invariant2Qeps& inv,const Invariant & invariant);
162 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux invariants
163 PoGrenobleAvecPhaseAvecVar PoGrenoblePhase_et_var
164     (const Invariant2QepsCosphi& inv,const Invariant & invariant);
165
166
167 protected :
168
169 // VARIABLES PROTEGEES :
170 double K; // module de dilatation
171 double Q0s;
172 double mu01,mu02,mu03; // les 3 pentes
173 double alpha1,alpha3; // paramètres réglants les courbures
174 double Q0e;
175
176 //variables utilisées dans le cas d'une dépendance à la phase
177 CourbelD* mu01_phi, * mu02_phi, *mu03_phi;
178 CourbelD* Q0s_phi, * Q0e_phi;
179 // cas d'une dépendance à une fonction nD
180 Fonction_nD* mu01_nD, * mu02_nD, *mu03_nD;
181 Fonction_nD* Q0s_nD, * Q0e_nD;
182 // variables intermédiaires fixes pour optimiser le calcul
183 double alpha1_2;
184 double alpha3_2;
185 // cas d'une thermo-dépendance de Q0s
186 int cas_Q0s_thermo_dependant; // =0 pas de thermo-dépendances
187 // =1 : thermo-dépendance du type de celle donnée par Laurent dans sa thèse
188 // page 154
189 // =2 : thermo-dépendance via une fonction de charge classique
190 double T0r,Gr,Qrmax,ar; // coeff pour la dépendance pour Qs: cf page 54 thèse Laurent
191 int cas_Q0e_thermo_dependant; // =0 pas de thermo-dépendances
192 // =1 : thermo-dépendance fonction de la pente mu3+mu2 -> Qe(T) = Qe(T0r) +
193 // (Qs(T)-Qs(T0r))/(3.*(mu3+mu2))
194 // =2 : thermo-dépendance via une fonction de charge classique
194 double T0rQe,Qe_T0rQe,h1,Qs_T0rQe; // coeff dans le cas cas_Q0e_thermo_dependant = 1
195 double Tc,Ts; // grandeurs intermédiaires qui sont fixes
196 CourbelD* Q0s_temperature; // courbe éventuelle d'évolution de Q0s
197 CourbelD* Q0e_temperature; // courbe éventuelle d'évolution de Q0e
198
199
200
201
202 ///===== fonctions pour la vérification et la mise au point =====
203
204 // vérif des dérivées du potentiels par rapport aux invariants, ceci par différences finies
205 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv
206     ,const PoGrenobleSansPhaseAvecVar& potret );
207 // idem mais avec la phase
208 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv,const double& cos3phi
209     ,const PoGrenobleAvecPhaseAvecVar& potret );
210 // vérif des dérivées du potentiels par rapport à l'invariants V, ceci par différences finies

```

```

211 void Verif_PoGrenoble_et_var_VV(const double & Qeps, const Invariant & inv
212     , const PoGrenoble_VV & potret );
213     // utilitaire de vérif pour comparaison avec des calculs effectués avec un programme
214     // de calcul formel: par exemple maxima
215 void CompaFormel();
216
217 static int indic_Verif_PoGrenobleorgeas2_et_var; // indicateur utilisé par Verif_Potentiel_et_var
218
219 static int indic_Verif_PoGrenobleorgeas2_et_var_VV; // indicateur utilisé par Verif_Potentiel_et_var_VV
220
221 static int indic_Verif_CompFormel; // indicateur utilisé par CompaFormel
222 };
223 /// @} // end of group
224 #endif

```

## 7.45 IsoHyperBulk3.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:          6/04/2008
33 *
34 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:        Herezh++
37 *
38 *      ****
39 *      BUT: Concernant un potentiel hyperélastique composé uniquement
40 *      d'un bulk, avec une écriture comme les potentiels grenoblois.
41 *      E = K/6 * (log(V))**2
42 *
43 *      *****
44 *
45 *      VERIFICATION:
46 *      ! date ! auteur ! but
47 *      -----
48 *      ! ! !
49 *      $
50 *      *****
51 *      MODIFICATIONS:
52 *      ! date ! auteur ! but
53 *      -----
54 *      $
55 *      *****/
56 #ifndef ISOHYPERBULK3_H
57 #define ISOHYPERBULK3_H
58
59 #include "Hyper3D.h"
60
61
62 /// @addtogroup Les_lois_hyperelastiques
63 /// @{
64 ///
65

```

```

66
67 class IsoHyperBulk3 :
68     public Hyper3D
69 {
70     public :
71         // VARIABLES PUBLIQUES :
72
73     // CONSTRUCTEURS :
74     // Constructeur par défaut
75     IsoHyperBulk3 ();
76     // Constructeur de copie
77     IsoHyperBulk3 (const IsoHyperBulk3& loi) ;
78     // DESTRUCTEUR :
79     ~IsoHyperBulk3 ();
80
81     // Lecture des donnees de la classe sur fichier
82     void LectureDonneesParticulieres
83         (UtilLecture * , LesCourbes1D& lesCourbes1D, LesFonctions_nD& lesFonctionsnD);
84
85     // affichage de la loi
86     void Affiche() const ;
87     // test si la loi est complete
88     // = 1 tout est ok, =0 loi incomplete
89     int TestComplet();
90
91
92 //----- lecture écriture de restart -----
93 // cas donne le niveau de la récupération
94 // = 1 : on récupère tout
95 // = 2 : on récupère uniquement les données variables (supposées comme telles)
96 void Lecture_base_info_loi(ifstream& ent, const int cas, LesReferences& lesRef, LesCourbes1D& lesCourbes1D
97     , LesFonctions_nD& lesFonctionsnD);
98 // cas donne le niveau de sauvegarde
99 // = 1 : on sauvegarde tout
100 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
101 void Ecriture_base_info_loi(ofstream& sort, const int cas);
102
103 // calcul d'un module d'young équivalent à la loi, ceci pour un
104 // chargement nul
105 double Module_young_equivalent(Enum_dure temps, const Deformation & , SaveResul * saveResul);
106
107 // récupération de la variation relative d'épaisseur calculée: h/h0
108 // cette variation n'est utile que pour des lois en contraintes planes
109 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
110 // - pour les lois 2D def planes: retour de 0
111 // les infos nécessaires à la récupération , sont stockées dans saveResul
112 // qui est le conteneur spécifique au point où a été calculé la loi
113 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
114
115 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
116 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
117 double Module_compressibilite_equivalent(Enum_dure temps, const Deformation & def, SaveResul *
118     saveResul);
119
120 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
121 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new IsoHyperBulk3(*this)); };
122
123 // affichage et definition interactive des commandes particulières à chaque lois
124 void Info_commande_LoisDeComp(UtilLecture& lec);
125
126 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
127 // méthodes virtuelles de Hyper3D
128
129 // calcul du potentiel tout seul sans la phase car Qeps est nul
130 // ou très proche de 0
131 double PoGrenoble
132     (const double & Qeps, const Invariant & invariant);
133 // calcul du potentiel tout seul avec la phase donc dans le cas où Qeps est non nul
134 double PoGrenoble
135     (const Invariant0QepsCosphi & inv, const Invariant & invariant);
136 // calcul du potentiel tout seul sans la phase car Qeps est nul
137 // ou très proche de 0, et de sa variation suivant V uniquement
138 PoGrenoble_V PoGrenoble_et_V
139     (const double & Qeps, const Invariant & invariant);
140 // calcul du potentiel et de sa variation suivant V uniquement
141 PoGrenoble_V PoGrenoble_et_V
142     (const Invariant0QepsCosphi & inv, const Invariant & invariant);
143 // calcul du potentiel tout seul sans la phase car Qeps est nul
144 // ou très proche de 0, et de ses variations première et seconde suivant V uniquement
145 PoGrenoble_VV PoGrenoble_et_VV
146     (const double & Qeps, const Invariant & invariant);
147 // calcul du potentiel et de sa variation première et seconde suivant V uniquement
148 PoGrenoble_VV PoGrenoble_et_VV
149     (const Invariant0QepsCosphi & inv, const Invariant & invariant);
150 // calcul du potentiel et de ses dérivées non compris la phase
151 PoGrenobleSansPhaseSansVar PoGrenoble

```

```

152         (const InvariantQeps & inv,const Invariant & invariant);
153 // calcul du potentiel et de ses dérivées avec la phase
154 PoGrenobleAvecPhaseSansVar PoGrenoblePhase
155         (const InvariantQepsCosphi & inv,const Invariant & invariant);
156 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux invariants
157 PoGrenobleSansPhaseAvecVar PoGrenoble_et_var
158         (const Invariant2Qeps & inv,const Invariant & invariant);
159 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux invariants
160 PoGrenobleAvecPhaseAvecVar PoGrenoblePhase_et_var
161         (const Invariant2QepsCosphi & inv,const Invariant & invariant);
162
163
164 protected :
165
166 // VARIABLES PROTEGEES :
167 double K; // module de dilatation
168 //fonction de dépendance éventuelle
169 CourbelD* F_K_V, * F_K_T; // dépendance à V, dépendance à la température
170
171 //===== fonctions pour la vérification et la mise au point =====
172
173 // vérif des dérivées du potentiels par rapport aux invariants, ceci par différences finies
174 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv
175         ,const PoGrenobleSansPhaseAvecVar & potret );
176 // idem mais avec la phase
177 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv,const double & cos3phi
178         ,const PoGrenobleAvecPhaseAvecVar & potret );
179
180 static int indic_Verif_PoGrenoble_et_var; // indicateur utilisé par Verif_Potentiel_et_var
181
182 };
183 /// @} // end of group
184
185 #endif

```

## 7.46 IsoHyperBulk\_gene.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           6/04/2008
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 *****/
39 *   BUT: Concernant un potentiel hyperélastique composé uniquement
40 *   d'une fonction bulk, avec une écriture comme les potentiels .
41 *   grenoblois E = K(V)
42 *
43 *   *****
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !           but
47 *   -----

```

```

48 *      !      !      !      !      *
49 *      !      !      !      !      $      *
50 *      !      !      !      !      *
51 *      MODIFICATIONS:      *
52 *      ! date ! auteur ! but      !      *
53 *      !      !      !      !      *
54 *      !      !      !      !      $      *
55 *      !      !      !      !      *
56 #ifndef ISOHYPERBULK_GENE_H
57 #define ISOHYPERBULK_GENE_H
58
59 #include "Hyper3D.h"
60
61
62
63 /// @addtogroup Les_lois_hyperelastiques
64 /// @{
65 ///
66
67 class IsoHyperBulk_gene :
68 public Hyper3D
69 {
70 public :
71 // VARIABLES PUBLIQUES :
72
73 // CONSTRUCTEURS :
74 // Constructeur par défaut
75 IsoHyperBulk_gene ();
76 // Constructeur de copie
77 IsoHyperBulk_gene (const IsoHyperBulk_gene& loi) ;
78 // DESTRUCTEUR :
79 ~IsoHyperBulk_gene ();
80
81 // Lecture des donnees de la classe sur fichier
82 void LectureDonneesParticulieres
83 (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD& lesFonctionsnD);
84
85 // affichage de la loi
86 void Affiche() const ;
87 // test si la loi est complete
88 // = 1 tout est ok, =0 loi incomplete
89 int TestComplet();
90
91
92 //----- lecture écriture de restart -----
93 // cas donne le niveau de la récupération
94 // = 1 : on récupère tout
95 // = 2 : on récupère uniquement les données variables (supposées comme telles)
96 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
97 ,LesFonctions_nD& lesFonctionsnD);
98 // cas donne le niveau de sauvegarde
99 // = 1 : on sauvegarde tout
100 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
101 void Ecriture_base_info_loi(ofstream& sort,const int cas);
102
103 // calcul d'un module d'young équivalent à la loi, ceci pour un
104 // chargement nul
105 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
106
107 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
108 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
109 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
110 saveResul);
111 // récupération de la variation relative d'épaisseur calculée: h/h0
112 // cette variation n'est utile que pour des lois en contraintes planes
113 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
114 // - pour les lois 2D def planes: retour de 0
115 // les infos nécessaires à la récupération , sont stockées dans saveResul
116 // qui est le conteneur spécifique au point où a été calculé la loi
117 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
118
119 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
120 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new IsoHyperBulk_gene(*this)); };
121
122 // affichage et definition interactive des commandes particulières à chaque lois
123 void Info_commande_LoisDeComp(UtilLecture& lec);
124
125 // METHODES internes spécifiques à l'hyperélasticité isotrope découlant de
126 // méthodes virtuelles de Hyper3D
127
128
129 // calcul du potentiel tout seul sans la phase car Qeps est nul
130 // ou très proche de 0
131 double PoGrenoble
132 (const double & Qeps,const Invariant & invariant);
133 // calcul du potentiel tout seul avec la phase donc dans le cas où Qeps est non nul

```

```

134 double PoGrenoble
135     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
136 // calcul du potentiel tout seul sans la phase car Qeps est nul
137 // ou très proche de 0, et de sa variation suivant V uniquement
138 PoGrenoble_V PoGrenoble_et_V
139     (const double & Qeps,const Invariant & invariant);
140 // calcul du potentiel et de sa variation suivant V uniquement
141 PoGrenoble_V PoGrenoble_et_V
142     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
143 // calcul du potentiel tout seul sans la phase car Qeps est nul
144 // ou très proche de 0, et de ses variations première et seconde suivant V uniquement
145 PoGrenoble_VV PoGrenoble_et_VV
146     (const double & Qeps,const Invariant & invariant);
147 // calcul du potentiel et de sa variation première et seconde suivant V uniquement
148 PoGrenoble_VV PoGrenoble_et_VV
149     (const Invariant0QepsCosphi & inv,const Invariant & invariant);
150 // calcul du potentiel et de ses dérivées non compris la phase
151 PoGrenobleSansPhaseSansVar PoGrenoble
152     (const InvariantQeps & inv,const Invariant & invariant);
153 // calcul du potentiel et de ses dérivées avec la phase
154 PoGrenobleAvecPhaseSansVar PoGrenoblePhase
155     (const InvariantQepsCosphi & inv,const Invariant & invariant);
156 // calcul du potentiel sans phase et dérivées avec ses variations par rapport aux invariants
157 PoGrenobleSansPhaseAvecVar PoGrenoble_et_var
158     (const Invariant2Qeps & inv,const Invariant & invariant);
159 // calcul du potentiel avec phase et dérivées avec ses variations par rapport aux invariants
160 PoGrenobleAvecPhaseAvecVar PoGrenoblePhase_et_var
161     (const Invariant2QepsCosphi & inv,const Invariant & invariant);
162
163
164 protected :
165
166 // VARIABLES PROTEGEES :
167 //fonction de dépendance éventuelle
168 CourbelD* F_w_V, * F_w_T; // dépendance à V, dépendance à la température
169
170 ///===== fonctions pour la vérification et la mise au point =====
171
172 // vérif des dérivées du potentiels par rapport aux invariants, ceci par différences finies
173 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv
174     ,const PoGrenobleSansPhaseAvecVar& potret );
175 // idem mais avec la phase
176 void Verif_PoGrenoble_et_var(const double & Qeps,const Invariant & inv,const double& cos3phi
177     ,const PoGrenobleAvecPhaseAvecVar& potret );
178
179 static int indic_Verif_PoGrenoble_et_var; // indicateur utilisé par Verif_Potentiel_et_var
180
181 };
182 /// @} // end of group
183
184 #endif

```

## 7.47 Maheo\_hyper.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30

```



```

31 /*****
32 *   DATE:      26/01/2015
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++
37 *
38 *   *****/
39 *   BUT:      La classe Maheo_hyper permet de calculer la contrainte
40 *            et ses derivees pour une loi isotrope hyper élastique
41 *            de type extension de la loi de Mooney Rivlin en 3D
42 *            sous forme d'un log cosh de l'invariant I1
43 *            de Mooney Rivlin. La loi est purement déviatorique.
44 *            Il s'agit d'une classe derivatee de la classe Loi_comp_abstraite.
45 *
46 *            *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date ! auteur ! but
51 *   -----
52 *   ! ! !
53 *   *****
54 *   MODIFICATIONS:
55 *   ! date ! auteur ! but
56 *   -----
57 *   $
58 *****/
59 #ifndef MAHEO_HYPER_H
60 #define MAHEO_HYPER_H
61
62
63
64
65 #include "Loi_comp_abstraite.h"
66 #include "Courbe1D.h"
67 #include "MathUtil.h"
68 #include "Hyper_W_gene_3D.h"
69
70
71 /// @addtogroup Les_lois_hyperelastiques
72 /// @{
73 ///
74
75
76 class Maheo_hyper : public Hyper_W_gene_3D
77 {
78
79
80 public :
81
82
83 // CONSTRUCTEURS :
84
85 // Constructeur par défaut
86 Maheo_hyper ();
87
88
89 // Constructeur de copie
90 Maheo_hyper (const Maheo_hyper& loi) ;
91
92 // DESTRUCTEUR :
93
94 ~Maheo_hyper ();
95
96 // Lecture des donnees de la classe sur fichier
97 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
98 ,LesFonctions_nD& lesFonctionsnD);
99
100 // affichage de la loi
101 void Affiche() const ;
102 // test si la loi est complete
103 // = 1 tout est ok, =0 loi incomplete
104 int TestComplet();
105
106 //----- lecture écriture de restart -----
107 // cas donne le niveau de la récupération
108 // = 1 : on récupère tout
109 // = 2 : on récupère uniquement les données variables (supposées comme telles)
110 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
111 lesCourbes1D ,LesFonctions_nD& lesFonctionsnD);
112 // cas donne le niveau de sauvegarde
113 // = 1 : on sauvegarde tout
114 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
115 void Ecriture_base_info_loi(ofstream& sort,const int cas);

```

```

116 // calcul d'un module d'young equivalent à la loi,
117 // c'est sans doute complètement débile mais c'est pour pouvoir avancer !!
118 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
119
120 // récupération de la variation relative d'épaisseur calculée: h/h0
121 // cette variation n'est utile que pour des lois en contraintes planes
122 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
123 // - pour les lois 2D def planes: retour de 0
124 // les infos nécessaires à la récupération , sont stockées dans saveResul
125 // qui est le conteneur spécifique au point où a été calculé la loi
126 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
127
128 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
129 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Maheo_hyper(*this)); };
130
131 // affichage et definition interactive des commandes particulières à chaque lois
132 void Info_commande_LoisDeComp(UtilLecture& lec);
133
134 protected :
135 // données de la loi
136
137 double Qsig_rev, mu1_rev,mu2_rev;
138 CourbelD* Qsig_rev_temperature; // courbe éventuelle d'évolution de Qsig_rev en fonction de la
température
139 CourbelD* mu1_rev_temperature; // courbe éventuelle d'évolution de mu1_rev en fonction de la
température
140 CourbelD* mu2_rev_temperature; // courbe éventuelle d'évolution de mu2_rev en fonction de la
température
141 double fact_regularisation; // spécifie le facteur de régularisation
142
143 double W_d,W_v; // le potentiel: partie déviatorique, partie sphérique
144 Vecteur W_r; // dérivées premières du potentiel par rapport aux J_r
145 double W_d_J1,W_d_J2; // dérivées premières du potentiel déviatoire par rapport aux J_1 et J_2
146 double W_d_J1_2,W_d_J1_J2,W_d_J2_2; // dérivées secondes
147 double W_v_J3,W_v_J3J3; // dérivées premières et seconde du potentiel volumique / J3
148 Tableau2 <double> W_rs; // dérivées secondes du potentiel par rapport aux J_r
149
150
151 // --- codage des METHODES VIRTUELLES protegees:
152 // calcul des contraintes a t+dt
153 // calcul des contraintes
154 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
155 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB & giB,BaseH & gi_H, TenseurBB & epsBB_
156 ,TenseurBB & delta_epsBB_
157 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
158 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
159 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
160 ,const Met_abstraite::Expli_t_tdt& ex);
161
162 // calcul des contraintes et de ses variations a t+dt
163 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
164 ,BaseB & giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
165 ,BaseB & giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH & giH_tdt,Tableau <BaseH> & d_giH_tdt
166 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
167 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
168 ,Tableau <TenseurBB *>& d_gijBB_tdt
169 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
170 ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
171 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
172 ,const Met_abstraite::Impli& ex);
173
174 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
175 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
176 // le tenseur de déformation et son incrémentsont également en orthonormees
177 // si = false: les bases transmises sont utilisées
178 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
179 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
180 ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
181 ,TenseurHH& sigHH,TenseurHH& d_sigma_deps
182 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
183 ,const Met_abstraite::Umat_cont& ex) ; // = 0;
184
185
186 // fonction surchargée dans les classes dérivée si besoin est
187 virtual void CalculGrandeurTravail
188 (const PtIntegMecaInterne& ,const Deformation &
189 ,Enum_dure,const ThermoDonnee&
190 ,const Met_abstraite::Impli* ex_impli
191 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
192 ,const Met_abstraite::Umat_cont* ex_umat
193 ,const List_io<Ddl_etendu>* exclure_dd_etend
194 ,const List_io<const TypeQuelconque *>* exclure_Q
195 ) {};
196

```

```

197 private:
198 // calcul du potentiel et de ses dérivées premières / aux invariants J_r
199 void Potentiel_et_var();
200 // calcul du potentiel et de ses dérivées premières et secondes / aux invariants J_r
201 void Potentiel_et_var2();
202 // calcul de la dérivée numérique de la contrainte
203 void Cal_dsigma_deps_num (const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
204 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
205 ,const double& jacobien_0,const double& jacobien
206 ,Tenseur3HHHH& dSigdepsHHHH);
207 // calcul de la contrainte avec le minimum de variable de passage, utilisé pour le numérique
208 void Cal_sigma_pour_num(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
209 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
210 ,const double& jacobien_0,const double& jacobien,TenseurHH & sigHH_);
211 // idem avec la variation
212 void Cal_sigmaEtDer_pour_num(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
213 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
214 ,const double& jacobien_0,const double& jacobien
215 ,TenseurHH & sigHH_,Tenseur3HHHH& dSigdepsHHHH);
216
217
218 };
219 /// @} // end of group
220
221
222 #endif
223
224
225

```

## 7.48 MooneyRivlin1D.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:      26/4/2004
33 *
34 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:    Herezh++
37 *
38 *
39 *      BUT:      La classe MooneyRivlin1D permet de calculer la contrainte
40 *               et ses derivees pour une loi isotrope hyper élastique
41 *               de type Mooney Rivlin en 1D. Ici on ne considère pas
42 *               la variation de volume.
43 *               S contrainte de cauchy et e def d'almanisi
44 *               S = 2*C10*(1/(1-2*e) - racine(1-2*e)) + 2*C01*(1/racine(1-2*e)
45 *               - (1-2*e))
46 *               Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
47 *
48 *               *****
49 *
50 *      VERIFICATION:
51 *
52 *      ! date !   auteur !           but
53 *      -----

```

```

53 *      !      !      !      !      *
54 *      *      *      *      *      *
55 *      *      *      *      *      *
56 *      *      *      *      *      *
57 *      *      *      *      *      *
58 *      *      *      *      *      *
59 *      *      *      *      *      *
60 *      *      *      *      *      *
61 #ifndef MOONEY_RIVLIN_1D_H
62 #define MOONEY_RIVLIN_1D_H
63
64
65
66
67 #include "Loi_comp_abstraite.h"
68 #include "Courbe1D.h"
69 #include "MathUtil.h"
70
71 /// @addtogroup Les_lois_hyperelastiques
72 /// @{
73 ///
74
75
76
77 class MooneyRivlin1D : public Loi_comp_abstraite
78 {
79
80
81     public :
82
83
84     // CONSTRUCTEURS :
85
86     // Constructeur par défaut
87     MooneyRivlin1D ();
88
89
90     // Constructeur de copie
91     MooneyRivlin1D (const MooneyRivlin1D& loi) ;
92
93     // DESTRUCTEUR :
94
95     ~MooneyRivlin1D ();
96
97     // Lecture des donnees de la classe sur fichier
98     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D lesCourbes1D
99                                     ,LesFonctions_nD& lesFonctionsnD);
100
101     // affichage de la loi
102     void Affiche() const ;
103     // test si la loi est complete
104     // = 1 tout est ok, =0 loi incomplete
105     int TestComplet();
106
107     //----- lecture écriture de restart -----
108     // cas donne le niveau de la récupération
109     // = 1 : on récupère tout
110     // = 2 : on récupère uniquement les données variables (supposées comme telles)
111     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
112                               lesCourbes1D
113                               ,LesFonctions_nD& lesFonctionsnD);
114
115     // cas donne le niveau de sauvegarde
116     // = 1 : on sauvegarde tout
117     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
118     void Ecriture_base_info_loi(ofstream& sort,const int cas);
119
120     // calcul d'un module d'young équivalent à la loi,
121     // c'est sans doute complètement débile mais c'est pour pouvoir avancer !!
122     double Module_young_equivalent(Enum_dure ,const Deformation & ,SaveResul * ) {return 6.* C10;};
123
124     // récupération de la variation relative d'épaisseur calculée: h/h0
125     // cette variation n'est utile que pour des lois en contraintes planes
126     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
127     // - pour les lois 2D def planes: retour de 0
128     // les infos nécessaires à la récupération , sont stockées dans saveResul
129     // qui est le conteneur spécifique au point où a été calculé la loi
130     virtual double HsurH0(SaveResul * saveResul) const
131     { cout << "\n MooneyRivlin1D::HsurH0(.. , methode non implante pour l'instant ";
132       Sortie(1);return 0.;
133     };
134
135     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
136     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new MooneyRivlin1D(*this)); };
137
138     // affichage et definition interactive des commandes particulières à chaque lois
139     void Info_commande_LoisDeComp(UtilLecture& lec);
140
141     protected :

```

```

139 // donnée de la loi
140 double C10,C01; // deux coeff
141 CourbelD* C10_temperature; // courbe éventuelle d'évolution de C10 en fonction de la température
142 CourbelD* C01_temperature; // courbe éventuelle d'évolution de C01 en fonction de la température
143
144 // codage des METHODES VIRTUELLES protegees:
145 // calcul des contraintes a t+dt
146 // calcul des contraintes
147 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
148 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB & giB,BaseH & gi_H, TenseurBB & epsBB_
149 ,TenseurBB & delta_epsBB_
150 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
151 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
152 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
153 ,const Met_abstraite::Expli_t_tdt& ex);
154
155 // calcul des contraintes et de ses variations a t+dt
156 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
157 ,BaseB & giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
158 ,BaseB & giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH & giH_tdt,Tableau <BaseH> & d_giH_tdt
159 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
160 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
161 ,Tableau <TenseurBB *>& d_gijBB_tdt
162 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
163 ,Vecteur& d_jacobien_tdt,TenseurHH & sigHH,Tableau <TenseurHH *>& d_sigHH
164 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
165 ,const Met_abstraite::Impli& ex);
166
167
168 // fonction surchargée dans les classes dérivée si besoin est
169 virtual void CalculGrandeurTravail
170 (const PtIntegMecaInterne& ,const Deformation &
171 ,Enum_dure,const ThermoDonnee&
172 ,const Met_abstraite::Impli* ex_impli
173 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
174 ,const Met_abstraite::Umat_cont* ex_umat
175 ,const List_io<Ddl_etendu>* ex_lure_dd_etend
176 ,const List_io<const TypeQuelconque *>* ex_lure_Q
177 ) {};
178
179 };
180 /// @} // end of group
181
182
183 #endif
184
185
186

```

## 7.49 MooneyRivlin3D.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 * DATE: 30/5/2005 *

```

```

33 *
34 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)           $   *
35 *
36 *   PROJET:      Herezh++                                     $   *
37 *
38 * *****
39 *   BUT:   La classe MooneyRivlin3D permet de calculer la contrainte *
40 *         et ses derivees pour une loi isotrope hyper élastique *
41 *         de type Mooney Rivlin en 3D. Ici on considère *
42 *         la variation de volume. *
43 *         S contrainte de cauchy et e def d'almanzi *
44 *          $S = 2 * C10 * (1 / (1 - 2 * e) - \text{racine}(1 - 2 * e)) + 2 * C01 * (1 / \text{racine}(1 - 2 * e) - (1 - 2 * e))$  *
45 *
46 *   Il s'agit d'une classe derivee de la classe Loi_comp_abstraite. *
47 *
48 *   *****
49 *
50 *   VERIFICATION:
51 *
52 *   ! date ! auteur ! but !
53 *   -----
54 *   ! ! ! ! $
55 *   *****
56 *   MODIFICATIONS:
57 *   ! date ! auteur ! but !
58 *   -----
59 *   $
60 * *****/
61 #ifndef MOONEY_RIVLIN_3D_H
62 #define MOONEY_RIVLIN_3D_H
63
64
65
66
67 #include "Loi_comp_abstraite.h"
68 #include "CourbelD.h"
69 #include "MathUtil.h"
70 #include "Hyper_W_gene_3D.h"
71
72
73 /// @addtogroup Les_lois_hyperelastiques
74 /// @{
75 ///
76
77
78 class MooneyRivlin3D : public Hyper_W_gene_3D
79 {
80
81
82 public :
83
84
85 // CONSTRUCTEURS :
86
87 // Constructeur par défaut
88 MooneyRivlin3D ();
89
90
91 // Constructeur de copie
92 MooneyRivlin3D (const MooneyRivlin3D & loi) ;
93
94 // DESTRUCTEUR :
95
96 ~MooneyRivlin3D ();
97
98 // Lecture des donnees de la classe sur fichier
99 void LectureDonneesParticulieres (UtilLecture * , LesCourbes1D & lesCourbes1D
100 , LesFonctions_nD & lesFonctionsnD);
101
102 // affichage de la loi
103 void Affiche() const ;
104 // test si la loi est complete
105 // = 1 tout est ok, =0 loi incomplete
106 int TestComplet();
107
108 //----- lecture écriture de restart -----
109 // cas donne le niveau de la récupération
110 // = 1 : on récupère tout
111 // = 2 : on récupère uniquement les données variables (supposées comme telles)
112 void Lecture_base_info_loi(ifstream & ent, const int cas, LesReferences & lesRef, LesCourbes1D &
113 lesCourbes1D , LesFonctions_nD & lesFonctionsnD);
114
115 // cas donne le niveau de sauvegarde
116 // = 1 : on sauvegarde tout
117 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
118 void Ecriture_base_info_loi(ofstream & sort, const int cas);
119
120
121

```

```

118 // calcul d'un module d'young équivalent à la loi,
119 // c'est sans doute complètement débile mais c'est pour pouvoir avancer !!
120 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * ) {return 6.*
121 C10;};
122 // récupération de la variation relative d'épaisseur calculée: h/h0
123 // cette variation n'est utile que pour des lois en contraintes planes
124 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
125 // - pour les lois 2D def planes: retour de 0
126 // les infos nécessaires à la récupération , sont stockées dans saveResul
127 // qui est le conteneur spécifique au point où a été calculé la loi
128 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
129
130 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
131 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new MooneyRivlin3D(*this)); };
132
133 // affichage et definition interactive des commandes particulières à chaque lois
134 void Info_commande_LoisDeComp(UtilLecture& lec);
135
136 protected :
137 // donnée de la loi
138 double C10,C01,K; // 3 coeffs
139 CourbelD* C10_temperature; // courbe éventuelle d'évolution de C10 en fonction de la température
140 CourbelD* C01_temperature; // courbe éventuelle d'évolution de C01 en fonction de la température
141 CourbelD* K_temperature; // courbe éventuelle d'évolution de K en fonction de la température
142 int type_pot_vol; // indique le type de potentiel volumique, par défaut 2
143 bool avec_courbure; // indique s'il y a un potentiel de courbure ou non
144 double a_courbure; // para a utilisé que dans le cas avec courbure
145 double r_courbure; // para r utilisé que dans le cas avec courbure
146 CourbelD* a_temperature; // courbe éventuelle d'évolution de a en fonction de la température
147 CourbelD* r_temperature; // courbe éventuelle d'évolution de r en fonction de la température
148
149 double W_d,W_v; // le potentiel: partie déviatorique, partie sphérique
150 Vecteur W_r; // dérivées premières du potentiel par rapport aux J_r
151 double W_d_J1,W_d_J2; // dérivées premières du potentiel déviatoire par rapport aux J_1 et J_2
152 double W_v_J3,W_v_J3J3; // dérivées premières et seconde du potentiel volumique / J3
153 Tableau2 <double> W_rs; // dérivées secondes du potentiel par rapport aux J_r
154 // cas éventuel de potentiel additionnel de raidissement: variables intermédiaires de passage
155 double W_c,W_c_J1,W_c_J3,W_c_J1_2,W_c_J3_2,W_c_J1_J3; // potentiel et dérivées 1 et 2
156
157
158 // codage des METHODES VIRTUELLES protegees:
159 // calcul des contraintes a t+dt
160 // calcul des contraintes
161 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
162 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB & giB,BaseH & gi_H, TenseurBB & epsBB_
163 ,TenseurBB & delta_epsBB_
164 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
165 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
166 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
167 module_cisaillement
168 ,const Met_abstraite::Expli_t_tdt& ex);
169
170 // calcul des contraintes et de ses variations a t+dt
171 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
172 ,BaseB & giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
173 ,BaseB & giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH & giH_tdt,Tableau <BaseH> & d_giH_tdt
174 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
175 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
176 ,Tableau <TenseurBB *>& d_gijBB_tdt
177 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
178 ,Vecteur & d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
179 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
180 module_cisaillement
181 ,const Met_abstraite::Impli& ex);
182
183 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
184 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
185 // le tenseur de déformation et son incrémentsont également en orthonormees
186 // si = false: les bases transmises sont utilisées
187 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
188 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
189 ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
190 ,TenseurHH& sigHH,TenseurHH& d_sigma_deps
191 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
192 module_cisaillement
193 ,const Met_abstraite::Umat_cont& ex) ; // = 0;
194
195 // fonction surchargée dans les classes dérivée si besoin est
196 virtual void CalculGrandeurTravail
197 (const PtIntegMecaInterne& ,const Deformation &
198 ,Enum_dure,const ThermoDonnee&
199 ,const Met_abstraite::Impli* ex_impli
200 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
201 ,const Met_abstraite::Umat_cont* ex_umat
202 ,const List_io<Ddl_etendu>* excludre_dd_etend

```

```

201         ,const List_io<const TypeQuelconque *>* exclure_Q
202     ) {};
203
204 private:
205     // calcul du potentiel et de ses dérivées premières / aux invariants J_r
206     void Potentiel_et_var(double & module_compressibilite);
207     // calcul du potentiel et de ses dérivées premières et secondes / aux invariants J_r
208     void Potentiel_et_var2(double & module_compressibilite);
209     // calcul de la dérivée numérique de la contrainte
210     void Cal_dsigma_deps_num (const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
211                             ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
212                             ,const double& jacobien_0,const double& jacobien
213                             ,Tenseur3HHHH& dSigdepsHHHH);
214     // calcul de la contrainte avec le minimum de variable de passage, utilisé pour le numérique
215     void Cal_sigma_pour_num(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
216                             ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
217                             ,const double& jacobien_0,const double& jacobien,TenseurHH & sigHH_);
218
219     // idem avec la variation
220     void Cal_sigmaEtDer_pour_num(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
221                                 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
222                                 ,const double& jacobien_0,const double& jacobien
223                                 ,TenseurHH & sigHH_,Tenseur3HHHH& dSigdepsHHHH);
224 };
225 /// @} // end of group
226
227
228 #endif
229
230
231

```

## 7.50 Poly\_hyper3D.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           30/5/2005
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 *****/
39 *   BUT:   La classe Poly_hyper3D permet de calculer la contrainte
40 *          et ses derivees pour une loi isotrope hyper élastique
41 *          de type extension de la loi de Mooney Rivlin en 3D
42 *          sous forme polynomiale, en fonction des memes trois inva-
43 *          viants que ceux de Mooney Rivlin.
44 *   Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
45 *
46 *   *****
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !           but

```



```

50 * ----- *
51 * ! ! ! ! *
52 * $ *
53 * ***** *
54 * MODIFICATIONS: *
55 * ! date ! auteur ! but ! *
56 * ----- *
57 * $ *
58 *****/
59 #ifndef POLY_HYPER_3D_H
60 #define POLY_HYPER_3D_H
61
62
63
64
65 #include "Courbe1D.h"
66 #include "MathUtil.h"
67 #include "Hyper_W_gene_3D.h"
68
69
70 /// @addtogroup Les_lois_hyperelastiques
71 /// @{
72 ///
73
74
75 class Poly_hyper3D : public Hyper_W_gene_3D
76 {
77
78     public :
79
80
81
82     // CONSTRUCTEURS :
83
84     // Constructeur par default
85     Poly_hyper3D ();
86
87
88     // Constructeur de copie
89     Poly_hyper3D (const Poly_hyper3D& loi) ;
90
91     // DESTRUCTEUR :
92
93     ~Poly_hyper3D ();
94
95     // Lecture des donnees de la classe sur fichier
96     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
97                                     ,LesFonctions_nD& lesFonctionsnD);
98
99     // affichage de la loi
100    void Affiche() const ;
101    // test si la loi est complete
102    // = 1 tout est ok, =0 loi incomplete
103    int TestComplet();
104
105    //----- lecture écriture de restart -----
106    // cas donne le niveau de la récupération
107    // = 1 : on récupère tout
108    // = 2 : on récupère uniquement les données variables (supposées comme telles)
109    void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
110                             lesCourbes1D
111                             ,LesFonctions_nD& lesFonctionsnD);
112
113    // cas donne le niveau de sauvegarde
114    // = 1 : on sauvegarde tout
115    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
116    void Ecriture_base_info_loi(ofstream& sort,const int cas);
117
118    // calcul d'un module d'young équivalent à la loi,
119    // c'est sans doute complètement débile mais c'est pour pouvoir avancer !!
120    double Module_young_equivalent (Enum_dure ,const Deformation & ,SaveResul * ) {return 6.* Cij(1)(2)};
121
122    // récupération de la variation relative d'épaisseur calculée: h/h0
123    // cette variation n'est utile que pour des lois en contraintes planes
124    // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
125    // - pour les lois 2D def planes: retour de 0
126    // les infos nécessaires à la récupération , sont stockées dans saveResul
127    // qui est le conteneur spécifique au point où a été calculé la loi
128    virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand};
129
130    // création d'une loi à l'identique et ramène un pointeur sur la loi créée
131    Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Poly_hyper3D(*this)); };
132
133    // affichage et definition interactive des commandes particulières à chaque lois
134    void Info_commande_LoisDeComp(UtilLecture& lec);
135
136     protected :
137
138     // donnée de la loi
139     double K; // le module de compressibilité

```

```

136 // on utilise des tableaux de tableaux car les dimensions des sous tableaux sont toutes
différentes
137 Tableau <Tableau <double> > Cij; // Cij(i)(j) : contient les coefficients du polynome tel que:
138 // Ckl : est stocké dans Cij(l+k)(k+1)
139 // et Cij(i)(j) : contient C_(j-1)_(i-j+1)
140 Tableau <Tableau < CourbelD* > > Cij_temperature; // les courbe éventuelle d'évolution
141 // des coefficients en fonction de la température
142 CourbelD* K_temperature; // courbe éventuelle d'évolution de K en fonction de la température
143 int type_pot_vol; // indique le type de potentiel volumique, par défaut 2
144 bool avec_courbure; // indique s'il y a un potentiel de courbure ou non
145 double a_courbure; // para a utilisé que dans le cas avec courbure
146 double r_courbure; // para r utilisé que dans le cas avec courbure
147 CourbelD* a_temperature; // courbe éventuelle d'évolution de a en fonction de la température
148 CourbelD* r_temperature; // courbe éventuelle d'évolution de r en fonction de la température
149
150 double W_d,W_v; // le potentiel: partie déviatorique, partie sphérique
151 Vecteur W_r; // dérivées premières du potentiel par rapport aux J_r
152 double W_d_J1,W_d_J2; // dérivées premières du potentiel déviatoire par rapport aux J_1 et J_2
153 double W_d_J1_2,W_d_J1_J2,W_d_J2_2; // dérivées secondes
154 double W_v_J3,W_v_J3J3; // dérivées premières et seconde du potentiel volumique / J3
155 Tableau2 <double> W_rs; // dérivées secondes du potentiel par rapport aux J_r
156 // cas éventuel de potentiel additionnel de raidissement: variables intermédiaires de passage
157 double W_c,W_c_J1,W_c_J3,W_c_J1_2,W_c_J3_2,W_c_J1_J3; // potentiel et dérivées 1 et 2
158
159
160 // codage des METHODES VIRTUELLES protegees:
161 // calcul des contraintes a t+dt
162 // calcul des contraintes
163 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
164 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
165 ,TenseurBB & delta_epsBB_
166 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
167 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
168 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
169 ,const Met_abstraite::Expli_t_tdt& ex);
170
171 // calcul des contraintes et de ses variations a t+dt
172 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
173 ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
174 ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
175 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
176 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
177 ,Tableau <TenseurBB *>& d_gijBB_tdt
178 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
179 ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
180 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
181 ,const Met_abstraite::Impli& ex);
182
183 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
184 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
185 // le tenseur de déformation et son incrémentsont également en orthonormees
186 // si = false: les bases transmises sont utilisées
187 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
188 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
189 ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
190 ,TenseurHH& sigHH,TenseurHHH& d_sigma_deps
191 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
192 ,const Met_abstraite::Umat_cont& ex) ; // = 0;
193
194
195 // fonction surchargée dans les classes dérivée si besoin est
196 virtual void CalculGrandeurTravail
197 (const PtIntegMecaInterne& ,const Deformation &
198 ,Enum_dure,const ThermoDonnee&
199 ,const Met_abstraite::Impli* ex_impli
200 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
201 ,const Met_abstraite::Umat_cont* ex_umat
202 ,const List_io<Ddl_etendu>* exclure_dd_etend
203 ,const List_io<const TypeQuelconque *>* exclure_Q
204 ) {};
205
206 private:
207 // calcul du potentiel et de ses dérivées premières / aux invariants J_r
208 void Potentiel_et_var(double & module_compressibilite);
209 // calcul du potentiel et de ses dérivées premières et secondes / aux invariants J_r
210 void Potentiel_et_var2(double & module_compressibilite);
211 // calcul de la dérivée numérique de la contrainte
212 void Cal_dsigma_deps_num (const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
213 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
214 ,const double& jacobien_0,const double& jacobien
215 ,Tenseur3HHH& dSigdepsHHH);
216 // calcul de la contrainte avec le minimum de variable de passage, utilisé pour le numérique
217 void Cal_sigma_pour_num(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
218 ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt

```

```

219                                     ,const double& jacobien_0,const double& jacobien,TenseurHH & sigHH_);
220
221 // idem avec la variation
222 void Cal_sigmaEtDer_pour_num(const TenseurBB & gijBB_0,const TenseurHH & gijHH_0
223                               ,const TenseurBB & gijBB_tdt,const TenseurHH & gijHH_tdt
224                               ,const double& jacobien_0,const double& jacobien
225                               ,TenseurHH & sigHH_,Tenseur3HHHH& dSigdepsHHHH);
226
227 // calcul de puissances particulières, utilisées dans le calcul du potentiel
228 // notamment pour les dérivées du potentielles, on a des exposants nulles
229 // en fait cela signifie que la dérivée est nulle, adapte donc le calcul
230 inline double PuissPoten(double a,const int n)
231 { if (n >=1) {return PUISSN(a,n);} // cas normal
232   else if (n==1) {return a;} // cas simple
233   else if (n==0) {return 1.;} // cas exposent nul
234   else {return 0.;;} // cas négatif, on met à 0
235 };
236 };
237 /// @} // end of group
238
239
240 #endif
241
242
243

```

## 7.51 TreloarN.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:      22/03/99
33 *
34 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:    Herezh++
37 *
38 *      *****
39 *      BUT: Loi Hyper élastique fondée sur le potentiel développé par
40 *      Tréloar, mais en incorporant une déformation volumique.
41 *
42 *      *****
43 *
44 *      VERIFICATION:
45 *      ! date ! auteur ! but
46 *      -----
47 *      ! ! !
48 *      $
49 *      *****
50 *      MODIFICATIONS:
51 *      ! date ! auteur ! but
52 *      -----
53 *      $
54 *      *****/
55 #ifndef TRELOARN_H

```

```

56 #define TRELOARN_H
57
58 #include "Hyper3DN.h"
59
60
61 /// @addtogroup Les_lois_hyperelastiques
62 /// @{
63 ///
64
65
66 class TreloarN : public Hyper3DN
67 {
68 public :
69     // VARIABLES PUBLIQUES :
70
71 // CONSTRUCTEURS :
72 // Constructeur par défaut
73 TreloarN ();
74 // Constructeur de copie
75 TreloarN (const TreloarN& loi) ;
76 // DESTRUCTEUR :
77 ~TreloarN ()
78     {};
79
80 // Lecture des donnees de la classe sur fichier
81 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
82                                     ,LesFonctions_nD& lesFonctionsnD);
83
84 // affichage de la loi
85 void Affiche() const ;
86 // test si la loi est complete
87 // = 1 tout est ok, =0 loi incomplete
88 int TestComplet ();
89
90 //----- lecture écriture de restart -----
91 // cas donne le niveau de la récupération
92 // = 1 : on récupère tout
93 // = 2 : on récupère uniquement les données variables (supposées comme telles)
94 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
95                             ,LesFonctions_nD& lesFonctionsnD);
96 // cas donne le niveau de sauvegarde
97 // = 1 : on sauvegarde tout
98 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
99 void Ecriture_base_info_loi(ofstream& sort,const int cas);
100
101 // affichage et definition interactive des commandes particulières à chaque lois
102 void Info_commande_LoisDeComp(UtilLecture& lec);
103
104 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
105 Loi_comp_abstraite* Nouvelle_loi_identique() const {return (new TreloarN(*this));};
106
107 // METHODES découlant de méthodes virtuelles définies dans HyperDN
108
109 // calcul du potentiel et de ses dérivées premières
110 void Potentiel (double& jacobien_0,double& IEps,double& V,double& bIIb,
111                double& E,double& EV,double& EbIIb,double& EIEps);
112 // calcul du potentiel et de ses dérivées premières et secondes
113 void Potentiel_et_var( double& jacobien_0,double& IEps,double& V,double& bIIb,
114                       double& E,double& EV,double& EbIIb,double& EIEps ,
115                       double& EVV,double& EbIIb2,double& EIEps2,
116                       double& EVbIIb,double& EVIEps,double& EbIIbIEps );
117
118
119 // VARIABLES PROTEGEES :
120 // paramètres de la loi de comportement
121 double K; // coefficient de compressibilité volumique
122 double C; // coefficient de Treloar
123
124 // METHODES PROTEGEES :
125
126 };
127 /// @} // end of group
128
129 #endif

```

## 7.52 Hypo\_hooke1D.h

```

1 // FICHER : Hypo1D.h
2 // CLASSE : Hypo1D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.

```

```

9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *      DATE:      18/07/2020
35 *
36 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *
41 *      BUT:      La classe Hypo1D definit une loi 1D hypo-élastique
42 *              qui sous forme intégrée peut dans certain cas être
43 *              équivalente à hooke.
44 *              On a donc :
45 *              sigma_point = f(..) * D
46 *
47 *      *****
48 *
49 *      VERIFICATION:
50 *      ! date !   auteur !   but
51 *      -----
52 *      !       !       !
53 *
54 *      *****
55 *
56 *      MODIFICATIONS:
57 *      ! date !   auteur !   but
58 *      -----
59 *
60 *
61
62 #ifndef HYPO_ELAS1D_H
63 #define HYPO_ELAS1D_H
64
65
66 #include "Loi_comp_abstraite.h"
67
68 /** @defgroup Les_lois_hypoelastiques
69 *
70 *      BUT:   groupe des lois hypoélastiques
71 *
72 *
73 * \author   Gérard Rio
74 * \version  1.0
75 * \date    28/06/2004
76 * \brief   Définition des lois hypoélastiques
77 *
78 */
79
80 /// @addtogroup Les_lois_hypoelastiques
81 /// @{
82 ///
83
84
85 class Hypo_hooke1D : public Loi_comp_abstraite
86 {
87
88
89     public :
90
91
92         // CONSTRUCTEURS :
93
94         // Constructeur par défaut

```

```

95     Hypo_hooke1D ();
96
97
98     // Constructeur de copie
99     Hypo_hooke1D (const Hypo_hooke1D& loi) ;
100
101     // DESTRUCTEUR :
102
103     ~Hypo_hooke1D ();
104
105
106     // initialise les donnees particulieres a l'elements
107     // de matiere traite ( c-a-dire au pt calcule)
108     // Il y a creation d'une instance de SaveResul particuliere
109     // a la loi concernee
110     // la SaveResul classe est remplie par les instances heritantes
111     // le pointeur de SaveResul est sauvegarde au niveau de l'element
112     // c'a-d que les info particulieres au point considere sont stocke
113     // au niveau de l'element et non de la loi.
114     class SaveResulLoi_HypolD: public SaveResul
115     { public :
116         SaveResulLoi_HypolD(); // constructeur par défaut à ne pas utiliser
117         // le constructeur courant
118         SaveResulLoi_HypolD (SaveResul* );
119         // de copie
120         SaveResulLoi_HypolD(const SaveResulLoi_HypolD& sav): // de copie
121             Kc(sav.Kc),Kc_t(sav.Kc_t),f(sav.f),f_t(sav.f_t)
122             ,eps22(sav.eps22),eps22_t(sav.eps22_t),eps33(sav.eps33),eps33_t(sav.eps33_t)
123             ,eps_cumulBB(sav.eps_cumulBB),eps_cumulBB_t(sav.eps_cumulBB_t)
124             {};
125
126         virtual ~SaveResulLoi_HypolD() {}; // destructeur
127         // définition d'une nouvelle instance identique
128         // appelle du constructeur via new
129         SaveResul * Nevez_SaveResul() const{return (new SaveResulLoi_HypolD(*this));};
130         // affectation
131         virtual SaveResul & operator = ( const SaveResul & a)
132         { SaveResulLoi_HypolD& sav = *((SaveResulLoi_HypolD*) &a);
133           Kc=sav.Kc;Kc_t=sav.Kc_t;f=sav.f;f_t=sav.f_t;
134           eps22=sav.eps22;eps22_t=sav.eps22_t;eps33=sav.eps33;eps33_t=sav.eps33_t;
135           eps_cumulBB=sav.eps_cumulBB;eps_cumulBB_t=sav.eps_cumulBB_t;
136           return *this;
137         };
138         //===== lecture écriture dans base info =====
139         // cas donne le niveau de la récupération
140         // = 1 : on récupère tout
141         // = 2 : on récupère uniquement les données variables (supposées comme telles)
142         void Lecture_base_info (ifstream& ent,const int cas);
143
144         // cas donne le niveau de sauvegarde
145         // = 1 : on sauvegarde tout
146         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
147         void Ecriture_base_info(ofstream& sort,const int cas);
148
149         // mise à jour des informations transitoires
150         void TdtversT()
151         {Kc_t = Kc; f_t=f; eps22_t=eps22;eps33_t=eps33;
152          eps_cumulBB_t = eps_cumulBB;
153         };
154         void TversTdt ()
155         {Kc = Kc_t; f=f_t;eps22=eps22_t;eps33=eps33_t;
156          eps_cumulBB = eps_cumulBB_t;
157         };
158
159         // affichage à l'écran des infos
160         void Affiche() const
161         { cout <<"\n Kc= " << Kc << " f= " << f
162           << " eps22= " << eps22 << " eps33= " << eps33
163           << " eps_cumulBB= " << eps_cumulBB
164           << " ";
165         };
166
167         //changement de base de toutes les grandeurs internes tensorielles stockées
168         // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
169         // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
170         // gpH(i) = gamma(i,j) * gH(j)
171         virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) ;
172
173         // procedure permettant de completer éventuellement les données particulières
174         // de la loi stockées
175         // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
176         // completer est appelé apres sa creation avec les donnees du bloc transmis
177         // peut etre appeler plusieurs fois
178         virtual SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>&
tab_coor
179                                     ,const Loi_comp_abstraite* loi) {};
180

```

```

181
182 //-----
183 // données
184 //-----
185 double Kc,Kc_t; // les paramètres matériaux réellement utilisés
186 double f,f_t;
187 double eps33,eps22; // déformations transversale courantes
188 double eps33_t,eps22_t; // les dernières enregistrées
189
190 Tenseur1BB eps_cumulBB,eps_cumulBB_t; // déformation cumulée associée à la loi
191
192 };
193
194 SaveResul * New_et_Initialise();
195
196 friend class SaveResulLoi_Hypo1D;
197
198 // Lecture des donnees de la classe sur fichier
199 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
200                                   ,LesFonctions_nD& lesFonctionsnD);
201
202 // affichage de la loi
203 void Affiche() const ;
204 // test si la loi est complete
205 // = 1 tout est ok, =0 loi incomplete
206 int TestComplet();
207
208 //----- lecture écriture de restart -----
209 // cas donne le niveau de la récupération
210 // = 1 : on récupère tout
211 // = 2 : on récupère uniquement les données variables (supposées comme telles)
212 void Lecture_base_info_loi(istream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
213                             lesCourbes1D
214                             ,LesFonctions_nD& lesFonctionsnD);
215
216 // cas donne le niveau de sauvegarde
217 // = 1 : on sauvegarde tout
218 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
219 void Ecriture_base_info_loi(ofstream& sort,const int cas);
220
221 // récupération des grandeurs particulière (hors ddl )
222 // correspondant à liTQ
223 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
224 void Grandeur_particuliere
225 (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal)
226 const;
227
228 // récupération de la liste de tous les grandeurs particulières
229 // ces grandeurs sont ajoutées à la liste passées en paramètres
230 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
231 void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
232
233 // calcul d'un module d'young équivalent à la loi
234 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
235
236 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
237 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
238 // >> en fait ici il s'agit du dernier module tangent calculé !!
239 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
240     saveResul);
241
242 // récupération de la dernière déformation d'épaisseur calculée: cette déformaion n'est utile que pour
243 // des lois en contraintes planes ou doublement planes
244 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
245 // - pour les lois 2D def planes: retour de 0
246 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
247 // qui est le conteneur spécifique au point où a été calculé la loi
248 double Eps33BH(SaveResul * saveDon) const
249 { SaveResulLoi_Hypo1D & save_resul = *((SaveResulLoi_Hypo1D*) saveDon);
250   return save_resul.eps33;
251 };
252
253 // récupération de la dernière déformation de largeur calculée: cette déformaion n'est utile que pour
254 // des lois en contraintes doublement planes
255 // - pour les lois 3D et 2D : retour d'un nombre très grand, indiquant que cette fonction est invalide
256 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
257 // qui est le conteneur spécifique au point où a été calculé la loi
258 double Eps22BH(SaveResul * saveDon) const
259 { SaveResulLoi_Hypo1D & save_resul = *((SaveResulLoi_Hypo1D*) saveDon);
260   return save_resul.eps22;
261 };
262
263 // récupération de la variation relative d'épaisseur calculée: h/h0
264 // cette variation n'est utile que pour des lois en contraintes planes
265 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
266 // - pour les lois 2D def planes: retour de 0
267 // les infos nécessaires à la récupération , sont stockées dans saveResul
268 // qui est le conteneur spécifique au point où a été calculé la loi
269 double HsurH0(SaveResul * saveResul) const;
270
271

```

```

263 // récupération de la variation relative d'épaisseur calculée: h/h0
264 // et de sa variation par rapport aux ddls la concernant: d_hsurh0
265 // cette variation n'est utile que pour des lois en contraintes planes
266 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
267 // - pour les lois 2D def planes: retour de 0
268 // les infos nécessaires à la récupération , sont stockées dans saveResul
269 // qui est le conteneur spécifique au point où a été calculé la loi
270 // pour l'instant en attente *** virtual double d_HsurH0(SaveResul * saveResul,Vecteur & d_hsurh0)
    const ;
271
272
273 // récupération de la variation relative de largeur calculée: b/b0
274 // cette variation n'est utile que pour des lois en contraintes planes double
275 // - pour les lois 3D et 2D : retour d'un nombre très grand, indiquant que cette fonction est
    invalide
276 // les infos nécessaires à la récupération , sont stockées dans saveResul
277 // qui est le conteneur spécifique au point où a été calculé la loi
278 double BsurB0(SaveResul * saveResul) const ;
279
280 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
281 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hypo_hooke1D(*this)); };
282
283 // affichage et definition interactive des commandes particulières à chaque lois
284 void Info_commande_LoisDeComp(UtilLecture& lec);
285 // calcul de grandeurs de travail aux points d'intégration via la def et autres
286 // ici permet de récupérer la compressibilité
287 // fonction surchargée dans les classes dérivée si besoin est
288 virtual void CalculGrandeurTravail
289     (const PtIntegMecaInterne& ptintmeca
290     ,const Deformation & def,Enum_dure temps,const ThermoDonnee& dTP
291     ,const Met_abstraite::Impli* ex_impli
292     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
293     ,const Met_abstraite::Umat_cont* ex_umat
294     ,const List_io<Ddl_etendu>* exclure_dd_etendu
295     ,const List_io<const TypeQuelconque *>* exclure_Q
296     )
297 {if (compress_thermophysique) Kc = 3./dTP.Compressibilite(); };
298
299 protected :
300     // donnée de la loi
301     double f; // coef de proportionalité entre sig1{.1} et D1{.1}
302     CourbelD* f_temperature; // courbe éventuelle d'évolution de f en fonction de la température
303     CourbelD* f_IIeps; // courbe éventuelle d'évolution de f en fonction du deuxième invariant
    d'epsilon
304     Fonction_nD* f_nD; // fonction nD éventuelle pour f
305
306     double Kc; // 3 * coefficient de compressibilité tangent
307     CourbelD* Kc_temperature; // courbe éventuelle d'évolution de Kc en fonction de la température
308     CourbelD* Kc_IIeps; // courbe éventuelle d'évolution de Kc en fonction du deuxième invariant
    d'epsilon
309     Fonction_nD * Kc_nD; // fonction nD éventuelle pour Kc
310
311     bool compress_thermophysique; // indique si oui ou non la compressibilité est calculée par une
    loi
312         // thermophysique et donc
313         // récupéré par la fonction "CalculGrandeurTravail"
314     int type_derive; // type de dérivée objective utilisée pour sigma
315
316     int restriction_traction_compression; // =0 -> pas de restriction
317         // = -1 : traction uniquement autorisée, la compression est mise à 0
318         // = 1 : compression uniquement autorisée, la traction est mise à 0
319
320     // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
321     // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonormee
322     Tenseur3HHHH I_x_I_HHHH,I_xbarre_I_HHHH,I_x_eps_HHHH,I_x_D_HHHH,I_xbarre_D_HHHH,d_sig_t_HHHH;
323     Tenseur3HHHH d_spherique_sig_t_HHHH;
324
325     // codage des METHODES VIRTUELLES protegees:
326 // calcul des contraintes a t+dt
327 // calcul des contraintes
328 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
329     ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
330     ,TenseurBB & delta_epsBB_
331     ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
332     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
333     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
334     ,const Met_abstraite::Expli_t_tdt& ex);
335
336 // calcul des contraintes et de ses variations a t+dt
337 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
338     ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
339     ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
340     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
341     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
342     ,Tableau <TenseurBB *>& d_gijBB_tdt
343     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien

```



```

344         ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
345         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
346         ,const Met_abstraite::Impli& ex);
347
348         // calcul des contraintes et ses variations par rapport aux déformations a t+dt
349         // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
350         // le tenseur de déformation et son incrémentsont également en orthonormees
351         // si = false: les bases transmises sont utilisées
352         // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
353 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
354         ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
355         ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
356         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
357         ,const Met_abstraite::Umat_cont& ex) ; // = 0;
358
359
360 };
361 /// @} // end of group
362
363
364 #endif
365
366
367

```

## 7.53 Hypo\_hooke2D\_C.h

```

1 // FICHER : Hypo_hooke2D_C.h
2 // CLASSE : Hypo_hooke2D_C
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPLY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:      30/12/2006
35 *
36 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 * *****/
41 * BUT:   La classe Hypo_hooke2D_C definit une loi 2D hypo_élastique*
42 * qui sous forme intégrée peut-être équivalente à hooke. *
43 * viscosité non linéaire éventuelle. *
44 * On a donc : *
45 *     S_point = mu D_b *
46 *     I_point_sigma = K I_D_b *
47 *
48 * *****
49 * VERIFICATION: *
50 *
51 * ! date ! auteur ! but ! *
52 * ----- *
53 * ! ! ! ! *
54 * $ *
55 * *****

```

```

56 *      MODIFICATIONS:
57 *      ! date !      auteur !      but
58 *      -----
59 *
60 *****/
61
62
63 #ifndef HYPO_HOOKE_2D_C_H
64 #define HYPO_HOOKE_2D_C_H
65
66
67 #include "Loi_comp_abstraite.h"
68
69
70 /// @addtogroup Les_lois_hypoelastiques
71 /// @
72 ///
73
74
75 class Hypo_hooke2D_C : public Loi_comp_abstraite
76 {
77
78     public :
79
80
81         // CONSTRUCTEURS :
82
83         // Constructeur par défaut
84         Hypo_hooke2D_C ();
85
86
87         // Constructeur de copie
88         Hypo_hooke2D_C (const Hypo_hooke2D_C& loi) ;
89
90         // DESTRUCTEUR :
91
92         ~Hypo_hooke2D_C ();
93
94
95
96 // initialise les donnees particulieres a l'elements
97 // de matiere traite ( c-a-dire au pt calcule)
98 class SaveResul_Hypo_hooke2D_C: public SaveResul
99 { public :
100     SaveResul_Hypo_hooke2D_C(); // constructeur par défaut (a ne pas utiliser)
101     // le constructeur courant
102     SaveResul_Hypo_hooke2D_C(SaveResul* l_des_SaveResul);
103     // constructeur de copie
104     SaveResul_Hypo_hooke2D_C(const SaveResul_Hypo_hooke2D_C& sav );
105     // destructeur
106     ~SaveResul_Hypo_hooke2D_C();
107     // définition d'une nouvelle instance identique
108     // appelle du constructeur via new
109     SaveResul * Nevez_SaveResul() const {return (new SaveResul_Hypo_hooke2D_C(*this));};
110     // affectation
111     virtual SaveResul & operator = ( const SaveResul & a)
112     {SaveResul_Hypo_hooke2D_C& sav = *((SaveResul_Hypo_hooke2D_C*) &a);
113       Kc=sav.Kc;Kc_t=sav.Kc_t;mu=sav.mu;mu_t=sav.mu_t;
114       eps33=sav.eps33;eps33_t=sav.eps33_t;
115       eps_cumulBB = sav.eps_cumulBB;eps_cumulBB_t=sav.eps_cumulBB_t;
116       return *this;
117     };
118     //===== lecture écriture dans base info =====
119     // cas donne le niveau de la récupération
120     // = 1 : on récupère tout
121     // = 2 : on récupère uniquement les données variables (supposées comme telles)
122     void Lecture_base_info (ifstream& ent,const int cas);
123     // cas donne le niveau de sauvegarde
124     // = 1 : on sauvegarde tout
125     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
126     void Ecriture_base_info(ofstream& sort,const int cas);
127
128     // mise à jour des informations transitoires en définitif s'il y a convergence
129     // par exemple (pour la plasticité par exemple)
130     void TdtversT()
131     {Kc_t = Kc; mu_t=mu;eps33_t=eps33;eps_cumulBB_t=eps_cumulBB;} ;
132     void TversTdt()
133     {Kc = Kc_t; mu = mu_t; eps33=eps33_t;eps_cumulBB=eps_cumulBB_t;} ;
134
135     // affichage à l'écran des infos
136     void Affiche() const;
137
138     //changement de base de toutes les grandeurs internes tensorielles stockées
139     // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
140     // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
141     // ici il n'y a pas de données tensorielles donc rien n'a faire
142     // gpH(i) = gamma(i,j) * gH(j)

```

```

143     virtual void ChBase_des_grandeurs(const Mat_pleine& beta, const Mat_pleine& gamma);
144
145     // procedure permettant de completer éventuellement les données particulières
146     // de la loi stockées
147     // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
148     // completer est appelé apres sa creation avec les donnees du bloc transmis
149     // peut etre appeler plusieurs fois
150     SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
151                                   , const Loi_comp_abstraite* loi) {};
152
153     // ---- récupération d'information: spécifique à certaine classe dérivée
154     double Deformation_plastique();
155
156     // données protégées
157     double Kc, Kc_t; // les paramètres matériaux réellement utilisés
158     double mu, mu_t;
159     double eps33, eps33_t; // déformation d'épaisseur
160     Tenseur2BB eps_cumulBB, eps_cumulBB_t; // déformation cumulée associée à la loi
161 };
162
163     // def d'une instance de données spécifiques, et initialisation
164     SaveResul * New_et_Initialise();
165
166     // Lecture des donnees de la classe sur fichier
167     void LectureDonneesParticulieres (UtilLecture * , LesCourbes1D& lesCourbes1D, LesFonctions_nD&
168     lesFonctionsnD);
169     // affichage de la loi
170     void Affiche() const ;
171     // test si la loi est complete
172     // = 1 tout est ok, =0 loi incomplete
173     int TestComplet();
174
175     //----- lecture écriture de restart -----
176     // cas donne le niveau de la récupération
177     // = 1 : on récupère tout
178     // = 2 : on récupère uniquement les données variables (supposées comme telles)
179     void Lecture_base_info_loi(ifstream& ent, const int cas, LesReferences& lesRef, LesCourbes1D&
180     lesCourbes1D
181                                   , LesFonctions_nD& lesFonctionsnD);
182     // cas donne le niveau de sauvegarde
183     // = 1 : on sauvegarde tout
184     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
185     void Ecriture_base_info_loi(ofstream& sort, const int cas);
186
187     // récupération des grandeurs particulière (hors ddl )
188     // correspondant à liTQ
189     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
190     void Grandeur_particuliere
191     (bool absolue, List_io<TypeQuelconque>& , Loi_comp_abstraite::SaveResul * , list<int>& decal)
192     const;
193     // récupération de la liste de tous les grandeurs particulières
194     // ces grandeurs sont ajoutées à la liste passées en paramètres
195     void ListeGrandeurs_particulieres(List_io<TypeQuelconque>& ) const;
196
197     // calcul d'un module d'young équivalent à la loi, ceci pour un
198     // chargement nul
199     double Module_young_equivalent(Enum_dure temps, const Deformation & , SaveResul * saveResul );
200     // récupération d'un module de compressibilité équivalent à la loi pour un chargement nul
201     // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
202     double Module_compressibilite_equivalent(Enum_dure temps, const Deformation & , SaveResul * saveDon);
203
204     // récupération de la variation relative d'épaisseur calculée: h/h0
205     // cette variation n'est utile que pour des lois en contraintes planes
206     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
207     // - pour les lois 2D def planes: retour de 0
208     // les infos nécessaires à la récupération , sont stockées dans saveResul
209     // qui est le conteneur spécifique au point où a été calculé la loi
210     virtual double HsurH0(SaveResul * saveResul) const;
211
212     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
213     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hypo_hooke2D_C(*this)); };
214
215     // affichage et definition interactive des commandes particulières à chaque lois
216     void Info_commande_LoisDeComp(UtilLecture& lec);
217     // calcul de grandeurs de travail aux points d'intégration via la def et autres
218     // ici permet de récupérer la compressibilité
219     // fonction surchargée dans les classes dérivée si besoin est
220     virtual void CalculGrandeurTravail
221     (const PtIntegMecaInterne& ptintmeca
222     , const Deformation & def, Enum_dure temps, const ThermoDonnee& dTP
223     , const Met_abstraite::Impli* ex_impli
224     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
225     , const Met_abstraite::Umat_cont* ex_umat
226     , const List_io<Ddl_etendu>* exclure_dd_etend
227     , const List_io<const TypeQuelconque *>* exclure_Q
228     )
229     {if (compress_thermophysique) Kc = 3./dTP.Compressibilite();};

```

```

227
228 // ----- methode propre a une loi en contraintes planes -----
229 // récupération de la dernière déformation d'épaisseur calculée: cette déformaion n'est utile que pour
    des lois en contraintes planes ou doublement planes
230 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
231 // - pour les lois 2D def planes: retour de 0
232 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
233 // qui est le conteneur spécifique au point où a été calculé la loi
234 virtual double Eps33BH(SaveResul * saveResul) const ;
235
236 // indique si la loi est en contraintes planes en s'appuyant sur un comportement 3D
237 virtual bool Contraintes_planes_de_3D() const {return true;};
238
239 // calcul de la vitesse de deformation eps33_point
240 double Deps33BH(TenseurBB & epsBB_, TenseurBB & DepsBB_, TenseurHH & gijHH_);
241
242 protected :
243     // donnée de la loi
244     double mu; // coef de proportionalité entre S_point et D_barre
245     CourbelD* mu_temperature; // courbe éventuelle d'évolution de mu en fonction de la température
246     CourbelD* mu_IIeps; // courbe éventuelle d'évolution de mu en fonction du deuxième invariant
    d'epsilon
247     Fonction_nD* mu_nD; // fonction nD éventuelle pour mu
248
249     double Kc; // coefficient de compressibilité instantané
250     CourbelD* Kc_temperature; // courbe éventuelle d'évolution de Kc en fonction de la température
251     CourbelD* Kc_IIeps; // courbe éventuelle d'évolution de Kc en fonction du deuxième invariant
    d'epsilon
252     Fonction_nD * Kc_nD; // fonction nD éventuelle pour Kc
253
254     bool compress_thermophysique; // indique si oui ou non la compressibilité est calculée par une loi
    // thermophysique et donc
255     // récupéré par la fonction "CalculGrandeurTravail"
256     int type_derive; // type de dérivée objective utilisée pour sigma
257     // -1: dérivée de Jauman (par défaut)
258     // 0 : dérivée deux fois covariante
259     // 1 : dérivée deux fois contravariante
260     short int cas_calcul; // indique le choix entre différents types de calcul possible
261     // = 0 : calcul normal
262     // = 1 : calcul seulement déviatorique (la partie sphérique est mise à
263     // zéro)
264     // = 2 : calcul seulement sphérique (la partie déviatorique est mise à
    zéro)
265
266     // codage des METHODES VIRTUELLES protegees:
267 // calcul des contraintes a t+dt
268 // calcul des contraintes
269 void Calcul_SigmaHH (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
270     , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB & giB, BaseH & gi_H, TenseurBB & epsBB_
271     , TenseurBB & delta_epsBB_
272     , TenseurBB & gijBB_, TenseurHH & gijHH_, Tableau <TenseurBB *>& d_gijBB_
273     , double& jacobien_0, double& jacobien, TenseurHH & sigHH
274     , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
275     , const Met_abstraite::Expli_t_tdt& ex);
276
277 // calcul des contraintes et de ses variations a t+dt
278 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
279     , BaseB & giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
280     , BaseB & giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH & giH_tdt, Tableau <BaseH> & d_giH_tdt
281     , TenseurBB & epsBB_tdt, Tableau <TenseurBB *>& d_epsBB
282     , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
283     , Tableau <TenseurBB *>& d_gijBB_tdt
284     , Tableau <TenseurHH *>& d_gijHH_tdt, double& jacobien_0, double& jacobien
285     , Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *>& d_sigHH
286     , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
287     , const Met_abstraite::Impli& ex);
288
289
290
291 };
292 /// @} // end of group
293
294
295 #endif
296
297
298

```

## 7.54 Hypo\_hooke3D.h

```

1 // FICHER : Hypo_hooke3D.h
2 // CLASSE : Hypo_hooke3D
3

```

```

4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32 //
33 /*****
34 *   DATE:      28/06/2004
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   *****/
41 *   BUT:   La classe Hypo_hooke3D definit une loi 3D hypo_élastique
42 *         qui sous forme intégrée peut-être équivalente à hooke.
43 *         viscosité non linéaire éventuelle.
44 *         On a donc :
45 *             S_point = mu  D_b
46 *             I_point_sigma = K  I_D_b
47 *
48 *   *****
49 *   VERIFICATION:
50 *
51 *   ! date !   auteur !           but
52 *   -----
53 *   !       !           !
54 *   $
55 *   *****
56 *   MODIFICATIONS:
57 *   ! date !   auteur !           but
58 *   -----
59 *   $
60 *****/
61
62
63 #ifndef HYPO_HOOKE_3D_H
64 #define HYPO_HOOKE_3D_H
65
66
67 #include "Loi_comp_abstraite.h"
68
69
70
71 /// @addtogroup Les_lois_hypoelastiques
72 /// @{
73 ///
74
75 class Hypo_hooke3D : public Loi_comp_abstraite
76 {
77
78     public :
79
80         // CONSTRUCTEURS :
81
82         // Constructeur par défaut
83         Hypo_hooke3D ();
84
85         // Constructeur de copie
86         Hypo_hooke3D (const Hypo_hooke3D& loi) ;

```

```

90
91     // DESTRUCTEUR :
92
93     ~Hypo_hooke3D ();
94
95     // initialise les donnees particulieres a l'elements
96     // de matiere traite ( c-a-dire au pt calcule)
97     // Il y a creation d'une instance de SaveResul particuliere
98     // a la loi concernee
99     // la SaveResul classe est remplie par les instances heritantes
100    // le pointeur de SaveResul est sauvegarde au niveau de l'element
101    // c'a-d que les info particulieres au point considere sont stocke
102    // au niveau de l'element et non de la loi.
103    class SaveResulLoi_Hypo3D: public SaveResul
104    { public :
105
106        SaveResulLoi_Hypo3D(); // constructeur par défaut (a ne pas utiliser)
107        // le constructeur courant
108        SaveResulLoi_Hypo3D(SaveResul* l_des_SaveResul);
109        // de copie
110        SaveResulLoi_Hypo3D(const SaveResulLoi_Hypo3D& sav): // de copie
111            Kc(sav.Kc), Kc_t(sav.Kc_t), mu(sav.mu), mu_t(sav.mu_t)
112            , eps_cumulBB(sav.eps_cumulBB), eps_cumulBB_t(sav.eps_cumulBB_t)
113            {};
114        virtual ~SaveResulLoi_Hypo3D() {}; // destructeur
115        // définition d'une nouvelle instance identique
116        // appelle du constructeur via new
117        SaveResul * Nevez_SaveResul() const{return (new SaveResulLoi_Hypo3D(*this));};
118        // affectation
119        virtual SaveResul & operator = ( const SaveResul & a)
120        { SaveResulLoi_Hypo3D& sav = *((SaveResulLoi_Hypo3D*) &a);
121          Kc=sav.Kc;Kc_t=sav.Kc_t;mu=sav.mu;mu_t=sav.mu_t;
122          eps_cumulBB=sav.eps_cumulBB;eps_cumulBB_t=sav.eps_cumulBB_t;
123          return *this;
124        };
125        //===== lecture écriture dans base info =====
126        // cas donne le niveau de la récupération
127        // = 1 : on récupère tout
128        // = 2 : on récupère uniquement les données variables (supposées comme telles)
129        void Lecture_base_info (ifstream& ent,const int cas);
130
131        // cas donne le niveau de sauvegarde
132        // = 1 : on sauvegarde tout
133        // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
134        void Ecriture_base_info(ofstream& sort,const int cas);
135
136        // mise à jour des informations transitoires
137        void TdtversT()
138        {Kc_t = Kc; mu_t=mu; eps_cumulBB_t = eps_cumulBB;};
139        void TversTdt()
140        {Kc = Kc_t; mu=mu_t;eps_cumulBB = eps_cumulBB_t;};
141
142        // affichage à l'écran des infos
143        void Affiche() const
144        { cout <<"\n Kc= " << Kc << " mu= " << mu << " eps_cumulBB= " << eps_cumulBB
145          << " ";
146        };
147
148        //changement de base de toutes les grandeurs internes tensorielles stockées
149        // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gb
150        // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
151        // gpH(i) = gamma(i,j) * gH(j)
152        virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
153
154        // procedure permettant de completer éventuellement les données particulières
155        // de la loi stockées
156        // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
157        // completer est appelé apres sa creation avec les donnees du bloc transmis
158        // peut etre appeler plusieurs fois
159        virtual SaveResul* Complete_SaveResul(const BlocGen & bloc
160        , const Tableau <Coordonnee>& tab_coor
161        ,const Loi_comp_abstraite* loi) {return NULL;};
162
163
164        //-----
165        // données
166        //-----
167        double Kc,Kc_t; // les paramètres matériaux réellement utilisés
168        double mu,mu_t;
169        Tenseur3BB eps_cumulBB,eps_cumulBB_t; // déformation cumulée associée à la loi
170
171    };
172    // def d'une instance de données spécifiques, et initialisation
173    SaveResul * New_et_Initialise();
174
175    // Lecture des donnees de la classe sur fichier
176    void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D

```

```

177                                     ,LesFonctions_nD& lesFonctionsnD);
178 // affichage de la loi
179 void Affiche() const ;
180 // test si la loi est complete
181 // = 1 tout est ok, =0 loi incomplete
182 int TestComplet();
183
184 //----- lecture écriture de restart -----
185 // cas donne le niveau de la récupération
186 // = 1 : on récupère tout
187 // = 2 : on récupère uniquement les données variables (supposées comme telles)
188 void Lecture_base_info_loi(ifstream& ent,const int cas
189                             ,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
190                             ,LesFonctions_nD& lesFonctionsnD);
191 // cas donne le niveau de sauvegarde
192 // = 1 : on sauvegarde tout
193 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
194 void Ecriture_base_info_loi(ofstream& sort,const int cas);
195
196 // récupération des grandeurs particulière (hors ddl )
197 // correspondant à liTQ
198 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
199 void Grandeur_particuliere
200     (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal)
201     const;
202 // récupération de la liste de tous les grandeurs particulières
203 // ces grandeurs sont ajoutées à la liste passées en paramètres
204 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
205 void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
206
207 // calcul d'un module d'young équivalent à la loi, ceci pour un
208 // chargement nul
209 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
210
211 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
212 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
213 // >> en fait ici il s'agit du dernier module tangent calculé !!
214 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
215     saveResul);
216
217 // récupération de la variation relative d'épaisseur calculée: h/h0
218 // cette variation n'est utile que pour des lois en contraintes planes
219 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
220 // - pour les lois 2D def planes: retour de 0
221 // les infos nécessaires à la récupération , sont stockées dans saveResul
222 // qui est le conteneur spécifique au point où a été calculé la loi
223 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
224
225 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
226 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hypo_hooke3D(*this)); };
227
228 // affichage et definition interactive des commandes particulières à chaque lois
229 void Info_commande_LoisDeComp(UtilLecture& lec);
230 // calcul de grandeurs de travail aux points d'intégration via la def et autres
231 // ici permet de récupérer la compressibilité
232 // fonction surchargée dans les classes dérivée si besoin est
233 virtual void CalculGrandeurTravail
234     (const PtIntegMecaInterne& ptintmeca
235      ,const Deformation & def,Enum_dure temps,const ThermoDonnee& dTP
236      ,const Met_abstraite::Impli* ex_impli
237      ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
238      ,const Met_abstraite::Umat_cont* ex_umat
239      ,const List_io<Ddl_etendu>* exclure_dd_etend
240      ,const List_io<const TypeQuelconque *>* exclure_Q
241      )
242     {if (compress_thermophysique) Kc = 3./dTP.Compressibilite(); };
243
244 protected :
245     // donnée de la loi
246     double mu; // coef de proportionalité entre S_point et D_barre
247     CourbelD* mu_temperature; // courbe éventuelle d'évolution de mu en fonction de la température
248     CourbelD* mu_IIEps; // courbe éventuelle d'évolution de mu en fonction du deuxième invariant
249     d'epsilon
250     Fonction_nD* mu_nD; // fonction nD éventuelle pour mu
251
252     double Kc; // coefficient de compressibilité instantané
253     CourbelD* Kc_temperature; // courbe éventuelle d'évolution de Kc en fonction de la température
254     CourbelD* Kc_IIEps; // courbe éventuelle d'évolution de Kc en fonction du deuxième invariant
255     d'epsilon
256     Fonction_nD * Kc_nD; // fonction nD éventuelle pour Kc
257
258     bool compress_thermophysique; // indique si oui ou non la compressibilité est calculée par une loi
259     // thermophysique et donc
260     // récupéré par la fonction "CalculGrandeurTravail"
261     int type_derive; // type de dérivée objective utilisée pour sigma
262     // -1: dérivée de Jauman (par défaut)
263     // 0 : dérivée deux fois covariante

```

```

260 // 1 : dérivée deux fois contravariante
261 short int cas_calcul; // indique le choix entre différents types de calcul possible
262 // = 0 : calcul normal
263 // = 1 : calcul seulement déviatorique (la partie sphérique est mise à
zéro)
264 // = 2 : calcul seulement sphérique (la partie déviatorique est mise à
zéro)
265
266 // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
267 // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonormee
268 Tenseur3HHHH I_x_I_HHHH,I_xbarre_I_HHHH,I_x_eps_HHHH,I_x_D_HHHH,I_xbarre_D_HHHH,d_sig_t_HHHH;
269 Tenseur3HHHH d_spherique_sig_t_HHHH;
270
271 // codage des METHODES VIRTUELLES protegees:
272 // calcul des contraintes a t+dt
273 // calcul des contraintes
274 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
275 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
276 ,TenseurBB & delta_epsBB_
277 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
278 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
279 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
280 ,const Met_abstraite::Expli_t_tdt& ex);
281
282 // calcul des contraintes et de ses variations a t+dt
283 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
284 ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
285 ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
286 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
287 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
288 ,Tableau <TenseurBB *>& d_gijBB_tdt
289 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
290 ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
291 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
292 ,const Met_abstraite::Impli& ex);
293
294 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
295 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
296 // le tenseur de déformation et son incrémentsont également en orthonormees
297 // si = false: les bases transmises sont utilisées
298 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
299 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
300 ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
301 ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
302 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
303 ,const Met_abstraite::Umat_cont& ex) ; // = 0;
304
305
306 };
307 /// @} // end of group
308
309
310 #endif
311
312
313

```

## 7.55 Copie\_de\_Hysteresis3D.h

```

1 // FICHER : Hysteresis3D.h
2 // CLASSE : Hysteresis3D
3 /*****
4 * UNIVERSITE DE BRETAGNE SUD (UBS) --- I.U.P/I.U.T. DE LORIENT *
5 *****/
6 * LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M) *
7 * Centre de Recherche Rue de Saint Maud - 56325 Lorient cedex *
8 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
9 *****/
10 * DATE: 30/06/2004 *
11 * $ *
12 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
13 * Tel 0297874571 fax : 02.97.87.45.72 *
14 * $ *
15 * PROJET: Herezh++ *
16 * $ *
17 *****/
18 * BUT: La classe Hysteresis3D permet de calculer la contrainte *
19 * et ses derivees pour une loi d'hysteresis 3D, type *
20 * celle dveloppe par Guelin, Favier, Pegon. *
21 * $ *
22 * ***** *

```



```

23 *      VERIFICATION:                                     *
24 *      ! date !   auteur !           but                ! *
25 *      -----                                         ! *
26 *      !           !           !                       ! *
27 *      !           !           !                       ! *
28 *      !           !           !                       ! *
29 *      !           !           !                       ! *
30 *      !           !           !                       ! *
31 *      !           !           !                       ! *
32 *      !           !           !                       ! *
33 *      !           !           !                       ! *
34 *      !           !           !                       ! *
35 *      !           !           !                       ! *
36 *      !           !           !                       ! *
37 *      !           !           !                       ! *
38 *      !           !           !                       ! *
39 *      !           !           !                       ! *
40 *      !           !           !                       ! *
41 *      !           !           !                       ! *
42 *      !           !           !                       ! *
43 *      !           !           !                       ! *
44 *      !           !           !                       ! *
45 *      !           !           !                       ! *
46 *      !           !           !                       ! *
47 *      !           !           !                       ! *
48 *      !           !           !                       ! *
49 *      !           !           !                       ! *
50 *      !           !           !                       ! *
51 *      !           !           !                       ! *
52 *      !           !           !                       ! *
53 *      !           !           !                       ! *
54 *      !           !           !                       ! *
55 *      !           !           !                       ! *
56 *      !           !           !                       ! *
57 *      !           !           !                       ! *
58 *      !           !           !                       ! *
59 *      !           !           !                       ! *
60 *      !           !           !                       ! *
61 *      !           !           !                       ! *
62 *      !           !           !                       ! *
63 *      !           !           !                       ! *
64 *      !           !           !                       ! *
65 *      !           !           !                       ! *
66 *      !           !           !                       ! *
67 *      !           !           !                       ! *
68 *      !           !           !                       ! *
69 *      !           !           !                       ! *
70 *      !           !           !                       ! *
71 *      !           !           !                       ! *
72 *      !           !           !                       ! *
73 *      !           !           !                       ! *
74 *      !           !           !                       ! *
75 *      !           !           !                       ! *
76 *      !           !           !                       ! *
77 *      !           !           !                       ! *
78 *      !           !           !                       ! *
79 *      !           !           !                       ! *
80 *      !           !           !                       ! *
81 *      !           !           !                       ! *
82 *      !           !           !                       ! *
83 *      !           !           !                       ! *
84 *      !           !           !                       ! *
85 *      !           !           !                       ! *
86 *      !           !           !                       ! *
87 *      !           !           !                       ! *
88 *      !           !           !                       ! *
89 *      !           !           !                       ! *
90 *      !           !           !                       ! *
91 *      !           !           !                       ! *
92 *      !           !           !                       ! *
93 *      !           !           !                       ! *
94 *      !           !           !                       ! *
95 *      !           !           !                       ! *
96 *      !           !           !                       ! *
97 *      !           !           !                       ! *
98 *      !           !           !                       ! *
99 *      !           !           !                       ! *
100 *      !           !           !                       ! *
101 *      !           !           !                       ! *
102 *      !           !           !                       ! *
103 *      !           !           !                       ! *
104 *      !           !           !                       ! *
105 *      !           !           !                       ! *
106 *      !           !           !                       ! *
107 *      !           !           !                       ! *
108 *      !           !           !                       ! *
109 *      !           !           !                       ! *

```

```

37 #ifndef HYSTERESIS_3D_H
38 #define HYSTERESIS_3D_H
39
40
41 #include "Loi_comp_abstraite.h"
42 #include "CourbelD.h"
43 #include "TypeConsTens.h"
44 #include "Algo_zero.h"
45 #include "TenseurQlgene.h"
46
47 class Hysteresis3D : public Loi_comp_abstraite
48 {
49     public :
50
51         // CONSTRUCTEURS :
52
53         // Constructeur par défaut
54         Hysteresis3D ();
55
56
57         // Constructeur de copie
58         Hysteresis3D (const Hysteresis3D& loi) ;
59
60         // DESTRUCTEUR :
61
62         ~Hysteresis3D ();
63
64         // initialise les donnees particulieres a l'elements
65         // de matiere traite ( c-a-dire au pt calcule)
66         // Il y a creation d'une instance de SaveResul particuliere
67         // a la loi concerne
68         // la SaveResul classe est remplie par les instances heritantes
69         // le pointeur de SaveResul est sauvegarde au niveau de l'element
70         // c'a-d que les info particulieres au point considere sont stocke
71         // au niveau de l'element et non de la loi.
72         class SaveResulHysteresis3D: public SaveResul
73         { public :
74             SaveResulHysteresis3D(); // constructeur par dfaut :
75             SaveResulHysteresis3D(const SaveResulHysteresis3D& sav); // de copie
76             ~SaveResulHysteresis3D(){}; // destructeur
77             // d'finition d'une nouvelle instance identique
78             // appelle du constructeur via new
79             SaveResul * Nevez_SaveResul() const{return (new SaveResulHysteresis3D(*this));};
80             //===== lecture criture dans base info =====
81             // cas donne le niveau de la rcupration
82             // = 1 : on rcupre tout
83             // = 2 : on rcupre uniquement les donnees variables (supposes comme telles)
84             void Lecture_base_info (ifstream& ent,const int cas);
85             // cas donne le niveau de sauvegarde
86             // = 1 : on sauvegarde tout
87             // = 2 : on sauvegarde uniquement les donnees variables (supposes comme telles)
88             void Ecriture_base_info(ofstream& sort,const int cas);
89
90             // mise jour des informations transitoires
91             void TdtversT();
92             void TversTdt();
93
94             // ---- mthodes spcifiques
95             // initialise les informations de travail concernant le pas de temps en cours
96             void Init_debut_calcul();
97
98             // donnees protges
99             Tenseur3BH sigma_barre_BH_t; // derniere contrainte en BH t
100            Tenseur3BH sigma_barre_BH_tdt; // contrainte en cours BH tdt
101            double fonction_aide_t; // derniere valeur de la fonction d'aide
102            double fonction_aide_tdt; // valeur de la fonction d'aide en cours
103            int wBase_t,wBase_tdt; // paramtre de masing
104            List_io <double>::iterator ip2; // adresse ventuelle du 2 lments de fct_aide
105
106            // --- informations de travail concernant le pas de temps en cours ---
107            int modif; // = 0 rien de chang, =1 coincidence(s), =2 inversion(s), =3 coin et inver
108            // liste des nouvelles contraintes de rfrence, qui sont apparu
109            List_io <Tenseur3BH> sigma_barre_BH_R_t_a_tdt;

```

```

110         int nb_coincidence; // nombre de coincidence durant le pas de temps
111         List_io <double> fct_aide_t_a_tdt; // liste des valeurs de la fonction d'aide durant le pas
112         // de temps
113         List_io <bool> indic_coin; // liste d'indicateurs indiquant la suite des coincidences et
114         // inversion, = true -> indique que c'est une coincidence, sinon inversion
115         // --- fin informations de travail concernant le pas de temps en cours ---
116
117         // --- informations de mmorisation discrte de 0 t
118         // le dernier lment rang est en .begin() (c-a-d front())
119         List_io <Tenseur3BH> sigma_barre_BH_R; // liste des contraintes de rfrence
120         List_io <double> fct_aide; // liste des valeurs de la fonction d'aide
121
122     };
123
124     SaveResul * New_et_Initialise()
125     { SaveResulHysteresis3D * pt = new SaveResulHysteresis3D();
126       return pt;};
127
128     // Lecture des donnees de la classe sur fichier
129     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D);
130     // affichage de la loi
131     void Affiche() const ;
132     // test si la loi est complete
133     // = 1 tout est ok, =0 loi incomplete
134     int TestComplet();
135
136     // calcul d'un module d'young equivalent la loi, ceci pour un
137     // chargement nul
138     double Module_young_equivalent(Deformation & def);
139
140     // cration d'une loi l'identique et ramne un pointeur sur la loi cre
141     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hysteresis3D(*this)); };
142
143     //----- lecture criture de restart -----
144     // cas donne le niveau de la rcupration
145     // = 1 : on rcupre tout
146     // = 2 : on rcupre uniquement les donnees variables (supposes comme telles)
147     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D);
148
149     // cas donne le niveau de sauvegarde
150     // = 1 : on sauvegarde tout
151     // = 2 : on sauvegarde uniquement les donnees variables (supposes comme telles)
152     void Ecriture_base_info_loi(ofstream& sort,const int cas);
153
154     // affichage et definition interactive des commandes particulieres chaque lois
155     void Info_commande_LoisDeComp(UtilLecture& lec);
156
157 protected :
158
159     // donnees protegees
160     double xnp; // paramtre de Prager
161     int cas_prager; // indique si xnp =2 (=1), ou est compris entre 2 et 3 (=2), ou est sup 3 (=3)
162     double Qzero; // limite de plasticit du critre de von mises
163     double xmu; // paramtre de lame
164     // paramtres de l'algorithme
165     double tolerance_residu; // tolrance sur la rsolution de la plasticit
166     double tolerance_coincidence; // tolrance sur la prcision de la coincidence
167     int nb_boucle_maxi; // le maximum d'itration de plasticit permis
168     int nb_sous_increment; // le maxi de sous incrment prvu
169
170     // variables de travail pour l'change entre les diffrentes mthodes en internes
171     Tenseur3BH sigma_t_barre_tdt; // sigma barre finale
172     Tenseur3BH sigma_i_barre_BH; // sigma barre de dbut de calcul ( t au dbut)
173     Tenseur3BH sigma_barre_BH_R; // sigma barre de Rfrence en cours
174     Tenseur3BH delta_sigma_barre_BH_Rat; // deltat sigma barre de R a t
175     Tenseur3BH delta_sigma_barre_BH_Ratdt; // deltat sigma barre de R a tdt
176     Tenseur3BH delta_sigma_barre_tdt_BH; // delta sigma barre de t tdt
177     Tenseur3BH residuBH;
178     Tenseur3BH delta_barre_epsBH; // delta_barre epsilon totale
179     Tenseur3BH delta_barre_alpha_epsBH; // delta_barre epsilon intermediaire (avec alpha de 0 1)
180     int wBase; // paramtre de masing
181     double wprime; // paramtre de masing modifi
182     Tenseur3BH d_wprime; // variation de wprime par rapport deltat sigma barre de R a tdt
183     Vecteur residu; // rsidu de l'quation pour la rsolution de l'quation constitutive
184     Mat_pleine derResidu; // driv du rsidu de l'quation pour la rsolution de l'quation constitutive
185
186     Algo_zero alg_zero; // algo pour la recherche de zero
187
188     // -- variables de travail internes Residu_constitutif() et Mat_tangente_constitutif()
189     // dfinit ici pour viter de les dfinir chaque passage ds la mthode,
190     // ne doivent pas tre utilise en dehors de ces deux routines
191     Tenseur3BH rdelta_sigma_barre_BH_Ratdt; // deltat sigma barre de R a tdt
192     Tenseur3BH rdelta_sigma_barre_tdt_BH; // delta sigma barre de t tdt
193
194     // ----- mthodes internes -----
195     // affinage d'un point de coincidence

```

```

195 // ramne true si le traitement est exactement termin, sinon false, ce qui signifie qu'il
196 // faut encore continuer utiliser l'equation d'volution
197 // premiere_charge : indique si c'est oui ou non une coincidence avec la premiere charge
198 // pt_sur_principal : indique si oui ou non les pointeurs iaftc et iatens pointent sur les
listes
199 // principales
200 // iatens_princ et iaftc_princ: pointeurs sur les listes principales
201 bool Coincidence(double& unSur_wBaseCarre,bool premiere_charge
202 ,SaveResulHysteresis3D & save_resul,double& W_a
203 ,List_io <Tenseur3BH>::iterator& iatens,List_io <double>::iterator& iaftc
204 ,bool& pt_sur_principal,List_io <Tenseur3BH>::iterator& iatens_princ
205 ,List_io <double>::iterator& iaftc_princ,double& delta_W_a);
206
207
208 // ----- codage des METHODES VIRTUELLES protegees -----
209 // calcul des contraintes
210 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl,
211 TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_,
212 TenseurBB & delta_epsBB_,
213 TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_,
214 double& jacobien_0,double& jacobien,TenseurHH & sigHH);
215
216 // calcul des variations des contraintes a t+dt
217 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
218 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,
219 BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt,
220 TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB,
221 TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt,
222 Tableau <TenseurBB *>& d_gijBB_tdt,
223 Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien,
224 Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH);
225 public:
226 // calcul de la fonction rsidu de la resolution de l'equation constitutive
227 Vecteur& Residu_constitutif (const double & alpha, const Vecteur & x);
228 // calcul de la matrice tangente de la resolution de l'equation constitutive
229 Mat_pleine& Mat_tangente_constitutif(const double & alpha,const Vecteur & x, Vecteur& resi);
230 // calcul de l'oprateur tangent : dsigma/depsilon
231 TenseurQlgeneBHHB& Dsig_depsilon(TenseurQlgeneBHHB& dsig_deps);
232
233 protected:
234 // --- definition de differentes fonctions utiles pour le calcul final
235 // calcul de wprime en fonction de wBase, et des angles de phase
236 void Cal_wprime();
237 // calcul de wprime et de sa variation par rapport aux coordonnes de delta sigma barre
238 void Cal_wprime_et_der();
239
240
241 };
242
243
244 #endif
245
246
247

```

## 7.56 Hysteresis1D.h

```

1 // FICHER : Hysteresis1D.h
2 // CLASSE : Hysteresis1D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License

```

```

29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:      10/02/2004
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 * *****/
41 *   BUT:   La classe Hysteresis1D permet de calculer la contrainte
42 *   et ses derivees pour une loi d'hysteresis 1D, type
43 *   celle développée par Guelin, Favier, Pegon.
44 *
45 *   *****
46 *
47 *   VERIFICATION:
48 *   ! date !   auteur !           but
49 *   -----
50 *   !           !           !
51 *
52 *   *****
53 *   MODIFICATIONS:
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *****/
58
59
60 #ifndef HYSTERESIS_1D_H
61 #define HYSTERESIS_1D_H
62
63
64 #include "Loi_comp_abstraite.h"
65 #include "Courbe1D.h"
66 #include "TypeConsTens.h"
67 #include "Algo_zero.h"
68 #include "TenseurQlgene.h"
69 #include "Algo_edp.h"
70
71 /** @defgroup Les_lois_hysteresis
72 *
73 *   BUT:   groupe des lois de type hystérésis
74 *
75 *
76 * \author   Gérard Rio
77 * \version  1.0
78 * \date    10/02/2004
79 * \brief   Définition des lois de type hystérésis
80 *
81 */
82
83 /// @addtogroup Les_lois_hysteresis
84 /// @{
85 ///
86
87
88 class Hysteresis1D : public Loi_comp_abstraite
89 {
90 public :
91
92     // CONSTRUCTEURS :
93
94     // Constructeur par défaut
95     Hysteresis1D ();
96
97
98     // Constructeur de copie
99     Hysteresis1D (const Hysteresis1D& loi) ;
100
101     // DESTRUCTEUR :
102
103     ~Hysteresis1D ();
104
105     // initialise les donnees particulieres a l'elements
106     // de matiere traite ( c-a-dire au pt calcule)
107     // Il y a creation d'une instance de SaveResul particuliere
108     // a la loi concerne
109     // la SaveResul classe est remplie par les instances heritantes
110     // le pointeur de SaveResul est sauvegarde au niveau de l'element
111     // c'a-d que les info particulieres au point considere sont stocke
112     // au niveau de l'element et non de la loi.
113     class SaveResulHysteresis1D: public SaveResul
114     { public :

```

```

115         SaveResulHysteresis1D(); // constructeur par défaut :
116         SaveResulHysteresis1D(const SaveResulHysteresis1D& sav); // de copie
117         ~SaveResulHysteresis1D(){}; // destructeur
118         // définition d'une nouvelle instance identique
119         // appelle du constructeur via new
120         SaveResul * Nevez_SaveResul() const{return (new SaveResulHysteresis1D(*this));};
121         // affectation
122         virtual SaveResul & operator = ( const SaveResul & a);
123         //===== lecture écriture dans base info =====
124         // cas donne le niveau de la récupération
125         // = 1 : on récupère tout
126         // = 2 : on récupère uniquement les données variables (supposées comme telles)
127         void Lecture_base_info (ifstream& ent,const int cas);
128         // cas donne le niveau de sauvegarde
129         // = 1 : on sauvegarde tout
130         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
131         void Ecriture_base_info(ofstream& sort,const int cas);
132
133         // mise à jour des informations transitoires
134         void TdtversT();
135         void TversTdt();
136
137         // affichage à l'écran des infos
138         void Affiche() const;
139
140         //changement de base de toutes les grandeurs internes tensorielles stockées
141         // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gpB
142         // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
143         // gpH(i) = gamma(i,j) * gH(j)
144         virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
145
146         // procedure permettant de completer éventuellement les données particulières
147         // de la loi stockées
148         // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
149         // completer est appelé apres sa creation avec les donnees du bloc transmis
150         // peut etre appeler plusieurs fois
151         SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
152             ,const Loi_comp_abstraite* loi) {};
153
154         // ---- méthodes spécifiques
155         // initialise les informations de travail concernant le pas de temps en cours
156         void Init_debut_calcul();
157
158         // données protégées
159         Tenseur1BH sigma_barre_BH_t; // dernière contrainte en BH à t
160         Tenseur1BH sigma_barre_BH_tdt; // contrainte en cours BH à tdt
161         double fonction_aide_t; // dernière valeur de la fonction d'aide
162         double fonction_aide_tdt; // valeur de la fonction d'aide en cours
163         double wprime_t,wprime_tdt; // paramètre de masing
164         List_io <double>::iterator ip2; // adresse éventuelle du 2 éléments de fct_aide
165
166         // --- informations de travail concernant le pas de temps en cours ---
167         int modif; // = 0 rien de changé, =1 coincidence(s), =2 inversion(s), =3 coin et inver
168         // liste des nouvelles contraintes de référence, qui sont apparu
169         List_io <Tenseur1BH> sigma_barre_BH_R_t_a_tdt;
170         int nb_coincidence; // nombre de coincidence durant le pas de temps
171         List_io <double> fct_aide_t_a_tdt; // liste des valeurs de la fonction d'aide durant le pas
172         // de temps
173         List_io <bool> indic_coin; // liste d'indicateurs indiquant la suite des coincidences et
174         // inversion, = true -> indique que c'est une coincidence, sinon inversion
175         // --- fin informations de travail concernant le pas de temps en cours ---
176
177         // --- informations de mémorisation discrète de 0 à t
178         // le dernier élément rangé est en .begin() (c-a-d front())
179         List_io <Tenseur1BH> sigma_barre_BH_R; // liste des contraintes de référence
180         List_io <double> fct_aide; // liste des valeurs de la fonction d'aide
181
182         // 5) --- tableau d'indicateur de la résolution, éventuellement vide
183         // cela dépend de sortie_post
184         // indicateurs_resolution(1) : nb d'incrément utilisé pour la résolution de l'équation
185         // linéarisée
186         // (2) : nb total d'itération " " " " " "
187         // (3) : cas Runge Kutta: nb d'appel de la fonction
188         // (4) : cas Runge Kutta: nb de step de calcul
189         // (5) : cas Runge Kutta: erreur globale de la résolution
190         Tableau <double> indicateurs_resolution,indicateurs_resolution_t;
191     };
192
193     SaveResul * New_et_Initialise()
194     { SaveResulHysteresis1D * pt = new SaveResulHysteresis1D();
195       return pt;};
196
197     // Lecture des donnees de la classe sur fichier
198     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD&
199         lesFonctionsnD);
200     // affichage de la loi

```

```

200     void Affiche() const ;
201     // test si la loi est complete
202     // = 1 tout est ok, =0 loi incomplete
203     int TestComplet();
204
205     // calcul d'un module d'young equivalent à la loi, ceci pour un
206     // chargement nul
207     double Module_young_equivalent(Enum_dure temps,const Deformation & def ,SaveResul * saveResul);
208
209     // récupération de la variation relative d'épaisseur calculée: h/h0
210     // cette variation n'est utile que pour des lois en contraintes planes
211     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
212     // - pour les lois 2D def planes: retour de 0
213     // les infos nécessaires à la récupération , sont stockées dans saveResul
214     // qui est le conteneur spécifique au point où a été calculé la loi
215     virtual double HsurH0(SaveResul * saveResul) const
216     { cout << "\n HysteresisID::HsurH0(.. , methode non implante pour l'instant ";
217       Sortie(1);
218     };
219
220     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
221     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new HysteresisID(*this)); };
222
223     //----- lecture écriture de restart -----
224     // cas donne le niveau de la récupération
225     // = 1 : on récupère tout
226     // = 2 : on récupère uniquement les données variables (supposées comme telles)
227     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbesID&
lesCourbesID
228                                     ,LesFonctions_nD& lesFonctionsnD);
229
230     // cas donne le niveau de sauvegarde
231     // = 1 : on sauvegarde tout
232     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
233     void Ecriture_base_info_loi(ofstream& sort,const int cas);
234
235     // affichage et definition interactive des commandes particulières à chaque lois
236     void Info_commande_LoisDeComp(UtilLecture& lec);
237
238     // récupération des grandeurs particulière (hors ddl )
239     // correspondant à liTQ
240     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
241     virtual void Grandeur_particuliere
242     (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
243     ;
244     // récupération de la liste de tous les grandeurs particulières
245     // ces grandeurs sont ajoutées à la liste passées en paramètres
246     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
247     virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
248
249     protected :
250
251     // donnees protegees
252     // ---- paramètres matériaux ----
253     double xnp; // paramètre de Prager
254     CourbeID* xnp_temperature; // courbe éventuelle d'évolution de xnp en fonction de la température
255     int cas_prager; // indique si xnp =2 (=1), ou est compris entre 2 et 3 (=2), ou est sup à 3
(=3)
256     double Qzero; // limite de plasticité du critère de von mises
257     CourbeID* Qzero_temperature; // courbe éventuelle d'évolution de Qzero en fonction de la
température
258     double xmu; // paramètre de lame
259     CourbeID* xmu_temperature; // courbe éventuelle d'évolution de xmu en fonction de la température
260
261     // ---- paramètres de l'algorithme de Newton ----
262     double tolerance_residu; // tolérance absolue
263     double tolerance_residu_rel; // tolérance relative
264     double maxi_delta_var_sig_sur_iter_pour_Newton; // le maxi de variation que l'on tolère d'une
itération à l'autre
265     double tolerance_coincidence; // tolérance sur la précision de la coincidence
266     int nb_boucle_maxi; // le maximum d'itération de plasticité permis
267     int nb_sous_increment; // le maxi de sous incrément prévu
268     int type_resolution_equa_constitutive; // linéarisation ou kutta par exemple
269     int nb_maxInvCoinSurUnPas; // nombre maximum d'inversion ou de coincidence sur un pas
270
271     // ----- controle de la sortie des informations
272     // -> maintenant définit dans LoiAbstraiteGeneral
273     // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les
erreurs et warning
274     int sortie_post; // permet d'accéder au nombre d'itération, d'incrément, de précision etc. des
résolutions
275     // = 0 par défaut,
276     // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
277
278
279     // variables de travail pour l'échange entre les différentes méthodes en internes

```

```

280     Tenseur1BH sigma_t_barre_tdt; // sigma barre finale
281     Tenseur1BH sigma_i_barre_BH; // sigma barre de début de calcul (à t au début)
282     Tenseur1BH sigma_barre_BH_R; // sigma barre de Référence en cours
283     Tenseur1BH delta_sigma_barre_BH_Rat; // deltat sigma barre de R a t
284     Tenseur1BH delta_sigma_barre_BH_Ratdt; // deltat sigma barre de R a tdt
285     Tenseur1BH delta_sigma_barre_tdt_BH; // delta sigma barre de t à tdt
286     Tenseur1BH residuBH;
287     Tenseur1BH delta_barre_epsBH; // delta_barre epsilon totale
288     Tenseur1BH delta_barre_alpha_epsBH; // delta_barre epsilon intermediaire (avec alpha de 0 à 1)
289     double wprime; // paramètre de masing
290     Vecteur residu; // résidu de l'équation pour la résolution de l'équation constitutive
291     Mat_pleine derResidu; // dérivé du résidu de l'équation pour la résolution de l'équation
constitutive
292     Algo_zero alg_zero; // algo pour la recherche de zero
293     // ... partie relative à une résolution de l'avancement par une intégration de l'équation
différentielle
294     Algo_edp alg_edp;
295     int cas_kutta; // indique le type de runge_kutta que l'on veut utiliser
296     double erreurAbsolue,erreurRelative; // précision absolue et relative que l'on désire sur le
calcul de sig_tdt
297     int nbMaxiAppel; // nombre maxi d'appel de la fonction dérivée
298     Vecteur sig_point; // vitesse de sig: version vecteur de sigma_point
299     Tenseur1BH sigma_pointBH; // idem mais en tenseur
300     Tenseur1BH sigma_tauBH; // valeur de sigma pour le temps tau
301     Vecteur sigma_tau; // valeur de sigma pour le temps tau, en vecteur
302     Tenseur1BH delta_sigma_barre_R_a_tauBH; // delta sigma de R à tau
303     Tenseur1BH betaphideltasigHB,deuxmodeltaepsHB; // pour le calcul de la dérivée
304
305     // -- variables de travail internes à Residu_constitutif() et Mat_tangente_constitutif()
306     // définit ici pour éviter de les définir à chaque passage ds la méthode,
307     // ne doivent pas être utilisées en dehors de ces deux routines
308     Tenseur1BH rdelta_sigma_barre_BH_Ratdt; // deltat sigma barre de R a tdt
309     Tenseur1BH rdelta_sigma_barre_tdt_BH; // delta sigma barre de t à tdt
310
311     // ----- méthodes internes -----:
312     // affinage d'un point de coïncidence
313     // ramène true si le traitement est exactement terminé, sinon false, ce qui signifie qu'il
314     // faut encore continuer à utiliser l'équation d'évolution
315     // premiere_charge : indique si c'est oui ou non une coïncidence avec la première charge
316     // pt_sur_principal : indique si oui ou non les pointeurs iaftc et iatens pointent sur les
listes
317     // principales
318     // iatens_princ et iaftc_princ: pointeurs sur les listes principales
319     bool Coincidence(double& unSur_wprimeCarre,bool premiere_charge
,SaveResulHysteresis1D & save_resul,double& W_a
, List_io <Tenseur1BH>::iterator& iatens, List_io <double>::iterator& iaftc
, bool& pt_sur_principal, List_io <Tenseur1BH>::iterator& iatens_princ
, List_io <double>::iterator& iaftc_princ,double& delta_W_a);
324
325
326     // ----- codage des METHODES VIRTUELLES protegees -----:
327     // calcul des contraintes a t+dt
328     // calcul des contraintes a t+dt
329     // calcul des contraintes
330 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
331 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
332 ,TenseurBB & delta_epsBB_
333 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
334 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
335 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
, const Met_abstraite::Expli_t_tdt& ex);
337
338     // calcul des contraintes et de ses variations a t+dt
339 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
340 ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
341 ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
342 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
343 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
344 ,Tableau <TenseurBB *>& d_gijBB_tdt
345 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
346 ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
347 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
, const Met_abstraite::Impli& ex);
349
350
351
352     // fonction surchargée dans les classes dérivée si besoin est
353 virtual void CalculGrandeurTravail
354 (const PtIntegMecaInterne& ,const Deformation &
355 , Enum_dure,const ThermoDonnee&
356 ,const Met_abstraite::Impli* ex_impli
357 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
358 ,const Met_abstraite::Umat_cont* ex_umat
359 ,const List_io<Ddl_etendu>* exclure_dd_etend
360 ,const List_io<const TypeQuelconque *>* exclure_Q

```

```

361         ) {});
362
363         // initialisation éventuelle des variables thermo-dépendantes
364         void Init_thermo_dependance();
365         // méthode permettant le calcul de sigma à tdt par différentes méthodes: linéarisation
366         // ou kutta
367         void CalculContrainte_tdt(Tableau<double>& indicateurs_resolution);
368         // calcul de l'avancement temporel sur 1 pas,
369         // utilisé par les 3 programmes principaux:
370         // Calcul_SigmaHH, Calcul_DsigmaHH_tdt, Calcul_dsigma_deps,
371         void Avancement_temporel(const Tenseur1BB & delta_epsBB, const Tenseur1HH & gijHH
372             , SaveResulHysteresis1D & save_resul
373             , Tenseur1HH & sigHH);
374     public:
375         // calcul de la fonction résidu de la résolution de l'équation constitutive
376         // l'argument test ramène
377         //     . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
378         //     fatal, qui invalide le calcul du résidu
379         Vecteur& Residu_constitutif (const double & alpha, const Vecteur & x, int& test);
380         // calcul de la matrice tangente de la résolution de l'équation constitutive
381         // l'argument test ramène
382         //     . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
383         //     fatal, qui invalide le calcul du résidu et de la dérivée
384         Mat_abstraite& Mat_tangente_constitutif(const double & alpha, const Vecteur & x, Vecteur& resi, int&
385             test);
386         // calcul de l'opérateur tangent : dsigma/epsilon
387         TenseurQlgeneBBBH& Dsig_depsilon(TenseurQlgeneBBBH& dsig_deps);
388         // calcul de l'expression permettant d'obtenir la dérivée temporelle de la contrainte
389         // à un instant tau quelconque, en fait il s'agit de l'équation constitutive
390         // utilisée dans la résolution explicite (runge par exemple) de l'équation constitutive
391         // erreur : =0: le calcul est licite, si diff de 0, indique qu'il y a eu une erreur
392         //     =1: la norme de sigma est supérieure à la valeur limite de saturation
393         Vecteur& Sigma_point(const double & tau, const Vecteur & sigma_tau, Vecteur& sig_point, int &
394             erreur);
395         // vérification de l'intégrité du sigma calculé
396         // erreur : =0: le calcul est licite, si diff de 0, indique qu'il y a eu une erreur
397         //     =1: la norme de sigma est supérieure à la valeur limite de saturation
398         void Verif_integrite_Sigma(const double & tau, const Vecteur & sigma_tau, int & erreur);
399     };
400     /// @} // end of group
401
402 #endif
403
404
405

```

## 7.57 Hysteresis3D.h

```

1 // FICHER : Hysteresis3D.h
2 // CLASSE : Hysteresis3D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:      30/06/2004
35 *

```



```

36 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)           *
37 *                                                     $   *
38 *   PROJET:      Herezh++                                     *
39 *                                                     $   *
40 * *****
41 *   BUT:   La classe Hysteresis3D permet de calculer la contrainte *
42 *   et ses derivees pour une loi d'hysteresis 3D, type           *
43 *   celle développée par Guelin, Favier, Pegon.                 *
44 *                                                     $   *
45 *   *****
46 *   VERIFICATION:                                             *
47 *   ! date ! auteur ! but ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
48 *   ----- ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
49 *   ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
50 *   ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
51 *   ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
52 *   *****
53 *   MODIFICATIONS:                                           *
54 *   ! date ! auteur ! but ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
55 *   ----- ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
56 *   ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
57 * *****/
58
59
60 #ifndef HYSTERESIS_3D_H
61 #define HYSTERESIS_3D_H
62
63 #include "Vecteur.h"
64 #include "Loi_comp_abstraite.h"
65 #include "CourbelD.h"
66 #include "TypeConsTens.h"
67 #include "Algo_zero.h"
68 #include "TenseurQlgene.h"
69 #include "MatLapack.h"
70 #include "Algo_edp.h"
71 #include "TenseurQ3gene.h"
72
73
74 /// @addtogroup Les_lois_hysteresis
75 /// @{
76 ///
77
78 class Hysteresis3D : public Loi_comp_abstraite
79 {
80 public :
81
82     // CONSTRUCTEURS :
83
84     // Constructeur par défaut
85     Hysteresis3D ();
86
87
88     // Constructeur de copie
89     Hysteresis3D (const Hysteresis3D& loi) ;
90
91     // DESTRUCTEUR :
92
93     ~Hysteresis3D ();
94
95     // initialise les donnees particulieres a l'elements
96     // de matiere traite ( c-a-dire au pt calcule)
97     // Il y a creation d'une instance de SaveResul particuliere
98     // a la loi concerne
99     // la SaveResul classe est remplie par les instances heritantes
100    // le pointeur de SaveResul est sauvegarde au niveau de l'element
101    // c'a-d que les info particulieres au point considere sont stocke
102    // au niveau de l'element et non de la loi.
103    class SaveResulHysteresis3D: public SaveResul
104    { public :
105        SaveResulHysteresis3D(); // constructeur par défaut :
106        SaveResulHysteresis3D(const SaveResulHysteresis3D& sav); // de copie
107        virtual ~SaveResulHysteresis3D(); // destructeur
108        // définition d'une nouvelle instance identique
109        // appelle du constructeur via new
110        SaveResul * Nevez_SaveResul() const{return (new SaveResulHysteresis3D(*this));};
111    // affectation
112    virtual SaveResul & operator = ( const SaveResul & a);
113    //===== lecture écriture dans base info =====
114    // cas donne le niveau de la récupération
115    // = 1 : on récupère tout
116    // = 2 : on récupère uniquement les données variables (supposées comme telles)
117    void Lecture_base_info (ifstream& ent,const int cas);
118    // cas donne le niveau de sauvegarde
119    // = 1 : on sauvegarde tout
120    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
121    void Ecriture_base_info(ofstream& sort,const int cas);

```

```

122
123 // mise à jour des informations transitoires
124 void TdtversT();
125 void TversTdt();
126
127 // affichage à l'écran des infos
128 void Affiche() const;
129
130 //changement de base de toutes les grandeurs internes tensorielles stockées
131 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
132 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
133 // gpH(i) = gamma(i,j) * gH(j)
134 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
135
136 // procedure permettant de completer éventuellement les données particulières
137 // de la loi stockées
138 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
139 // completer est appelé apres sa creation avec les donnees du bloc transmis
140 // peut etre appeler plusieurs fois
141 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
142 ,const Loi_comp_abstraite* loi) {};
143
144 // ---- méthodes spécifiques
145 // initialise les informations de travail concernant le pas de temps en cours
146 void Init_debut_calcul();
147 // vérif de la cohérence des centres et références: les rayons associés doivent être de taille
148 // décroissante: programme utilisé uniquement en debug
149 void Verif_centre_reference(const int& permet_affichage);
150
151 //-----
152 // données
153 //-----
154
155 // 1) ===== valeurs courantes (que l'on pourra peut-être économiser par la suite)
156
157 Tenseur3BH sigma_barre_BH_t; // dernière contrainte en BH à t
158 Tenseur3BH sigma_barre_BH_tdt; // contrainte en cours BH à tdt
159 // les contraintes sigma_barre_BH_t et sigma_barre_BH_tdt sont celles obtenues
160 // à partir de l'équation constitutive en mixte BH,
161 // ce n'est donc pas la contrainte finale (qui utilise la dérivée de jauman)
162 double fonction_aide_t; // dernière valeur de la fonction d'aide
163 double fonction_aide_tdt; // valeur de la fonction d'aide en cours
164 List_io <double>::iterator ip2; // adresse éventuelle du 2 éléments de fct_aide
165
166 Tenseur3BH oc_BH_t; // dernière valeur du centre de référence
167 Tenseur3BH oc_BH_tdt; // valeur du centre de référence en cours
168 int wBase_t,wBase_tdt; // paramètre de masing
169 double def_equi_at; // valeur de la def équivalente à t
170 double def_equi_atdt; // valeur de la def équivalente à tdt
171 // fonction d'aide
172 // double fonction_aide_t; // dernière valeur de la fonction d'aide
173 // double fonction_aide_tdt; // valeur de la fonction d'aide en cours
174 // List_io <double>::iterator ip2; // adresse éventuelle du 2 éléments de fct_aide
175
176 // 2) a) ===== informations de travail concernant le pas de temps en cours, c-a-d de t à
177 t+tdt
178 int modif; // = 0 rien de changé,
179 // = 1 une ou plusieurs coïncidence(s) (sans inversion),
180 // = 2 une ou plusieurs inversion(s) (sans coïncidence),
181 // = 3 un ensemble d'une ou plusieurs coïncidences de inversions
182 // liste des nouvelles contraintes de référence, qui sont apparus
183 List_io <Tenseur3BH> sigma_barre_BH_R_t_a_tdt;
184 int nb_coïncidence; // nombre de coïncidence durant le pas de temps
185 // liste des centres (contraintes) de référence qui sont apparues
186 List_io <Tenseur3BH> oc_BH_t_a_tdt;
187 List_io <double> def_equi_t_a_tdt; // liste des def équivalentes
188 //aux points d'inversions durant le pas de temps
189 List_io <double> fct_aide_t_a_tdt; // liste des valeurs de la fonction d'aide durant le pas
190 // de temps
191 // 2) b) ===== sauvegarde des variable de 2) a) pour pouvoir revenir en arrière : de tdt
192 vers t
193 // ces variables ne sont pas sauvegardées dans les .BI
194 // int modif_sauve; // liste des nouvelles contraintes de référence, qui sont apparus
195 // List_io <Tenseur3BH> sigma_barre_BH_R_sauve;
196 // int nb_ref_new_sauve;
197 // int nb_coïncidence_sauve; // nombre de coïncidence durant le pas de temps
198 // // liste des centres (contraintes) de référence qui sont apparues
199 // List_io <Tenseur3BH> oc_BH_R_sauve;
200 // int nb_centre_new_sauve;
201 // List_io <double> def_equi_sauve; // liste des def équivalentes
202 // //aux points d'inversions durant le pas de temps
203
204 // 3) --- informations de mémorisation discrète de 0 à t -----
205 // le dernier élément rangé est en .begin() (c-a-d front())
206 List_io <Tenseur3BH> sigma_barre_BH_R; // liste des contraintes de référence
207 List_io <Tenseur3BH> oc_BH_R; // liste des centres (contraintes) de référence
208 // il y a un élément de moins dans la liste oc_BH_R que dans la liste sigma_barre_BH_R

```

```

207         // car sur la première charge, le centre c'est 0, et la ref de sigma = 0 aussi
208         // sur la deuxième charge: le centre c'est toujours 0, et il y a un sigma de ref non nulle
209         // sur une autre charge : on a un nouveau centre et 2 sig de ref, puis etc par la suite
210         List_io <double> def_equi; // liste des def équivalentes aux points d'inversions
211         List_io <double> fct_aide; // liste des valeurs de la fonction d'aide
212
213         // 4) --- sauvegarde des informations de mémorisation discrète de 0 à t avant transport
214         -----
215         // le transport sur toute la mémorisation tensorielle à lieu à chaque calcul !!
216         // donc il faut pouvoir revenir en arrière
217         // ces grandeurs ne sont pas sauvegardées, elles sont utilisée pour t->tdt uniquement
218         List_io <Tenseur3BH> sigma_barre_BH_R_atrans; // liste des contraintes de référence
219         List_io <Tenseur3BH> oc_BH_R_atrans; // liste des centres (contraintes) de référence
220
221         // 5) --- tableau d'indicateur de la résolution, éventuellement vide
222         // cela dépend de sortie_post
223         // indicateurs_resolution(1) : nb d'incrément utilisé pour la résolution de l'équation
224         // linéarisée
225         // (2) : nb total d'itération " " " " " "
226         // (3) : cas Runge Kutta: nb d'appel de la fonction
227         // (4) : cas Runge Kutta: nb de step de calcul
228         // (5) : cas Runge Kutta: erreur globale de la résolution
229         Tableau <double> indicateurs_resolution,indicateurs_resolution_t;
230
231     };
232
233     SaveResul * New_et_Initialise()
234     { SaveResulHysteresis3D * pt = new SaveResulHysteresis3D();
235       return pt;
236     };
237
238     friend class SaveResulHysteresis3D;
239
240     // Lecture des donnees de la classe sur fichier
241     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD&
242     lesFonctionsnD);
243     // affichage de la loi
244     void Affiche() const ;
245     // test si la loi est complete
246     // = 1 tout est ok, =0 loi incomplete
247     int TestComplet();
248
249     // calcul d'un module d'young équivalent à la loi, ceci pour un
250     // chargement nul
251     double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
252
253     // récupération de la variation relative d'épaisseur calculée: h/h0
254     // cette variation n'est utile que pour des lois en contraintes planes
255     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
256     // - pour les lois 2D def planes: retour de 0
257     // les infos nécessaires à la récupération , sont stockées dans saveResul
258     // qui est le conteneur spécifique au point où a été calculé la loi
259     virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
260
261     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
262     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hysteresis3D(*this)); };
263
264     //----- lecture écriture de restart -----
265     // cas donne le niveau de la récupération
266     // = 1 : on récupère tout
267     // = 2 : on récupère uniquement les données variables (supposées comme telles)
268     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
269     lesCourbes1D
270     ,LesFonctions_nD& lesFonctionsnD);
271
272     // cas donne le niveau de sauvegarde
273     // = 1 : on sauvegarde tout
274     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
275     void Ecriture_base_info_loi(ofstream& sort,const int cas);
276
277     // affichage et definition interactive des commandes particulières à chaque lois
278     void Info_commande_LoisDeComp(UtilLecture& lec);
279
280     // récupération des grandeurs particulière (hors ddl )
281     // correspondant à liTQ
282     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
283     virtual void Grandeur_particuliere
284     (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
285     ;
286     // récupération de la liste de tous les grandeurs particulières
287     // ces grandeurs sont ajoutées à la liste passées en paramètres
288     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
289     virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
290
291     protected :
292
293     // donnees protegees

```

```

289 // ---- paramètres matériaux -----
290 double xnp,xnp_lue; // paramètre de Prager
291 CourbelD* xnp_temperature; // courbe éventuelle d'évolution de xnp en fonction de la température
292 CourbelD* xnp_phase; // courbe éventuelle d'évolution de xnp en fonction de la phase
293 CourbelD* xnp_defEqui; // courbe éventuelle d'évolution de xnp en fonction de la déformation
    équivalente
294 double Qzero,Qzero_lue; // limite de plasticité du critère de von mises
295 CourbelD* Qzero_temperature; // courbe éventuelle d'évolution de Qzero en fonction de la
    température
296 CourbelD* Qzero_phase; // courbe éventuelle d'évolution de Qzero en fonction de la phase
297 CourbelD* Qzero_defEqui; // courbe éventuelle d'évolution de Qzero en fonction de la déformation
    équivalente
298 double xmu,xmu_lue; // paramètre de lame
299 CourbelD* xmu_temperature; // courbe éventuelle d'évolution de xmu en fonction de la température
300 CourbelD* xmu_phase; // courbe éventuelle d'évolution de xmu en fonction de la phase
301 // deux types de dépendance à la phase: soit Delta_S_Oi_tdt, soit la déformation totale, ceci pour
    xnp,Qzero,xmu
302 // 1 - prise en compte éventuelle de la phase (de Delta_S_Oi_tdt par rapport aux axes principaux
    dans le plan déviat)
303 bool avec_phaseDeltaSig; // indique si oui ou non on a de la phase
304 // 2 - prise en compte éventuelle de la phase de la déformation
305 bool avec_phaseEps;
306 // dépendance éventuelle à la déformation équivalente
307 bool avec_defEqui;
308 // dépendance éventuelle à la phase de la déformation totale pour un calcul de val_coef_paradefEqui,
309 // s'emploie en conjonction avec defEauI
310 bool avec_phase_paradefEqui;
311 CourbelD* coef_paradefEqui;
312 double val_coef_paradefEqui; // valeur courante du coef
313 double mini_QepsilonBH; // le minimum de Qsig pour le calcul de la phase de delta_sigma_Oi_tdt
314
315
316 // ----- paramètres de l'algorithme -----
317 double tolerance_residu; // tolérance absolu sur la résolution de la plasticité
318 double tolerance_residu_rel; // tolérance relative sur la résolution de la plasticité
319 double maxi_delta_var_sig_sur_iter_pour_Newton; // le maxi de variation que l'on tolère d'une
    itération à l'autre
320 double tolerance_coincidence; // tolérance sur la précision de la coïncidence
321 double tolerance_centre; // tolérance sur la précision du calcul des centres
322 int nb_boucle_maxi; // le maximum d'itération de plasticité permis
323 int nb_dichotomie; // le maxi de dichotomie prévu pour l'équation de Newton
324 double mini_rayon; // mini en dessous duquel on considère les rayons et accroissements nuls
325 int type_resolution_equa_constitutive; // linéarisation ou kutta par exemple
326 int type_calcul_comportement_tangent; // indique le type de calcul du comportement tangent
327 bool avecVarWprimeS; // prise en compte ou non des variations du cos et du cosPoint pour le calcul
    tangent
328 int nb_maxInvCoinSurUnPas; // nombre maximum d'inversion ou de coïncidence sur un pas
329 double min_angle_trajet_neutre; // le minimum du cos(d'angle) en dessous duquel le trajet est
    considéré neutre
330 bool possibilite_cosAlphaNegatif; // indique si oui ou non on tolère transitoirement un cosAlpha
    négatif (près d'une inversion par axe)
331 double mini_Qsig_pour_phase_sigma_Oi_tdt; // le minimum de Qsig pour le calcul de la phase de
    delta_sigma_Oi_tdt
332 double force_phi_zero_si_negatif; // para pour forcer le phi à 0 si ça valeur négative devient sup
333 double depassement_Q0; // valeur en relatif de dépassement permis sur la saturation: devrait = 1, mais
    dans les faits il faut laisser une marge (1.2 par axe), par défaut très grand pour l'instant, je ne
    sais pas pourquoi !!
334 // def du type de transport des grandeurs mémorisée: = 0, transport en mixte (historique)
335 int type_de_transport_mémorisation; //=-1 : transport cohérent avec la dérivée de Jauman
336 // def le niveau (entre 0 et 1) du rayon par rapport au rayon maxi de Q0 à partir duquel on utilise
337 double niveau_bascule_fct_aide; // la fonction d'aide
338
339 // ---- controle de la sortie des informations
340 // -> maintenant définit dans LoiAbstraiteGeneral
341 // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les erreurs
    et warning
342 int sortie_post; // permet d'accéder au nombre d'itération, d'incrément, de précision etc. des
    résolutions
343 // = 0 par défaut,
344 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
345
346 // variables de travail pour l'échange entre les différentes méthodes en internes
347 Tenseur3BH sigma_t_barre_tdt; // sigma barre finale
348 Tenseur3BH sigma_i_barre_BH; // sigma barre de début de calcul (à t au début)
349 Tenseur3BH sigma_barre_BH_R; // sigma barre de Référence en cours
350 double def_equi_R; // déformation équivalente à R
351 double def_i_equi; // déformation équivalente de de début de calcul (comme sigma_i_barre_BH)
352 double defEqui_maxi; // defEqui maxi sur la courbe de première charge
353 double def_equi_atdt; // déformation équivalente à tdt
354 Tenseur3BH delta_sigma_barre_BH_Rat; // deltat sigma barre de R a t
355 Tenseur3BH delta_sigma_barre_BH_Ra_i; // deltat sigma barre de R a i
356 Tenseur3BH delta_sigma_barre_BH_Ratdt; // deltat sigma barre de R a tdt
357 Tenseur3BH delta_sigma_barre_tdt_BH; // delta sigma barre de t à tdt
358
359 Tenseur3BH delta_sigma_barre_BH_OiAR; // delta sigma barre de Oi à R
360 double Q_OiAR; // norme de delta_sigma_barre_BH_OiAR
361 double QdeltaSigmaRatdt; // norme de rdelta_sigma_barre_BH_Ratdt

```

```

362     Tenseur3BH oc_BH_R;
363
364     // -- variables internes à Calcul_SigmaHH et Calcul_DsigmaHH_tdt
365 //     algo de newton
366 Vecteur val_initiale; // val initiale
367 Vecteur racine; // racine finale
368 MatLapack der_at_racine; // dérivée du résidu locale
369
370 bool centre_initial; // pour gérer le fait qu'au début le centre initial = tenseur nul
371 bool aucun_pt_inversion; // pour gérer le fait qu'au début il n'y a pas de pt d'inversion !
372
373 Tenseur3BH residuBH;
374 Tenseur3BH delta_epsBH; // delta epsilon totale
375 Tenseur3BH delta_barre_epsBH; // delta_barre epsilon totale
376 Tenseur3BH delta_barre_alpha_epsBH; // delta_barre epsilon intermediaire (avec alpha de 0 à 1)
377 int wBase; // paramètre de masing
378 double wprime,wprime_point; // paramètre de masing modifié et sa dérivée temporelle
379 bool trajet_neutre; // indique si oui ou non le trajet est neutre: associé au calcul de
wprime
380 Tenseur3BH d_wprime; // variation de wprime par rapport à deltat sigma barre de R a tdt
381 Tenseur3BH d_wprime_point; // variation de wprime_point par rapport à deltat sigma barre de R a
tdt
382 Vecteur residu; // résidu de l'équation pour la résolution de l'équation constitutive
383 MatLapack derResidu; // dérivé du résidu de l'équation pour la résolution de l'équation
constitutive
384 // correspond également à la dérivée du résidu à déformation BH constante
385 MatLapack derResidu_adeConstant; // dérivé du résidu de l'équation constitutive, à contrainte cte
386 Algo_zero alg_zero; // algo pour la recherche de zero
387
388 // ... partie relative à une résolution de l'avancement par une intégration de l'équation
différentielle
389 Algo_edp alg_edp;
390 int cas_kutta; // indique le type de runge_kutta que l'on veut utiliser
391 double erreurAbsolue,erreurRelative; // précision absolue et relative que l'on désire sur le calcul de
sig_tdt
392 int nbMaxiAppel; // nombre maxi d'appel de la fonction dérivée
393 Vecteur sig_point; // vitesse de sig: version vecteur de sigma_point
394 Tenseur3BH sigma_pointBH; // idem mais en tenseur
395 Tenseur3BH sigma_tauBH; // valeur de sigma pour le temps tau
396 Vecteur sigma_tau; // valeur de sigma pour le temps tau, en vecteur
397 Tenseur3BH delta_sigma_barre_R_a_tauBH; // delta sigma de R à tau
398 Tenseur3BH delta_sigma_barre_t_a_tauBH; // delta sigma barre de t à tau
399 Tenseur3BH betaphideltasigHB,deuxmudeltaepsHB; // pour le calcul de la dérivée
400 Vecteur sig_initiale,dersig_initiale; // pour CalculContrainte_tdt
401 Vecteur sig_finale,dersig_finale; // " "
402 Vecteur estime_erreur;
403
404 // -- variables de travail internes à Residu_constitutif() et Mat_tangente_constitutif()
405 // définit ici pour éviter de les définir à chaque passage ds la méthode,
406 // ne doivent pas être utilisées en dehors de ces deux routines
407 Tenseur3BH rdelta_sigma_barre_BH_Ratdt; // deltat sigma barre de R a tdt
408 Tenseur3BH rdelta_sigma_barre_tdt_BH; // delta sigma barre de t à tdt
409
410 // variable de travail commune à Mat_tangente_constitutif() et Dsig_d_ddl et Calcul_DsigmaHH_tdt
411 bool calcul_dResi_dsig; // indique si oui ou non le calcul de la matrice a été effectué
412
413 // -- variables de travail pour la méthode: Dsig_d_ddl
414 MatLapack dRdsigMoinsun; // inverse de der_at_racine
415 MatLapack dRdeps; // variation du résidu constitutif / aux déformations
416 MatLapack MPC; // produit de dRdsigMoinsun et de dRdeps
417 // -- variable de travail pour Hysteresis3D:Dsig_depsilon
418 TenseurQ3geneHHHH dsig_ojef_HHHH;
419
420
421 // -- variables internes à la méthode Dsig_d_Runge
422 MatLapack matriceH, matriceM,matHmoinsun;
423
424 // ----- méthodes internes -----:
425 // 1) affinage d'un point de coïncidence à l'aide des rayons
426 // ramène true si le traitement est exactement terminé, sinon false, ce qui signifie qu'il
427 // faut encore continuer à utiliser l'équation d'évolution
428 // pt_sur_principal : indique si oui ou non le pointeur iatens pointe sur sa liste principale
429 // oi_sur_principal : indique si oui ou non le pointeur iaOi pointe sur sa liste principale
430 // iatens_princ et iaftct_princ: pointeurs sur les listes principales
431 // force_coïncidence : on force la coïncidence à la position actuelle
432 // cas très particulier où l'algo normal n'aboutie pas, et que le rayon finale est
433 // plus grand que le dernier rayon de référence
434 // le rayon peut-être modifié dans le cas particulier d'une coïncidence à la précision près
435 bool Coïncidence(bool bascule_fct_aide,double& unSur_wBaseCarre
436 ,const Tenseur3HH & gijHH,const Tenseur3BB & gijBB
437 ,SaveResulHysteresis3D & save_resul,double& W_a
438 ,List_io <Tenseur3BH>::iterator& iatens
439 ,List_io <double>::iterator& iadef_equi
440 ,List_io <double>::iterator& iaftct
441 ,bool& pt_sur_principal
442 ,List_io <Tenseur3BH>::iterator& iaOi,bool& oi_sur_principal

```

```

443         ,List_io <Tenseur3BH>::iterator& iatens_princ
444         ,List_io <double>::iterator& iadef_equi_princ
445         ,List_io <Tenseur3BH>::iterator& iaOi_princ
446         ,List_io <double>::iterator& iafct_princ
447         ,const double& rayon_tdt,bool force_coincidence);
448
449     // 2) affinage d'un point de coincidence à l'aide de la fonction d'aide
450     // ramène true si le traitement est exactement terminé, sinon false, ce qui signifie qu'il
451     // faut encore continuer à utiliser l'équation d'évolution
452     // premiere_charge : indique si c'est oui ou non une coincidence avec la première charge
453     // pt_sur_principal : indique si oui ou non les pointeurs iafct et iatens pointent sur les listes
454     // principales
455     // iatens_princ et iafct_princ: pointeurs sur les listes principales
456     bool Coincidence(bool bascule_fct_aide, double& unSur_wprimeCarre
457         ,const Tenseur3HH & gijHH,const Tenseur3BB & gijBB
458         ,SaveResulHysteresis3D & save_resul,double& W_a
459         ,List_io <Tenseur3BH>::iterator& iatens
460         ,List_io <double>::iterator& iadef_equi
461         ,List_io <double>::iterator& iafct
462         ,bool& pt_sur_principal
463         ,List_io <Tenseur3BH>::iterator& iaOi,bool& oi_sur_principal
464         ,List_io <Tenseur3BH>::iterator& iatens_princ
465         ,List_io <double>::iterator& iadef_equi_princ
466         ,double& position_coincidence
467         ,List_io <Tenseur3BH>::iterator& iaOi_princ
468         ,List_io <double>::iterator& iafct_princ,double& delta_W_a,bool
469         force_coincidence);
470
471
472     // ----- codage des METHODES VIRTUELLES protegees -----:
473     // calcul des contraintes a t+dt
474     // calcul des contraintes
475     void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
476         ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH & gi_H, TenseurBB & epsBB_
477         ,TenseurBB & delta_epsBB_
478         ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
479         ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
480         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
481         module_cisaillement
482         ,const Met_abstraite::Expli_t_tdt& ex);
483
484     // calcul des contraintes et de ses variations a t+dt
485     void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
486         ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
487         ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH & giH_tdt,Tableau <BaseH> & d_giH_tdt
488         ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
489         ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
490         ,Tableau <TenseurBB *>& d_gijBB_tdt
491         ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
492         ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
493         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
494         module_cisaillement
495         ,const Met_abstraite::Impli& ex);
496
497     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
498     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
499     // le tenseur de déformation et son incrémentsont également en orthonormees
500     // si = false: les bases transmises sont utilisées
501     // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
502     void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
503         ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
504         ,TenseurHH& sigHH,TenseurHHH& d_sigma_deps
505         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
506         module_cisaillement
507         ,const Met_abstraite::Umat_cont& ex) ; // = 0;
508
509     // calcul de grandeurs de travail aux points d'intégration via la def
510     // fonction surchargée dans les classes dérivée si besoin est
511     virtual void CalculGrandeurTravail
512         (const PtIntegMecaInterne& ,const Deformation &
513         ,Enum_dure,const ThermoDonnee&
514         ,const Met_abstraite::Impli* ex_impli
515         ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
516         ,const Met_abstraite::Umat_cont* ex_umat
517         ,const List_io<Ddl_etendu>* exclure_dd_etend
518         ,const List_io<const TypeQuelconque *>* exclure_0
519         ) {};
520
521 public:
522     // calcul de la fonction résidu de la résolution de l'équation constitutive
523     // l'argument test ramène
524     // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
525     // fatal, qui invalide le calcul du résidu
526     Vecteur& Residu_constitutif (const double & alpha, const Vecteur & x, int& test);
527     // calcul de la matrice tangente de la résolution de l'équation constitutive

```

```

526 // l'argument test ramène
527 // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
528 // fatal, qui invalide le calcul du résidu et de la dérivée
529 Mat_abstraite& Mat_tangente_constitutif(const double & alpha,const Vecteur & x, Vecteur& resi,
int& test);
530 // calcul de l'opérateur tangent : dsigma/epsilon, dans le cas d'un repère ortho-normé ou non
531 TenseurHHHH& Dsig_depsilon(bool en_base_orthonormee,const Tenseur3HH & gijHH_tdt , TenseurHHHH&
dsig_deps
532 ,TenseurHH & sigHH);
533 // calcul de l'opérateur tangent : dsigma/dddl
534 void Dsig_d_ddl(Tableau <TenseurHH *>& d_sigHH,TenseurBB & epsBB_tdt
535 ,TenseurHH & gijHH_tdt,Tableau <TenseurBB *>& d_gijBB_tdt
536 ,Tableau <TenseurHH *>& d_gijHH_tdt,Tableau <TenseurBB *>& d_epsBB );
537 // calcul de l'expression permettant d'obtenir la dérivée temporelle de la contrainte
538 // à un instant tau quelconque, en fait il s'agit de l'équation constitutive
539 // utilisée dans la résolution explicite (runge par exemple) de l'équation constitutive
540 // sinon (diff de 0) indique que les valeurs calculées ne sont pas licite
541 // erreur : =0: le calcul est licite, si diff de 0, indique qu'il y a eu une erreur
542 // =1: la norme de sigma est supérieure à la valeur limite de saturation
543 Vecteur& Sigma_point(const double & tau, const Vecteur & sigma_tau,Vecteur& sig_point,int&
erreur);
544 // vérification de l'intégrité du sigma calculé
545 // erreur : =0: le calcul est licite, si diff de 0, indique qu'il y a eu une erreur
546 // =1: la norme de sigma est supérieure à la valeur limite de saturation
547 void Verif_integrite_Sigma(const double & tau, const Vecteur & sigma_tau,int & erreur);
548 // ramène le type de transport
549 int Type_de_transport_tenseur() const {return type_de_transport_memorisation;};
550
551 protected:
552 // --- définition de différentes fonctions utiles pour le calcul final
553 // calcul de wprime en fonction de grandeurs supposées déjà calculées
554 // utilise : Q_OiaR, QdeltaSigmaRatdt,
555 // delta_sigma_barre_BH_Ratdt, delta_sigma_barre_BH_OiaR
556 // wBase
557 // trajet_neutre: indique si oui ou non, on est dans un chargement neutre à la précision près
558 void Cal_wprime(const double& QdeltaSiRatdt,const Tenseur3BH& delta_sig_barre_BH_Ratdt
559 ,const double& Q_OiaR,const Tenseur3BH& delta_sig_barre_BH_OiaR
560 ,const double& w_Base,Tenseur3BH& rdelta_sigma_barre_tdt_BH
561 ,double& w_prime, double& w_prime_point,bool& trajet_neutre) const;
562 // calcul de wprime et de sa variation par rapport aux coordonnées de delta sigma barre
563 // trajet_neutre: indique si oui ou non, on est dans un chargement neutre à la précision près
564 void Cal_wprime_et_der(const double& QdeltaSiRatdt,const Tenseur3BH& delta_sig_barre_BH_Ratdt
565 ,const double& Q_OiaR,const Tenseur3BH& delta_sig_barre_BH_OiaR
566 ,const double& w_Base,Tenseur3BH& rdelta_sigma_barre_tdt_BH
567 ,double& w_prime, Tenseur3BH& d_w_prime
568 ,double& w_prime_point,Tenseur3BH& d_wprime_point,bool& trajet_neutre) const;
569 // calcul du centre d'inversion avec les données actuelles
570 // ramène un indicateur informant comment l'opération s'est passé et si elle est valide (= 0)
571 int CentreCercle(Tenseur3BH& sigO1_BH,const Tenseur3BH& oc_BH_R
572 ,const Tenseur3BH & delta_sigma_barre_BH_OiaR
573 ,const Tenseur3BH& delta_sigma_barre_BH_Rat);
574 // gestion d'un dépilement des pointeurs dans les listes, dans le cas d'une coïncidence
575 // met à jour les booléens interne à l'instance : aucun_pt_inversion et centre_initial
576 void Gestion_pointeur_coïncidence(double& unSur_wBaseCarre
577 ,SaveResulHysteresis3D & save_resul,double& W_a
578 ,List_io <Tenseur3BH>::iterator& iatens
579 ,List_io <double>::iterator& iadef_equi
580 ,List_io <double>::iterator& iaftct
581 ,bool& pt_sur_principal
582 ,List_io <Tenseur3BH>::iterator& iaO1,bool& oi_sur_principal
583 ,List_io <Tenseur3BH>::iterator& iatens_princ
584 ,List_io <double>::iterator& iadef_equi_princ
585 ,List_io <Tenseur3BH>::iterator& iaO1_princ
586 ,List_io <double>::iterator& iaftct_princ);
587 // gestion du paramètre "modif" de saveresult
588 // inversion = true: on met à jour après une inversion
589 // = false : on met à jour après une coïncidence
590 void Gestion_para_Saveresult_Modif
591 (const bool& pt_sur_principal,SaveResulHysteresis3D & save_resul
592 ,const bool& inversion
593 );
594
595 // traitement des cas où des autres coïncidences existent à la précision près
596 // ramène true si effectivement il y a une nouvelle coïncidence
597 bool Autre_coïncidence_a_la_tolerance_pres(bool bascule_fct_aide
598 ,const Tenseur3HH & gijHH,const Tenseur3BB & gijBB
599 ,const bool& pt_sur_principal,const bool& oi_sur_principal
600 ,List_io <Tenseur3BH>::iterator& iatens
601 ,List_io <Tenseur3BH>::iterator& iaO1);
602 // calcul de la première phase de l'opérateur tangent, est utilisé par Dsig_d_ddl et
603 // Dsig_depsilon
604 // -> calcul de la matrice M
605 void Dsig_d_(MatLapack& MPC);
606 // idem que Dsig_d_, mais pour le cas où l'on calcul l'avancement à l'aide d'un runge_Kutta
607 // -> calcul de la matrice M
608 void Dsig_d_Runge(MatLapack& MPC);
609

```



```

610 // calcul de l'avancement temporel sur 1 pas,
611 // utilisé par les 3 programmes principaux:
612 // Calcul_SigmaHH, Calcul_DsigmaHH_tdt, Calcul_dsigma_deps,
613 // int cas: =1 : normal, on symétrise le tenseur des contraintes en fonction de la dérivée
614 // de Jauman
615 // int cas = 2 : pas de symétrisation du tenseur des contraintes à la fin du calcul
616 void Avancement_temporel(const Tenseur3BB & delta_epsBB, const Tenseur3HH & gijHH
617 , const Tenseur3BB & gijBB, int cas, SaveResulHysteresis3D & save_resul
618 , Tenseur3HH & sigHH, const EnergieMeca & energ_t, EnergieMeca & energ);
619
620 // initialisation éventuelle des variables thermo-dépendantes
621 void Init_thermo_dependance();
622 // méthode permettant le calcul de sigma à tdt par différentes méthodes: linéarisation
623 // ou kutta
624 // en sortie calcul de :
625 // sigma_t_barre_tdt, delta_sigma_barre_tdt_BH, delta_sigma_barre_BH_Ratdt
626 // également les grandeurs: QdeltaSigmaRatdt, et wprime, wprime_point, calculées à tdt
627 // ramène en retour la valeur de phi_2_dt
628 void CalculContrainte_tdt(double & phi_2_dt, Tableau<double>& indicateurs_resolution);
629 // update des différentes énergies sur le pas de temps ou la partie de pas de temps
630 // effectué, et calcul de defEqui finalisée
631 void UpdateEnergieHyste(EnergieMeca & energ
632 , const Tenseur3BH & sigma_deb_BH, const Tenseur3BH & sigma_fin_BH
633 , const Tenseur3BH & delta_sigma_sur_tau_BH
634 , const Tenseur3BH & delta_sigma_R_tau_BH
635 , const double & phi_2_dt
636 , const Tenseur3BH & delta_eps_sur_deltaTau_BH, double & QdeltaSigmaRatdt
637 , const double & wprime, const double & w_prime_point, const bool & trajet_neutre
638 , const double & def_equi_R, const double & def_i_equi, double & def_equi);
639 // calcul de la déformation cumulée, et de la déformation maxi à prendre en compte
640 void DefEqui_et_maxi(double & QdeltaSigmaRatdt, const bool & trajet_neutre, double & defEquiMaxi
641 , const double & def_equi_R, const double & def_i_equi
642 , double & def_equi, const double & phi_2_dt, const double & wprime);
643
644 // calcul des paramètres matériaux dépendants de la phase de deltaSig_Oi^tdt
645 // en entrée: les paramètres sans phase = paramètres lues ou fonction de la température
646 // en sortie: les paramètres en tenant compte de la phase de delta_sigma_Oi_tdt
647 void ParamateriauxAvecPhaseDeltaSig(double & xnp, const Tenseur3BH & delta_sigma_barre_BH_OiAR
648 , const Tenseur3BH & delta_sig_barre_BH_Ratdt
649 , double & xmu, double & Qzero );
650 // calcul des paramètres matériaux dépendants de la déformation équivalente
651 // en entrée: les paramètres sans dépendance = paramètres lues ou fonction de la température ou
652 // phase
653 // en sortie: les paramètres en tenant compte de la dépendance
654 void ParamateriauxAvecDefEqui(double & xnp, const double & QdeltaSigmaRatdt, const double & phi_2_dt
655 , const double & wprime, const double & def_equi_R
656 , const bool & trajet_neutre, const double & def_i_equi, const double &
657 defEqui_maxi);
658 // calcul des paramètres matériaux dépendants de la phase de la déformation totale
659 // en entrée: les paramètres sans phase = paramètres lues ou fonction de la température
660 // en sortie: les paramètres en tenant compte de la phase de epsBH
661 void ParamateriauxAvecPhaseEpsilon(const Tenseur3HH & gijHH, const Tenseur3BB & epsBB
662 , double & xnp, double & xmu, double & Qzero );
663
664 // affichage des informations pour le debug
665 void Affiche_debug(double & unSur_wBaseCarre, SaveResulHysteresis3D & save_resul
666 , List_io <Tenseur3BH>::iterator & iatens, List_io <double>::iterator & iadef_equi, bool &
667 pt_sur_principal
668 , List_io <Tenseur3BH>::iterator & iaOi, bool & oi_sur_principal, const
669 List_io <Tenseur3BH>::iterator & iatens_princ
670 , const List_io <double>::iterator & iadef_equi_princ, const List_io <Tenseur3BH>::iterator &
671 iaOi_princ);
672 // vérif de la coïncidence: prog de mise au point
673 // premiere_fois: comme il y a une récursion, pour arrêter la boucle infinie
674 void Verif_coïncidence(const Tenseur3BB & delta_epsBB, const Tenseur3HH & gijHH
675 , const Tenseur3BB & gijBB, int cas, SaveResulHysteresis3D & save_resul
676 , Tenseur3HH & sigHH, const EnergieMeca & energ_t, EnergieMeca & energ
677 , bool premiere_fois);
678 // Transport des différentes grandeurs en fonction de la variable : type_de_transport_memorisation
679 // coordonnées initiales du tenseur en mixte: aBH,
680 // coordonnées finales: rBH : donc correspondant au tenseur transporté en l'état actuel
681 void Transport_grandeur(const Tenseur3BB & gijBB, const Tenseur3HH & gijHH
682 , const Tenseur3BH & aBH, Tenseur3BH & rBH);
683 // transport éventuel de toutes les grandeurs tensorielles mémorisées dans save_resul
684 void Transport_grandeur(const Tenseur3BB & gijBB, const Tenseur3HH & gijHH, SaveResulHysteresis3D &
685 save_resul);
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```



## 7.58 Hysteresis\_bulk.h

```

1 // FICHER : Hysteresis_bulk.h
2 // CLASSE : Hysteresis_bulk
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:      06/03/2023
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   ****
41 *   BUT:      La classe Hysteresis_bulk permet de calculer une contrainte
42 *   avec une évolution hystérétique non visqueuse et ses derivees
43 *   pour comportement purement sphérique.
44 *   (cf celle développée par Guelin, Favier, Pegon pour le déviatorique.)
45 *
46 *   *****
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !           but
50 *   -----
51 *   !       !           !
52 *
53 *   *****
54 *   MODIFICATIONS:
55 *   ! date !   auteur !           but
56 *   -----
57 *
58 *   *****/
59
60
61 #ifndef HYSTERESIS_BULK_H
62 #define HYSTERESIS_BULK_H
63
64
65 #include "Loi_comp_abstraite.h"
66 #include "CourbelD.h"
67 #include "TypeConsTens.h"
68 #include "Algo_zero.h"
69 #include "TenseurQlgene.h"
70 #include "Algo_edp.h"
71
72
73 /// @addtogroup Les_lois_hysteresis
74 /// @{
75 ///
76
77 class Hysteresis_bulk : public Loi_comp_abstraite
78 {
79     public :
80
81         // CONSTRUCTEURS :
82
83         // Constructeur par défaut
84         Hysteresis_bulk ();
85

```

```

86
87 // Constructeur de copie
88 Hysteresis_bulk (const Hysteresis_bulk& loi) ;
89
90 // DESTRUCTEUR :
91
92 ~Hysteresis_bulk ();
93
94 // initialise les donnees particulieres a l'elements
95 // de matiere traite ( c-a-dire au pt calcule)
96 // Il y a creation d'une instance de SaveResul particuliere
97 // a la loi concerne
98 // la SaveResul classe est remplie par les instances heritantes
99 // le pointeur de SaveResul est sauvegarde au niveau de l'element
100 // c'a-d que les info particulieres au point considere sont stocke
101 // au niveau de l'element et non de la loi.
102 class SaveResulHysteresis_bulk: public SaveResul
103 { public :
104     SaveResulHysteresis_bulk(); // constructeur par défaut :
105     SaveResulHysteresis_bulk(const SaveResulHysteresis_bulk& sav); // de copie
106     ~SaveResulHysteresis_bulk(); // destructeur
107     // définition d'une nouvelle instance identique
108     // appelle du constructeur via new
109     SaveResul * Nevez_SaveResul() const{return (new SaveResulHysteresis_bulk(*this));};
110 // affectation
111 virtual SaveResul & operator = ( const SaveResul & a);
112 //===== lecture écriture dans base info =====
113 // cas donne le niveau de la récupération
114 // = 1 : on récupère tout
115 // = 2 : on récupère uniquement les données variables (supposées comme telles)
116 void Lecture_base_info (ifstream& ent,const int cas);
117 // cas donne le niveau de sauvegarde
118 // = 1 : on sauvegarde tout
119 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
120 void Ecriture_base_info(ofstream& sort,const int cas);
121
122 // mise à jour des informations transitoires
123 void TdtversT();
124 void TversTdt ();
125
126 // affichage à l'écran des infos
127 void Affiche() const;
128
129 //changement de base de toutes les grandeurs internes tensorielles stockées
130 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gb
131 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
132 // ici il n'y a pas de données tensorielles donc rien n'a faire
133 // gpH(i) = gamma(i,j) * gH(j)
134 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma){};
135
136 // procedure permettant de completer éventuellement les données particulières
137 // de la loi stockées
138 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
139 // completer est appelé apres sa creation avec les donnees du bloc transmis
140 // peut etre appeler plusieurs fois
141 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
142 ,const Loi_comp_abstraite* loi) {};
143
144 // ---- méthodes spécifiques
145 // initialise les informations de travail concernant le pas de temps en cours
146 void Init_debut_calcul();
147
148 // données protégées
149 double MPr_t; // dernière MPr à t
150 double MPr_tdt; // MPr en cours à tdt
151 double fonction_aide_t; // dernière valeur de la fonction d'aide
152 double fonction_aide_tdt; // valeur de la fonction d'aide en cours
153 double wprime_t,wprime_tdt; // paramètre de masing
154 List_io <double>::iterator ip2; // adresse éventuelle du 2 éléments de fct_aide
155
156 // --- informations de travail concernant le pas de temps en cours ---
157 int modif; // = 0 rien de changé, =1 coincidence(s), =2 inversion(s), =3 coin et inver
158 // liste des nouvelles MPrs de référence, qui sont apparu
159 List_io <double> MPr_R_t_a_tdt;
160 int nb_coincidence; // nombre de coincidence durant le pas de temps
161 List_io <double> fct_aide_t_a_tdt; // liste des valeurs de la fonction d'aide durant le pas
162 // de temps
163 List_io <int> indic_coin; // liste d'indicateurs indiquant la suite des coincidences et
164 // inversions,
165 // = 1 -> indique que c'est une coincidence avec un seul dépilage
166 // = 2 -> indique que c'est une coincidence avec deux dépilages
167 // = 0 -> c'est une inversion
168 // --- fin informations de travail concernant le pas de temps en cours ---
169
170 // --- informations de mémorisation discrète de 0 à t
171 // le dernier élément rangé est en .begin() (c-a-d front())
172 List_io <double> MPr_R; // liste des MPrs de référence

```

```

172         List_io <double> fct_aide; // liste des valeurs de la fonction d'aide
173
174         // 5) --- tableau d'indicateur de la résolution, éventuellement vide
175         // cela dépend de sortie_post
176         // indicateurs_resolution(1) : nb d'incrément utilisé pour la résolution de l'équation
linéarisée
177         // (2) : nb total d'itération " " " " " "
178         // (3) : cas Runge Kutta: nb d'appel de la fonction
179         // (4) : cas Runge Kutta: nb de step de calcul
180         // (5) : cas Runge Kutta: erreur globale de la résolution
181         Tableau <double> indicateurs_resolution,indicateurs_resolution_t;
182
183         // --- gestion d'une map de grandeurs quelconques éventuelles ---
184         // une map de grandeurs quelconques particulière qui peut servir aux classes appelantes
185         // il s'agit ici d'une map interne qui a priori ne doit servir qu'aux class loi de comportement
186         // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
187         // -> n'est pas sauvegardé, car a priori il s'agit de grandeurs redondantes
188         map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >
map_type_quelconque;
189
190         // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
191         const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >*
Map_type_quelconque()
192         const {return &map_type_quelconque;};
193     private:
194         void Mise_a_jour_map_type_quelconque();
195
196         // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ---
197
198     };
199
200     SaveResul * New_et_Initialise();
201
202     // Lecture des donnees de la classe sur fichier
203     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD&
lesFonctionsnD);
204     // affichage de la loi
205     void Affiche() const ;
206     // test si la loi est complete
207     // = 1 tout est ok, =0 loi incomplete
208     int TestComplet();
209
210     // calcul d'un module d'young équivalent à la loi, ceci pour un
211     // chargement nul
212     double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
213     // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
214     // il s'agit ici de la relation - MPR = sigma_trace/3. = module de compressibilité * I_eps
215     virtual double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def);
216
217     // récupération de la variation relative d'épaisseur calculée: h/h0
218     // cette variation n'est utile que pour des lois en contraintes planes
219     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
220     // - pour les lois 2D def planes: retour de 0
221     // les infos nécessaires à la récupération , sont stockées dans saveResul
222     // qui est le conteneur spécifique au point où a été calculé la loi
223     virtual double HsurH0(SaveResul * saveResul) const
224     { cout << "\n Hysteresis_bulk::HsurH0(.. , methode non implante pour l'instant ";
225     Sortie(1);
226     };
227     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
228     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Hysteresis_bulk(*this)); };
229
230     //----- lecture écriture de restart -----
231     // cas donne le niveau de la récupération
232     // = 1 : on récupère tout
233     // = 2 : on récupère uniquement les données variables (supposées comme telles)
234     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
235     ,LesFonctions_nD& lesFonctionsnD);
236
237     // cas donne le niveau de sauvegarde
238     // = 1 : on sauvegarde tout
239     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
240     void Ecriture_base_info_loi(ofstream& sort,const int cas);
241
242     // affichage et definition interactive des commandes particulières à chaque lois
243     void Info_commande_LoisDeComp(UtilLecture& lec);
244
245     // récupération des grandeurs particulière (hors ddl )
246     // correspondant à liTQ
247     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
248     virtual void Grandeur_particuliere
249     (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
250     ;
251     // récupération de la liste de tous les grandeurs particulières
252     // ces grandeurs sont ajoutées à la liste passées en paramètres
253     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière

```

```

253 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
254
255 // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
256 // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
257 virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >&
    listEnuQuelc);
258
259 // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
260 // passée en paramètre, dans le save result: ces conteneurs doivent être valides
261 // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
262 virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul);
263
264 protected :
265
266 // donnees protegees
267 // ---- paramètres matériaux ----
268 double xnp; // paramètre de Prager
269 CourbeID* xnp_temperature; // courbe éventuelle d'évolution de xnp en fonction de la température
270 // int cas_prager; // indique si xnp =2 (=1), ou est compris entre 2 et 3 (=2), ou est sup à 3
    (=3)
271 double Qzero; // limite de plasticité du critère de von mises
272 CourbeID* Qzero_temperature; // courbe éventuelle d'évolution de Qzero en fonction de la
    température
273 double xmu; // pente à l'origine
274 CourbeID* xmu_temperature; // courbe éventuelle d'évolution de xmu en fonction de la température
275
276 // ---- paramètres de l'algorithme ----
277 double tolerance_residu; // tolérance absolu sur la résolution de la plasticité
278 double tolerance_residu_rel; // tolérance relative sur la résolution de la plasticité
279 double maxi_delta_var_sig_sur_iter_pour_Newton; // le maxi de variation que l'on tolère d'une
    itération à l'autre
280 double tolerance_coincidence; // tolérance sur la précision de la coïncidence
281 int nb_boucle_maxi; // le maximum d'itération de plasticité permis
282 int nb_sous_increment; // le maxi de sous incrément prévu
283 int type_resolution_equa_constitutive; // linéarisation ou kutta par exemple
284 int nb_maxInvCoinSurUnPas; // nombre maximum d'inversion ou de coïncidence sur un pas
285 double depassement_Q0; // valeur en relatif de dépassement permis sur la saturation: devrait = 1,
    mais dans les faits il faut laisser une marge (1.2 par exe)
286
287 // ----- controle de la sortie des informations
288 // -> maintenant définit dans LoiAbstraiteGeneral
289 // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les
    erreurs et warning
290 int sortie_post; // permet d'accéder au nombre d'itération, d'incrément, de précision etc. des
    résolutions
291 // = 0 par défaut,
292 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
293
294 // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
295 // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonomie
296 Tenseur3HHHH I_x_I_HHHH,I_xbarre_I_HHHH,I_x_eps_HHHH,Ixbarre_eps_HHHH;
297
298 // variables de travail pour l'échange entre les différentes méthodes en internes
299 double MPr_t_tdt; // - pression finale
300 double MPr_i__; // - pression de début de calcul (à t au début)
301 double MPr_R; // - pression de Référence en cours
302 double delta_MPr_Rat; // deltat (- pression) de R a t
303 double delta_MPr_Ratdt; // deltat (- pression) de R a tdt
304 double delta_MPr_tatdt; // delta (- pression) de t à tdt
305 double residuBH;
306 double delta_V; // delta V totale
307 double delta_alpha_V; // delta V intermediaire (avec alpha de 0 à 1)
308 double wprime; // paramètre de masing
309 Vecteur residu; // résidu de l'équation pour la résolution de l'équation constitutive
310 Mat_pleine derResidu; // dérivé du résidu de l'équation pour la résolution de l'équation
    constitutive
311 Algo_zero alg_zero; // algo pour la recherche de zero
312 // ... partie relative à une résolution de l'avancement par une intégration de l'équation
    différentielle
313 Algo_edp alg_edp;
314 int cas_kutta; // indique le type de runge_kutta que l'on veut utiliser
315 double erreurAbsolue,erreurRelative; // précision absolue et relative que l'on désire sur le
    calcul de sig_tdt
316 int nbMaxiAppel; // nombre maxi d'appel de la fonction dérivée
317 Vecteur MPr_point_; // vitesse de - pression: version vecteur de - pression_point
318 double MPr_point; // idem mais en scalaire
319 double MPr_tau; // valeur de la MPr pour le temps tau
320 Vecteur MPr_tau_vect; // valeur de la MPr pour le temps tau, en vecteur
321 double delta_MPr_R_a_tau; // delta MPr de R à tau
322
323 // -- variables de travail internes à Residu_constitutif() et Mat_tangente_constitutif()
324 // définit ici pour éviter de les définir à chaque passage ds la méthode,
325 // ne doivent pas être utilisée en dehors de ces deux routines
326 double rdelta_MPr_Ratdt; // deltat MPr de R a tdt
327 double rdelta_MPr_tatdt; // delta MPr de t à tdt
328
329 bool aucun_pt_inversion; // pour gérer le fait qu'au début il n'y a pas de pt d'inversion !

```

```

330
331 // ----- méthodes internes -----:
332 // affinage d'un point de coïncidence
333 // ramène true si le traitement est exactement terminé, sinon false, ce qui signifie qu'il
334 // faut encore continuer à utiliser l'équation d'évolution
335 // premiere_charge : indique si c'est oui ou non une coïncidence avec la première charge
336 // pt_sur_principal : indique si oui ou non les pointeurs iaftc et iatens pointent sur les
listes
337 // principales
338 // iatens_princ et iaftc_princ: pointeurs sur les listes principales
339 bool Coincidence(bool & aucun_pt_inversion, double& unSur_wprimeCarre, bool premiere_charge
340 , SaveResulHysteresis_bulk & save_resul, double& W_a
341 , List_io <double>::iterator& iatens, List_io <double>::iterator& iaftc
342 , bool& pt_sur_principal, List_io <double>::iterator& iatens_princ
343 , List_io <double>::iterator& iaftc_princ, double& delta_W_a
344 , bool force_coïncidence, const double& MPr_tdt );
345 // cas particulier d'une inversion et coïncidence : affinage d'un point de coïncidence
346 // ramène true si le traitement est exactement terminé, sinon false, ce qui signifie qu'il
347 // faut encore continuer à utiliser l'équation d'évolution
348 // premiere_charge : indique si c'est oui ou non une coïncidence avec la première charge
349 // pt_sur_principal : indique si oui ou non les pointeurs iaftc et iatens pointent sur les
listes
350 // principales
351 // iatens_princ et iaftc_princ: pointeurs sur les listes principales
352 bool Inversion_et_Coincidence(bool & aucun_pt_inversion, double& unSur_wprimeCarre
353 , SaveResulHysteresis_bulk & save_resul, double& W_a
354 , List_io <double>::iterator& iatens, const double& delta_MPr_tatdt
355 , List_io <double>::iterator& iaftc
356 , bool& pt_sur_principal, List_io <double>::iterator& iatens_princ
357 , List_io <double>::iterator& iaftc_princ, double& delta_W_a
358 , bool force_coïncidence, const double& MPr_tdt );
359
360 // gestion d'un dépilement des pointeurs dans les listes, dans le cas d'une coïncidence
361 // met à jour les booléens interne à l'instance : aucun_pt_inversion et centre_initial
362 void Gestion_pointeur_coïncidence(double& unSur_wprimeCarre
363 , SaveResulHysteresis_bulk & save_resul, double& W_a
364 , bool & aucun_pt_inversion
365 , List_io <double>::iterator& iatens
366 , List_io <double>::iterator& iaftc
367 , bool& pt_sur_principal
368 , List_io <double>::iterator& iatens_princ
369 , List_io <double>::iterator& iaftc_princ
370 , const double& MPr_tdt);
371
372
373 // gestion d'un dépilement des pointeurs dans les listes, dans le cas particulier
374 // d'une inversion - coïncidence
375 // met à jour les booléens interne à l'instance : aucun_pt_inversion et centre_initial
376 void Gestion_pointeur_Inversion_et_Coincidence(double& unSur_wprimeCarre
377 , SaveResulHysteresis_bulk & save_resul, double& W_a
378 , bool & aucun_pt_inversion
379 , List_io <double>::iterator& iatens
380 , List_io <double>::iterator& iaftc
381 , bool& pt_sur_principal
382 , List_io <double>::iterator& iatens_princ
383 , List_io <double>::iterator& iaftc_princ
384 , const double& MPr_tdt);
385
386 // gestion du paramètre "modif" de saveresult
387 // inversion = true: on met à jour après une inversion
388 // = false : on met à jour après une coïncidence
389 void Gestion_para_Saveresult_Modif
390 (const bool& pt_sur_principal, SaveResulHysteresis_bulk & save_resul
391 , const bool& inversion
392 );
393
394 // ----- codage des METHODES VIRTUELLES protegees -----:
395 // calcul des contraintes a t+dt
396 // calcul des contraintes
397 void Calcul_SigmaHH (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
398 , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB & giB, BaseH & giH, TenseurBB & epsBB_
399 , TenseurBB & delta_epsBB_
400 , TenseurBB & gijBB_, TenseurHH & gijHH_, Tableau <TenseurBB *> & d_gijBB_
401 , double& jacobien_0, double& jacobien, TenseurHH & sigHH
402 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
module_cisaillement
403 , const Met_abstraite::Expli_t_tdt& ex);
404
405 // calcul des contraintes et de ses variations a t+dt
406 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
407 , BaseB & giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
408 , BaseB & giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH & giH_tdt, Tableau <BaseH> & d_giH_tdt
409 , TenseurBB & epsBB_tdt, Tableau <TenseurBB *> & d_epsBB
410 , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
411 , Tableau <TenseurBB *> & d_gijBB_tdt
412 , Tableau <TenseurHH *> & d_gijHH_tdt, double& jacobien_0, double& jacobien
413 , Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *> & d_sigHH

```

```

414         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
415         ,const Met_abstraite::Impli& ex);
416
417         // calcul des contraintes et ses variations par rapport aux déformations a t+dt
418         // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
419         // le tenseur de déformation et son incrémentsont également en orthonormees
420         // si = false: les bases transmises sont utilisées
421         // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
422 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
423         ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
424         ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
425         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
426         ,const Met_abstraite::Umat_cont& ex) ; // = 0;
427
428
429
430 // fonction surchargée dans les classes dérivée si besoin est
431 virtual void CalculGrandeurTravail
432         (const PtIntegMecaInterne& ,const Deformation &
433         ,Enum_dure,const ThermoDonnee&
434         ,const Met_abstraite::Impli* ex_impli
435         ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
436         ,const Met_abstraite::Umat_cont* ex_umat
437         ,const List_io<Ddl_etendu>* exclure_dd_etendu
438         ,const List_io<const TypeQuelconque *>* exclure_Q
439         ) {};
440
441 // initialisation éventuelle des variables thermo-dépendantes
442 void Init_thermo_dependance();
443 // méthode permettant le calcul de la MPr à tdt par différente méthodes: linéarisation
444 // ou kutta
445 void CalculPression_tdt(Tableau<double>& indicateurs_resolution);
446
447 // calcul de l'avancement temporel sur 1 pas,
448 // utilisé par les 3 programmes principaux:
449 // Calcul_SigmaHH, Calcul_DsigmaHH_tdt, Calcul_dsigma_deps,
450 // int cas: =1 : normal, on symétrise le tenseur des contraintes en fonction de la dérivée
451 // de Jauman
452 // int cas = 2 : pas de symétrisation du tenseur des contraintes à la fin du calcul
453 void Avancement_temporel(const Tenseur3HH & gijHH
454         ,const Tenseur3BB & gijBB,int cas,SaveResulHysteresis_bulk & save_resul
455         ,Tenseur3HH & sigHH,const EnergieMeca & energ_t,EnergieMeca & energ);
456 public:
457 // calcul de la fonction résidu de la résolution de l'équation constitutive
458 // l'argument test ramène
459 // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
460 // fatal, qui invalide le calcul du résidu
461 Vecteur& Residu_constitutif (const double & alpha, const Vecteur & x, int& test);
462 // calcul de la matrice tangente de la résolution de l'équation constitutive
463 // l'argument test ramène
464 // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
465 // fatal, qui invalide le calcul du résidu et de la dérivée
466 Mat_abstraite& Mat_tangente_constitutif(const double & alpha,const Vecteur & x, Vecteur& resi, int&
test);
467 // calcul de l'opérateur tangent : dsigma/epsilon, dans le cas d'un repère ortho-normé ou non
468 // T_d_pres_d_V: représente la variation de la MPr par rapport à V
469 // dsig_deps : version 3D
470 void Dsig_depsilon(double& T_d_pres_d_V,bool en_base_orthonormee,const Tenseur3HH & gijHH_tdt
471         , TenseurHHHH* dsig_deps,const double & V_tdt);
472 // calcul de l'expression permettant d'obtenir la dérivée temporelle de la MPr
473 // à un instant tau quelconque, en fait il s'agit de l'équation constitutive
474 // utilisée dans la résolution explicite (runge par exemple) de l'équation constitutive
475 // erreur : =0: le calcul est licite, si diff de 0, indique qu'il y a eu une erreur
476 // =1: la norme de sigma est supérieure à la valeur limite de saturation
477 Vecteur& Pression_point(const double & tau, const Vecteur & sigma_tau,Vecteur& sig_point,int &
erreur);
478 // vérification de l'intégrité de la MPr calculée
479 // erreur : =0: le calcul est licite, si diff de 0, indique qu'il y a eu une erreur
480 // =1: la norme de la MPr est supérieure à la valeur limite de saturation
481 void Verif_integrite_Pression(const double & tau, const Vecteur & MPr_tau,int & erreur);
482
483 // affichage des informations pour le debug
484 void Affiche_debug(double& unSur_wBaseCarre,SaveResulHysteresis_bulk & save_resul
485         ,List_io <double>::iterator& iatens,bool& pt_sur_principal
486         ,const List_io <double>::iterator& iatens_princ
487         );
488
489 };
490 /// @} // end of group
491
492
493 #endif
494
495
496

```

## 7.59 Loi\_iso\_elas1D.h

```

1 // FICHER : Loi_iso_elas.h
2 // CLASSE : Loi_iso_elas
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33
34 /*****
35 *      DATE:          15/09/2020                      *
36 *                                                           $ *
37 *      AUTEUR:       G RIO   (mailto:gerardrio56@free.fr) *
38 *                                                           $ *
39 *      PROJET:       Herezh++                          *
40 *                                                           $ *
41 *****/
42 *      BUT: La classe Loi_iso_elas permet de calculer la contrainte *
43 *      et ses derivees pour une loi isotrope elastique.          *
44 *      Il s'agit d'une classe derivee de la classe Loi_comp_abstraite. *
45 *                                                           $ *
46 *      ***** *
47 *
48 *      VERIFICATION: *
49 *
50 *      ! date !   auteur !           but *
51 *      ----- *
52 *      !           !           !           ! *
53 *                                                           $ *
54 *      ***** *
55 *      MODIFICATIONS: *
56 *      ! date !   auteur !           but *
57 *      ----- *
58 *                                                           $ *
59 *****/
60
61
62 #ifndef LOI_ISO_ELAS1D_H
63 #define LOI_ISO_ELAS1D_H
64
65
66 #include "Loi_comp_abstraite.h"
67
68 /** @defgroup Les_lois_hooke
69 *
70 *      BUT:   groupe des lois de type Hooke
71 *
72 *
73 * \author   Gérard Rio
74 * \version  1.0
75 * \date    15/09/2020
76 * \brief   Définition des lois de type Hooke
77 *
78 */
79
80 /// @addtogroup Les_lois_hooke
81 /// @{
82 ///
83
84 class Loi_iso_elas1D : public Loi_comp_abstraite
85 {

```

```

86
87
88 public :
89
90
91 // CONSTRUCTEURS :
92
93 // Constructeur par défaut
94 Loi_iso_elas1D ();
95
96 // Constructeur fonction du E et du nu
97 Loi_iso_elas1D(const double& EE,const double& nunu);
98
99 // Constructeur de copie
100 Loi_iso_elas1D (const Loi_iso_elas1D& loi) ;
101
102 // DESTRUCTEUR :
103
104 ~Loi_iso_elas1D ();
105
106 // initialise les donnees particulieres a l'elements
107 // de matiere traite ( c-a-dire au pt calcule)
108 // Il y a creation d'une instance de SaveResul particuliere
109 // a la loi concernee
110 // la SaveResul classe est remplie par les instances heritantes
111 // le pointeur de SaveResul est sauvegarde au niveau de l'element
112 // c'a-d que les info particulieres au point considere sont stocke
113 // au niveau de l'element et non de la loi.
114 class SaveResulLoi_iso_elas1D: public SaveResul
115 { public :
116     SaveResulLoi_iso_elas1D(): // constructeur par défaut :
117         E(-ConstMath::trespetit),nu(-2.*ConstMath::trespetit)
118         ,E_t(-ConstMath::trespetit),nu_t(-2.*ConstMath::trespetit),map_type_quelconque() {} ;
119     SaveResulLoi_iso_elas1D(const SaveResulLoi_iso_elas1D& sav): // de copie
120         E(sav.E),nu(sav.nu),E_t(sav.E_t),nu_t(sav.nu_t)
121         ,map_type_quelconque(sav.map_type_quelconque) {} ;
122
123 virtual ~SaveResulLoi_iso_elas1D() {} ; // destructeur
124 // définition d'une nouvelle instance identique
125 // appelle du constructeur via new
126 SaveResul * Nevez_SaveResul() const{return (new SaveResulLoi_iso_elas1D(*this));};
127 // affectation
128 virtual SaveResul & operator = ( const SaveResul & a)
129 { SaveResulLoi_iso_elas1D& sav = *((SaveResulLoi_iso_elas1D*) &a);
130   E=sav.E; nu=sav.nu;E_t=sav.E_t; nu_t=sav.nu_t;
131   map_type_quelconque = sav.map_type_quelconque;
132   return *this;
133 };
134 //===== lecture écriture dans base info =====
135 // cas donne le niveau de la récupération
136 // = 1 : on récupère tout
137 // = 2 : on récupère uniquement les données variables (supposées comme telles)
138 void Lecture_base_info (ifstream& ent,const int cas)
139 {string toto; ent >> toto >> E >> toto >> nu; E_t=E; nu_t = nu; };
140 // cas donne le niveau de sauvegarde
141 // = 1 : on sauvegarde tout
142 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
143 void Ecriture_base_info(ofstream& sort,const int cas)
144 {sort << "\n E= " << E << " nu= " << nu << " ";};
145
146 // mise à jour des informations transitoires
147 void TdtversT()
148 {E_t = E; nu_t = nu;
149 // mise à jour de la liste des grandeurs quelconques internes
150 Mise_a_jour_map_type_quelconque();
151 };
152 void TversTdt ()
153 {E = E_t; nu = nu_t;
154 // mise à jour de la liste des grandeurs quelconques internes
155 Mise_a_jour_map_type_quelconque();
156 };
157
158 // affichage à l'écran des infos
159 void Affiche() const
160 { cout << "\n E= " << E << " nu= " << nu << " ";};
161
162 //changement de base de toutes les grandeurs internes tensorielles stockées
163 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
164 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
165 // gpH(i) = gamma(i,j) * gH(j)
166 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) {} ;
167
168 // procedure permettant de completer éventuellement les données particulières
169 // de la loi stockées
170 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
171 // completer est appelé apres sa creation avec les donnees du bloc transmis
172 // peut etre appeler plusieurs fois

```



```

173     virtual SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>&
tab_coor
174                                     ,const Loi_comp_abstraite* loi) {return NULL;};
175
176
177     //-----
178     // données
179     //-----
180     double E,E_t,nu,nu_t; // les paramètres matériaux réellement utilisés
181
182     // --- gestion d'une map de grandeurs quelconques éventuelles ---
183
184     // une map de grandeurs quelconques particulière qui peut servir aux classes appelantes
185     // il s'agit ici d'une map interne qui a priori ne doit servir qu'aux class loi de
comportement
186     // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
187     // -> n'est pas sauvegardé, car a priori il s'agit de grandeurs redondantes
188     map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque > >
map_type_quelconque;
189
190     // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
191     const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque > >
Map_type_quelconque()
192     const {return &map_type_quelconque;};
193     private:
194     void Mise_a_jour_map_type_quelconque();
195
196     // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ---
197 };
198
199 SaveResul * New_et_Initialise()
200 { SaveResulLoi_iso_elas1D * pt = new SaveResulLoi_iso_elas1D();
201 // insertion éventuelle de conteneurs de grandeurs quelconque
202 Insertion_conteneur_dans_save_result(pt);
203 return pt;
204 };
205
206 friend class SaveResulLoi_iso_elas1D;
207
208
209 // Lecture des donnees de la classe sur fichier
210 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
211                                 ,LesFonctions_nD& lesFonctionsnD);
212
213 // affichage de la loi
214 void Affiche() const ;
215 // test si la loi est complete
216 // = 1 tout est ok, =0 loi incomplete
217 int TestComplet();
218
219 //----- lecture écriture de restart -----
220 // cas donne le niveau de la récupération
221 // = 1 : on récupère tout
222 // = 2 : on récupère uniquement les données variables (supposées comme telles)
223 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
224                                 ,LesFonctions_nD& lesFonctionsnD);
225
226 // cas donne le niveau de sauvegarde
227 // = 1 : on sauvegarde tout
228 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
229 void Ecriture_base_info_loi(ofstream& sort,const int cas);
230
231 // calcul d'un module d'young équivalent la loi, ceci pour un
232 // chargement nul
233 double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
234 // récupération d'un module de compressibilité équivalent à la loi pour un chargement nul
235 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
236 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
saveResul);
237
238 // récupération de la variation relative d'épaisseur calculée: h/h0
239 // cette variation n'est utile que pour des lois en contraintes planes
240 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
241 // - pour les lois 2D def planes: retour de 0
242 // les infos nécessaires à la récupération , sont stockées dans saveResul
243 // qui est le conteneur spécifique au point où a été calculé la loi
244 virtual double HsurH0(SaveResul * saveResul) const
245 { cout << "\n Loi_iso_elas1D::HsurH0(.. , methode non implante pour l'instant ";
246 Sortie(1);
247 return 0.; // pour taire le compilo
248 };
249
250 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
251 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_iso_elas1D(*this)); };
252
253 // affichage et definition interactive des commandes particulières a chaque lois
254 void Info_commande_LoisDeComp(UtilLecture& lec);
255

```

```

254 // récupération des grandeurs particulière (hors ddl )
255 // correspondant à liTQ
256 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
257 virtual void Grandeur_particuliere
258     (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
    ;
259 // récupération de la liste de tous les grandeurs particulières
260 // ces grandeurs sont ajoutées à la liste passées en paramètres
261 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
262 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
263
264
265 // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
266 // passée en paramètre, dans le save result: ces conteneurs doivent être valides
267 // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
268 virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul);
269
270 // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
271 // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
272 virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >& listEnuQuelc);
273
274
275 protected :
276     // donnée de la loi
277     double E,nu; // module d'young et coef de poisson
278     CourbeId* E_temperature; // courbe éventuelle d'évolution de E en fonction de la température
279     Fonction_nD * E_nD; // fonction nD éventuelle pour E
280     Fonction_nD * nu_nD; // fonction nD éventuelle pour nu
281
282     // codage des METHODES VIRTUELLES protegees:
283 // calcul des contraintes a t+dt
284 // calcul des contraintes
285 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
286     ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
287     ,TenseurBB & delta_epsBB_
288     ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
289     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
290     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
291     module_cisaillement
292     ,const Met_abstraite::Expli_t_tdt& ex);
293
294 // calcul des contraintes et de ses variations a t+dt
295 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
296     ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
297     ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
298     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
299     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
300     ,Tableau <TenseurBB *>& d_gijBB_tdt
301     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
302     ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
303     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
304     module_cisaillement
305     ,const Met_abstraite::Impli& ex);
306
307 // fonction surchargée dans les classes dérivée si besoin est
308 virtual void CalculGrandeurTravail
309     (const PtIntegMecaInterne& ,const Deformation &
310     ,Enum_dure,const ThermoDonnee&
311     ,const Met_abstraite::Impli* ex_impli
312     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
313     ,const Met_abstraite::Umat_cont* ex_umat
314     ,const List_io<Ddl_etendu>* exclure_dd_etend
315     ,const List_io<const TypeQuelconque *>* exclure_Q
316     ) {};
317 };
318 /// @} // end of group
319
320
321 #endif
322
323
324

```

## 7.60 Loi\_iso\_elas2D\_C.h

```

1 // FICHER : Loi_iso_elas2D_C.h
2 // CLASSE : Loi_iso_elas2D_C
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.

```

```

9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *      DATE:          15/09/2020
35 *
36 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:        Herezh++
39 *
40 *****/
41 *      BUT:   La classe Loi_iso_elas2D_C permet de calculer la
42 *            contrainte et ses derivees pour une loi isotrope elastique
43 *            2D en contraintes planes.
44 *            Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
45 *
46 *            *****
47 *
48 *      VERIFICATION:
49 *
50 *      ! date !   auteur !           but
51 *      -----
52 *      !       !           !
53 *
54 *            *****
55 *      MODIFICATIONS:
56 *      ! date !   auteur !           but
57 *      -----
58 *
59 *****/
60
61
62
63 #ifndef LOI_ISO_ELAS_2D_C_H
64 #define LOI_ISO_ELAS_2D_C_H
65
66
67 #include "Loi_comp_abstraite.h"
68
69
70 /// @addtogroup Les_lois_hooke
71 /// @{
72 ///
73
74
75 class Loi_iso_elas2D_C : public Loi_comp_abstraite
76 {
77
78
79     public :
80
81
82         // CONSTRUCTEURS :
83
84         // Constructeur par default
85         Loi_iso_elas2D_C ();
86
87         // Constructeur fonction du E et du nu
88         Loi_iso_elas2D_C(const double& EE,const double& nu);
89
90         // Constructeur de copie
91         Loi_iso_elas2D_C (const Loi_iso_elas2D_C& loi) ;
92
93         // DESTRUCTEUR :
94
95         ~Loi_iso_elas2D_C ();

```

```

96
97
98
99 // initialise les donnees particulieres a l'elements
100 // de matiere traite ( c-a-dire au pt calcule)
101 class SaveResul_Loi_iso_elas2D_C: public SaveResul
102 { public :
103     SaveResul_Loi_iso_elas2D_C(); // constructeur par défaut (a ne pas utiliser)
104     // le constructeur courant
105     SaveResul_Loi_iso_elas2D_C(SaveResul* l_des_SaveResul);
106     // constructeur de copie
107     SaveResul_Loi_iso_elas2D_C(const SaveResul_Loi_iso_elas2D_C& sav );
108     // destructeur
109     ~SaveResul_Loi_iso_elas2D_C();
110     // définition d'une nouvelle instance identique
111     // appelle du constructeur via new
112     SaveResul * Nevez_SaveResul() const {return (new SaveResul_Loi_iso_elas2D_C(*this));};
113 // affectation
114 virtual SaveResul & operator = ( const SaveResul & a)
115 {SaveResul_Loi_iso_elas2D_C& sav = *((SaveResul_Loi_iso_elas2D_C*) &a);
116   eps33=sav.eps33;eps33_t=sav.eps33_t;
117   E=sav.E; nu=sav.nu;E_t=sav.E_t; nu_t=sav.nu_t;
118   map_type_quelconque = sav.map_type_quelconque;
119   return *this;};
120 //===== lecture écriture dans base info =====
121 // cas donne le niveau de la récupération
122 // = 1 : on récupère tout
123 // = 2 : on récupère uniquement les données variables (supposées comme telles)
124 void Lecture_base_info (ifstream& ent,const int cas);
125 // cas donne le niveau de sauvegarde
126 // = 1 : on sauvegarde tout
127 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
128 void Ecriture_base_info(ofstream& sort,const int cas);
129
130 // mise à jour des informations transitoires en définitif s'il y a convergence
131 // par exemple (pour la plasticité par exemple)
132 void TdtversT()
133 {eps33=eps33_t;
134   E_t = E; nu_t = nu;
135   // mise à jour de la liste des grandeurs quelconques internes
136   Mise_a_jour_map_type_quelconque();
137 };
138 void TversTdt()
139 {eps33_t=eps33;E = E_t; nu = nu_t;
140   // mise à jour de la liste des grandeurs quelconques internes
141   Mise_a_jour_map_type_quelconque();
142 };
143
144 // affichage à l'écran des infos
145 void Affiche() const;
146
147 //changement de base de toutes les grandeurs internes tensorielles stockées
148 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
149 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
150 // ici il n'y a pas de données tensorielles donc rien n'a faire
151 // gpH(i) = gamma(i,j) * gH(j)
152 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma){};
153
154 // procedure permettant de completer éventuellement les données particulières
155 // de la loi stockées
156 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
157 // completer est appelé apres sa creation avec les donnees du bloc transmis
158 // peut etre appeler plusieurs fois
159 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
160 ,const Loi_comp_abstraite* loi) {return NULL;};
161
162 // ---- récupération d'information: spécifique à certaine classe dérivée
163 double Deformation_plastique();
164
165 //-----
166 // données
167 //-----
168 double E,E_t,nu,nu_t; // les paramètres matériaux réellement utilisés
169 double eps33,eps33_t; // déformation d'épaisseur
170
171 // --- gestion d'une map de grandeurs quelconques éventuelles ---
172
173 // une map de grandeurs quelconques particulière qui peut servir aux classes appelantes
174 // il s'agit ici d'une map interne qui a priori ne doit servir qu'aux class loi de comportement
175 // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
176 // -> n'est pas sauvegardé, car a priori il s'agit de grandeurs redondantes
177 map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >
178 map_type_quelconque;
179
180 // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
181 const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >*
182 Map_type_quelconque()

```

```

181         const {return &map_type_quelconque;};
182     private:
183     void Mise_a_jour_map_type_quelconque();
184
185     // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ---
186 };
187
188     // def d'une instance de données spécifiques, et initialisation
189     SaveResul * New_et_Initialise()
190     { SaveResul_Loi_iso_elas2D_C * pt = new SaveResul_Loi_iso_elas2D_C();
191     // insertion éventuelle de conteneurs de grandeurs quelconque
192     Insertion_conteneur_dans_save_result(pt);
193     return pt;
194     };
195
196     friend class SaveResul_Loi_iso_elas2D_C;
197
198
199     // Lecture des donnees de la classe sur fichier
200     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
201     ,LesFonctions_nD& lesFonctionsnD);
202
203     // affichage de la loi
204     void Affiche() const ;
205     // test si la loi est complete
206     // = 1 tout est ok, =0 loi incomplete
207     int TestComplet();
208
209     // calcul d'un module d'young équivalent à la loi, ceci pour un
210     // chargement nul
211     double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
212     // récupération d'un module de compressibilité équivalent à la loi pour un chargement nul
213     // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
214     double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
215     saveResul);
216
217     // récupération de la variation relative d'épaisseur calculée: h/h0
218     // cette variation n'est utile que pour des lois en contraintes planes
219     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
220     // - pour les lois 2D def planes: retour de 0
221     // les infos nécessaires à la récupération , sont stockées dans saveResul
222     // qui est le conteneur spécifique au point où a été calculé la loi
223     virtual double HsurH0(SaveResul * saveResul) const;
224
225     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
226     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_iso_elas2D_C(*this)); };
227
228     // récupération des grandeurs particulière (hors ddl )
229     // correspondant à liTQ
230     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
231     virtual void Grandeur_particuliere
232     (bool absolue,List_io<TypeQuelconque>& liTQ,Loi_comp_abstraite::SaveResul * saveDon,list<int>&
233     decal) const;
234
235     // récupération de la liste de tous les grandeurs particulières
236     // ces grandeurs sont ajoutées à la liste passées en paramètres
237     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
238     virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& liTQ) const;
239
240     //----- lecture écriture de restart -----
241     // cas donne le niveau de la récupération
242     // = 1 : on récupère tout
243     // = 2 : on récupère uniquement les données variables (supposées comme telles)
244     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
245     lesCourbes1D
246     ,LesFonctions_nD& lesFonctionsnD);
247
248     // cas donne le niveau de sauvegarde
249     // = 1 : on sauvegarde toutg,const VariablesTemps&
250     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
251     void Ecriture_base_info_loi(ofstream& sort,const int cas);
252
253     // affichage et definition interactive des commandes particulières à chaque lois
254     void Info_commande_LoisDeComp(UtilLecture& lec);
255
256
257     // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
258     // passée en paramètre, dans le save result: ces conteneurs doivent être valides
259     // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
260     virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul);
261
262     // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
263     // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
264     virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >& listEnuQuelc);
265
266
267     // ----- methode propre a une loi en contraintes planes -----
268     // récupération de la dernière déformation d'épaisseur calculée: cette déformaion n'est utile que pour
269     // des lois en contraintes planes ou doublement planes
270     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide

```

```

264 // - pour les lois 2D def planes: retour de 0
265 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
266 // qui est le conteneur spécifique au point où a été calculé la loi
267 virtual double Eps33BH(SaveResul * saveResul) const
268 {SaveResul_Loi_iso_elas2D_C & save_resul = *((SaveResul_Loi_iso_elas2D_C*) saveResul);
269   return save_resul.eps33;
270 };
271
272 // indique si la loi est en contraintes planes en s'appuyant sur un comportement 3D
273 virtual bool Contraintes_planes_de_3D() const {return true;};
274
275
276 protected :
277   // paramètres de la loi
278   double E,nu; // module d'young et coefficient de poisson
279   CourbeID* E_temperature; // courbe éventuelle d'évolution de E en fonction de la température
280   Fonction_nD * E_nD; // fonction nD éventuelle pour E
281   Fonction_nD * nu_nD; // fonction nD éventuelle pour nu
282   short int cas_calcul; // indique le choix entre différents types de calcul possible
283   // = 0 : calcul normal
284   // = 1 : calcul seulement déviatorique (la partie sphérique est mise à zéro)
285   // = 2 : calcul seulement sphérique (la partie déviatorique est mise à zéro)
286
287 // codage des METHODES VIRTUELLES protegees:
288 // calcul des contraintes a t+dt
289 // calcul des contraintes
290 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
291   ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
292   ,TenseurBB & delta_epsBB_
293   ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *> & d_gijBB_
294   ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
295   ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
   module_cisaillement
296   ,const Met_abstraite::Expli_t_tdt& ex );
297
298 // calcul des contraintes et de ses variations a t+dt
299 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
300   ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
301   ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
302   ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
303   ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
304   ,Tableau <TenseurBB *> & d_gijBB_tdt
305   ,Tableau <TenseurHH *> & d_gijHH_tdt,double& jacobien_0,double& jacobien
306   ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *> & d_sigHH
307   ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
   module_cisaillement
308   ,const Met_abstraite::Impli& ex );
309
310
311 // fonction surchargée dans les classes dérivée si besoin est
312 virtual void CalculGrandeurTravail
313   (const PtIntegMecaInterne& ,const Deformation &
314   ,Enum_dure,const ThermoDonnee&
315   ,const Met_abstraite::Impli* ex_impli
316   ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
317   ,const Met_abstraite::Umat_cont* ex_umat
318   ,const List_io<Ddl_etendu>* exclure_dd_etend
319   ,const List_io<const TypeQuelconque *>* exclure_Q
320   ) {};
321
322 };
323 /// @} // end of group
324
325
326 #endif
327
328
329

```

## 7.61 Loi\_iso\_elas2D\_D.h

```

1 // FICHER : Loi_iso_elas2D_D.h
2 // CLASSE : Loi_iso_elas2D_D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)

```

```

15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32 //
33 /*****
34 *      DATE:          15/09/2020
35 *
36 *      AUTEUR:        G RIO (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:        Herezh++
39 *
40 *      $
41 *
42 *      BUT:           La classe Loi_iso_elas2D_D permet de calculer la
43 *                   contrainte et ses derivees pour une loi isotrope elastique 2D
44 *                   en deformations planes.
45 *                   Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
46 *
47 *      $
48 *
49 *      VERIFICATION:
50 *
51 *      ! date !   auteur !   but
52 *      -----
53 *      !       !       !
54 *
55 *      $
56 *
57 *      MODIFICATIONS:
58 *
59 *      ! date !   auteur !   but
60 *      -----
61 *      !       !       !
62 *
63 *      $
64 *
65 *      *****/
66
67 #ifndef LOI_ISO_ELAS_2D_D_H
68 #define LOI_ISO_ELAS_2D_D_H
69
70 #include "Loi_comp_abstraite.h"
71
72 /// @addtogroup Les_lois_hooke
73 /// @
74 ///
75
76 class Loi_iso_elas2D_D : public Loi_comp_abstraite
77 {
78 public :
79
80     // CONSTRUCTEURS :
81
82     // Constructeur par defaut
83     Loi_iso_elas2D_D ();
84
85     // Constructeur fonction du E et du nu
86     Loi_iso_elas2D_D(const double& EE,const double& nunu);
87
88     // Constructeur de copie
89     Loi_iso_elas2D_D (const Loi_iso_elas2D_D& loi) ;
90
91     // DESTRUCTEUR :
92
93     ~Loi_iso_elas2D_D ();
94
95     // initialise les donnees particulieres a l'elements
96     // de matiere traite ( c-a-dire au pt calcule)
97     // Il y a creation d'une instance de SaveResul particuliere
98     // a la loi concernee
99     // la SaveResul classe est remplie par les instances heritantes

```

```

102 // le pointeur de SaveResul est sauvegarde au niveau de l'element
103 // c'a-d que les info particulieres au point considere sont stocke
104 // au niveau de l'element et non de la loi.
105 class SaveResulLoi_iso_elas2D_D: public SaveResul
106 { public :
107     SaveResulLoi_iso_elas2D_D(): // constructeur par défaut :
108         E(-ConstMath::trespetit),nu(-2.*ConstMath::trespetit)
109         ,E_t(-ConstMath::trespetit),nu_t(-2.*ConstMath::trespetit),map_type_quelconque() {};
110     SaveResulLoi_iso_elas2D_D(const SaveResulLoi_iso_elas2D_D& sav): // de copie
111         E(sav.E),nu(sav.nu),E_t(sav.E_t),nu_t(sav.nu_t)
112         ,map_type_quelconque(sav.map_type_quelconque) {};
113
114     virtual ~SaveResulLoi_iso_elas2D_D() {}; // destructeur
115     // définition d'une nouvelle instance identique
116     // appelle du constructeur via new
117     SaveResul * Nevez_SaveResul() const{return (new SaveResulLoi_iso_elas2D_D(*this));};
118     // affectation
119     virtual SaveResul & operator = ( const SaveResul & a)
120     { SaveResulLoi_iso_elas2D_D& sav = *((SaveResulLoi_iso_elas2D_D*) &a);
121       E=sav.E; nu=sav.nu;E_t=sav.E_t; nu_t=sav.nu_t;
122       map_type_quelconque = sav.map_type_quelconque;
123       return *this;
124     };
125     //===== lecture écriture dans base info =====
126     // cas donne le niveau de la récupération
127     // = 1 : on récupère tout
128     // = 2 : on récupère uniquement les données variables (supposées comme telles)
129     void Lecture_base_info (ifstream& ent,const int cas)
130     {string toto; ent >> toto >> E >> toto >> nu; E_t=E; nu_t = nu; };
131     // cas donne le niveau de sauvegarde
132     // = 1 : on sauvegarde tout
133     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
134     void Ecriture_base_info(ofstream& sort,const int cas)
135     {sort << "\n E= " << E << " nu= " << nu << " ";};
136
137     // mise à jour des informations transitoires
138     void TdtversT()
139     {E_t = E; nu_t = nu;
140     // mise à jour de la liste des grandeurs quelconques internes
141     Mise_a_jour_map_type_quelconque();
142     };
143     void TversTdt()
144     {E = E_t; nu = nu_t;
145     // mise à jour de la liste des grandeurs quelconques internes
146     Mise_a_jour_map_type_quelconque();
147     };
148
149     // affichage à l'écran des infos
150     void Affiche() const
151     { cout << "\n E= " << E << " nu= " << nu << " ";};
152
153     //changement de base de toutes les grandeurs internes stockées
154     // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
155     // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
156     // gpH(i) = gamma(i,j) * gH(j)
157     virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) {};
158
159     // procedure permettant de completer éventuellement les données particulières
160     // de la loi stockées
161     // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
162     // completer est appelé apres sa creation avec les donnees du bloc transmis
163     // peut etre appeler plusieurs fois
164     virtual SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>&
165     tab_coor
166         ,const Loi_comp_abstraite* loi) {return NULL;};
167
168     //-----
169     // données
170     //-----
171     double E,E_t,nu,nu_t; // les paramètres matériaux réellement utilisés
172
173     // --- gestion d'une map de grandeurs quelconques éventuelles ---
174
175     // une map de grandeurs quelconques particulière qui peut servir aux classes appelantes
176     // il s'agit ici d'une map interne qui a priori ne doit servir qu'aux class loi de comportement
177     // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
178     // -> n'est pas sauvegardé, car a priori il s'agit de grandeurs redondantes
179     map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >
180     map_type_quelconque;
181
182     // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
183     const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >*
184     Map_type_quelconque()
185     const {return &map_type_quelconque;};
186
187     private:
188     void Mise_a_jour_map_type_quelconque();

```



```

186
187     // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ---
188 };
189
190 SaveResul * New_et_Initialise()
191 { SaveResulLoi_iso_elas2D_D * pt = new SaveResulLoi_iso_elas2D_D();
192   // insertion éventuelle de conteneurs de grandeurs quelconque
193   Insertion_conteneur_dans_save_result(pt);
194   return pt;
195 };
196 friend class SaveResulLoi_iso_elas2D_D;
197
198 // Lecture des donnees de la classe sur fichier
199 void LectureDonneesParticulieres (UtilLecture *,LesCourbes1D& lesCourbes1D ,LesFonctions_nD&
lesFonctionsnD);
200 // affichage de la loi
201 void Affiche() const ;
202 // test si la loi est complete
203 // = 1 tout est ok, =0 loi incomplete
204 int TestComplet();
205
206 // calcul d'un module d'young equivalent à la loi, ceci pour un
207 // chargement nul
208 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
209 // récupération d'un module de compressibilité equivalent à la loi pour un chargement nul
210 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
211 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
saveResul);
212
213 // récupération de la variation relative d'épaisseur calculée: h/h0
214 // cette variation n'est utile que pour des lois en contraintes planes
215 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
216 // - pour les lois 2D def planes: retour de 0
217 // les infos nécessaires à la récupération , sont stockées dans saveResul
218 // qui est le conteneur spécifique au point où a été calculé la loi
219 virtual double HsurH0(SaveResul * saveResul) const {return 0.};
220
221 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
222 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_iso_elas2D_D(*this)); };
223
224 //----- lecture écriture de restart -----
225 // cas donne le niveau de la récupération
226 // = 1 : on récupère tout
227 // = 2 : on récupère uniquement les données variables (supposées comme telles)
228 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
229                               ,LesFonctions_nD& lesFonctionsnD);
230 // cas donne le niveau de sauvegarde
231 // = 1 : on sauvegarde tout
232 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
233 void Ecriture_base_info_loi(ofstream& sort,const int cas);
234
235 // affichage et definition interactive des commandes particulières à chaque lois
236 void Info_commande_LoisDeComp(UtilLecture& lec);
237
238 // récupération des grandeurs particulière (hors ddl )
239 // correspondant à liTQ
240 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
241 virtual void Grandeur_particuliere
242     (bool absolue,List_io<TypeQuelconque> ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
243 ;
244 // récupération de la liste de tous les grandeurs particulières
245 // ces grandeurs sont ajoutées à la liste passées en paramètres
246 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
247 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
248
249 // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
250 // passée en paramètre, dans le save result: ces conteneurs doivent être valides
251 // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
252 virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul);
253
254 // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
255 // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
256 virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >& listEnuQuelc);
257
258
259 // ----- methode propre a la loi -----
260 // calcul de la contrainte sigma33
261 double Sig33BH(TenseurBB & epsBB,TenseurHH & gijHH);
262
263 protected :
264     // paramètres de la loi
265     double E,nu; // module d'young et coefficient de poisson
266     Courbe1D * E_temperature; // courbe éventuelle d'évolution de E en fonction de la température
267     Fonction_nD * E_nD; // fonction nD éventuelle pour E
268     Fonction_nD * nu_nD; // fonction nD éventuelle pour nu

```

```

269     short int cas_calcul; // indique le choix entre différents types de calcul possible
270         // = 0 : calcul normal
271     // = 1 : calcul seulement déviatorique (la partie sphérique est mise à
    zéro)
272     // = 2 : calcul seulement sphérique (la partie déviatorique est mise à
    zéro)
273
274     // codage des METHODES VIRTUELLES protegees:
275 // calcul des contraintes a t+dt
276 // calcul des contraintes
277 void Calcul_SigmaHH (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
278     , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB& giB, BaseH& gi_H, TenseurBB & epsBB_
279     , TenseurBB & delta_epsBB_
280     , TenseurBB & gijBB_, TenseurHH & gijHH_, Tableau <TenseurBB *> & d_gijBB_
281     , double& jacobien_0, double& jacobien, TenseurHH & sigHH
282     , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
283     , const Met_abstraite::Expli_t_tdt& ex);
284
285     // calcul des contraintes et de ses variations a t+dt
286 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
287     , BaseB& giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
288     , BaseB& giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH& giH_tdt, Tableau <BaseH> & d_giH_tdt
289     , TenseurBB & epsBB_tdt, Tableau <TenseurBB *> & d_epsBB
290     , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
291     , Tableau <TenseurBB *> & d_gijBB_tdt
292     , Tableau <TenseurHH *> & d_gijHH_tdt, double& jacobien_0, double& jacobien
293     , Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *> & d_sigHH
294     , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
295     , const Met_abstraite::Impli& ex);
296
297
298     // fonction surchargée dans les classes dérivée si besoin est
299 virtual void CalculGrandeurTravail
300     (const PtIntegMecaInterne& , const Deformation &
301     , Enum_dure, const ThermoDonnee&
302     , const Met_abstraite::Impli* ex_impli
303     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
304     , const Met_abstraite::Umat_cont* ex_umat
305     , const List_io<Ddl_etendu>* exclure_dd_etend
306     , const List_io<const TypeQueIconque *>* exclure_Q
307     ) {};
308
309 };
310 /// @} // end of group
311
312
313 #endif
314
315
316

```

## 7.62 Loi\_iso\_elas3D.h

```

1 // FICHER : Loi_iso_elas3D.h
2 // CLASSE : Loi_iso_elas3D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //

```

```

31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:      15/09/2020
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *****/
41 * La classe Loi_iso_elas3D permet de calculer la contrainte et ses
42 * derivees pour une loi isotrope elastique en 3D sans condition
43 * particulieres.
44 * Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
45 *
46 *   *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !       but
51 *   -----
52 *   !       !       !
53 *   $
54 *   *****
55 *   MODIFICATIONS:
56 *   ! date !   auteur !       but
57 *   -----
58 *   $
59 *****/
60
61 // La classe Loi_iso_elas3D permet de calculer la contrainte et ses derivees pour une loi
62 // isotrope elastique en 3D sans condition particulieres.
63 // Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
64
65
66 #ifndef LOI_ISO_ELAS3D_H
67 #define LOI_ISO_ELAS3D_H
68
69
70 #include "Loi_comp_abstraite.h"
71
72 /// @addtogroup Les_lois_hooke
73 /// @{
74 ///
75
76
77
78 class Loi_iso_elas3D : public Loi_comp_abstraite
79 {
80
81
82     public :
83
84
85         // CONSTRUCTEURS :
86
87         // Constructeur par default
88         Loi_iso_elas3D ();
89
90         // Constructeur fonction du E et du nu
91         Loi_iso_elas3D(const double& EE,const double& nunu);
92
93         // Constructeur de copie
94         Loi_iso_elas3D (const Loi_iso_elas3D& loi) ;
95
96         // DESTRUCTEUR :
97         ~Loi_iso_elas3D ();
98
99         // initialise les donnees particulieres a l'elements
100        // de matiere traite ( c-a-dire au pt calcule)
101        // Il y a creation d'une instance de SaveResul particuliere
102        // a la loi concernee
103        // la SaveResul classe est remplie par les instances heritantes
104        // le pointeur de SaveResul est sauvegarde au niveau de l'element
105        // c'a-d que les info particulieres au point considere sont stocke
106        // au niveau de l'element et non de la loi.
107        class SaveResulLoi_iso_elas3D: public SaveResul
108        { public :
109            SaveResulLoi_iso_elas3D(): // constructeur par default :
110                E(-ConstMath::trespetit),nu(-2.*ConstMath::trespetit)
111                ,E_t(-ConstMath::trespetit),nu_t(-2.*ConstMath::trespetit),map_type_quelconque() {};
112            SaveResulLoi_iso_elas3D(const SaveResulLoi_iso_elas3D& sav): // de copie
113                E(sav.E),nu(sav.nu),E_t(sav.E_t),nu_t(sav.nu_t)
114                ,map_type_quelconque(sav.map_type_quelconque) {};
115
116            virtual ~SaveResulLoi_iso_elas3D() {}; // destructeur
117            // definition d'une nouvelle instance identique

```

```

118 // appelle du constructeur via new
119 SaveResul * Nevez_SaveResul() const{return (new SaveResulLoi_iso_elas3D(*this));};
120 // affectation
121 virtual SaveResul & operator = ( const SaveResul & a)
122 { SaveResulLoi_iso_elas3D& sav = *((SaveResulLoi_iso_elas3D*) &a);
123   E=sav.E; nu=sav.nu;E_t=sav.E_t; nu_t=sav.nu_t;
124   map_type_quelconque = sav.map_type_quelconque;
125   return *this;
126 };
127 //===== lecture écriture dans base info =====
128 // cas donne le niveau de la récupération
129 // = 1 : on récupère tout
130 // = 2 : on récupère uniquement les données variables (supposées comme telles)
131 void Lecture_base_info (ifstream& ent,const int cas)
132 {string toto; ent >> toto >> E >> toto >> nu; E_t=E; nu_t = nu; };
133 // cas donne le niveau de sauvegarde
134 // = 1 : on sauvegarde tout
135 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
136 void Ecriture_base_info(ofstream& sort,const int cas)
137 {sort << "\n E= " << E << " nu= " << nu << " ";};
138
139 // mise à jour des informations transitoires
140 void TdtversT()
141 {E_t = E; nu_t = nu;
142 // mise à jour de la liste des grandeurs quelconques internes
143 Mise_a_jour_map_type_quelconque();
144 };
145 void TversTdt()
146 {E = E_t; nu = nu_t;
147 // mise à jour de la liste des grandeurs quelconques internes
148 Mise_a_jour_map_type_quelconque();
149 };
150
151 // affichage à l'écran des infos
152 void Affiche() const
153 { cout << "\n E= " << E << " nu= " << nu << " ";};
154
155 //changement de base de toutes les grandeurs internes tensorielles stockées
156 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gb
157 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
158 // gpH(i) = gamma(i,j) * gH(j)
159 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) {};
160
161 // procedure permettant de completer éventuellement les données particulières
162 // de la loi stockées
163 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
164 // completer est appelé apres sa creation avec les donnees du bloc transmis
165 // peut etre appeler plusieurs fois
166 virtual SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>&
tab_coor
167 ,const Loi_comp_abstraite* loi) {return NULL;};
168
169
170 //-----
171 // données
172 //-----
173 double E,E_t,nu,nu_t; // les paramètres matériaux réellement utilisés
174
175 // --- gestion d'une map de grandeurs quelconques éventuelles ---
176
177 // une map de grandeurs quelconques particulière qui peut servir aux classes appelantes
178 // il s'agit ici d'une map interne qui a priori ne doit servir qu'aux class loi de comportement
179 // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
180 // -> n'est pas sauvegardé, car a priori il s'agit de grandeurs redondantes
181 map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >
map_type_quelconque;
182
183 // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
184 const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >*
Map_type_quelconque()
185 {return &map_type_quelconque;};
186 private:
187 void Mise_a_jour_map_type_quelconque();
188
189 // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ----
190 };
191
192 SaveResul * New_et_Initialise()
193 { SaveResulLoi_iso_elas3D * pt = new SaveResulLoi_iso_elas3D();
194 // insertion éventuelle de conteneurs de grandeurs quelconque
195 Insertion_conteneur_dans_save_result(pt);
196 return pt;
197 };
198
199 friend class SaveResulLoi_iso_elas3D;
200
201 // Lecture des donnees de la classe sur fichier

```

```

202     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
203                                     ,LesFonctions_nD& lesFonctionsnD);
204     // affichage de la loi
205     void Affiche() const ;
206     // test si la loi est complete
207     // = 1 tout est ok, =0 loi incomplete
208     int TestComplet();
209
210     // calcul d'un module d'young equivalent à la loi, ceci pour un
211     // chargement nul
212     double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
213     // récupération d'un module de compressibilité equivalent à la loi pour un chargement nul
214     // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
215     double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
216         saveResul);
217
218     // récupération de la variation relative d'épaisseur calculée: h/h0
219     // cette variation n'est utile que pour des lois en contraintes planes
220     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
221     // - pour les lois 2D def planes: retour de 0
222     // les infos nécessaires à la récupération , sont stockées dans saveResul
223     // qui est le conteneur spécifique au point où a été calculé la loi
224     virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
225
226     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
227     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_iso_elas3D(*this)); };
228
229     //----- lecture écriture de restart -----
230     // cas donne le niveau de la récupération
231     // = 1 : on récupère tout
232     // = 2 : on récupère uniquement les données variables (supposées comme telles)
233     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
234         lesCourbes1D
235                                     ,LesFonctions_nD& lesFonctionsnD);
236
237     // cas donne le niveau de sauvegarde
238     // = 1 : on sauvegarde tout
239     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
240     void Ecriture_base_info_loi(ofstream& sort,const int cas);
241
242     // affichage et definition interactive des commandes particulières à chaque lois
243     void Info_commande_LoisDeComp(UtilLecture& lec);
244
245     // récupération des grandeurs particulière (hors ddl )
246     // correspondant à liTQ
247     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
248     virtual void Grandeur_particuliere
249         (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
250         ;
251     // récupération de la liste de tous les grandeurs particulières
252     // ces grandeurs sont ajoutées à la liste passées en paramètres
253     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
254     virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
255
256     // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
257     // passée en paramètre, dans le save result: ces conteneurs doivent être valides
258     // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
259     virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul);
260
261     // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
262     // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
263     virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >& listEnuQuelc);
264
265     protected :
266
267     // donnees protegees
268     double E,nu; // paramètres de la loi
269     CourbelD* E_temperature; // courbe éventuelle d'évolution de E en fonction de la température
270     Fonction_nD * E_nD; // fonction nD éventuelle pour E
271     Fonction_nD * nu_nD; // fonction nD éventuelle pour nu
272     short int cas_calcul; // indique le choix entre différents types de calcul possible
273     // = 0 : calcul normal
274     // = 1 : calcul seulement déviatorique (la partie sphérique est mise à zéro)
275     // = 2 : calcul seulement sphérique (la partie déviatorique est mise à zéro)
276
277     // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
278     // Calcul_sigma_deps, dans le cas où on n'est pas en orthonormee
279     Tenseur3HHHH I_x_I_HHHH,I_xbarre_I_HHHH,I_x_eps_HHHH,Ixbarre_eps_HHHH;
280
281     // codage des METHODES VIRTUELLES protegees:
282     // calcul des contraintes a t+dt
283     // calcul des contraintes
284     void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
285         ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB & giB,BaseH & gi_H, TenseurBB & epsBB_
286         ,TenseurBB & delta_epsBB_
287         ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
288         ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
289         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&

```

```

module_cisaillement
286     ,const Met_abstraite::Expli_t_tdt& ex);
287
288     // calcul des contraintes et de ses variations a t+dt
289 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
290     ,BaseB& giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
291     ,BaseB& giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH& giH_tdt, Tableau <BaseH> & d_giH_tdt
292     ,TenseurBB & epsBB_tdt, Tableau <TenseurBB *> & d_epsBB
293     ,TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
294     ,Tableau <TenseurBB *> & d_gijBB_tdt
295     ,Tableau <TenseurHH *> & d_gijHH_tdt, double& jacobien_0, double& jacobien
296     ,Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *> & d_sigHH
297     ,EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
298 module_cisaillement
299     ,const Met_abstraite::Impli& ex);
300
301     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
302     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
303     // le tenseur de déformation et son incrémentsont également en orthonormees
304     // si = false: les bases transmises sont utilisées
305     // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
306 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
307     ,TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
308     ,TenseurHH& sigHH, TenseurHHH& d_sigma_deps
309     ,EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
310 module_cisaillement
311     ,const Met_abstraite::Umat_cont& ex) ; // = 0;
312
313 // fonction surchargée dans les classes dérivée si besoin est
314 virtual void CalculGrandeurTravail
315     (const PtIntegMecaInterne& ,const Deformation &
316     ,Enum_dure, const ThermoDonnee&
317     ,const Met_abstraite::Impli* ex_impli
318     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
319     ,const Met_abstraite::Umat_cont* ex_umat
320     ,const List_io<Ddl_etendu>* exclure_dd_etendu
321     ,const List_io<const TypeQuelconque *>* exclure_Q
322     ) {};
323 };
324 /// @} // end of group
325
326
327 #endif
328
329
330

```

## 7.63 Iso\_elas\_expo1D.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      3/5/2002
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *

```

```

35 *                                     $ *
36 *   PROJET:      Herezh++                                     $ *
37 *                                     $ *
38 *****
39 *   BUT:   La classe Iso_elas_exp01D permet de calculer la contrainte*
40 *          et ses derivees pour une loi isotrope elastique non *
41 *          lineaire du type : sig = f(invariants eps) E*eps *
42 *   Il s'agit d'une classe derivee de la classe Loi_comp_abstraite. *
43 *                                     $ *
44 *   ***** *
45 *   VERIFICATION: *
46 * *
47 *   ! date !   auteur !           but *
48 *   ----- *
49 *   !           !           !           ! *
50 *                                     $ *
51 *   ***** *
52 *   MODIFICATIONS: *
53 *   ! date !   auteur !           but *
54 *   ----- *
55 *                                     $ *
56 *****/
57 #ifndef ISO_ELAS_ESP01D_H
58 #define ISO_ELAS_ESP01D_H
59
60
61
62
63 #include "Loi_comp_abstraite.h"
64 #include "CourbelD.h"
65 #include "MathUtil.h"
66
67
68 /// @addtogroup Les_lois_iso_non_lineaire
69 /// @{
70 ///
71
72 class Iso_elas_exp01D : public Loi_comp_abstraite
73 {
74
75
76     public :
77
78
79         // CONSTRUCTEURS :
80
81         // Constructeur par defaut
82         Iso_elas_exp01D ();
83
84
85         // Constructeur de copie
86         Iso_elas_exp01D (const Iso_elas_exp01D& loi) ;
87
88         // DESTRUCTEUR :
89
90         ~Iso_elas_exp01D ();
91
92     // initialise les donnees particulieres a l'elements
93     // de matiere traite ( c-a-dire au pt calcule)
94     // Il y a creation d'une instance de SaveResul particuliere
95     // a la loi concerne
96     // la SaveResul classe est remplie par les instances heritantes
97     // le pointeur de SaveResul est sauvegarde au niveau de l'element
98     // c'a-d que les info particulieres au point considere sont stocke
99     // au niveau de l'element et non de la loi.
100     class SaveResulIso_elas_exp01D: public SaveResul
101     { public :
102         SaveResulIso_elas_exp01D(): // constructeur par defaut :
103             E(0.),nu(0.),E_t(0.),nu_t(0.),map_type_quelconque() {};
104         SaveResulIso_elas_exp01D(const SaveResulIso_elas_exp01D& sav): // de copie
105             E(sav.E),nu(sav.nu),E_t(sav.E_t),nu_t(sav.nu_t)
106             ,map_type_quelconque(sav.map_type_quelconque) {};
107
108         virtual ~SaveResulIso_elas_exp01D() {}; // destructeur
109         // definition d'une nouvelle instance identique
110         // appelle du constructeur via new
111         SaveResul * Nevez_SaveResul() const{return (new SaveResulIso_elas_exp01D(*this));};
112         // affectation
113         virtual SaveResul & operator = ( const SaveResul & a)
114         { SaveResulIso_elas_exp01D& sav = *((SaveResulIso_elas_exp01D*) &a);
115           E=sav.E; nu=sav.nu;E_t=sav.E_t; nu_t=sav.nu_t;
116           map_type_quelconque = sav.map_type_quelconque;
117           return *this;
118         };
119         //===== lecture écriture dans base info =====
120         // cas donne le niveau de la récupération

```

```

121         // = 1 : on récupère tout
122         // = 2 : on récupère uniquement les données variables (supposées comme telles)
123 void Lecture_base_info (ifstream& ent,const int cas)
124 {string toto; ent >> toto >> E >> toto >> nu; E_t=E; nu_t = nu; };
125         // cas donne le niveau de sauvegarde
126         // = 1 : on sauvegarde tout
127         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
128 void Ecrire_base_info(ofstream& sort,const int cas)
129 {sort << "\n E= " << E << " nu= " << nu << " "};};
130
131 // mise à jour des informations transitoires
132 void TdtversT()
133 {E_t = E; nu_t = nu;
134 // mise à jour de la liste des grandeurs quelconques internes
135 Mise_a_jour_map_type_quelconque();
136 };
137 void TversTdt()
138 {E = E_t; nu = nu_t;
139 // mise à jour de la liste des grandeurs quelconques internes
140 Mise_a_jour_map_type_quelconque();
141 };
142
143 // affichage à l'écran des infos
144 void Affiche() const
145 { cout << "\n E= " << E << " nu= " << nu << " "};};
146
147 //changement de base de toutes les grandeurs internes tensorielles stockées
148 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
149 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
150 // gpH(i) = gamma(i,j) * gH(j)
151 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) {};
152
153 // procedure permettant de completer éventuellement les données particulières
154 // de la loi stockées
155 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
156 // completer est appelé apres sa creation avec les donnees du bloc transmis
157 // peut etre appeler plusieurs fois
158 virtual SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>&
tab_coor
159                                     ,const Loi_comp_abstraite* loi) {return NULL;};
160
161
162 //-----
163 // données
164 //-----
165 double E,E_t,nu,nu_t; // les paramètres matériaux réellement utilisés
166
167 // --- gestion d'une map de grandeurs quelconques éventuelles ---
168
169 // une map de grandeurs quelconques particulière qui peut servir aux classes appelantes
170 // il s'agit ici d'une map interne qui a priori ne doit servir qu'aux class loi de
comportement
171 // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
172 // -> n'est pas sauvegardé, car a priori il s'agit de grandeurs redondantes
173 map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque > >
map_type_quelconque;
174
175 // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
176 const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque > >*
Map_type_quelconque()
177 {const {return &map_type_quelconque;};
178 private:
179 void Mise_a_jour_map_type_quelconque();
180
181 // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ---
182 };
183
184 SaveResul * New_et_Initialise()
185 { SaveResulIso_elas_exp0D * pt = new SaveResulIso_elas_exp0D();
186 // insertion éventuelle de conteneurs de grandeurs quelconque
187 Insertion_conteneur_dans_save_result(pt);
188 return pt;
189 };
190
191 friend class SaveResulIso_elas_exp0D;
192
193
194
195 // Lecture des donnees de la classe sur fichier
196 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
197                                     ,LesFonctions_nD& lesFonctionsnD);
198
199 // affichage de la loi
200 void Affiche() const ;
201 // test si la loi est complete
202 // = 1 tout est ok, =0 loi incomplete
203 int TestComplet();

```



```

204 //----- lecture écriture de restart -----
205 // cas donne le niveau de la récupération
206 // = 1 : on récupère tout
207 // = 2 : on récupère uniquement les données variables (supposées comme telles)
208 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
209                                     ,LesFonctions_nD& lesFonctionsnD);
210 // cas donne le niveau de sauvegarde
211 // = 1 : on sauvegarde tout
212 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
213 void Ecriture_base_info_loi(ofstream& sort,const int cas);
214
215
216 // calcul d'un module d'young équivalent la loi, ceci pour un
217 // chargement nul
218 double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
219 // récupération d'un module de compressibilité équivalent à la loi pour un chargement nul
220 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
221 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
saveResul);
222
223
224 // récupération de la variation relative d'épaisseur calculée: h/h0
225 // cette variation n'est utile que pour des lois en contraintes planes
226 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
227 // - pour les lois 2D def planes: retour de 0
228 // les infos nécessaires à la récupération , sont stockées dans saveResul
229 // qui est le conteneur spécifique au point où a été calculé la loi
230 virtual double HsurH0(SaveResul * saveResul) const
231 { cout << "\n Iso_elas_expo1D::HsurH0(.. , methode non implante pour l'instant ";
232   Sortie(1);
233 };
234
235 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
236 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Iso_elas_expo1D(*this)); };
237
238 // affichage et definition interactive des commandes particulières à chaque lois
239 void Info_commande_LoisDeComp(UtilLecture& lec);
240
241 // récupération des grandeurs particulière (hors ddl )
242 // correspondant à liTQ
243 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
244 virtual void Grandeur_particuliere
245 (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
246 ;
247 // récupération de la liste de tous les grandeurs particulières
248 // ces grandeurs sont ajoutées à la liste passées en paramètres
249 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
250 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const;
251
252 // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
253 // passée en paramètre, dans le save result: ces conteneurs doivent être valides
254 // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
255 virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul);
256
257 // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
258 // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
259 virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >& listEnuQuelc);
260
261 protected :
262 // donnée de la loi
263 double E,nu; // module d'young et coef de poisson qui sert pour les modules de compressibilité et
cisaillement
264 Courbe1D* f_coefficient; // courbe coefficient
265 Fonction_nD * E_nD; // fonction nD éventuelle pour E
266 Fonction_nD * nu_nD; // fonction nD éventuelle pour nu
267
268 // codage des METHODES VIRTUELLES proteges:
269 // calcul des contraintes a t+dt
270 // calcul des contraintes
271 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
272 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
273 ,TenseurBB & delta_epsBB_
274 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
275 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
276 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
277 ,const Met_abstraite::Expli_t_tdt& ex);
278
279 // calcul des contraintes et de ses variations a t+dt
280 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
281 ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
282 ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
283 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
284 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
285 ,Tableau <TenseurBB *>& d_gijBB_tdt

```

```

286         ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
287         ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
288         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
289         ,const Met_abstraite::Impli& ex);
290
291
292
293         // fonction surchargée dans les classes dérivée si besoin est
294         virtual void CalculGrandeurTravail
295             (const PtIntegMecaInterne& ,const Deformation &
296             ,Enum_dure,const ThermoDonnee&
297             ,const Met_abstraite::Impli* ex_impli
298             ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
299             ,const Met_abstraite::Umat_cont* ex_umat
300             ,const List_io<Ddl_etendu>* exclure_dd_etend
301             ,const List_io<const TypeQuelconque *>* exclure_Q
302             ) {};
303
304 };
305 /// @} // end of group
306
307
308 #endif
309
310
311

```

## 7.64 Iso\_elas\_expo3D.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:          27/04/2004
33 *
34 *      AUTEUR:        G RIO      (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:        Herezh++
37 *
38 *
39 *      BUT:           La classe Iso_elas_expo3D permet de calculer la contrainte
40 *                    et ses derivees pour une loi isotrope elastique non
41 *                    lineaire du type : sig = f(II_eps) E*eps
42 *                    Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
43 *
44 *                    *****
45 *
46 *      VERIFICATION:
47 *
48 *      ! date !      auteur !      but
49 *      -----
50 *
51 *      *****
52 *
53 *      MODIFICATIONS:
54 *      ! date !      auteur !      but
55 *      -----

```

```

55  *
56  *****
57  #ifndef ISO_ELAS_ESPO3D_H
58  #define ISO_ELAS_ESPO3D_H
59
60
61
62
63  #include "Loi_comp_abstraite.h"
64  #include "CourbelD.h"
65  #include "MathUtil.h"
66
67
68  /// @addtogroup Les_lois_iso_non_lineaire
69  /// @{
70  ///
71
72  class Iso_elas_expo3D : public Loi_comp_abstraite
73  {
74
75
76      public :
77
78
79          // CONSTRUCTEURS :
80
81          // Constructeur par défaut
82          Iso_elas_expo3D ();
83
84
85          // Constructeur de copie
86          Iso_elas_expo3D (const Iso_elas_expo3D& loi) ;
87
88          // DESTRUCTEUR :
89
90          ~Iso_elas_expo3D ();
91
92          // Lecture des donnees de la classe sur fichier
93          void LectureDonneesParticulieres
94              (UtilLecture * ,LesCourbes1D& lesCourbes1D,LesFonctions_nD& lesFonctionsnD);
95          // affichage de la loi
96          void Affiche() const ;
97          // test si la loi est complete
98          // = 1 tout est ok, =0 loi incomplete
99          int TestCompleet();
100
101          //----- lecture écriture de restart -----
102          // cas donne le niveau de la récupération
103          // = 1 : on récupère tout
104          // = 2 : on récupère uniquement les données variables (supposées comme telles)
105          void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
106              lesCourbes1D
107              ,LesFonctions_nD& lesFonctionsnD);
108          // cas donne le niveau de sauvegarde
109          // = 1 : on sauvegarde tout
110          // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
111          void Ecriture_base_info_loi(ofstream& sort,const int cas);
112
113          // calcul d'un module d'young équivalent à la loi pour un chargement nul
114          double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
115          // récupération d'un module de compressibilité équivalent à la loi pour un chargement nul
116          // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
117          double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def);
118
119          // récupération de la variation relative d'épaisseur calculée: h/h0
120          // cette variation n'est utile que pour des lois en contraintes planes
121          // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
122          // - pour les lois 2D def planes: retour de 0
123          // les infos nécessaires à la récupération , sont stockées dans saveResul
124          // qui est le conteneur spécifique au point où a été calculé la loi
125          virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
126
127          // création d'une loi à l'identique et ramène un pointeur sur la loi créée
128          Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Iso_elas_expo3D(*this)); };
129
130          // affichage et definition interactive des commandes particulières à chaque lois
131          void Info_commande_LoisDeComp(UtilLecture& lec);
132
133      protected :
134          // donnée de la loi
135          double E,nu; // module d'young et coef de poisson
136          CourbelD* f_coefficient; // courbe coefficient
137          // on introduit un certain nombre de tenseur du quatrième ordre, qui vont nous servir pour
138          // Calcul_dsigma_deps, dans le cas où on n'est pas en orthonormee
139          Tenseur3HHHH I_x_I_HHHH,I_xbarre_I_HHHH,I_x_eps_HHHH,Ixbarre_eps_HHHH;
140
141          // codage des METHODES VIRTUELLES protegees:

```

```

141 // calcul des contraintes a t+dt
142 // calcul des contraintes
143 void Calcul_SigmaHH (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
144 , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB & giB, BaseH & gi_H, TenseurBB & epsBB_
145 , TenseurBB & delta_epsBB_
146 , TenseurBB & gijBB_, TenseurHH & gijHH_, Tableau <TenseurBB *> & d_gijBB_
147 , double& jacobien_0, double& jacobien, TenseurHH & sigHH
148 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
149 , const Met_abstraite::Expli_t_tdt& ex);
150
151 // calcul des contraintes et de ses variations a t+dt
152 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
153 , BaseB & giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
154 , BaseB & giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH & giH_tdt, Tableau <BaseH> & d_giH_tdt
155 , TenseurBB & epsBB_tdt, Tableau <TenseurBB *> & d_epsBB
156 , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
157 , Tableau <TenseurBB *> & d_gijBB_tdt
158 , Tableau <TenseurHH *> & d_gijHH_tdt, double& jacobien_0, double& jacobien
159 , Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *> & d_sigHH
160 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
161 , const Met_abstraite::Impli& ex);
162
163 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
164 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
165 // le tenseur de déformation et son incrémentsont également en orthonormees
166 // si = false: les bases transmises sont utilisées
167 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
168 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
169 , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
170 , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps
171 , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
172 , const Met_abstraite::Umat_cont& ex) ; // = 0;
173
174 // fonction surchargée dans les classes dérivée si besoin est
175 virtual void CalculGrandeurTravail
176 (const PtIntegMecaInterne& , const Deformation &
177 , Enum_dure, const ThermoDonnee&
178 , const Met_abstraite::Impli* ex_impli
179 , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
180 , const Met_abstraite::Umat_cont* ex_umat
181 , const List_io<Ddl_etendu>* excludre_dd_etend
182 , const List_io<const TypeQueIconque *>* excludre_0
183 ) {};
184
185 };
186 /// @} // end of group
187
188
189 #endif
190
191
192

```

## 7.65 Iso\_elas\_SE1D.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.

```

```

28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      15/9/2003
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++
37 *
38 *
39 *   BUT:       La classe Iso_elas_SE1D permet de calculer la contrainte
40 *             et ses derivees pour une loi isotrope elastique non
41 *             lineaire du type : sig = f(eps) en 1D
42 *             Il s'agit d'une classe derivee de la classe Loi_comp_abstraite.
43 *
44 *   *****
45 *
46 *   VERIFICATION:
47 *
48 *   ! date !   auteur !           but
49 *   -----
50 *   !           !           !
51 *   *****
52 *
53 *   MODIFICATIONS:
54 *
55 *   ! date !   auteur !           but
56 *   -----
57 *   $
58 *
59 *   *****/
60
61 #ifndef ISO_ELAS_SE1D_H
62 #define ISO_ELAS_SE1D_H
63
64 #include "Loi_comp_abstraite.h"
65 #include "Courbe1D.h"
66 #include "MathUtil.h"
67
68 /** @defgroup Les_lois_iso_non_lineaire
69 *
70 *   BUT:   groupe des lois de type iso non linéaire
71 *
72 *
73 *   \author   Gérard Rio
74 *   \version  1.0
75 *   \date    15/9/2003
76 *   \brief   Définition des lois de type iso non linéaire
77 *
78 */
79
80 /// @addtogroup Les_lois_iso_non_lineaire
81 /// @{
82 ///
83
84 class Iso_elas_SE1D : public Loi_comp_abstraite
85 {
86
87     public :
88
89         // CONSTRUCTEURS :
90
91         // Constructeur par défaut
92         Iso_elas_SE1D ();
93
94         // Constructeur de copie
95         Iso_elas_SE1D (const Iso_elas_SE1D& loi) ;
96
97         // DESTRUCTEUR :
98         ~Iso_elas_SE1D ();
99
100        // Lecture des donnees de la classe sur fichier
101        void LectureDonneesParticulieres
102            (UtilLecture * , LesCourbes1D& lesCourbes1D, LesFonctions_nD& lesFonctionsnD);
103
104        // affichage de la loi
105        void Affiche() const ;
106
107        // test si la loi est complete
108        // = 1 tout est ok, =0 loi incomplete
109        int TestCompleet();
110
111        //----- lecture écriture de restart -----
112
113

```

```

114 // cas donne le niveau de la récupération
115 // = 1 : on récupère tout
116 // = 2 : on récupère uniquement les données variables (supposées comme telles)
117 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
118                                     ,LesFonctions_nD& lesFonctionsnD);
119 // cas donne le niveau de sauvegarde
120 // = 1 : on sauvegarde tout
121 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
122 void Ecriture_base_info_loi(ofstream& sort,const int cas);
123
124 // calcul d'un module d'young équivalent à la loi, ceci pour un
125 // chargement nul
126 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * )
127 { return f_coefficient->Valeur(0.); };
128
129 // récupération de la variation relative d'épaisseur calculée: h/h0
130 // cette variation n'est utile que pour des lois en contraintes planes
131 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
132 // - pour les lois 2D def planes: retour de 0
133 // les infos nécessaires à la récupération , sont stockées dans saveResul
134 // qui est le conteneur spécifique au point où a été calculé la loi
135 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
136
137 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
138 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Iso_elas_SE1D(*this)); };
139
140 // affichage et definition interactive des commandes particulières à chaque lois
141 void Info_commande_LoisDeComp(UtilLecture& lec);
142
143 protected :
144 // donnée de la loi
145 CourbelD* f_coefficient; // courbe coefficient
146 bool symetrique; // indique si la courbe est symétrique / 0 ou non
147 double nu; // par défaut = 0
148
149 // codage des METHODES VIRTUELLES protegees:
150 // calcul des contraintes a t+dt
151 // calcul des contraintes
152 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
153 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
154 ,TenseurBB & delta_epsBB_
155 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
156 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
157 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
158 ,const Met_abstraite::Expli_t_tdt& ex);
159
160 // calcul des contraintes et de ses variations a t+dt
161 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
162 ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
163 ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
164 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
165 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
166 ,Tableau <TenseurBB *>& d_gijBB_tdt
167 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
168 ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
169 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
170 ,const Met_abstraite::Impli& ex);
171
172
173 // fonction surchargée dans les classes dérivée si besoin est
174 virtual void CalculGrandeurTravail
175 (const PtIntegMecaInterne& ,const Deformation &
176 ,Enum_dure,const ThermoDonnee&
177 ,const Met_abstraite::Impli* ex_impli
178 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
179 ,const Met_abstraite::Umat_cont* ex_umat
180 ,const List_io<Ddl_etendu>* exclure_dd_etend
181 ,const List_io<const TypeQuelconque *>* exclure_Q
182 ) {};
183
184 };
185 /// @} // end of group
186
187
188 #endif
189
190
191

```

## 7.66 LesLoisDeComp.h

```

2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:      23/01/97
33 *
34 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:    Herezh++
37 *
38 *      BUT:      Gestion des differentes lois de comportement.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *
44 *      ! date !   auteur !           but
45 *      -----
46 *      !       !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      -----
52 *
53 *      *****/
54 #ifndef LESLOISDECOMP_H
55 #define LESLOISDECOMP_H
56
57
58 #include "LoiAbstraiteGeneral.h"
59 #include "UtilLecture.h"
60 #include "string"
61 #include "LesReferences.h"
62 #include "MotCle.h"
63 #include <list>
64 #include "LesCourbes1D.h"
65
66 class LesLoisDeComp
67 {
68 public :
69     // classe de stockage
70     class RefLoi
71     {
72     friend istream & operator » (istream &, LesLoisDeComp::RefLoi &);
73     // surcharge de l'operator d'écriture typée
74     friend ostream & operator « (ostream &, const LesLoisDeComp::RefLoi &);
75     public : string* nom_maillage; // nom du maillage associé
76             string st1; // nom d'une reference
77             string st2; // nom d'une loi
78     RefLoi() : st1(),st2(),nom_maillage(NULL) {};
79     // RefLoi(string & s1,string & s2) : nom_maillage(NULL),st1(s1), st2(s2) {};
80     RefLoi(string* nom, string & s1,string & s2)
81     : nom_maillage(NULL),st1(s1), st2(s2) { if (nom != NULL)nom_maillage=new string(*nom); };
82     RefLoi (const RefLoi & a)
83     : nom_maillage(NULL),st1(a.st1), st2(a.st2)
84     { if (a.nom_maillage != NULL) nom_maillage=new string(*(a.nom_maillage)); };
85     ~RefLoi() {if (nom_maillage != NULL) delete nom_maillage;};
86     RefLoi& operator = (const RefLoi & a);
87     // changement de nom de maillage

```

```

88     void Change_nom_maillage(const string& nouveau);
89     //----- lecture écriture de restart -----
90     // cas donne le niveau de la récupération
91     // = 1 : on récupère tout
92     // = 2 : on récupère uniquement les données variables (supposées comme telles)
93     void Lecture_base_info(istream& ent,const int cas);
94     // cas donne le niveau de sauvegarde
95     // = 1 : on sauvegarde tout
96     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
97     void Ecriture_base_info(ofstream& sort,const int cas) const;
98 };
99 class Loi
100 { // surcharge de l'operator de lecture typée
101     friend istream & operator » (istream &, LesLoisDeComp::Loi &);
102     // surcharge de l'operator d'écriture typée
103     friend ostream & operator « (ostream &, const LesLoisDeComp::Loi &);
104     public : string st1; // nom d'une loi
105     LoiAbstraiteGeneral * pt ; // pointeur de loi
106     Loi() {};
107     Loi(string & s1,LoiAbstraiteGeneral * p ) :
108         st1(s1)
109         { pt = p;};
110     Loi (const Loi & a) :
111         st1(a.st1)
112         { pt = a.pt;};
113     Loi& operator = (const Loi & a)
114     { st1 = a.st1;pt = a.pt; return *this; };
115     //----- lecture écriture de restart -----
116     // cas donne le niveau de la récupération
117     // = 1 : on récupère tout
118     // = 2 : on récupère uniquement les données variables (supposées comme telles)
119     void Lecture_base_info(istream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
120                                     ,LesFonctions_nD& lesFonctionsnD);
121     // cas donne le niveau de sauvegarde
122     // = 1 : on sauvegarde tout
123     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
124     void Ecriture_base_info(ofstream& sort,const int cas);
125 };
126
127 friend class Loi;
128 // CONSTRUCTEURS :
129 LesLoisDeComp (); // par défaut
130 // DESTRUCTEUR :
131
132 // METHODES PUBLIQUES :
133
134 // lecture des lois de comportement
135 void LecturelesLoisDeComp(UtilLecture& lec,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
136                                     ,LesFonctions_nD& lesFonctionsnD);
137
138 // affichage des lois de comportement
139 void Affiche() const ;
140
141 // affichage et definition interactive des commandes
142 void Info_commande_lesLoisDeComp(UtilLecture& lec,LesReferences& lesRef);
143
144 // ramene le tableau de reference de loi
145 // tableau des références de loi qui servent directement dans le calcul avec les éléments
146 inline const Tableau <RefLoi> & TabRefLoi() const
147     { return tabRefLoi; };
148
149 // ramene le tableau des loi
150 // tableau de toutes les lois lues: même celles qui ne servent pas directement, mais qui
151 // soit ne servent pas du tout, ou soit servent indirectement via par exemple des umat
152 inline const Tableau <Loi> & TabLoi() const
153     { return tabLoi; };
154
155 // ramene le pointeur de LoiAbstraiteGeneral en fonction
156 // 1) du nom de la loi, 2) du nom d'une reference
157 // ou null si la recherche n'abouti pas
158 LoiAbstraiteGeneral * PtLoi_abstraite(const string& st) const ;
159
160 // indique aux différentes loi que l'on doit utiliser une loi tangente simplifiée
161 // ou non suivant la valeur du paramètre de passage
162 void Loi_simplifie(bool ordre);
163 // récup du type de loi: simplifiée ou non
164 bool Test_loi_simplife();
165
166 // lecture de donnée en fonction d'un indicateur : int type
167 // pour l'instant ne fait rien
168 void LectureDonneesExternes(UtilLecture& ,LesReferences& ,const int ,const string&) {};
169
170 //indique aux lois Umat éventuelles le numéro d'itération
171 void MiseAJour_umat_nbiter(const int& num_iter);
172 // ou le numéro d'incrément
173 void MiseAJour_umat_nbincr(const int& num_incr);

```



```

174
175 //----- lecture écriture de restart -----
176 // cas donne le niveau de la récupération
177 // = 1 : on récupère tout
178 // = 2 : on récupère uniquement les données variables (supposées comme telles)
179 void Lecture_base_info(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
180                      ,LesFonctions_nD& lesFonctionsnD);
181 // cas donne le niveau de sauvegarde
182 // = 1 : on sauvegarde tout
183 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
184 void Ecriture_base_info(ofstream& sort,const int cas);
185
186 // définition d'un pointeur de loi en fonction d'une chaîne de caractères
187 static LoiAbstraiteGeneral * Def_loi(string & nom);
188
189 // sortie sur fichier des temps cpu spécifiques à toutes les lois
190 void Sortie_temps_cpu(UtilLecture& lec);
191
192 private :
193 // VARIABLES PROTEGEES :
194 // tableau des références de loi qui servent directement dans le calcul avec les éléments
195 Tableau <RefLoi> tabRefLoi;
196 // tableau de toutes les lois lues: même celles qui ne servent pas directement, mais qui
197 // soit ne servent pas du tout, ou soit servent indirectement via par exemple des umat
198 Tableau <Loi> tabLoi;
199 // la liste des lois umat éventuelles pour optimiser les routines MiseAJour_umat_nbiter
200 // et MiseAJour_umat_nbincr
201 list <LoiAbstraiteGeneral *> list_de_umat;
202
203 // la liste de tous les types de lois actuellement possibles (mais pas forcément lue)
204 static list <LoiAbstraiteGeneral *> list_de_loi;
205
206 // METHODES PROTEGEES :
207 // lecture des references de materiaux
208 void LecRef(UtilLecture& lec,LesReferences& lesRef);
209 // lecture des materiaux
210 void LecMateriaux(UtilLecture& lec,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
211                  ,LesFonctions_nD& lesFonctionsnD);
212
213 };
214
215 #endif

```

## 7.67 Loi\_comp\_abstraite.h

```

1 // FICHER : Loi_comp_abstraite.h
2 // CLASSE : Loi_comp_abstraite
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:      19/01/2001
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 * *****/

```

```

41 *      BUT: Loi générique pour les comportements mecaniques.      *
42 *                                                                 $ *
43 *      ***** *
44 *      *
45 *      VERIFICATION: *
46 *      ! date ! auteur ! but ! *
47 *      ----- *
48 *      ! ! ! ! *
49 *      $ *
50 *      ***** *
51 *      MODIFICATIONS: *
52 *      ! date ! auteur ! but ! *
53 *      ----- *
54 *      $ *
55 *****/
56
57 #ifndef LOI_COMP_ABSTRAITE_H
58 #define LOI_COMP_ABSTRAITE_H
59
60
61 #include "Enum_comp.h"
62 #include "Tableau_T.h"
63 #include "Tenseur.h"
64 #include "Deformation.h"
65 #include "LoiAbstraiteGeneral.h"
66 #include "ThermoDonnee.h"
67 #include "CompThermoPhysiqueAbstraite.h"
68 #include "TypeQuelconque.h"
69 #include "UmatAbaqus.h"
70 #include "EnergieMeca.h"
71 #include "LesPtIntegMecaInterne.h"
72 #include "Temps_CPU_H2pp.h"
73 #include "bloc.h"
74
75 class Loi_comp_abstraite : public LoiAbstraiteGeneral
76 {
77
78     public :
79     friend class LoiAdditiveEnSigma;
80     friend class LoiDesMelangesEnSigma;
81     friend class Loi_Umat;
82     friend class LoiContraintesPlanes;
83     friend class LoiContraintesPlanesDouble;
84     friend class LoiDeformationsPlanes;
85     friend class LoiCritere;
86     friend class Projection_anisotrope_3D;
87     friend class ElemMeca;
88
89     // CONSTRUCTEURS :
90
91     // Constructeur par défaut
92     Loi_comp_abstraite () ;
93
94     // Constructeur utile si l'identificateur du nom de la loi
95     // de comportement et la dimension sont connus
96     // vit_def indique si oui ou non la loi utilise la vitesse de déformation
97     Loi_comp_abstraite (Enum_comp id_compor,Enum_categorie_loi_comp categorie_comp
98     ,int dimension,bool vit_def = false);
99
100    // Constructeur utile si l'identificateur du nom de la loi
101    // de comportement et la dimension sont connus
102    // vit_def indique si oui ou non la loi utilise la vitesse de déformation
103    Loi_comp_abstraite (char* nom,Enum_categorie_loi_comp categorie_comp
104    ,int dimension,bool vit_def = false);
105
106    // Constructeur de copie
107    Loi_comp_abstraite (const Loi_comp_abstraite & a );
108
109    // DESTRUCTEUR VIRTUEL :
110
111    virtual ~Loi_comp_abstraite ();
112
113
114 // 2) METHODES VIRTUELLES public:
115
116 //----- classe SaveResul virtuelle pure
117 //-----
118 // initialise les donnees particulieres a l'elements
119 // de matiere traite ( c-a-dire au pt calcule)
120 // Il y a creation d'une instance de SaveResul particuliere
121 // a la loi concernee
122 // la SaveResul classe est remplie par les instances heritantes
123 // le pointeur de SaveResul est sauvegarde au niveau de l'element
124 // c'a-d que les info particulieres au point considere sont stocke
125 // au niveau de l'element et non de la loi.
126 class SaveResul

```

```

126     { public :
127         // destructeur
128         virtual ~SaveResul() {};
129         // définition d'une nouvelle instance identique
130         // appelle du constructeur via new
131         virtual SaveResul * Nevez_SaveResul() const =0;
132     // affectation
133     // *** attention : dans le cas de grandeurs internes pointées:
134     // les grandeurs pointées sont affectées, mais si elles sont de nature différente -> erreur
135     // donc la surcharge n'est utilisable que pour des grandeurs que l'on sait de même nature !!!!!
136     virtual SaveResul & operator = ( const SaveResul & ) = 0;
137
138         //===== lecture écriture dans base info =====
139         // cas donne le niveau de la récupération
140         // = 1 : on récupère tout
141         // = 2 : on récupère uniquement les données variables (supposées comme telles)
142         virtual void Lecture_base_info (ifstream& ent,const int cas) = 0;
143         // cas donne le niveau de sauvegarde
144         // = 1 : on sauvegarde tout
145         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
146         virtual void Ecriture_base_info(ofstream& sort,const int cas) = 0;
147
148         // mise à jour des informations transitoires en définitif s'il y a convergence
149         // par exemple (pour la plasticité par exemple)
150         virtual void TdtversT() {};
151         virtual void TversTdt() {};
152
153         // affichage à l'écran des infos
154         virtual void Affiche() const = 0;
155
156         //changement de base de toutes les grandeurs internes tensorielles stockées
157         // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
158         // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
159         // gpH(i) = gamma(i,j) * gH(j)
160         virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) = 0;
161
162         // procedure permettant de completer éventuellement les données particulières
163         // de la loi stockées
164         // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
165         // completer est appelé apres sa creation avec les donnees du bloc transmis
166         // peut etre appeler plusieurs fois
167         virtual SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
168             ,const Loi_comp_abstraite* loi) = 0;
169
170         // test si le conteneur est complet
171         // = 1 tout est ok, =0 conteneur incomplet
172         virtual int TestCompleter()const {return 1;};
173
174         // ramène la liste des types de grandeurs qui sont stocké, et éventuellement consultables
175
176         // ---- récupération d'information: spécifique à certaine classe dérivée
177         virtual double Deformation_plastique()
178         { cout << "\n méthode non implanté pour cette loi de comportement"
179           << "\n SaveResul::Deformation_plastique()";
180           Sortie(1); return 0.;
181         }
182
183         // --- gestion d'une map de grandeurs quelconques éventuelles ---
184         // la map de grandeurs quelconques qui est alimentée par les classes dérivées
185         // il s'agit ici d'une map interne qui ne doit servir qu'aux classes loi de comportement
186         // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
187
188         // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
189         virtual const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >*
190         Map_type_quelconque()
191         const {return NULL;};
192
193         // pour info, pour que la map soit fonctionnelle : il faut: (cf. exemple de Hysteresis_bulk)
194         // - définir pour le stockage la méthode : Map_type_quelconque()
195         // - définir pour le stockage la méthode : Mise_a_jour_map_type_quelconque() et s'en
196         // servir dans le stockage local
197         // - définir pour la loi la méthode: Insertion_conteneur_dans_save_result()
198         // - définir pour la loi la méthode: Activation_stockage_grandeurs_quelconques()
199
200         // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ----
201     };
202
203     virtual SaveResul * New_et_Initialise() { return NULL;};
204
205     // affichage des donnees particulieres a l'elements
206     // de matiere traite ( c-a-dire au pt calcul)
207     virtual void AfficheDataSpecif(ofstream& ,SaveResul * ) const {};
208
209     //----- classe SaveResul_C -----
210     // cette classe permet de gérer une ou plusieurs contraintes, qui peuvent conduire à annuler l'action
211     // des contraintes

```

```

210 // par exemple au-dessus d'une déformation seuil, on considère qu'il y a rupture, on annule donc la
      contraintes et la raideur éventuelle
211
212     class SaveResul_C
213     { public :
214         SaveResul_C():actif_t(true),actif(true) {}; // constructeur par défaut :
215         SaveResul_C(const SaveResul_C& sav):actif_t(sav.actif_t),actif(sav.actif){}; // de copie
216     virtual ~SaveResul_C(){}; // destructeur
217         // définition d'une nouvelle instance identique
218         // appelle du constructeur via new
219         virtual SaveResul_C * Nevez_SaveResul_C()
220         { SaveResul_C * pt = new SaveResul_C(); return pt;};
221         //===== lecture écriture dans base info =====
222         // cas donne le niveau de la récupération
223         // = 1 : on récupère tout
224         // = 2 : on récupère uniquement les données variables (supposées comme telles)
225     virtual void Lecture_base_info (ifstream& ent,const int )
226         {string toto; ent > toto > actif_t ;};
227         // cas donne le niveau de sauvegarde
228         // = 1 : on sauvegarde tout
229         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
230     virtual void Ecriture_base_info(ofstream& sort,const int )
231         {sort << "\n actif_t= " << actif_t << " ";};
232
233         // mise à jour des informations transitoires en définitif s'il y a convergence
234         // par exemple (pour la plasticité par exemple)
235     virtual void TdtversT() {actif_t=actif;};
236     virtual void TversTdt() {actif=actif_t;};
237
238         // ramène la liste des types de grandeurs qui sont stocké, et éventuellement consultables
239     public :
240         bool actif,actif_t; // indique si actuellement c'est actif ou pas, et idem au pas précédent
241
242     };
243
244
245 //----- fin classe SaveResul_C -----
246
247     // définition du type de calcul de déformation sur une instance de déformation passée en
      paramètre
248     void Def_type_deformation(Deformation & def);
249
250     // schema de calcul explicite à t
251     virtual const Met_abstraite::Expli& Cal_explicit_t
252         (Loi_comp_abstraite::SaveResul * saveDon
253         ,Deformation & def, DdlElement & tab_ddl
254         ,PtIntegMecaInterne& ptintmeca,Tableau <TenseurBB *> & d_epsBB,double& Jacobien
255         ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite* loiTP
256         ,bool dilatation,EnergieMeca & energ,const EnergieMeca & energ_t,bool premier_calcul
257         );
258
259     // schema de calcul explicite à tdt
260     virtual const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt
261         (Loi_comp_abstraite::SaveResul * saveDon
262         ,Deformation & def, DdlElement & tab_ddl
263         ,PtIntegMecaInterne& ptintmeca,Tableau <TenseurBB *> & d_epsBB,double& Jacobien
264         ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite* loiTP
265         ,bool dilatation,EnergieMeca & energ,const EnergieMeca & energ_t,bool premier_calcul
266         );
267
268     // schema implicite
269     virtual const Met_abstraite::Impli& Cal_implicit
270         (Loi_comp_abstraite::SaveResul * saveDon
271         ,Deformation & def,DdlElement & tab_ddl
272         ,PtIntegMecaInterne& ptintmeca, Tableau <TenseurBB *>& d_epsBB_tdt,double& jacobien
273         ,Vecteur& d_jacobien_tdt,Tableau <TenseurHH *>& d_sigHH,const ParaAlgoControle
274         & pa
275         ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite*
276         loiTP
277         ,bool dilatation,EnergieMeca & energ,const EnergieMeca & energ_t,bool
278         premier_calcul
279         );
280
281     // schema pour le flambage linéaire
282     virtual void Cal_flamb_lin
283         (Loi_comp_abstraite::SaveResul * saveDon,Deformation & def
284         ,DdlElement & tab_ddl
285         ,PtIntegMecaInterne& ptintmeca, Tableau <TenseurBB *>& d_epsBB_tdt,double& jacobien
286         ,Vecteur& d_jacobien_tdt,Tableau <TenseurHH *>& d_sigHH,const ParaAlgoControle & pa
287         ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite*
288         loiTP
289         ,bool dilatation,EnergieMeca & energ,const EnergieMeca & energ_t,bool
290         premier_calcul
291         );
292
293     // schema pour le calcul de la loi de comportement dans le cas de l'umat

```

```

289 virtual void ComportementUmat
290     ( Loi_comp_abstraite::SaveResul * saveDon
291       ,Deformation & def,PtIntegMecaInterne& ptintmeca
292       ,ParaAlgoControle & pa
293       ,CompThermoPhysiqueAbstraite::SaveResul * saveTP,CompThermoPhysiqueAbstraite*
294       loiTP
295       ,bool dilatation,UmatAbaqus& umatAbaqusqus,bool premier_calcul
296     );
297
298 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
299 // exemple: mise en service des ddl de température aux noeuds
300 virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud,bool
301 dilatation,LesPtIntegMecaInterne& lesPtMecaInt)
302     {Activ_donnees(tabnoeud,dilatation,lesPtMecaInt)};
303
304 // besoin de grandeurs particulières: la lois fournies la liste de grandeurs particulières dont elle
305 // aimerait disposer des valeurs (qui ne sont donc pas directement disponibles)
306 virtual void Besoin_de_grandeurs_particuliere(list <EnumTypeQuelconque >& listEnuQuelc) const {};
307
308 // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
309 // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
310 virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >& listEnuQuelc)
311     {};
312
313 // test pour savoir si une grandeur possiblement accessible en lecture
314 // via le conteneur SaveResul,
315 // ramène true si elle la grandeur existe pour la loi
316 bool Existe_stockage_grandeurs_quelconques(EnumTypeQuelconque enuQuelc) const
317     {if
318     (find(listdeTouslesQuelc_dispo_localement.begin(),listdeTouslesQuelc_dispo_localement.end(),enuQuelc)
319     == listdeTouslesQuelc_dispo_localement.end())
320     return false; else return true;
321     };
322
323 // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
324 // passée en paramètre, dans le save result: ces conteneurs doivent être valides
325 // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
326 virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul) {} ;
327
328 // acces en lecture à la liste des grandeurs locales qui sont dispo localement via saveResul
329 const list <EnumTypeQuelconque >& ListQuelc_mis_en_acces_localement() const {return
330 listQuelc_mis_en_acces_localement;};
331
332 // acces en lecture à la liste de tous les grandeurs locales qui pourraient être dispo localement via
333 saveResul
334 const list <EnumTypeQuelconque >& ListdeTouslesQuelc_dispo_localement() const {return
335 listdeTouslesQuelc_dispo_localement;};
336
337 // modification de l'indicateur de comportement tangent
338 void Modif_comp_tangent_simplifie(bool modif)
339     { comp_tangent_simplifie = modif;};
340
341 // test pour connaître l'état du comportement : simplifié ou non
342 bool Test_loi_simplifie()
343     { return comp_tangent_simplifie;};
344
345 // calcul d'un module d'young équivalent à la loi, ceci pour un
346 // chargement nul
347 virtual double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
348 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
349 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
350 virtual double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
351 saveResul);
352 // récupération d'un module de cisaillement équivalent à la loi
353 // virtual double Module_cisaillement_equivalent(Deformation & def,PtIntegMecaInterne&
354 ptintmeca);
355
356 // récupération de la dernière déformation d'épaisseur calculée: cette déformaion n'est utile que pour
357 des lois en contraintes planes ou doublement planes
358 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
359 // - pour les lois 2D def planes: retour de 0
360 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
361 // qui est le conteneur spécifique au point où a été calculé la loi
362 virtual double Eps33BH(SaveResul * saveResul) const ;
363
364 // récupération de la dernière déformation de largeur calculée: cette déformaion n'est utile que pour
365 des lois en contraintes doublement planes
366 // - pour les lois 3D et 2D : retour d'un nombre très grand, indiquant que cette fonction est invalide
367 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
368 // qui est le conteneur spécifique au point où a été calculé la loi
369 virtual double Eps22BH(SaveResul * saveResul) const ;
370
371 // récupération de la variation relative d'épaisseur calculée: h/h0
372 // cette variation n'est utile que pour des lois en contraintes planes
373 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
374 // - pour les lois 2D def planes: retour de 0

```

```

365 // les infos nécessaires à la récupération , sont stockées dans saveResul
366 // qui est le conteneur spécifique au point où a été calculé la loi
367 virtual double HsurH0(SaveResul * saveResul) const = 0;
368
369 // récupération de la variation relative d'épaisseur calculée: h/h0
370 // et de sa variation par rapport aux ddls la concernant: d_hsurh0
371 // cette variation n'est utile que pour des lois en contraintes planes
372 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
373 // - pour les lois 2D def planes: retour de 0
374 // les infos nécessaires à la récupération , sont stockées dans saveResul
375 // qui est le conteneur spécifique au point où a été calculé la loi
376 virtual double d_HsurH0(SaveResul * saveResul, Vecteur & d_hsurh0) const ;
377
378 // récupération de la variation relative de largeur calculée: b/b0
379 // cette variation n'est utile que pour des lois en contraintes planes double
380 // - pour les lois 3D et 2D : retour d'un nombre très grand, indiquant que cette fonction est invalide
381 // les infos nécessaires à la récupération , sont stockées dans saveResul
382 // qui est le conteneur spécifique au point où a été calculé la loi
383 virtual double BsurB0(SaveResul * saveResul) const ;
384
385 // récupération de la variation relative de largeur calculée: b/b0
386 // et de sa variation par rapport aux ddls la concernant: d_bsurb0
387 // cette variation n'est utile que pour des lois en contraintes planes
388 // - pour les lois 3D et 2D : retour d'un nombre très grand, indiquant que cette fonction est invalide
389 // les infos nécessaires à la récupération , sont stockées dans saveResul
390 // qui est le conteneur spécifique au point où a été calculé la loi
391 virtual double d_BsurB0(SaveResul * saveResul, Vecteur & d_bsurb0) const ;
392
393 // indique si la loi est en contraintes planes en s'appuyant sur un comportement 3D
394 virtual bool Contraintes_planes_de_3D() const {return false;};
395
396 // signale si la loi est thermo dépendante ou pas
397 bool ThermoDependante() const {return thermo_dependant;};
398
399 // mise à jour des températures d'une manière unilatérale: sert par exemple pour
400 // le calcul du module d'young équivalent, quand il est calculé en dehors du calcul des contraintes
401 void Mise_a_jour_temperature(Enum_dure temps, Deformation & def);
402
403 // récupération des grandeurs particulière (hors ddl )
404 // correspondant à liTQ
405 // la liste d'entiers correspond à un décalage éventuel des tableaux de retour
406 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
407 virtual void Grandeur_particuliere
408     (bool absolue, List_io<TypeQuelconque>& , Loi_comp_abstraite::SaveResul * , list<int>& ) const {};
409 // récupération de la liste de tous les grandeurs particulières
410 // ces grandeurs sont ajoutées à la liste passées en paramètres
411 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
412 virtual void ListeGrandeurs_particulieres(bool absolue, List_io<TypeQuelconque>& ) const {};
413
414 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
415 virtual Loi_comp_abstraite* Nouvelle_loi_identique() const = 0;
416
417 // récup du pointeur si différent de Null, peut-être utilisés par les méthodes
418 // internes
419 const PtIntegMecaInterne* Ptintmeca_en_cours() const {return ptintmeca_en_cours;};
420
421 // affichage de la signature du pti si c'est disponible
422 void Signature_pti_encours(ostream& sort) const
423     { if (ptintmeca_en_cours != NULL)
424         sort << " mail: " << ptintmeca_en_cours->Nb_mail() << " ele: " << ptintmeca_en_cours->Nb_ele()
425             << " npti: " << ptintmeca_en_cours->Nb_pti() ;
426     };
427
428 // affichage de la liste des grandeurs possible à calculer avec Valeur_multi_interpoler_ou_calculer
429 static void Affichage_grandeurs_Valeur_multi_interpoler_ou_calculer();
430
431 // affichage de la liste des grandeurs possible à calculer avec
432 // Valeurs_Tensorielles_interpoler_ou_calculer
433 static void Affichage_grandeurs_Valeurs_Tensorielles_interpoler_ou_calculer();
434
435 // affichage de la liste des grandeurs possible à calculer avec
436 // Valeurs_quelconque_interpoler_ou_calculer
437 static void Affichage_grandeurs_Valeurs_quelconque_interpoler_ou_calculer();
438
439 // retourne le temps cpu utilisé par la loi
440 const Temps_CPU_Hzpp& Temp_CPU_loi() const {return temps_loi;};
441
442 protected :
443 // 3) METHODES VIRTUELLES PURES protegees:
444 // calcul des contraintes à un instant t+deltat
445 // les indices t se rapporte au pas précédent, sans indice au temps actuel
446 virtual void Calcul_SigmaHH
447     (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
448     , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB& giB, BaseH& gi_H, TenseurBB & epsBB
449     , TenseurBB & delta_epsBB, TenseurBB & gijBB, TenseurHH & gijHH, Tableau <TenseurBB *>& d_gijBB
450     , double& jacobien_0, double& jacobien, TenseurHH & sigHH

```

```

450         ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
      module_cisaillement
451         ,const Met_abstraite::Expli_t_tdt& ex) = 0;
452
453 // calcul des contraintes et ses variations par rapport aux ddl a t+dt
454 virtual void Calcul_DsigmaHH_tdt
455     (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
456     ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt
457     ,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt,
458     TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,
459     TenseurHH & gijHH_tdt,Tableau <TenseurBB *>& d_gijBB_tdt,
460     Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien,
461     Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
462     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
      module_cisaillement
463     ,const Met_abstraite::Impli& ex) = 0;
464
465 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
466 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
467 // le tenseur de déformation et son incrémentsont également en orthonormees
468 // si = false: les bases transmises sont utilisées
469 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
470 virtual void Calcul_dsigma_deps
471     (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
472     ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
473     ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
474     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
      module_cisaillement
475     ,const Met_abstraite::Umat_cont& ex) ; //!= 0;
476
477 // lecture éventuelle du type de déformation et du niveau de commentaire
478 // on commence par passer une nouvelle donnée par défaut,
479 // sinon il faut mettre avec_passage_nouvelle_donnee a false
480 void Lecture_type_deformation_et_niveau_commentaire
481 (UtilLecture& lec,LesFonctions_nD& lesFonctionsnD,bool avec_passage_nouvelle_donnee=true);
482 // affichage des données de la classe comp_abstraite
483 void Affiche_don_classe_abstraite() const ;
484 // affichage et definition interactive des commandes particulières à la classe
      loi_comp_abstraite
485 void Info_commande_don LoisDeComp(UtilLecture& entreePrinc) const ;
486
487 //----- lecture écriture de restart spécifique aux données de la classe -----
488 // cas donne le niveau de la récupération
489 // = 1 : on récupère tout
490 // = 2 : on récupère uniquement les données variables (supposées comme telles)
491 void Lecture_don_base_info(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
      lesCourbes1D
492     ,LesFonctions_nD& lesFonctionsnD);
493 // cas donne le niveau de sauvegarde
494 // = 1 : on sauvegarde tout
495 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
496 void Ecriture_don_base_info(ofstream& sort,const int cas) const;
497 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
498 // exemple: mise en service des ddl de température aux noeuds
499 // méthode appelée par Activation_donnees principal, ou des classes dérivées
500 // ce qui permet de surcharger ces dernières
501 void Activ_donnees(Tableau<Noeud *>& tabnoeud,bool dilatation,LesPtIntegMecaInterne& lesPtMecaInt);
502 // calcul de grandeurs de travail aux points d'intégration via la def
503 // fonction surchargée dans les classes dérivée si besoin est
504 virtual void CalculGrandeurTravail
505     (const PtIntegMecaInterne& ptintmeca,const Deformation & ,Enum_dure,const ThermoDonnee&
506     ,const Met_abstraite::Impli* ex_impli
507     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
508     ,const Met_abstraite::Umat_cont* ex_umat
509     ,const List_io<Ddl_etendu>* exclure_dd_etend
510     ,const List_io<const TypeQuelconque *>* exclure_Q
511     ) = 0;
512 // permet d'indiquer à la classe à quelle valeur de saveResult il faut se référer
513 // en particulier est utilisé par les lois additives,
514 // par contre doit être utilisé avec prudence
515 void IndiqueSaveResult(SaveResul * saveR) {saveResul = saveR;};
516 // permet d'indiquer à la classe à quelle valeur de PtIntegMecaInterne il faut se référer
517 // en particulier est utilisé par les lois additives,
518 // par contre doit être utilisé avec prudence
519 virtual void IndiquePtIntegMecaInterne(const PtIntegMecaInterne * ptintmeca) {ptintmeca_en_cours =
      ptintmeca;};
520 // permet d'indiquer à la classe à quelle valeur de def_en_cours il faut se référer
521 // en particulier est utilisé par les lois additives,
522 // par contre doit être utilisé avec prudence
523 void IndiqueDef_en_cours(Deformation * def_en_cours_) {def_en_cours = def_en_cours_;};
524
525 // récupération de valeurs interpolées pour les grandeur enu ou directement calculées
526 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
527 // une seule des 3 métriques doit-être renseigné, les autres doivent être un pointeur nul
528 // exclure_dd_etend: donne une liste de Ddl_enum_etendu à exclure de la recherche
529 // parce que par exemple, ils sont calculés par ailleurs
530 // on peut également ne pas définir de métrique, dans ce cas on ne peut pas calculer certaines

```

```

    grandeurs
531 // -> il y a vérification
532 Tableau <double> Valeur_multi_interpoler_ou_calculer
533     (bool absolue, Enum_dure temps, const List_io<Ddl_enum_etendu>& enu
534     , const Met_abstraite::Impli* ex_impli
535     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
536     , const Met_abstraite::Umat_cont* ex_umat
537     , const List_io<Ddl_enum_etendu>* exclure_dd_etend
538     );
539
540 // récupération de valeurs interpolées pour les grandeur enu
541 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
542 // une seule des 3 métriques doit-être renseigné, les autres doivent être un pointeur nul
543 // exclure_Q: donne une liste de grandeur quelconque à exclure de la recherche
544 //         parce que par exemple, ils sont calculés par ailleurs
545 // on peut également ne pas définir de métrique, dans ce cas on ne peut pas calculer certaines
    grandeurs
546 // -> il y a vérification
547 void Valeurs_Tensorielles_interpoler_ou_calculer
548     (bool absolue, Enum_dure temps, List_io<TypeQuelconque>& enu
549     , const Met_abstraite::Impli* ex_impli
550     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
551     , const Met_abstraite::Umat_cont* ex_umat
552     , const List_io<EnumTypeQuelconque>* exclure_Q
553     );
554
555 // récupération de valeurs interpolées pour les grandeur ici considéré quelconque enu
556 // ces grandeurs ne sont pas définies dans la liste des Ddl_enum_etendu : ex mises à t
557 // ou le numéro de l'élément etc.
558 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
559 // une seule des 3 métriques doit-être renseigné, les autres doivent être un pointeur nul
560 // exclure_Q: donne une liste de grandeur quelconque à exclure de la recherche
561 //         parce que par exemple, ils sont calculés par ailleurs
562 // on peut également ne pas définir de métrique, dans ce cas on ne peut pas calculer certaines
    grandeurs
563 // -> il y a vérification
564 // retour: la list li_quelc
565 void Valeurs_quelconque_interpoler_ou_calculer
566     (bool absolue, Enum_dure temps
567     , const Tableau <EnumTypeQuelconque>& tqi
568     , List_io<TypeQuelconque>& li_quelc
569     , const Met_abstraite::Impli* ex_impli
570     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
571     , const Met_abstraite::Umat_cont* ex_umat
572     , const List_io<EnumTypeQuelconque>* exclure_Q
573     );
574
575 // calcul de la valeur et retour dans tab_ret d'une fonction nD
576 // à l'aide des grandeurs disponibles pour la loi de comportement
577 // nb_retour: nombre de composantes attendues en retour, si > 0
578 // deja_calculer: donne une liste de grandeur Ddl_enum_etendu et quelconque
579 //         à exclure de la recherche car ils doivent avoir été calculés par ailleurs
580 //         c'est une donnée d'entrée qui peut-être utilisée pour l'appel de la fonction nD
581 // list_save : est censé contenir la ou les save_result à consulter pour avoir
582 //         des infos supplémentaires
583 Tableau <double> & Loi_comp_Valeur_FnD_Evoluee
584     (Fonction_nD* fct, int nb_retour
585     , const Met_abstraite::Impli* ex_impli
586     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
587     , const Met_abstraite::Umat_cont* ex_umat
588     , const List_io<Ddl_etendu>* deja_calculer_etend = NULL
589     , const List_io<Const TypeQuelconque *>* deja_calculer_Q = NULL
590     , list <SaveResul*>* list_save = NULL
591     );
592
593 //retourne le niveau d'affichage
594 int Permet_affichage()
595     { return( (permet_affich_loi_nD == NULL) ?
596     (permet_affich_loi == 0) ? ParaGlob::NiveauImpression() : permet_affich_loi
597     : Cal_permet_affichage()); };
598
599 // lecture de l'affichage avec éventuellement une fonction nD
600 void Lecture_permet_affichage(UtilLecture * entreePrinc, LesFonctions_nD& lesFonctionsnD);
601 // sortie du niveau d'affichage
602 void Affiche_niveau_affichage() const;
603 void Affiche_niveau_affichage(ofstream& sort, const int cas);
604 void Lecture_permet_affichage(ifstream& ent, const int cas, LesFonctions_nD& lesFonctionsnD);
605
606 // VARIABLES PROTEGEES :
607 // pointeur de travail utilise par les classes derivantes
608 SaveResul * saveResul;
609 // indic pour définir si oui ou non on utilise un comportement tangent simplifié
610 bool comp_tangent_simplifie;
611 // indication si l'on utilise ou pas la vitesse de déformation
612 bool utilise_vitesse_deformation;
613 // indication du type de déformation utilisée
614 Enum_type_deformation type_de_deformation;

```



```

615
616
617 private: // -- on veut un acces via une méthode
618
619 // liste des grandeurs locales qui peuvent être accédé localement via saveResul
620 // cette liste nécessite d'être abonée par la méthode Activation_stockage_grandeurs_quelconques
621 // qui doit être implantée dans les méthodes qui hérites
622 list <EnumTypeQuelconque > listQuelc_mis_en_acces_localement;
623 // la liste totale qui est construite au moment de la définition de la loi
624 // cette liste est différente de listQuelc_mis_en_acces_localement qui est celle des grandeurs
625 // réellement demandés pour une mise en place : rempli par les classes dérivées
626 list <EnumTypeQuelconque > listdeTouslesQuelc_dispo_localement;
627
628 const PtIntegMecaInterne* ptintmeca_en_cours; // si différent de Null, peut-être utilisés par les
méthodes
629 // internes
630
631 int permet_affich_loi; // pour permettre un affichage spécifique dans les méthodes,
632 // pour les erreurs et des warnings
633 Fonction_nD * permet_affich_loi_nD; // fonction nD éventuelle pour l'affichage
634 List_io <TypeQuelconque > li_quelconque; // stockage inter des grandeurs vraiment quelconques
635 // qui sont ensuite transmise à *permet_affich_loi_nD s'il existe
636 Tableau <const TypeQuelconque * > tab_pt_li_quelconque; // stockage de pointeurs de li_quelconque
637 // pour le passage des infos à *permet_affich_loi_nD s'il existe
638
639 protected :
640
641 // ----- variables gérées en I/O par les classes dérivées -----
642 // et variables de travail
643
644 // indique si oui ou non la loi dépend de la température
645 bool thermo_dependant; // paramètre lue par les classes dérivées
646 double temperature_0,temperature_t,temperature_tdt; // variables valides que si l'on est
thermo_dependant
647 // utilisée par les classes dérivées
648 double* temperature; // pointeur sur la température de travail (à 0, à t ou tdt)
649 bool dilatation; // variable interne, qui est mise en route par l'appel de certaine fonction, comme
celles
650 Deformation * def_en_cours; // si différent de NULL, indique une déformation qui peut être utilisée
par les
651 // classes dérivées
652 Temps_CPU_HZpp temps_loi; // spécifique à ce type de loi: cumule tous les appels
653
654 // du calcul de la contrainte
655 // ----- variables de travail utilisées dans le cadre de la dilatation thermique : pour l'instant
en locales
656 // mais pourraient très bien être dimensionnées dans les éléments si on en avait besoin
657 TenseurBB * epsBB_totale, * epsBB_therm;
658 TenseurBB * DepsBB_totale, * DepsBB_therm;
659 TenseurBB * DepsBB_umat; // pour l'umat
660 // fonction interne utilisée par les classes dérivées
661 // répercussion éventuelle du changement de température dans les classes dérivées
662 // permet par exemple aux lois qui comprennent eux même plusieurs lois de répercuter la modification
de température
663 // l'Enum_dure: indique quel est la température courante : 0 t ou tdt
664 virtual void RepercuteChangeTemperature(Enum_dure );
665
666 // calcul éventuel des invariants liés aux contraintes
667 void CalculInvariants_contraintes(PtIntegMecaInterne& ptIntegMeca, TenseurBB & gijBB,TenseurHH &
gijHH);
668 // calcul éventuel des invariants liés uniquement à la cinématique
669 void CalculInvariants_cinematique(PtIntegMecaInterne& ptIntegMeca, TenseurBB & gijBB,TenseurHH &
gijHH);
670
671 // acces à listdeTouslesQuelc_dispo_localement
672 list <EnumTypeQuelconque > ListdeTouslesQuelc_dispo_localement()
673 {return listdeTouslesQuelc_dispo_localement;};
674 // idem pour listQuelc_mis_en_acces_localement
675 list <EnumTypeQuelconque > ListQuelc_mis_en_acces_localement()
676 {return listQuelc_mis_en_acces_localement;};
677
678 // calcul l'affichage, si celui-ci dépend d'une fonction nD
679 // ne doit être appelé que si la fonction nD existe (pas de vérification)
680 // en fait est utilisé uniquement par la méthode inline: Permet_affichage()
681 int Cal_permet_affichage();
682
683 };
684 //----- les fonctions templates -----
685
686 // #include "ComLoi_comp_abstraite.h"
687
688 #endif

```

## 7.68 Loi\_Umat.h

```

1 // FICHER : Loi_Umat.h
2 // CLASSE : Loi_Umat
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:      11/03/2005
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   BUT:       Loi Umat permet d'utiliser des loi_umat découlant d'un
41 *             d'un autre programme (processus). Le passage d'information*
42 *             s'effectue via deux pipes nommés.
43 *
44 *             $
45 *
46 *   VERIFICATION:
47 *
48 *   ! date !   auteur !           but
49 *   -----
50 *   !       !           !
51 *
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *****/
58
59
60 #ifndef LOI_UMAT_H
61 #define LOI_UMAT_H
62
63
64 #include "Loi_comp_abstraite.h"
65 #include "UmatAbaqus.h"
66 #include "Tenseur3.h"
67 #include "Tenseur2.h"
68 #include "Tenseur1.h"
69 #include "TenseurQ.h"
70 #include "TenseurQ-2.h"
71 #include "TenseurQ-1.h"
72
73
74 class Loi_Umat : public Loi_comp_abstraite
75 {
76
77     public :
78
79
80
81         // CONSTRUCTEURS :
82
83         // Constructeur par défaut
84         // par défaut c'est une loi 3D mais cela pourrait-être des CP

```

```

85     Loi_Umat (Enum_comp enu = LOI_VIA_UMAT);
86
87
88     // Constructeur de copie
89     Loi_Umat (const Loi_Umat& loi) ;
90
91     // DESTRUCTEUR :
92
93     ~Loi_Umat ();
94
95
96     // initialise les donnees particulieres a l'elements
97     // de matiere traite ( c-a-dire au pt calcule)
98     // Il y a creation d'une instance de SaveResul particuliere
99     // a la loi concerne
100    // la SaveResul classe est remplie par les instances heritantes
101    // le pointeur de SaveResul est sauvegarde au niveau de l'element
102    // c'a-d que les info particulieres au point considere sont stocke
103    // au niveau de l'element et non de la loi.
104    class SaveResul_Loi_Umat: public SaveResul
105    { public :
106        SaveResul_Loi_Umat(); // constructeur par défaut (a ne pas utiliser)
107        // le constructeur courant
108        SaveResul_Loi_Umat(list <SaveResul*>& l_des_SaveResul, list <TenseurHH* >& l_siHH
109                            , list <TenseurHH* >& l_siHH_t);
110
111        // constructeur de copie
112        SaveResul_Loi_Umat(const SaveResul_Loi_Umat& sav );
113        // destructeur
114        ~SaveResul_Loi_Umat(){};
115        // définition d'une nouvelle instance identique
116        // appelle du constructeur via new
117        SaveResul * Nevez_SaveResul() const {return (new SaveResul_Loi_Umat(*this));};
118
119    // affectation
120    virtual SaveResul & operator = ( const SaveResul & a);
121    //===== lecture écriture dans base info =====
122    // cas donne le niveau de la récupération
123    // = 1 : on récupère tout
124    // = 2 : on récupère uniquement les données variables (supposées comme telles)
125    void Lecture_base_info (ifstream& ent, const int cas);
126    // cas donne le niveau de sauvegarde
127    // = 1 : on sauvegarde tout
128    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
129    void Ecriture_base_info(ofstream& sort, const int cas);
130
131    // mise à jour des informations transitoires en définitif s'il y a convergence
132    // par exemple (pour la plasticité par exemple)
133    void TdtversT();
134    void TversTdt();
135
136    // affichage à l'écran des infos
137    void Affiche() const
138    { cout << "\n SaveResul_Loi_Umat: ";
139      save_pour_loi_ext->Affiche();
140    };
141
142    // changement de base de toutes les grandeurs internes tensorielles stockées
143    // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gb
144    // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
145    // gpH(i) = gamma(i,j) * gH(j)
146    virtual void ChBase_des_grandeurs(const Mat_pleine& beta, const Mat_pleine& gamma)
147    {return save_pour_loi_ext->ChBase_des_grandeurs(beta, gamma);};
148
149    // procedure permettant de completer éventuellement les données particulières
150    // de la loi stockées
151    // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
152    // completer est appelé apres sa creation avec les donnees du bloc transmis
153    // peut etre appeler plusieurs fois
154    SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
155                                  , const Loi_comp_abstraite* loi) {};
156
157    // données protégées
158    SaveResul* save_pour_loi_ext;
159    double hsurh0, h_tsurh0;
160 };
161
162 // def d'une instance de données spécifiques, et initialisation
163 SaveResul * New_et_Initialise();
164
165 // Lecture des donnees de la classe sur fichier
166 void LectureDonneesParticulieres (UtilLecture * , LesCourbes1D & lesCourbes1D
167                                   , LesFonctions_nD & lesFonctionsnD);
168
169 // affichage de la loi
170 void Affiche() const ;
171 // test si la loi est complete
172 // = 1 tout est ok, =0 loi incomplete
173 int TestComplet();
174
175

```

```

172 // calcul d'un module d'young équivalent à la loi, ceci pour un
173 // chargement nul
174 double Module_young_equivalent(Enum_dure temps,const Deformation & ,SaveResul * saveResul);
175
176 // récupération de la variation relative d'épaisseur calculée: h/h0
177 // cette variation n'est utile que pour des lois en contraintes planes
178 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
179 // - pour les lois 2D def planes: retour de 0
180 // les infos nécessaires à la récupération , sont stockées dans saveResul
181 // qui est le conteneur spécifique au point où a été calculé la loi
182 virtual double HsurH0(SaveResul * saveResul) const
183 {Loi_Umat::SaveResul_Loi_Umat& sav = *(Loi_Umat::SaveResul_Loi_Umat*) &saveResul);
184   return sav.hsurh0;};
185
186 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
187 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_Umat(*this)); };
188
189 // ----- gestion des incréments et itérations -----
190 // mise à jour des infos nécessaires pour l'umat
191
192 void Mise_a_jour_num_increment(int num) {umatAbaqus.nb_increment=num;};
193 void Mise_a_jour_num_iteration(int num) {umatAbaqus.nb_step=num;};
194
195 // idem pour le numéro d'élément et le numéro de point d'intégration
196 void Mise_a_jour_nbe_nbptinteg(int nbe, int nbptinteg)
197   {umatAbaqus.nb_elem = nbe;umatAbaqus.nb_pt_integ = nbptinteg;};
198
199 //----- lecture écriture de restart -----
200 // cas donne le niveau de la récupération
201 // = 1 : on récupère tout
202 // = 2 : on récupère uniquement les données variables (supposées comme telles)
203 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
204   ,LesFonctions_nD& lesFonctionsnD);
205
206 // cas donne le niveau de sauvegarde
207 // = 1 : on sauvegarde tout
208 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
209 void Ecriture_base_info_loi(ofstream& sort,const int cas);
210
211 // affichage et definition interactive des commandes particulières à chaque lois
212 void Info_commande_LoisDeComp(UtilLecture& lec);
213
214 // retour du nom de la loi associée
215 const string& NomDeLaLoi() const {return nom_de_la_loi;};
216
217 // définition de la loi ext dans le cas d'une umat directe sans pipe
218 void DefLoiExt(LoiAbstraiteGeneral* lext) {loi_ext=lext; };
219 // indique si oui ou non la loi utilise une umat interne
220 bool Utilise_une_umat_interne() const {return utilisation_umat_interne;};
221
222 protected :
223
224 // donnees protegees
225 string nom_de_la_loi; // nom de la loi
226 bool utilisation_umat_interne; // indique si oui ou non on utilise une umat interne ou alors via
pipe
227 // ----- controle de la sortie des informations
228 // -> maintenant définit dans LoiAbstraiteGeneral
229 // int permet affichage; // pour permettre un affichage spécifique dans les méthodes, pour les
erreurs et warning
230 UmatAbaqus umatAbaqus; // le conteneur d'information
231 // par défaut on définit 3 métriques ce qui nous permet de naviguer entre les 3 dimensions
232 // certaines ne servent pas, mais comme il s'agit de la définition de la loi, on imagine
233 // qu'il y aura peut d'exemplaires d'où un impact de stockage négligeable
234 Met_abstraite umat_met3D; // métrique 3D particulière à l'Umat
235 Met_abstraite umat_met2D; // métrique 2D particulière à l'Umat
236 Met_abstraite umat_met1D; // métrique 1D particulière à l'Umat
237 Tenseur2HHHH d_sigma_deps_2D; // un tenseur de travail pour la méthode Calcul_DsigmaHH_tdt en
CP
238 Tenseur1HHHH d_sigma_deps_1D; // un tenseur de travail pour la méthode Calcul_DsigmaHH_tdt en
CP2
239
240 // élément par défaut pour umat_met
241 // def des bases naturelles et duales par rapport aux coordonnées initiales
242 // l'indice x est pour rappeler que les coordonnées matérielles sont les x_0
243 BaseB gixB_0,gixB_t,gixB_tdt; BaseH gixH_0,gixH_t,gixH_tdt;
244 // idem pour les métriques, sachant que la métrique initiale est l'identité
245 TenseurBB* gabBB_tdt; TenseurHH* gabHH_tdt; // tenseurs dans Ia
246 TenseurBB* gabBB_t; TenseurHH* gabHH_t; // tenseurs dans Ia
247 // les grandeurs qui pour l'instant ne servent pas, on y met un pointeur nul
248 BaseB * ggaB_0null, * ggaB_tnull;BaseH * ggaH_0null;
249 TenseurBB * ggradVmoyBB_tnull, * ggradVmoyBB_tdtnull;
250 TenseurBB * ggradVBB_tdtnull;
251
252 // --//\-- grandeurs particulières pour le cas contraintes planes
253 Met_abstraite::Umat_cont* umat_cont_3D; // version 3D
254 Met_abstraite::Umat_cont* umat_cont_2D; // version 2D

```

```

254     BaseB Ip2_B; // base orthonormée locale
255
256     // -- variables nécessaires pour la création de umat_cont_3D
257     // certaines grandeurs sont associées à un pointeur qui peut soit être nulle soit pointer sur le
conteneur
258     // l'intérêt est que le fait d'avoir un pointeur nul est parfois utilisé pour éviter un calcul
259     BaseB giB_0_3D;
260     BaseH giH_0_3D;
261     BaseB giB_t_3D;
262     BaseH giH_t_3D;
263     BaseB giB_tdt_3D;
264     BaseH giH_tdt_3D;
265     Tenseur3BB gijBB_0_3D;
266     Tenseur3HH gijHH_0_3D;
267     Tenseur3BB gijBB_t_3D;
268     Tenseur3HH gijHH_t_3D;
269     Tenseur3BB gijBB_tdt_3D;
270     Tenseur3HH gijHH_tdt_3D;
271
272     TenseurBB * gradVmoyBB_t_3D_P; Tenseur_ns3BB gradVmoyBB_t_3D;
273     TenseurBB * gradVmoyBB_tdt_3D_P; Tenseur_ns3BB gradVmoyBB_tdt_3D;
274     TenseurBB * gradVBB_tdt_3D_P; Tenseur_ns3BB gradVBB_tdt_3D;
275     double jacobien_tdt_3D; double jacobien_t_3D; double jacobien_0_3D; // pour les jacobiens on
considère qu'ils existent toujours
276     // -- variables nécessaires pour la création de umat_cont_2D
277     BaseB Ip3B_0_3D; // coordonnées de la base locale ortho dans le repère globale
278     BaseH Ip3H_0_3D; // 3 vecteurs de dimensions 3, les deux premiers sont dans le plan des CP
279
280     // puis les grandeurs pointées dans umat_cont_2D
281     // certaines grandeurs sont associées à un pointeur qui peut soit être nulle soit pointer sur le
conteneur
282     // l'intérêt est que le fait d'avoir un pointeur nul est parfois utilisé pour éviter un calcul
283
284     BaseB giB_0_2D; // coordonnée de giB_0 localement dans la base Ip2B_0 = IP2H_0 car orthonormée
285     BaseH giH_0_2D; // coordonnée de giH_0 localement dans la base Ip2B_0 = IP2H_0 car orthonormée
286     BaseB giB_t_2D;
287     BaseH giH_t_2D;
288     BaseB giB_tdt_2D;
289     BaseH giH_tdt_2D;
290     Tenseur2BB gijBB_0_2D;
291     Tenseur2HH gijHH_0_2D;
292     Tenseur2BB gijBB_t_2D;
293     Tenseur2HH gijHH_t_2D;
294     Tenseur2BB gijBB_tdt_2D;
295     Tenseur2HH gijHH_tdt_2D;
296
297     TenseurBB * gradVmoyBB_t_2D_P; Tenseur_ns3BB gradVmoyBB_t_2D;
298     TenseurBB * gradVmoyBB_tdt_2D_P; Tenseur_ns3BB gradVmoyBB_tdt_2D;
299     TenseurBB * gradVBB_tdt_2D_P; Tenseur_ns3BB gradVBB_tdt_2D;
300     double jacobien_tdt_2D; double jacobien_t_2D; double jacobien_0_2D; // pour les jacobiens on
considère qu'ils existent toujours
301     // --//\-- fin grandeurs particulières pour le cas contraintes planes
302
303     // un compteur pour alouer un numéro d'ordre qui tiens lieu de numéro d'élément
304     static int compteur_simili_num_elemEtPtInteg,max_simili_num_ptinteg;
305     // un pointeur de loi qui est utilisé lorsque on n'utilise pas de pipe
306     // en développement ou dans le cas non unix
307     LoiAbstraiteGeneral* loi_ext;
308
309     // codage des METHODES VIRTUELLES proteges:
310     // calcul des contraintes a t+dt
311     // calcul des contraintes
312     void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
313     ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
314     ,TenseurBB & delta_epsBB_
315     ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
316     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
317     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
318     ,const Met_abstraite::Expli_t_tdt& ex);
319
320     // calcul des contraintes et de ses variations a t+dt
321     void Calcul_dsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
322     ,BaseB& giB_t,TenseurBB & gijBB_t & gijHH_t,TenseurHH & gijHH_t
323     ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
324     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
325     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
326     ,Tableau <TenseurBB *>& d_gijBB_tdt
327     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
328     ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
329     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
330     ,const Met_abstraite::Impli& ex);
331
332     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
333     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
334     // le tenseur de déformation et son incrémentsont également en orthonormees

```

```

335         //                               si = false: les bases transmises sont utilisées
336         // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
337 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB & DepsBB
338     , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double & jacobien_0, double & jacobien
339     , TenseurHH & sigHH, TenseurHHH & d_sigma_deps
340     , EnergieMeca & energ, const EnergieMeca & energ_t, double & module_compressibilite, double &
341     module_cisaillement
342     , const Met_abstraite::Umat_cont & ex) ; // = 0;
343
344 // fonction surchargée dans les classes dérivée si besoin est
345 virtual void CalculGrandeurTravail
346     (const PtIntegMecaInterne & , const Deformation &
347     , Enum_dure, const ThermoDonnee &
348     , const Met_abstraite::Impli* ex_impli
349     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
350     , const Met_abstraite::Umat_cont* ex_umat
351     , const List_io<Ddl_etendu>* exclure_dd_etendu
352     , const List_io<const TypeQuelconque *>* exclure_Q
353     ) {};
354
355 // fonction interne utilisée par les classes dérivées de Loi_comp_abstraite
356 // pour répercuter les modifications de la température
357 // ici utiliser pour modifier la température des lois élémentaires
358 // l'Enum_dure: indique quel est la température courante : 0 t ou tdt
359 void RepercuteChangeTemperature(Enum_dure temps);
360
361 //--- méthodes internes
362 // passage des grandeurs métriques de l'ordre 2 à 3: cas implicite
363 void Passage_métrique_ordre2_vers_3(const Met_abstraite::Umat_cont & ex);
364
365 // fonction de travail
366 static int Choix_dim(Enum_comp enu);
367
368 // calcul des modules de compressibilité et de cisaillement
369 // en fonction des résultats de l'umat
370 void Calcul_compressibilite_cisaillement(const Met_abstraite::Umat_cont & ex
371     , double & module_compressibilite, double &
372     module_cisaillement);
373
374 };
375 #endif
376
377
378

```

## 7.69 LoiAbstraiteGeneral.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      23/01/97
33 *
34 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++

```

```

37 *                                                                 $ *
38 *****
39 *   BUT:   Classe de base de tous les lois de comportement, permet *
40 *         la definition de fonctions generiques dans la class   *
41 *         Element.                                             *
42 *                                                                 $ *
43 *   ***** *
44 *   *
45 *   VERIFICATION: *
46 *   ! date ! auteur ! but ! *
47 *   ----- *
48 *   ! ! ! ! *
49 *   * *
50 *   ***** *
51 *   MODIFICATIONS: *
52 *   ! date ! auteur ! but ! *
53 *   ----- *
54 *   * *
55 *****/
56
57
58 #ifndef LOIABSTRAITEGENERAL_H
59 #define LOIABSTRAITEGENERAL_H
60
61
62 #include "Enum_comp.h"
63 #include "Tableau_T.h"
64 #include "UtilLecture.h"
65 #include "LesCourbes1D.h"
66 #include "Enum_categorie_loi_comp.h"
67 #include "LesReferences.h"
68 #include "Enum_ddl.h"
69 #include "LesFonctions_nD.h"
70 #include "Enum_GrandeurGlobale.h"
71
72
73 class LoiAbstraiteGeneral
74 {
75
76     public :
77
78         // CONSTRUCTEURS :
79         // Constructeur par defaut
80         LoiAbstraiteGeneral() ;
81
82         // Constructeur utile si l'identificateur du nom de la loi
83         // de comportement et la dimension sont connus ainsi que la catégorie de la loi
84         // ddl : identificateur donnant le type de degré de liberté
85         // de base, ou est calculé la loi, par défaut pt integ mécanique : SIG11
86
87         LoiAbstraiteGeneral(Enum_comp id_comp, int dimension, Enum_categorie_loi_comp id_categorie
88             , Enum_ddl ddl = SIG11);
89
90         // Constructeur utile si l'identificateur du nom de la loi
91         // de comportement et la dimension sont connus ainsi que la catégorie de la loi
92         LoiAbstraiteGeneral(string nom, int dimension, Enum_categorie_loi_comp id_categorie
93             , Enum_ddl ddl = SIG11);
94
95         // Constructeur de copie
96         LoiAbstraiteGeneral(const LoiAbstraiteGeneral& a);
97
98         // DESTRUCTEUR :
99
100         virtual ~LoiAbstraiteGeneral();
101
102
103
104
105 // 1) METHODES NON VIRTUELLES :
106
107         void Change_comport(string nouveau_nom) { id_comp=Id_nom_comp(nouveau_nom); };
108         // Remplace le nom de la loi de comportement par nouveau_nom
109         // (par l'intermediaire du type enumere associe)
110         // ATTENTION : Les modifications liees au changement de la loi de
111         // comportement sont a la charge de l'utilisateur.
112
113         void Change_comport(Enum_comp nouveau_id) { id_comp=nouveau_id;};
114         // Remplace l'identificateur de la loi de comportement
115         // ATTENTION : Les modifications liees au changement de la loi de
116         // comportement sont a la charge de l'utilisateur.
117
118         // Retourne l'identificateur de la loi de comportement (de type enumere)
119         Enum_comp Id_comport()const { return id_comp; };
120
121         // Retourne le nom de la loi de comportement
122         string Nom_comport() const { return Nom_comp(id_comp); };

```

```

123
124 // retourne la catégorie de la loi de comportement
125 Enum_categorie_loi_comp Id_categorie() {return categorie_loi_comp;};
126 // ramène l'identificateur donnant le type de degré de liberté
127 // de base, ou est calculé la loi, par exemple pour une loi mécanique
128 // pt d'integ SIG11
129 Enum_ddl EnumDdlTypePt() const {return enu_ddl_type_pt;};
130 // changement de l'identificateur de type de ddl de base ou est calculé la loi
131 void Change_EnumDdlTypePt(Enum_ddl enu) {enu_ddl_type_pt=enu;};
132
133
134 // 2) METHODES VIRTUELLES pures public:
135
136 // Lecture des donnees de la classe sur fichier
137 virtual void LectureDonneesParticulieres
138     (UtilLecture * ,LesCourbes1D& lesCourbes1D
139      ,LesFonctions_nD& lesFonctionsnD) = 0;
140
141 // affichage de la loi
142 virtual void Affiche() const = 0;
143
144 //----- lecture écriture de restart -----
145 // cas donne le niveau de la récupération
146 // = 1 : on récupère tout
147 // = 2 : on récupère uniquement les données variables (supposées comme telles)
148 virtual void Lecture_base_info_loi
149     (ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
150      ,LesFonctions_nD& lesFonctionsnD) = 0;
151 // cas donne le niveau de sauvegarde
152 // = 1 : on sauvegarde tout
153 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
154 virtual void Ecriture_base_info_loi(ofstream& sort,const int cas) =0;
155
156 // affichage et definition interactive des commandes particulières à chaque lois
157 virtual void Info_commande_LoisDeComp(UtilLecture& lec) = 0;
158
159 // 2) METHODES VIRTUELLES public:
160
161 // test si la loi est complete
162 // = 1 tout est ok, =0 loi incomplete
163 virtual int TestCompleet();
164
165 // retourne la dimension de la loi: en général 1 ou 2 ou 3
166 // si ramène 4 => valable quelque soit la dimension
167 virtual int Dimension_loi() const { return dim;};
168
169 // modification de l'indicateur de comportement tangent
170 virtual void Modif_comp_tangent_simplifie(bool )
171     { // doit être def dans les classes dérivées
172       cout << "erreur, non défini pour cette loi ";
173       Affiche();
174       Sortie(1);
175     };
176
177 // test pour connaître l'état du comportement : simplifié ou non
178 virtual bool Test_loi_simplife()
179     { // doit être def dans les classes dérivées
180       cout << "erreur, non défini pour cette loi ";
181       Affiche(); Sortie(1); return false; // pour éviter le warning
182     };
183
184 // indique le type Enum_comp_3D_CP_DP_1D correspondant à une loi de comportement
185 // la fonction est simple dans le cas d'une loi basique, par contre dans le cas
186 // d'une loi combinée, la réponse dépend des lois internes donc c'est redéfini
187 // dans les classes dérivées
188 virtual Enum_comp_3D_CP_DP_1D Comportement_3D_CP_DP_1D() {return Comp_3D_CP_DP_1D(id_comp);};
189
190 protected :
191
192 // ---- donnees protegees
193 Enum_comp id_comp; // identificateur de la loi de comportement
194 int dim ; // dimension de la loi
195 Enum_categorie_loi_comp categorie_loi_comp;
196 // identificateur donnant le type de degré de liberté
197 // de base, ou est calculé la loi, par exemple pour une loi mécanique
198 // pt d'integ SIG11
199 Enum_ddl enu_ddl_type_pt;
200
201 // ---- fonctions protégées appelées par les classes dérivées
202 // cas donne le niveau de la récupération
203 // = 1 : on récupère tout
204 // = 2 : on récupère uniquement les données variables (supposées comme telles)
205 void Lect_base_info_loi
206     (ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D& lesCourbes1D
207      ,LesFonctions_nD& lesFonctionsnD);
208 // cas donne le niveau de sauvegarde
209 // = 1 : on sauvegarde tout

```



```

210 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
211 void Ecrit_base_info_loi(ofstream& sort,const int cas) const;
212 // change la catégorie de la loi de comportement
213 void ChangeCategorie(Enum_categorie_loi_comp new_categorie)
214 {categorie_loi_comp=new_categorie;};
215 // change la dimension de la loi de comportement
216 void Change_dimension(int dime)
217 {dim=dime;};
218 // fonction utilitaire pour lire une grandeur qui dépend d'une fonction
219 // si la courbe existe déjà dans LesCourbes1D, ramène un pointeur dessus
220 // si la courbe n'exite pas déjà, création indépendamment de LesCourbes (donc via un new)
221 // et ramène un pointeur dessus
222 Courbe1D * Lecture_courbe(UtilLecture * entreePrinc,LesCourbes1D& lesCourbes1D);
223
224 };
225
226
227 #endif

```

## 7.70 LoiAdditiveEnSigma.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:      11/06/2003
33 *
34 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:     Herezh++
37 *
38 *****/
39 *      BUT:  Définir une loi telle que la contrainte résultante soit la
40 *           somme de contraintes élémentaires, eux-même définies à
41 *           partir de lois quelconques.
42 *
43 *           *****
44 *
45 *      VERIFICATION:
46 *
47 *      ! date !   auteur !           but
48 *      !-----!-----!-----!-----!
49 *      !           !           !           !
50 *      !           !           !           !
51 *
52 *      MODIFICATIONS:
53 *
54 *      ! date !   auteur !           but
55 *      !-----!-----!-----!-----!
56 *      !           !           !           !
57 *
58 *****/
59
60 // FICHER : LoiAdditiveEnSigma.h
61 // CLASSE : LoiAdditiveEnSigma
62
63 #ifndef LOIADDITIVEENSIGMA_H
64 #define LOIADDITIVEENSIGMA_H
65
66 #include "Loi_comp_abstraite.h"

```

```

64 #include "Ponderation.h"
65
66
67 /** @defgroup Les_lois_combinees
68 *
69 * BUT: groupe des lois combinées
70 *
71 *
72 * \author Gérard Rio
73 * \version 1.0
74 * \date 11/06/2003
75 * \brief Définition des lois combinées
76 *
77 */
78
79 /// @addtogroup Les_lois_combinees
80 /// @{
81 ///
82
83 class LoiAdditiveEnSigma : public Loi_comp_abstraite
84 {
85     public :
86
87         // CONSTRUCTEURS :
88
89         // Constructeur par défaut
90         LoiAdditiveEnSigma ();
91
92         // Constructeur de copie
93         LoiAdditiveEnSigma (const LoiAdditiveEnSigma& loi) ;
94
95         // DESTRUCTEUR :
96         ~LoiAdditiveEnSigma ();
97
98
99         // initialise les donnees particulieres a l'elements
100        // de matiere traite ( c-a-dire au pt calcule)
101        // Il y a creation d'une instance de SaveResul particuliere
102        // a la loi concernee
103        // la SaveResul classe est remplie par les instances heritantes
104        // le pointeur de SaveResul est sauvegarde au niveau de l'element
105        // c'a-d que les info particulieres au point considere sont stocke
106        // au niveau de l'element et non de la loi.
107        class SaveResul_LoiAdditiveEnSigma: public SaveResul
108        { public :
109            SaveResul_LoiAdditiveEnSigma(); // constructeur par défaut (a ne pas utiliser)
110            // le constructeur courant
111            SaveResul_LoiAdditiveEnSigma(list <SaveResul*> & l_des_SaveResul, list <TenseurHH* >& l_siHH
112                , list <TenseurHH* >& l_siHH_t
113                , list <EnergieMeca >& l_energ, list <EnergieMeca >& l_energ_t
114                , bool avec_ponderation);
115
116            // constructeur de copie
117            SaveResul_LoiAdditiveEnSigma(const SaveResul_LoiAdditiveEnSigma& sav );
118            // destructeur
119            ~SaveResul_LoiAdditiveEnSigma ();
120            // définition d'une nouvelle instance identique
121            // appelle du constructeur via new
122            SaveResul * Nevez_SaveResul() const {return (new SaveResul_LoiAdditiveEnSigma(*this));};
123
124
125        };
126
127        // affectation
128        virtual SaveResul & operator = ( const SaveResul & a);
129        //===== lecture écriture dans base info =====
130        // cas donne le niveau de la récupération
131        // = 1 : on récupère tout
132        // = 2 : on récupère uniquement les données variables (supposées comme telles)
133        void Lecture_base_info (ifstream& ent, const int cas);
134        // cas donne le niveau de sauvegarde
135        // = 1 : on sauvegarde tout
136        // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
137        void Ecriture_base_info(ofstream& sort, const int cas);
138
139
140        // mise à jour des informations transitoires en définitif s'il y a convergence
141        // par exemple (pour la plasticité par exemple)
142        void TdtversT() ;
143        void TversTdt() ;
144
145        // affichage à l'écran des infos
146        void Affiche() const;
147
148
149        // changement de base de toutes les grandeurs internes tensorielles stockées
150        // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gb
151        // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
152        // gpH(i) = gamma(i,j) * gH(j)
153        virtual void ChBase_des_grandeurs(const Mat_pleine& beta, const Mat_pleine& gamma);
154
155        // procedure permettant de completer éventuellement les données particulières
156        // de la loi stockées

```

```

150 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
151 // completer est appelé apres sa creation avec les donnees du bloc transmis
152 // peut etre appeler plusieurs fois
153 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
154 ,const Loi_comp_abstraite* loi);
155
156 // ---- récupération d'information: spécifique à certaine classe dérivée
157 double Deformation_plastique() ;
158
159 // données protégées
160 // la liste des données protégées de chaque loi
161 list <SaveResul*> liste_des_SaveResul;
162 // la liste des contraintes initiales particulières pour chaque loi
163 list <TenseurHH* > l_sigoHH,l_sigoHH_t; // valeur courante, et valeur sauvegardée au pas
précédent
164 // la liste des énergies pour chaque loi
165 list <EnergieMeca > l_energ,l_energ_t; // valeur courante, et valeur sauvegardée au pas
précédent
166 // listes éventuelles des fonctions de pondération
167 list <double> f_ponder,f_ponder_t; // le résultat des fonctions de pondérations
168 };
169
170 // def d'une instance de données spécifiques, et initialisation
171 SaveResul * New_et_Initialise() ;
172
173 friend class SaveResul_LoiAdditiveEnSigma;
174
175 // Lecture des lois de comportement
176 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
177 ,LesFonctions_nD& lesFonctionsnD);
178 // affichage de la loi
179 void Affiche() const ;
180 // test si la loi est complete
181 // = 1 tout est ok, =0 loi incomplete
182 int TestComplet();
183
184 // calcul d'un module d'young équivalent à la loi, ceci pour un
185 // chargement nul
186 double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
187 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
188 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
189 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
saveResul);
190
191 // récupération de la variation relative d'épaisseur calculée: h/h0
192 // cette variation n'est utile que pour des lois en contraintes planes
193 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
194 // - pour les lois 2D def planes: retour de 0
195 // les infos nécessaires à la récupération , sont stockées dans saveResul
196 // qui est le conteneur spécifique au point où a été calculé la loi
197 virtual double HsurH0(SaveResul * saveResul) const
198 { cout << "\n LoiAdditiveEnSigma::HsurH0(.. , methode non implante pour l'instant ";
199 Sortie(1);
200 };
201
202 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
203 // exemple: mise en service des ddl de température aux noeuds
204 virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud,bool
dilatation,LesPtIntegMecaInterne& lesPtMecaInt);
205 // récupération des grandeurs particulière (hors ddl )
206 // correspondant à liTQ
207 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
208 virtual void Grandeur_particuliere
209 (bool absolue,List_io<TypeQuelconque>& liTQ,Loi_comp_abstraite::SaveResul * saveDon,list<int>&)
const;
210 // récupération de la liste de tous les grandeurs particulières
211 // ces grandeurs sont ajoutées à la liste passées en paramètres
212 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
213 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& liTQ) const;
214
215 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
216 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new LoiAdditiveEnSigma(*this)); };
217
218 // indique le type Enum_comp_3D_CP_DP_1D correspondant à une loi de comportement
219 // la fonction est simple dans le cas d'une loi basique, par contre dans le cas
220 // d'une loi combinée, la réponse dépend des lois internes donc c'est redéfini
221 // dans les classes dérivées
222 virtual Enum_comp_3D_CP_DP_1D Comportement_3D_CP_DP_1D();
223
224 //----- lecture écriture de restart -----
225 // cas donne le niveau de la récupération
226 // = 1 : on récupère tout
227 // = 2 : on récupère uniquement les données variables (supposées comme telles)
228 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
229 ,LesFonctions_nD& lesFonctionsnD);
230

```

```

231 // cas donne le niveau de sauvegarde
232 // = 1 : on sauvegarde tout
233 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
234 void Ecriture_base_info_loi(ofstream& sort,const int cas);
235
236 // affichage et definition interactive des commandes particulières à chaque lois
237 void Info_commande_LoisDeComp(UtilLecture& lec);
238
239 protected :
240 // un type énuméré pour faciliter la lecture
241 enum Enumcompletudecalcul { CONTRAINTE_ET_TANGENT =0, CONTRAINTE_UNIQUEMENT, TANGENT_UNIQUEMENT};
242 // donnees protegees
243 list <Loi_comp_abstraite *> lois_interne; // liste des lois constitutives
244 list <Enumcompletudecalcul> list_completude_calcul; // pour savoir si on utilise tout ou une
partie
245
246 int type_calcul; // indique si l'on travail sur la contrainte ou l'incrément de contrainte
247 int tangent_ddl_via_eps; // par défaut false
248 // indique si true que l'on veut un calcul de l'opérateur tangent /ddl via
249 // tout d'abord l'opérateur tangent / eps utile a priori pour les tests pour
250 // voir si on a une convergence du même type
251
252 //-- partie optionnelle au cas d'une somme pondérée
253 bool avec_ponderation; // indique si oui ou non il y a des fonctions de ponderation
254 // pour chaque loi il y a un élément Ponderation associé, qui contient lui-même m fonctions 1D
dont le produit
255 // = la fonction finale de ponderation de la loi
256 list <Ponderation > list_ponderation; // liste éventuellement vide des fonctions de ponderation
257 // list <double> fonc_ponder; // le résultat des fonctions de pondérations
258
259 // idem avec une fonction nD via un objet: Ponderation_TypeQuelconque
260 // ici un pointeur nulle indique qu'il n'y a pas de fct, cependant la taille de la liste
261 // = celle de list_ponderation c-a-d du nombre de loi
262 // les grandeurs quelconque sont ceux de chaque loi, elles doivent donc être renseigné par
263 // les lois individuelles
264 list <Ponderation_TypeQuelconque* > list_ponderation_nD_quelconque;
265 // list <double> fonc_ponder_nD_quelconque; // le résultat des fonctions nD quelconques
266
267 // ---- tableau de travail
268 Tableau <TenseurHH *> d_sigtotalHH;
269 // tenseur du 4ième ordre de travail
270 TenseurHHHH* d_sigma_deps_inter;
271 TenseurHHHH* d_sigma_deps_pourCalcul_DsigmaHH_via_eps_tdt;
272
273 // codage des METHODES VIRTUELLES protegees:
274 // calcul des contraintes a t+dt
275 // calcul des contraintes
276 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
277 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB & giB,BaseH & gi_H, TenseurBB & epsBB_
278 ,TenseurBB & delta_epsBB_
279 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
280 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
281 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
282 ,const Met_abstraite::Expli_t_tdt& ex);
283
284 // calcul des contraintes et de ses variations a t+dt
285 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
286 ,BaseB & giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
287 ,BaseB & giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH & giH_tdt,Tableau <BaseH> & d_giH_tdt
288 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
289 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
290 ,Tableau <TenseurBB *>& d_gijBB_tdt
291 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
292 ,Vecteur & d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
293 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
294 ,const Met_abstraite::Impli& ex);
295
296 // calcul des contraintes et de ses variations a t+dt
297 // calcul particulier de l'opérateur tangent via dsig/deps
298 void Calcul_DsigmaHH_via_eps_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
299 ,BaseB & giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
300 ,BaseB & giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH & giH_tdt,Tableau <BaseH> & d_giH_tdt
301 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
302 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
303 ,Tableau <TenseurBB *>& d_gijBB_tdt
304 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
305 ,Vecteur & d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
306 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
307 ,const Met_abstraite::Impli& ex);
308
309 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
310 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
311 // le tenseur de déformation et son incrément sont également en orthonormees
312 // si = false: les bases transmises sont utilisées

```

```

313         // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
314 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB & DepsBB
315     , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double & jacobien_0, double & jacobien
316     , TenseurHH & sigHH, TenseurHHHH & d_sigma_deps
317     , EnergieMeca & energ, const EnergieMeca & energ_t, double & module_compressibilite, double &
    module_cisaillement
318     , const Met_abstraite::Umat_cont & ex) ; // = 0;
319
320
321 // fonction surchargée dans les classes dérivée si besoin est
322 virtual void CalculGrandeurTravail
323     (const PtIntegMecaInterne & ptintmeca
324     , const Deformation & def, Enum_dure temps, const ThermoDonnee & dTP
325     , const Met_abstraite::Impli* ex_impli
326     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
327     , const Met_abstraite::Umat_cont* ex_umat
328     , const List_io<Ddl_etendu>* exclure_dd_etend
329     , const List_io<const TypeQuelconque *>* exclure_Q
330     );
331 // permet d'indiquer à la classe à laquelle valeur de PtIntegMecaInterne il faut se référer
332 // en particulier est utilisé par les lois additives,
333 // par contre doit être utilisé avec prudence
334 virtual void IndiquePtIntegMecaInterne (const PtIntegMecaInterne * ptintmeca)
335     { list <Loi_comp_abstraite *>::iterator il, ilfin = lois_internes.end();
336     for ( il = lois_internes.begin(); il != ilfin; il++)
337         (*il)->IndiquePtIntegMecaInterne (ptintmeca);
338     // puis la classe mère
339     Loi_comp_abstraite::IndiquePtIntegMecaInterne (ptintmeca);
340     };
341
342 // fonction interne utilisée par les classes dérivées de Loi_comp_abstraite
343 // pour répercuter les modifications de la température
344 // ici utiliser pour modifier la température des lois élémentaires
345 // l'Enum_dure: indique quel est la température courante : 0 t ou tdt
346 void RepercuteChangeTemperature (Enum_dure temps);
347
348 // vérification et préparation de l'accès aux grandeurs locales
349 void Verif_et_preparation_acces_grandeurs_locale();
350
351 };
352 /// @} // end of group
353
354 #endif
355
356
357

```

## 7.71 LoiContraintesPlanes.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           8/02/2012
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++

```

```

37 *                                                                 $ *
38 *****
39 *      BUT: La loi est 2D_C et est associée à une loi 3D quelconque *
40 *      L'objectif est de transformer une loi 3D en 2D contraintes *
41 *      planes. *
42 *                                                                 $ *
43 *      ***** *
44 *      VERIFICATION: *
45 * *
46 *      ! date ! auteur ! but ! *
47 *      ----- *
48 *      ! ! ! ! *
49 * *
50 *      ***** *
51 *      MODIFICATIONS: *
52 *      ! date ! auteur ! but ! *
53 *      ----- *
54 * *
55 *      *****/
56
57 // FICHER : LoiContraintesPlanes.h
58 // CLASSE : LoiContraintesPlanes
59
60 #ifndef LOICONTRAINTEPLANES_H
61 #define LOICONTRAINTEPLANES_H
62
63 #include "Loi_comp_abstraite.h"
64 #include "Enum_contrainte_mathematique.h"
65 #include "Algo_edp.h"
66 #include "TenseurQ.h"
67 #include "Ponderation.h"
68
69
70 /// @addtogroup Les_lois_combinees
71 /// @{
72 ///
73
74
75 class LoiContraintesPlanes : public Loi_comp_abstraite
76 {
77     public :
78     friend class LoiContraintesPlanesDouble;
79     friend class LoiCritere;
80
81     // CONSTRUCTEURS :
82
83     // Constructeur par défaut
84     LoiContraintesPlanes ();
85
86     // Constructeur de copie
87     LoiContraintesPlanes (const LoiContraintesPlanes& loi) ;
88
89     // DESTRUCTEUR :
90     ~LoiContraintesPlanes ();
91
92
93
94     // initialise les donnees particulieres a l'elements
95     // de matiere traite ( c-a-dire au pt calcule)
96     // Il y a creation d'une instance de SaveResul particuliere
97     // a la loi concernee
98     // la SaveResul classe est remplie par les instances heritantes
99     // le pointeur de SaveResul est sauvegarde au niveau de l'element
100    // c'a-d que les info particulieres au point considere sont stocke
101    // au niveau de l'element et non de la loi.
102    class SaveResul_LoiContraintesPlanes: public SaveResul
103    { public :
104        SaveResul_LoiContraintesPlanes(); // constructeur par défaut (a ne pas utiliser)
105        // le constructeur courant
106        SaveResul_LoiContraintesPlanes(SaveResul* l_des_SaveResul);
107        // constructeur de copie
108        SaveResul_LoiContraintesPlanes(const SaveResul_LoiContraintesPlanes& sav );
109        // destructeur
110        ~SaveResul_LoiContraintesPlanes();
111        // définition d'une nouvelle instance identique
112        // appelle du constructeur via new
113        SaveResul * Nevez_SaveResul() const {return (new
114        SaveResul_LoiContraintesPlanes(*this));};
115    // affectation
116    virtual SaveResul & operator = ( const SaveResul & a);
117        //===== lecture écriture dans base info =====
118        // cas donne le niveau de la récupération
119        // = 1 : on récupère tout
120        // = 2 : on récupère uniquement les données variables (supposées comme telles)
121        void Lecture_base_info (ifstream& ent,const int cas);
122        // cas donne le niveau de sauvegarde

```

```

122         // = 1 : on sauvegarde tout
123         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
124         void Ecrire_base_info(ofstream& sort,const int cas);
125
126         // mise à jour des informations transitoires en définitif s'il y a convergence
127         // par exemple (pour la plasticité par exemple)
128         void TdtversT() ;
129         void TversTdt () ;
130
131         // affichage à l'écran des infos
132         void Affiche() const;
133
134         //changement de base de toutes les grandeurs internes tensorielles stockées
135         // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gb
136         // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
137         // gpH(i) = gamma(i,j) * gH(j)
138         virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
139
140         // procedure permettant de completer éventuellement les données particulières
141         // de la loi stockées
142         // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
143         // completer est appelé apres sa creation avec les donnees du bloc transmis
144         // peut etre appeler plusieurs fois
145         SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
146                                     ,const Loi_comp_abstraite* loi);
147
148         // ---- gestion d'informations spécifiques à certaine classe dérivée
149         double Deformation_plastique() ;
150
151         void Transfert_var_h(const SaveResul_LoiContraintesPlanes& sav)
152         {hsurh0 = sav.hsurh0;h_tsurh0 = sav.h_tsurh0;};
153
154         // création des conteneurs pour la déformation mécanique
155         void Creation_def_mecanique();
156
157         // -- données protégées
158
159         // les données protégées de la loi 3D associée
160         SaveResul* le_SaveResul;
161         // les contraintes qui servent d'entrée au calcul de la loi associée
162         TenseurHH* l_sigoHH, * l_sigoHH_t; // valeur courante, et valeur sauvegardée au pas
précédent
163         Vecteur epsInvar,epsInvar_t; // on sauvegarde les invariants ordre 3 à l'instat t
164         Vecteur depsInvar,depsInvar_t; // idem pour la vitesse
165         Tableau <double> def_equi, def_equi_t ; // les différentes def cumulées
166         // -- stockage éventuelle d'une déformation mécanique a priori différente de la def cinématique
167         TenseurBB* eps_mecaBB, * eps_mecaBB_t; // la déformation mécanique associée en g^i
168         TenseurBB* eps_P_mecaBB, * eps_P_mecaBB_t; // la déformation mécanique associée en ortho dans
le repère de traction
169         Met_abstraite::Umat_cont met_meca; // la métrique mécanique associée à la déformation méca
170         // les éléments de met_meca sont alimentés que si la déformation méca associée est
171         // elle-même alimentée
172
173         // les énergies pour la loi
174         EnergieMeca l_energ,l_energ_t; // valeur courante, et valeur sauvegardée au pas précédent
175         // les élongations suivant l'épaisseur:
176         double hsurh0,h_tsurh0; // valeur courante et valeur sauvegardée au pas précédent
177         Vecteur d_hsurh0; // vecteur de taille éventuellement nulle, contenant les variations de h / au
ddl
178         // c'est un vecteur de travail, il n'est pas sauvegardé d'un incrément à
l'autre
179         // il sert à mémoriser les choses d'une itération à l'autre
180
181         // --- tableau d'indicateur de la résolution, éventuellement vide
182         // cela dépend de sortie_post
183         // indicateurs_resolution(1) : nb total d'incrément utilisé pour le Newton
184         // (2) : nb total d'itération " " " "
185         Tableau <double> indicateurs_resolution,indicateurs_resolution_t;
186         // // niveau de sig 33
187         // double niveau_sig33,niveau_sig33_t; // le niveau qui a été retenue pour sig33
188     };
189
190     // def d'une instance de données spécifiques, et initialisation
191     SaveResul * New_et_Initialise() ;
192
193     friend class SaveResul_LoiContraintesPlanes;
194
195     // Lecture des donnees de la classe sur fichier
196     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
197                                     ,LesFonctions_nD& lesFonctionsnD);
198
199     // affichage de la loi
200     void Affiche() const ;
201     // test si la loi est complete
202     // = 1 tout est ok, =0 loi incomplete
203     int TestComplet();
204
205     // calcul d'un module d'young équivalent à la loi, ceci pour un

```

```

205 // chargement nul
206 double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
207 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
208 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
209 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
    saveResul);
210
211 // récupération de la dernière déformation d'épaisseur calculée: cette déformation n'est utile que
    pour des lois en contraintes planes
212 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
213 // - pour les lois 2D def planes: retour de 0
214 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
215 // qui est le conteneur spécifique au point où a été calculé la loi
216 virtual double Eps33BH(SaveResul * saveResul) const ;
217
218 // récupération de la variation relative d'épaisseur calculée: h/h0
219 // cette variation n'est utile que pour des lois en contraintes planes
220 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
221 // - pour les lois 2D def planes: retour de 0
222 // les infos nécessaires à la récupération , sont stockées dans saveResul
223 // qui est le conteneur spécifique au point où a été calculé la loi
224 virtual double HsurH0(SaveResul * saveResul) const;
225
226 // indique si la loi est en contraintes planes en s'appuyant sur un comportement 3D
227 virtual bool Contraintes_planes_de_3D() const {return true;};
228
229 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
230 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new LoiContraintesPlanes(*this)); };
231
232 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
233 // exemple: mise en service des ddl de température aux noeuds
234 virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud,bool
    dilatation,LesPtIntegMecaInterne& lesPtMecaInt);
235 // récupération des grandeurs particulière (hors ddl )
236 // correspondant à liTQ
237 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
238 virtual void Grandeur_particuliere
    (bool absolue,List_io<TypeQuelconque>& liTQ,Loi_comp_abstraite::SaveResul * saveDon,list<int>&
    decal) const;
239 // récupération de la liste de tous les grandeurs particulières
240 // ces grandeurs sont ajoutées à la liste passées en paramètres
241 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
242 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& liTQ) const;
243
244
245
246 //----- lecture écriture de restart -----
247 // cas donne le niveau de la récupération
248 // = 1 : on récupère tout
249 // = 2 : on récupère uniquement les données variables (supposées comme telles)
250 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
    lesCourbes1D
    ,LesFonctions_nD& lesFonctionsnD);
251
252
253 // cas donne le niveau de sauvegarde
254 // = 1 : on sauvegarde tout
255 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
256 void Ecriture_base_info_loi(ofstream& sort,const int cas);
257
258 // affichage et definition interactive des commandes particulières à chaque lois
259 void Info_commande_LoisDeComp(UtilLecture& lec);
260
261 protected :
262
263 // permet d'indiquer à la classe à quelle valeur de PtIntegMecaInterne il faut se référer
264 // en particulier est utilisé par les lois additives,
265 // par contre doit être utilisé avec prudence
266 virtual void IndiquePtIntegMecaInterne(const PtIntegMecaInterne * ptintmeca)
    { lois_interne->IndiquePtIntegMecaInterne(ptintmeca);
267 // puis la classe mère
268 Loi_comp_abstraite::IndiquePtIntegMecaInterne(ptintmeca);
269 };
270
271
272
273 // codage des METHODES VIRTUELLES protegees:
274 // calcul des contraintes a t+dt
275 // calcul des contraintes
276 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
    ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
    ,TenseurBB & delta_epsBB_
277 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
278 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
279 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
280 ,const Met_abstraite::Expli_t_tdt& ex);
281
282
283
284 // calcul des contraintes et de ses variations a t+dt
285 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl

```



```

286     ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
287     ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
288     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
289     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
290     ,Tableau <TenseurBB *>& d_gijBB_tdt
291     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
292     ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
293     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
294     ,const Met_abstraite::Impli& ex);
295
296     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
297     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
298     // le tenseur de déformation et son incrémentsont également en orthonormees
299     // si = false: les bases transmises sont utilisées
300     // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
301 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
302     ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
303     ,TenseurHH& sigHH,TenseurHHHH & d_sigma_deps
304     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
305     ,const Met_abstraite::Umat_cont& ex) ; // = 0;
306
307
308 // fonction surchargée dans les classes dérivée si besoin est
309 virtual void CalculGrandeurTravail
310     (const PtIntegMecaInterne& ptintmeca
311     ,const Deformation & def,Enum_dure temps,const ThermoDonnee& dTP
312     ,const Met_abstraite::Impli* ex_impli
313     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
314     ,const Met_abstraite::Umat_cont* ex_umat
315     ,const List_io<Ddl_etendu>* exclure_dd_etend
316     ,const List_io<const TypeQuelconque *>* exclure_Q
317     );
318
319 // fonction interne utilisée par les classes dérivées de Loi_comp_abstraite
320 // pour répercuter les modifications de la température
321 // ici utiliser pour modifier la température des lois élémentaires
322 // l'Enum_dure: indique quel est la température courante : 0 t ou tdt
323 void RepercuteChangeTemperature(Enum_dure temps);
324
325 //calcul d'une condition de contrainte plane dans une direction donnée, a priori qui n'est pas
326 // la direction 3. Il s'agit ici d'une I/O en loi 3D, dans un espace 3D
327
328 protected:
329 // passage des grandeurs métriques de l'ordre 2 à 3: cas implicite
330 // ramène un conteneur dont les éléments sont gérés par la loi CP, et ne peuvent être modifié que par
elle
331 const Met_abstraite::Impli* Passage_metrique_ordre2_vers_3(const Met_abstraite::Impli& ex);
332 // passage des grandeurs métriques de l'ordre 2 à 3: cas explicite
333 // ramène un conteneur dont les éléments sont gérés par la loi CP, et ne peuvent être modifié que par
elle
334 const Met_abstraite::Expli_t_tdt* Passage_metrique_ordre2_vers_3(const Met_abstraite::Expli_t_tdt&
ex);
335 // passage des grandeurs métriques de l'ordre 2 à 3: cas implicite
336 // ramène un conteneur dont les éléments sont gérés par la loi CP, et ne peuvent être modifié que par
elle
337 const Met_abstraite::Umat_cont* Passage_metrique_ordre2_vers_3(const Met_abstraite::Umat_cont& ex);
338 // passage des informations liées à la déformation de 2 vers 3: permet d'utiliser
339 // les conteneurs de l'instance CP, mais uniquement par les classes friend (a utiliser avec précaution
!)
340 void Passage_deformation_volume_ordre2_vers_3
341     (const TenseurBB& DepsBB,const TenseurBB & epsBB_tdt
342     ,const TenseurBB & delta_epsBB,const TenseurBB& Deps_BB_3D
343     ,const TenseurBB & eps_BB_3D,const TenseurBB & delta_eps_BB_3D
344     );
345
346
347 public:
348     //--- cas de la résolution de l'équation sig33(hsurh0)=0
349     // calcul de la fonction résidu de la résolution de l'équation constitutive
350 // l'argument test ramène
351 // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
352 // fatal, qui invalide le calcul du résidu
353 Vecteur& Residu_constitutif (const double & alpha, const Vecteur & x, int& test);
354 // calcul de la matrice tangente de la résolution de l'équation constitutive
355 // l'argument test ramène
356 // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
357 // fatal, qui invalide le calcul du résidu et de la dérivée
358 Mat_abstraite& Mat_tangente_constitutif(const double & alpha,const Vecteur & x, Vecteur& resi,
int& test);
359
360
361 private :
362     // donnees protegees
363     Enum_contrainte_mathematique type_de_contrainte; // def de la méthode utilisée pour imposer la
condition de

```

```

364 // de contrainte plane
365 // paramètre qui servent éventuellement pour la contrainte mathématique
366 double fac_penal; // le facteur de pénalisation relative,
367 double prec; // précision sur la contrainte
368
369 // double niveau_mini_sig33; // le niveau visé pour sig33, par défaut 0.
370 // // a) contrôle via une ou plusieurs grandeurs globales
371 // Ponderation_GGlobal* niveauF_grandeurGlobale;
372 // // b) via éventuellement un ddl étendu
373 // Ponderation * niveauF_ddlEtendu;
374 // // c) via éventuellement le temps
375 // Ponderation_temps * niveauF_temps;
376 // // d) contrôle via une ou plusieurs grandeurs consultables
377 // Ponderation_Consultable* niveauF_grandeurConsultable;
378
379 Loi_comp_abstraite * lois_interne; // loi 3D correspondante
380
381 int sortie_post; // permet d'accéder au nombre d'itération, d'incrément, de précision etc. des
résolutions
382 // = 0 par défaut,
383 // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
384 // --- variables particulières pour le cas ou on utilise une boucle de newton interne pour la
contrainte plane
385 Algo_zero alg_zero; // algo pour la recherche de zero
386 double maxi_delta_var_eps_sur_iter_pour_Newton; // le maxi de variation que l'on tolère d'une
itération à l'autre
387 // pilotage éventuel des précisions
388 Fonction_nD * fct_tolerance_residu, * fct_tolerance_residu_rel;
389
390 Vecteur val_initiale; // on démarre la recherche à la valeur à t
391 Vecteur racine; // dimensionnement init du résultat à 0.
392 Mat_pleine der_at_racine; // dimensionnement et init de la matrice dérivée à 0.
393 Vecteur residu; // résidu de l'équation c'est à dire sig33
394 Mat_pleine derResidu; // dérivé du résidu de l'équation c-a-d dsig33/dhsurh0
395 double mini_hsurh0; // limitation de la variation de hsurh0
396 double maxi_hsurh0; // limitation de la variation de hsurh0
397 int choix_calcul_epaisseur_si_relachement_complet;
398 // ----- controle de la sortie des informations
399 // -> maintenant définit dans LoiAbstraiteGeneral
400 // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes,
401 // pour les erreurs et des warnings
402 double module_compressibilite_3D;
403 double module_cisaillement_3D;
404 CoordonneeB giB_normer_3_tdt_3D_sauve; // sauvegarde du vecteur giB(3), normé
405 double sauve_jacobien2D_tdt;
406
407 // déclaration des variables internes nécessaires pour les passages 2D - 3D
408 // -- on définit des conteneurs pour le stockage des résultats des métriques, dimensionnés par défaut
non vide
409 // on utilise des pointeurs pour dimensionner après les variables internes
410 Met_abstraite::Expli_t_tdt* expli_3D;
411 Met_abstraite::Impli* impli_3D;
412 Met_abstraite::Umat_cont* umat_cont_3D;
413
414 // -- variables nécessaires pour la création de expli_3D, impli_3D et umat_cont_3D
415 // certaines grandeurs sont associées à un pointeur qui peut soit être nulle soit pointer sur le
conteneur
416 // l'intérêt est que le fait d'avoir un pointeur nul est parfois utilisé pour éviter un calcul
417 BaseB giB_0_3D;
418 BaseH giH_0_3D;
419 BaseB giB_t_3D;
420 BaseH giH_t_3D;
421 BaseB giB_tdt_3D;
422 BaseH giH_tdt_3D;
423 Tenseur3BB gijBB_0_3D;
424 Tenseur3HH gijHH_0_3D;
425 Tenseur3BB gijBB_t_3D;
426 Tenseur3HH gijHH_t_3D;
427 Tenseur3BB gijBB_tdt_3D;
428 Tenseur3HH gijHH_tdt_3D;
429
430 TenseurBB * gradVmoyBB_t_3D_P; Tenseur_ns3BB gradVmoyBB_t_3D;
431 TenseurBB * gradVmoyBB_tdt_3D_P; Tenseur_ns3BB gradVmoyBB_tdt_3D;
432 TenseurBB * gradVBB_tdt_3D_P; Tenseur_ns3BB gradVBB_tdt_3D;
433 double jacobien_tdt_3D; double jacobien_t_3D; double jacobien_0_3D; // pour les jacobiens on considère
qu'ils existent toujours
434 Vecteur d_jacobien_tdt_3D;
435 // pour tous les tableaux de pointeurs, on double le tableau en déclarant un vrai tableau en //
436 Tableau <BaseB> d_giB_tdt_3D;
437 Tableau <BaseH> d_giH_tdt_3D;
438 Tableau <TenseurBB *> d_gijBB_tdt_3D_P; Tableau <Tenseur3BB > d_gijBB_tdt_3D;
439 Tableau2 <TenseurBB *> d2_gijBB_tdt_3D_P; Tableau2 <Tenseur3BB > d2_gijBB_tdt_3D; // a priori ne
sera pas affecté, car ne sert
440 // dans les
lois de comportement
441 Tableau <TenseurHH *> d_gijHH_tdt_3D_P; Tableau <Tenseur3HH > d_gijHH_tdt_3D;
442 Tableau <TenseurBB *> d_gradVmoyBB_t_3D_P; Tableau <Tenseur_ns3BB > d_gradVmoyBB_t_3D;

```

```

443 Tableau <TenseurBB * >* d_gradVmoyBB_tdt_3D_P; Tableau <Tenseur_ns3BB > d_gradVmoyBB_tdt_3D;
444 Tableau <TenseurBB * >* d_gradVBB_t_3D_P; Tableau <Tenseur_ns3BB > d_gradVBB_t_3D;
445 Tableau <TenseurBB * >* d_gradVBB_tdt_3D_P; Tableau <Tenseur_ns3BB > d_gradVBB_tdt_3D;
446
447 // -- on définit les conteneurs pour les passages d'appels entrant de la loi 3D : donc en 3D par
    défaut
448 Tenseur3HH sig_HH_t_3D, sig_HH_3D ;
449 Tenseur3BB Deps_BB_3D, eps_BB_3D, delta_eps_BB_3D;
450 Tableau <TenseurBB * > d_eps_BB_3D_P; Tableau <Tenseur3BB > d_eps_BB_3D; // le tableau de
    pointeur puis les vrais grandeurs
451 Tableau <TenseurHH * > d_sig_HH_3D_P; Tableau <Tenseur3HH > d_sig_HH_3D; // """"
452 Tenseur3HHHH d_sigma_deps_3D;
453
454 Tenseur2HHHH d_sigma_deps_2D; // un tenseur de travail pour la méthode Calcul_dsigma_deps (
455 // pour les méthodes: Mat_tangente_constitutif et Residu_constitutif
456
457 // cas d'un point d'intégration locale (méthode CalculGrandeurTravail par exemple)
458 PtIntegMecaInterne ptintmeca;
459
460 //--- méthodes internes
461 // prise en compte de h et variation éventuelle sur métriques
462 void Prise_en_compte_h_sur_metriche();
463 void Prise_en_compte_h_et_variation_sur_metriche(const Met_abstraite::Impli& ex);
464 // passage des informations liées à la déformation de 2 vers 3, et variation de volume éventuelle
465 // si le pointeur d_jacobien_tdt est non nul
466 // idem pour d_epsBB
467 void Passage_deformation_volume_ordre2_vers_3(const TenseurBB& DepsBB
468 ,const TenseurBB & epsBB_tdt,const Tableau <TenseurBB * >* d_epsBB
469 ,const TenseurBB & delta_epsBB,const double& jacobien_0,const double& jacobien
470 ,const Vecteur* d_jacobien_tdt);
471 // idem mais sans variation: passage des informations liées à la déformation de 2 vers 3
472 void Passage_deformation_volume_ordre2_vers_3(const TenseurBB& DepsBB,const TenseurBB & epsBB_tdt
473 ,const TenseurBB & delta_epsBB,const double& jacobien_0,const double& jacobien);
474 // mise à jour des informations liées à la déformation de 2 vers 3
475 void Mise_a_jour_deformations_et_Jacobien_en_3D();
476
477 // calcul de la déformation d'épaisseur, en utilisant la variation de volume et la compressibilité
478 void Calcul_eps33_parVarVolume(double& jaco_2D_0,const double& module_compressibilite,double& jacobien
479 ,TenseurBH& sigHH);
480 // calcul de la déformation d'épaisseur et de sa variation
481 void Calcul_d_eps33_parVarVolume(double& jaco_2D_0,const double& module_compressibilite,double&
    jacobien
482 ,TenseurHH& sigHH_,Vecteur& d_jaco_2D
483 ,Tableau <TenseurHH * >& d_sigHH,Tableau <TenseurBB * >& d_gijBB_tdt
484 ,TenseurBB & gijBB_);
485 // calcul des invariants de déformation et de vitesse de déformation, et les def cumulées
486 // correspondant aux cas 3D
487 void Calcul_invariants_et_def_cumul();
488
489 // limitation des variations d'épaisseurs
490 // ramène true s'il y a eu une modif
491 bool Limitation_h(double& hsurh0);
492
493 };
494 /// @} // end of group
495
496 #endif
497
498
499

```

## 7.72 LoiContraintesPlanesDouble.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty

```

```

23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           06/03/2023
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 *
39 *   BUT:   La loi est 1D et est associée à une loi 3D quelconque
40 *          L'objectif est de transformer une loi 3D en 1D contraintes
41 *          planes dans les 2 autres directions.
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *   ! date ! auteur ! but
47 *   -----
48 *   ! ! !
49 *
50 *   *****
51 *   MODIFICATIONS:
52 *   ! date ! auteur ! but
53 *   -----
54 *
55 *   *****/
56
57 // FICHER : LoiContraintesPlanesDouble.h
58 // CLASSE : LoiContraintesPlanesDouble
59
60 #ifndef LOICONTRAINTEPLANESDOUBLE_H
61 #define LOICONTRAINTEPLANESDOUBLE_H
62
63 #include "Loi_comp_abstraite.h"
64 #include "Enum_contrainte_mathematique.h"
65 #include "Algo_edp.h"
66 #include "MatLapack.h"
67 #include "LoiContraintesPlanes.h"
68
69
70 /// @addtogroup Les_lois_combinees
71 /// @{
72 ///
73
74
75 class LoiContraintesPlanesDouble : public Loi_comp_abstraite
76 {
77     public :
78     friend class LoiContraintesPlanes;
79     friend class LoiCritere;
80
81     // CONSTRUCTEURS :
82
83     // Constructeur par défaut
84     LoiContraintesPlanesDouble ();
85
86     // constructeur à partir d'une instance de contraintes planes
87     // à condition que cette loi s'appuie sur un comportement 3D
88     // les deux lois étant proche, les paramètres semblables sont mis en commun
89     LoiContraintesPlanesDouble (LoiContraintesPlanes& loiCP_de3D
90     ,bool calcul_en_3D_via_direction_quelconque_);
91
92     // Constructeur de copie
93     LoiContraintesPlanesDouble (const LoiContraintesPlanesDouble& loi) ;
94
95     // DESTRUCTEUR :
96     ~LoiContraintesPlanesDouble ();
97
98
99
100     // initialise les donnees particulieres a l'elements
101     // de matiere traite ( c-a-dire au pt calcule)
102     // Il y a creation d'une instance de SaveResul particuliere
103     // a la loi concernee
104     // la SaveResul classe est remplie par les instances heritantes
105     // le pointeur de SaveResul est sauvegarde au niveau de l'element
106     // c'a-d que les info particulieres au point considere sont stocke
107     // au niveau de l'element et non de la loi.

```

```

108     class SaveResul_LoiContraintesPlanesDouble: public
LoiContraintesPlanes::SaveResul_LoiContraintesPlanes
109     { public :
110         SaveResul_LoiContraintesPlanesDouble(); // constructeur par défaut (a ne pas utiliser)
111         // le constructeur courant
112         SaveResul_LoiContraintesPlanesDouble(SaveResul* l_des_SaveResul);
113         // constructeur de copie
114         SaveResul_LoiContraintesPlanesDouble(const SaveResul_LoiContraintesPlanesDouble& sav );
115         // destructeur
116         ~SaveResul_LoiContraintesPlanesDouble();
117         // définition d'une nouvelle instance identique
118         // appelle du constructeur via new
119         SaveResul * Nevez_SaveResul() const {return (new
SaveResul_LoiContraintesPlanesDouble(*this));};
120     // affectation
121     virtual SaveResul & operator = ( const SaveResul & a);
122     //===== lecture écriture dans base info =====
123     // cas donne le niveau de la récupération
124     // = 1 : on récupère tout
125     // = 2 : on récupère uniquement les données variables (supposées comme telles)
126     void Lecture_base_info (ifstream& ent,const int cas);
127     // cas donne le niveau de sauvegarde
128     // = 1 : on sauvegarde tout
129     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
130     void Ecriture_base_info(ofstream& sort,const int cas);
131
132     // mise à jour des informations transitoires en définitif s'il y a convergence
133     // par exemple (pour la plasticité par exemple)
134     void TdtversT() ;
135     void TversTdt() ;
136
137     // affichage à l'écran des infos
138     void Affiche() const;
139
140     //changement de base de toutes les grandeurs internes tensorielles stockées
141     // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gpB
142     // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
143     // *** les métriques locales ne sont pas concernées: pour les modifier, il faut utiliser la
méthode
144     // *** Affectation_metriques_locales, et Recuperation_metriques_locales
145     // gpH(i) = gamma(i,j) * gH(j)
146     virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
147
148     // procedure permettant de completer éventuellement les données particulières
149     // de la loi stockées
150     // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
151     // completer est appelé apres sa creation avec les donnees du bloc transmis
152     // peut etre appeler plusieurs fois
153     SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
, const Loi_comp_abstraite* loi);
154
155     // ---- récupération d'information: spécifique à certaine classe dérivée
156     double Deformation_plastique() ;
157
158     // transfert des infos d'épaisseur et de largeur d'une instance de même type
159     void Transfert_var_hetb(const SaveResul_LoiContraintesPlanesDouble& sav)
160     {bsurb0 = sav.bsurb0;b_tsurb0 = sav.b_tsurb0;
161     LoiContraintesPlanes::SaveResul_LoiContraintesPlanes::Transfert_var_h(sav);
162     };
163
164     // mise à jour des métriques locales
165     void Affectation_metriques_locale(Deformation::Stmet& met_ref_00
,Deformation::Stmet& met_ref_t
,Deformation::Stmet& met_ref_tdt)
166     {meti_ref_00 = met_ref_00;meti_ref_t = met_ref_t;meti_ref_tdt=met_ref_tdt;};
167
168     // mise à 0 des variations de largeur
169     void Zero_var_largeur() {bsurb0=0.; };
170
171     // données protégées
172     // (1) toutes celles de LoiContraintesPlanes
173
174     // -- infos relatives au repere de référence 3D pour l'élément 1D associé
175     // les repères sont orthogonaux, la direction 1 est celle de l'élément 1D
176     Deformation::Stmet meti_ref_00,meti_ref_t,meti_ref_tdt; // les infos à 0 à t et à tdt
177
178     // def méca dans un repère ortho de traction: 1 axe de traction, 2 épaisseur, 3 largeur
179     // ordonnée suivant
180     // def_P(1) -> eps_P(2,2), def_P(2) -> eps_P(3,3), def_P(3) -> eps_P(1,2)
181     Vecteur def_P,def_P_t; // si taille nulle n'est pas alimenté
182     // est alimenté que si : calcul_en_3D_via_direction_quelconque == true
183
184     // les élongations suivant la largeur:
185     // valeur courante et valeur sauvegardée au pas précédent
186     double bsurb0,b_tsurb0; // largeur
187     // Vecteur d_hsurh0;
188     // vecteur de taille éventuellement nulle, contenant les variations de h / au ddl
189
190
191

```

```

192                                     // c'est un vecteur de travail, il n'est pas sauvegardé d'un incrément à
193 l'autre
193                                     // il sert à mémoriser les choses d'une itération à l'autre
194         Vecteur d_bsurb0; // idem pour la largeur
195     };
196
197     // def d'une instance de données spécifiques, et initialisation
198     SaveResul * New_et_Initialise() ;
199
200 friend class SaveResul_LoiContraintesPlanesDouble;
201
202 // Lecture des donnees de la classe sur fichier
203 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
204                                     ,LesFonctions_nD& lesFonctionsnD);
205
206 // Lecture des paramètres particuliers de l'objet sur fichier
207 // cette lecture est utilisée lorsque l'objet a été déjà défini
208 // il s'agit donc d'une lecture à l'intérieur d'une autre loi par exemple
209 void LectureParametres_controls (UtilLecture * ,LesCourbes1D& lesCourbes1D
210                                     ,LesFonctions_nD& lesFonctionsnD);
211
212     // affichage de la loi
213     void Affiche() const ;
214     // test si la loi est complete
215     // = 1 tout est ok, =0 loi incomplete
216     int TestCompleet();
217
218 // calcul d'un module d'young équivalent à la loi, ceci pour un
219 // chargement nul
220 double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
221 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
222 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
223 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
224     saveResul);
225
226 // récupération de la dernière déformation d'épaisseur calculée: cette déformaion n'est utile que pour
227 // des lois en contraintes planes
228 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
229 // - pour les lois 2D def planes: retour de 0
230 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
231 // qui est le conteneur spécifique au point où a été calculé la loi
232 virtual double Eps33BH(SaveResul * saveResul) const
233 { // retour de la def
234     return eps_BB_3D(3,3);
235 };
236
237 // récupération de la dernière déformation de largeur calculée: cette déformaion n'est utile que pour
238 // des lois en contraintes doublement planes
239 // - pour les lois 3D et 2D : retour d'un nombre très grand, indiquant que cette fonction est invalide
240 // les infos nécessaires à la récupération de la def, sont stockées dans saveResul
241 // qui est le conteneur spécifique au point où a été calculé la loi
242 virtual double Eps22BH(SaveResul * saveResul) const
243 { // retour de la def
244     return eps_BB_3D(2,2);
245 };
246
247 // récupération de la variation relative d'épaisseur calculée: h/h0
248 // cette variation n'est utile que pour des lois en contraintes planes
249 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
250 // - pour les lois 2D def planes: retour de 0
251 // les infos nécessaires à la récupération , sont stockées dans saveResul
252 // qui est le conteneur spécifique au point où a été calculé la loi
253 virtual double HsurH0(SaveResul * saveResul) const
254 { // récup du conteneur spécifique
255     SaveResul_LoiContraintesPlanesDouble & save_resul = *((SaveResul_LoiContraintesPlanesDouble*)
256     saveResul);
257     // retour de la variation relative
258     return save_resul.hsurh0;
259 };
260
261 // récupération de la variation relative d'épaisseur calculée: h/h0
262 // et de sa variation par rapport aux ddls la concernant: d_hsurh0
263 // cette variation n'est utile que pour des lois en contraintes planes
264 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
265 // - pour les lois 2D def planes: retour de 0
266 // les infos nécessaires à la récupération , sont stockées dans saveResul
267 // qui est le conteneur spécifique au point où a été calculé la loi
268 virtual double d_HsurH0(SaveResul * saveResul,Vecteur & d_hsurh0) const
269 { // récup du conteneur spécifique
270     SaveResul_LoiContraintesPlanesDouble & save_resul = *((SaveResul_LoiContraintesPlanesDouble*)
271     saveResul);
272     d_hsurh0 = save_resul.d_hsurh0;
273     // retour de la variation relative
274     return save_resul.hsurh0;
275 };
276
277 // récupération de la variation relative de largeur calculée: b/b0

```

```

273 // cette variation n'est utile que pour des lois en contraintes planes double
274 // - pour les lois 3D et 2D : retour d'un nombre très grand, indiquant que cette fonction est invalide
275 // les infos nécessaires à la récupération , sont stockées dans saveResul
276 // qui est le conteneur spécifique au point où a été calculé la loi
277 virtual double BsurB0(SaveResul * saveResul) const
278 { // récup du conteneur spécifique
279     SaveResul_LoiContraintesPlanesDouble & save_resul = *((SaveResul_LoiContraintesPlanesDouble*)
    saveResul);
280     // retour de la variation relative
281     return save_resul.bsurb0;
282 };
283
284 // récupération de la variation relative de largeur calculée: b/b0
285 // et de sa variation par rapport aux ddlS la concernant: d_bsurb0
286 // cette variation n'est utile que pour des lois en contraintes planes
287 // - pour les lois 3D et 2D : retour d'un nombre très grand, indiquant que cette fonction est invalide
288 // les infos nécessaires à la récupération , sont stockées dans saveResul
289 // qui est le conteneur spécifique au point où a été calculé la loi
290 virtual double d_BsurB0(SaveResul * saveResul, Vecteur & d_bsurb0) const
291 { // récup du conteneur spécifique
292     SaveResul_LoiContraintesPlanesDouble & save_resul = *((SaveResul_LoiContraintesPlanesDouble*)
    saveResul);
293     d_bsurb0 = save_resul.d_bsurb0;
294     // retour de la variation relative
295     return save_resul.bsurb0;
296 };
297
298 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
299 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new LoiContraintesPlanesDouble(*this));
    };
300
301 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
302 // exemple: mise en service des ddl de température aux noeuds
303 virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud, bool
    dilatation, LesPtIntegMecaInterne& lesPtMecaInt);
304 // récupération des grandeurs particulière (hors ddl )
305 // correspondant à liTQ
306 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
307 virtual void Grandeur_particuliere
    (bool absolue, List_io<TypeQuelconque>& liTQ, Loi_comp_abstraite::SaveResul * saveDon, list<int>&
    decal) const;
308 // récupération de la liste de tous les grandeurs particulières
309 // ces grandeurs sont ajoutées à la liste passées en paramètres
310 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
311 virtual void ListeGrandeurs_particulieres(bool absolue, List_io<TypeQuelconque>& liTQ) const;
312
313
314
315 //----- lecture écriture de restart -----
316 // cas donne le niveau de la récupération
317 // = 1 : on récupère tout
318 // = 2 : on récupère uniquement les données variables (supposées comme telles)
319 void Lecture_base_info_loi(ifstream& ent, const int cas, LesReferences& lesRef, LesCourbes1D&
    lesCourbes1D
    , LesFonctions_nD & lesFonctionsnD);
320
321
322 // cas donne le niveau de sauvegarde
323 // = 1 : on sauvegarde tout
324 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
325 void Ecriture_base_info_loi(ofstream& sort, const int cas);
326
327 // affichage et definition interactive des commandes particulières à chaque lois
328 void Info_commande_LoisDeComp(UtilLecture& lec);
329
330 protected :
331
332 // codage des METHODES VIRTUELLES protegees:
333 // calcul des contraintes a t+dt
334 // calcul des contraintes
335 void Calcul_SigmaHH (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
    , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB& giB, BaseH & gi_H, TenseurBB & epsBB_
    , TenseurBB & delta_epsBB_
    , TenseurBB & gijBB_, TenseurHH & gijHH_, Tableau <TenseurBB *>& d_gijBB_
    , double& jacobien_0, double& jacobien, TenseurHH & sigHH
    , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
    module_cisaillement
    , const Met_abstraite::Expli_t_tdt& ex);
336
337
338 // calcul des contraintes et de ses variations a t+dt
339 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
    , BaseB& giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
    , BaseB& giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH & giH_tdt, Tableau <BaseH> & d_giH_tdt
    , TenseurBB & epsBB_tdt, Tableau <TenseurBB *>& d_epsBB
    , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
    , Tableau <TenseurBB *>& d_gijBB_tdt
    , Tableau <TenseurHH *>& d_gijHH_tdt, double& jacobien_0, double& jacobien
    , Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *>& d_sigHH
    , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&

```

```

module_cisaillement
353     ,const Met_abstraite::Impli& ex);
354
355     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
356     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
357     // le tenseur de déformation et son incrémentsont également en orthonormees
358     // si = false: les bases transmises sont utilisées
359     // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
360 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
361     ,TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
362     , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps
363     ,EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
364     module_cisaillement
365     ,const Met_abstraite::Umat_cont& ex) ; // = 0;
366
367 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
368 // ceci dans le cas où l'axe de traction simple est quelconque
369 // ici on suppose que:
370 // - tous les calculs sont en dim 3
371 // - l'axe 3 est normal aux deux autres (ceci pour Vi et pour gi)
372 //
373 // ViB et ViH : correspond à la base de traction telle que ViB(1) = ViH(1) = l'axe de traction
374 // La base Vi est supposée orthonormée
375 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
376 // le tenseur de déformation et son incrémentsont également en orthonormees
377 // si = false: les bases transmises sont utilisées
378 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
379 void Calcul_dsigma_deps (bool en_base_orthonormee, const BaseB * ViB, const BaseH * ViH
380     ,const TenseurHH & sigHH_t, const TenseurBB& DepsBB
381     ,const TenseurBB & epsBB_tdt, const TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
382     , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps
383     ,EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
384     module_cisaillement
385     ,const Met_abstraite::Umat_cont& ex) ; // = 0;
386
387 // fonction surchargée dans les classes dérivée si besoin est
388 virtual void CalculGrandeurTravail
389     (const PtIntegMecaInterne& ptintmeca
390     ,const Deformation & def, Enum_dure temps, const ThermoDonnee& dTP
391     ,const Met_abstraite::Impli* ex_impli
392     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
393     ,const Met_abstraite::Umat_cont* ex_umat
394     ,const List_io<Ddl_etendu>* exclure_dd_etendu
395     ,const List_io<const TypeQuelconque +>* exclure_Q
396     );
397
398 // permet d'indiquer à la classe à quelle valeur de PtIntegMecaInterne il faut se référer
399 // en particulier est utilisé par les lois additives,
400 // par contre doit être utilisé avec prudence
401 virtual void IndiquePtIntegMecaInterne(const PtIntegMecaInterne * ptintmeca)
402     { lois_interne->IndiquePtIntegMecaInterne(ptintmeca);
403     // puis la classe mère
404     Loi_comp_abstraite::IndiquePtIntegMecaInterne(ptintmeca);
405     };
406
407 // fonction interne utilisée par les classes dérivées de Loi_comp_abstraite
408 // pour répercuter les modifications de la température
409 // ici utiliser pour modifier la température des lois élémentaires
410 // l'Enum_dure: indique quel est la température courante : 0 t ou tdt
411 void RepercuteChangeTemperature(Enum_dure temps);
412
413 // retourne le type de méthode utilisée pour imposer la condition de contrainte plane
414 Enum_contrainte_mathematique Type_de_contrainte() const {return type_de_contrainte;};
415
416 // permet de changer la valeur de l'indicateur: bool calcul_en_3D_via_direction_quelconque;
417 // *** cet indicateur agit sur les conteneurs de stockage, il faut donc le changer dès la construction
418 // de l'instance: donc est utile lorsque l'on ne connaît pas encore la valeur de
419 // calcul_en_3D_via_direction_quelconque au moment de sa création
420 void Change_calcul_en_3D_via_direction_quelconque (bool calcul_en_3D_via_direction_quelconque_)
421     {calcul_en_3D_via_direction_quelconque = calcul_en_3D_via_direction_quelconque_;}
422
423 public:
424     //--- cas de la résolution de l'équation sig33(hsurh0)=0
425     // calcul de la fonction résidu de la résolution de l'équation constitutive
426     // l'argument test ramène
427     // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
428     // fatal, qui invalide le calcul du résidu
429     Vecteur& Residu_constitutif (const double & alpha, const Vecteur & x, int& test);
430     // calcul de la matrice tangente de la résolution de l'équation constitutive
431     // l'argument test ramène
432     // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
433     // fatal, qui invalide le calcul du résidu et de la dérivée
434     Mat_abstraite& Mat_tangente_constitutif(const double & alpha, const Vecteur & x, Vecteur& resi,
435     int& test);
436     //--- cas de la résolution de l'équation sigij(eps_meca)*V_j{.1}=0, l=2 et 3

```



```

436 // calcul de la fonction résidu de la résolution de l'équation constitutive
437 // l'argument test ramène
438 // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
439 // fatal, qui invalide le calcul du résidu
440 Vecteur& Residu_constitutif_3D (const double & alpha, const Vecteur & x, int& test);
441 // calcul de la matrice tangente de la résolution de l'équation constitutive
442 // l'argument test ramène
443 // . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
444 // fatal, qui invalide le calcul du résidu et de la dérivée
445 Mat_abstraite& Mat_tangente_constitutif_3D(const double & alpha, const Vecteur & x, Vecteur& resi, int&
    test);
446
447 private :
448     // donnees protegees
449     Enum_contrainte_mathematique type_de_contrainte; // def de la méthode utilisée pour imposer la
    condition de
450
451     // de contrainte plane
452     // paramètre qui servent éventuellement pour la contrainte mathématique
453     double fac_penal; // le facteur de pénalisation relative,
454     double prec; // précision sur la contrainte
455
456     Loi_comp_abstraite * lois_interne; // loi 3D correspondante
457
458     int sortie_post; // permet d'accéder au nombre d'itération, d'incrément, de précision etc. des
    résolutions
459     // = 0 par défaut,
460     // = 1 : on stocke toutes les grandeurs et elles sont disponibles en sortie
461
462     bool calcul_en_3D_via_direction_quelconque; // indicateur qui permet d'utiliser la seconde méthode
463     // de calcul des contraintes planes doubles: par défaut = false
464     // --- variables particulières pour le cas ou on utilise une boucle de newton interne pour la
    contrainte plane
465     Algo_zero alg_zero; // algo pour la recherche de zero
466     double maxi_delta_var_eps_sur_iter_pour_Newton; // le maxi de variation que l'on tolère d'une
    itération à l'autre
467     // pilotage éventuel des précisions
468     Fonction_nD * fct_tolerance_residu, * fct_tolerance_residu_rel;
469
470     Vecteur val_initiale; // on démarre la recherche à la valeur à t
471     Vecteur racine; // dimensionnement init du résultat à 0.
472     //Mat_pleine
473     MatLapack der_at_racine; // dimensionnement et init de la matrice dérivée à 0.
474     Vecteur residu; // résidu de l'équation c'est à dire <sig22,sig33>
475     //Mat_pleine
476     MatLapack derResidu; // dérivé du résidu de l'équation c-a-d d<sig22,sig33>/<dbsurb0,dhsurb0>
477     double mini_hsurh0; // limitation de la variation de hsurh0
478     double maxi_hsurh0; // limitation de la variation de hsurh0
479     double mini_bsurb0; // limitation de la variation de bsurb0
480     double maxi_bsurb0; // limitation de la variation de bsurb0
481     // ---- controle de la sortie des informations
482     // -> maintenant définit dans LoiAbstraiteGeneral
483     int permet_affichage; // pour permettre un affichage spécifique dans les méthodes,
484     // pour les erreurs et des warnings
485     // -- pour le calcul de d_eps_eg_ll
486     Mat_pleine d_sig_ef_gh;
487     Vecteur d_eps_ef_ll;
488
489     //
490     double module_compressibilite_3D;
491     double module_cisaillement_3D;
492     CoordonneeB giB_normer_3_tdt_3D_sauve; // sauvegarde du vecteur giB(3), normé
493     CoordonneeB giB_normer_2_tdt_3D_sauve; // sauvegarde du vecteur giB(2), normé
494     double sauve_jacobien1D_tdt;
495
496     // déclaration des variables internes nécessaires pour les passages 1D - 3D
497     // -- on définit des conteneurs pour le stockage des résultats des métriques, dimensionnés par défaut
    non vide
498     // on utilise des pointeurs pour dimensionner après les variables internes
499     Met_abstraite::Expli_t_tdt* expli_3D;
500     Met_abstraite::Impli* impli_3D;
501     Met_abstraite::Umat_cont* umat_cont_3D;
502
503     // -- variables nécessaires pour la création de expli_3D, impli_3D et umat_cont_3D
504     // certaines grandeurs sont associées à un pointeur qui peut soit être nulle soit pointer sur le
    conteneur
505     // l'intérêt est que le fait d'avoir un pointeur nul est parfois utilisé pour éviter un calcul
506     BaseB giB_0_3D;
507     BaseH giH_0_3D;
508     BaseB giB_t_3D;
509     BaseH giH_t_3D;
510     BaseB giB_tdt_3D;
511     BaseH giH_tdt_3D;
512     Tenseur3BB gijBB_0_3D;
513     Tenseur3HH gijHH_0_3D;
514     Tenseur3BB gijBB_t_3D;
515     Tenseur3HH gijHH_t_3D;
516     Tenseur3BB gijBB_tdt_3D;
517     Tenseur3HH gijHH_tdt_3D;

```

```

516
517 TenseurBB * gradVmoyBB_t_3D_P;          Tenseur_ns3BB gradVmoyBB_t_3D;
518 TenseurBB * gradVmoyBB_tdt_3D_P; Tenseur_ns3BB gradVmoyBB_tdt_3D;
519 TenseurBB * gradVBB_tdt_3D_P;          Tenseur_ns3BB gradVBB_tdt_3D;
520 double jacobien_tdt_3D;double jacobien_t_3D;double jacobien_0_3D; // pour les jacobiens on considère
    qu'ils existent toujours
521 Vecteur d_jacobien_tdt_3D;
522 // pour tous les tableaux de pointeurs, on double le tableau en déclarant un vrai tableau en //
523 Tableau <BaseB> d_giB_tdt_3D;
524 Tableau <BaseH> d_giH_tdt_3D;
525 Tableau <TenseurBB *> d_gijBB_tdt_3D_P;          Tableau <Tenseur3BB > d_gijBB_tdt_3D;
526 Tableau2 <TenseurBB *>* d2_gijBB_tdt_3D_P;          Tableau2 <Tenseur3BB > d2_gijBB_tdt_3D; // a priori ne
    sera pas affecté, car ne sert
527 // dans les
    lois de comportement
528 Tableau <TenseurHH *> d_gijHH_tdt_3D_P;          Tableau <Tenseur3HH > d_gijHH_tdt_3D;
529 Tableau <TenseurBB *>* d_gradVmoyBB_t_3D_P;          Tableau <Tenseur_ns3BB > d_gradVmoyBB_t_3D;
530 Tableau <TenseurBB *>* d_gradVmoyBB_tdt_3D_P;          Tableau <Tenseur_ns3BB > d_gradVmoyBB_tdt_3D;
531 Tableau <TenseurBB *>* d_gradVBB_t_3D_P;          Tableau <Tenseur_ns3BB > d_gradVBB_t_3D;
532 Tableau <TenseurBB *>* d_gradVBB_tdt_3D_P;          Tableau <Tenseur_ns3BB > d_gradVBB_tdt_3D;
533
534 // -- on définit les conteneurs pour les passages d'appels entrant de la loi 3D : donc en 3D par
    défaut
535 Tenseur3HH sig_HH_t_3D, sig_HH_3D ;
536 Tenseur3BB Deps_BB_3D, eps_BB_3D, delta_eps_BB_3D;
537 Tableau <TenseurBB *> d_eps_BB_3D_P;          Tableau <Tenseur3BB > d_eps_BB_3D; // le tableau de
    pointeur puis les vrais grandeurs
538 Tableau <TenseurHH *> d_sig_HH_3D_P;          Tableau <Tenseur3HH > d_sig_HH_3D; // """"
539 Tenseur3HHHH d_sigma_deps_3D;
540
541 Tenseur1HHHH d_sigma_deps_1D; // un tenseur de travail pour la méthode Calcul_dsigma_deps (
542 // pour les méthodes: Mat_tangente_constitutif et Residu_constitutif
543
544 // cas d'un point d'intégration locale (méthode CalculGrandeurTravail par exemple)
545 PtIntegMecaInterne ptintmeca;
546
547 //----- spécifiques à Calcul_dsigma_deps hors axes -----
548 //grandeurs de travail, pour le passage d'infos en interne aux méthodes
549 // Residu_constitutif_3D et Mat_tangente_constitutif_3D, de l'équation
550 // sigij(eps_meca)+V_j{.1}=0, l=2 et 3
551 const BaseB* ViB_3D; const BaseH* ViH_3D;
552 Vecteur val_initiale_3D,racine_3D;
553 Vecteur residu_3D;
554 MatLapack derResidu_3D; // dérivé du résidu de l'équation c-a-d <résidu>/<def méca>
555 Tenseur3BB eps_P_BB,Deps_P_BB,delta_eps_P_BB; // def de travail: {epsilon'}_{kl}
556 Mat_pleine gamma3D,beta3D,gammaP_3D,betaP_3D; // matrices de travail pour les changements de base
557 const Tenseur3BB* DepsBB_cin; // vitesse cinématique d'entrée
558 const Tenseur3BB* epsBB_tdt_cin; // déformation cinématique d'entrée
559 const Tenseur3BB* delta_epsBB_cin; // delta def cinématique d'entrée
560 Mat_pleine mat_inter; // sert pour le calcul du jacobien mécanique
561 Tenseur3BB gij_meca_BB,gij_meca_BB_t; // un tenseur intermédiaire pour le calcul du jacobien mécanique
562 //double& jacobien_0,double& jacobien
563 Tableau <int> ind,jnd; // tableau d'index pour passer de (2,2) , (3,3) , (1,2) à 1,2,3
564 // ind le premier indice et jnd le second indice
565 // pour (m,n) = (2,2); (3,3); (1,2), et pour (g,h) = (2,2); (3,3); et (1,2)
566 // on va utiliser des matrices intermédiaires tels que:
567 // in = 1 pour (2,2) , 2 pour (3,3) et 3 pour (1,2)
568 Tenseur3BB V1V1_BB;
569 Tableau <Tenseur3BB > VhVg_BB;
570
571 //----- fin spécifiques à Calcul_dsigma_deps hors axes -----
572
573 //--- méthodes internes
574 // passage des grandeurs métriques de l'ordre 1 à 3: cas implicite
575 void Passage_métrique_ordre1_vers_3(const Met_abstraite::Implicite& ex);
576 // passage des grandeurs métriques de l'ordre 1 à 3: cas explicite
577 void Passage_métrique_ordre1_vers_3(const Met_abstraite::Explicite_t_tdt& ex);
578 // passage des grandeurs métriques de l'ordre 1 à 3: cas implicite
579 void Passage_métrique_ordre1_vers_3(const Met_abstraite::Umat_cont& ex);
580 // passage des informations liées à la déformation de l vers 3, et variation de volume éventuelle
581 // si le pointeur d_jacobien_tdt est non nul
582 // idem pour d_epsBB
583 void Passage_deformation_volume_ordre1_vers_3(TenseurBB& DepsBB
584 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>* d_epsBB
585 ,TenseurBB & delta_epsBB,const double& jacobien_0,double& jacobien
586 ,Vecteur* d_jacobien_tdt,const double& jacobien_t);
587 // mise à jour des informations liées à la déformation de l vers 3
588 void Mise_à_jour_deformations_et_Jacobien_en_3D();
589
590 // calcul des déformations d'épaisseur et de largeur, en utilisant la variation de volume et la
    compressibilité
591 // cas de la méthode 1
592 void Calcul_eps_trans_parVarVolumel(Tenseur3BH& sigBH_t,double& jaco_1D_0,const double&
    module_compressibilite,double& jacobien
593 ,Tenseur3BH& sigBH,double& jaco_1D_t);
594 // calcul des déformations d'épaisseur et de largeur, en utilisant la variation de volume et la

```

```

    compressibilité
596 // cas de la méthode 2
597 void Calcul_eps_trans_parVarVolume2(Tenseur3BH& sigBH_t,double& jaco_1D_0,const double&
    module_compressibilite,double& jacobien
598     ,Tenseur3BH& sigBH,double& jaco_1D_t);
599 // calcul des déformations d'épaisseur et de largeur et des variations
600 void Calcul_d_eps_trans_parVarVolume(double& jaco_1D_0,const double& module_compressibilite,double&
    jacobien
601     ,TenseurHH& sigHH_,Vecteur& d_jaco_1D
602     ,Tableau <TenseurHH *>& d_sigHH,Tableau <TenseurBB *>& d_gijBB_tdt
603     ,TenseurBB & gijBB_);
604 // calcul de la variation des déformations d'hauteur et de largeur, par rapport à la def_11, à cause
605 // des contraintes planes doubles
606 void Calcul_d_eps_eg_11();
607 // calcul de l'opérateur tangent (2ième méthode)
608 // d sigma^{ij} / d epsilon_{kl}
609 void Calcul2_dsigma_deps(TenseurHHHH& d_sigma_deps);
610
611 // calcul des invariants de déformation et de vitesse de déformation, et les def cumulées
612 // correspondant aux cas 3D
613 void Calcul_invariants_et_def_cumul();
614
615 // limitation des variations d'épaisseurs et largeurs
616 // ramène true s'il y a eu une modif
617 bool Limitation_h_b(double& hsurh0, double& bsub0);
618
619 // vérif des grandeurs et calcul de l'épaisseur et de largeur ainsi que calcul du jacobien final
620 // = 0 pas de modif
621 // = 1 modif due aux maxi permis de h et b, = 2 si def d'entrée incorrecte
622 // cas de la 2ième méthode
623 int Calcul_et_Limitation2_h_b(double& hsurh0,Tenseur3BB& eps_BB_3D, double& bsub0
624     ,double& jacobien_tdt_3DD, const double& jacobien_0_3DD);
625
626 // calcul de l'état final, dans le cas de la méthode de perturbation: version 2
627 void Cal_avec_perturbation2();
628
629 // passage entre le calcul classique en 1D de Calcul_dsigma_deps et le calcul 3D
630 void Passage_Calcul_1D_3D_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
631     ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
632     ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
633     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
634     ,const Met_abstraite::Umat_cont& ex) ; // = 0;
635
636
637 // passage entre le calcul classique en 1D de Calcul_DsigmaHH_tdt et le calcul 3D
638 void Passage_Calcul_1D_3D_dsigma_DsigmaHH_tdt
639     (TenseurHH& sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
640     ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
641     ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
642     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
643     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
644     ,Tableau <TenseurBB *>& d_gijBB_tdt
645     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
646     ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH_tdt,Tableau <TenseurHH *>& d_sigHH
647     ,EnergieMeca & energ,const EnergieMeca &
648     ,double& module_compressibilite,double& module_cisaillement
649     ,const Met_abstraite::Impli& ex);
650
651
652 };
653 /// @} // end of group
654
655 #endif
656
657
658

```

## 7.73 LoiCritere.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //

```

```

16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           11/06/2014
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 *   ****
39 *   BUT:   Définir une loi telle que la contrainte résultante intègre
40 *          une ou plusieurs contraintes: ex emdomagement, rupture
41 *          plissement ...
42 *          La loi contient a minima une loi classique interne.
43 *
44 *   *****
45 *   VERIFICATION:
46 *
47 *   ! date ! auteur ! but
48 *   -----
49 *   !     !     !
50 *
51 *   *****
52 *   MODIFICATIONS:
53 *   ! date ! auteur ! but
54 *   -----
55 *
56 *   *****/
57
58 // FICHER : LoiCritere.h
59 // CLASSE : LoiCritere
60
61 #ifndef LOICRITERE_H
62 #define LOICRITERE_H
63
64 #include "Loi_comp_abstraite.h"
65 #include "Enum_Critere_Loi.h"
66 #include "LoiContraintesPlanes.h"
67 #include "LoiContraintesPlanesDouble.h"
68 #include "Coordonnee2.h"
69 #include "Ponderation.h"
70
71
72 /// @addtogroup Les_lois_combinees
73 /// @
74 ///
75
76 class LoiCritere : public Loi_comp_abstraite
77 {
78     public :
79     friend class LoiContraintesPlanes;
80     friend class LoiContraintesPlanesDouble;
81
82     // CONSTRUCTEURS :
83
84     // Constructeur par défaut
85     LoiCritere ();
86
87     // Constructeur de copie
88     LoiCritere (const LoiCritere& loi) ;
89
90     // DESTRUCTEUR :
91     ~LoiCritere ();
92
93
94     // initialise les donnees particulieres a l'elements
95     // de matiere traite ( c-a-dire au pt calcule)
96     // Il y a creation d'une instance de SaveResul particuliere
97     // a la loi concerne
98     // la SaveResul classe est remplie par les instances heritantes
99     // le pointeur de SaveResul est sauvegarde au niveau de l'element
100     // c'a-d que les info particulieres au point considere sont stocke
101     // au niveau de l'element et non de la loi.
102     class SaveResul_LoiCritere: public SaveResul

```

```

103     { public :
104         SaveResul_LoiCritere(); // constructeur par défaut (a ne pas utiliser)
105         // le constructeur courant
106         SaveResul_LoiCritere(list <SaveResul*>& l_des_SaveResul,list <TenseurHH* >& l_siHH
107             ,list <TenseurHH* >& l_siHH_t
108             ,list <EnergieMeca >& l_energ,list <EnergieMeca >& l_energ_t
109             ,bool avec_ponderation,Enum_Critere_Loi type_crite);
110         // constructeur de copie
111         SaveResul_LoiCritere(const SaveResul_LoiCritere& sav );
112         // destructeur
113         ~SaveResul_LoiCritere();
114         // définition d'une nouvelle instance identique
115         // appelle du constructeur via new
116         SaveResul * Nevez_SaveResul() const {return (new SaveResul_LoiCritere(*this));};
117         // affectation
118         virtual SaveResul & operator = ( const SaveResul & a);
119         //===== lecture écriture dans base info =====
120         // cas donne le niveau de la récupération
121         // = 1 : on récupère tout
122         // = 2 : on récupère uniquement les données variables (supposées comme telles)
123         void Lecture_base_info (ifstream& ent,const int cas);
124         // cas donne le niveau de sauvegarde
125         // = 1 : on sauvegarde tout
126         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
127         void Ecriture_base_info(ofstream& sort,const int cas);
128
129         // mise à jour des informations transitoires en définitif s'il y a convergence
130         // par exemple (pour la plasticité par exemple)
131         void TdtversT() ;
132         void TversTdt() ;
133
134         // affichage à l'écran des infos
135         void Affiche() const;
136
137         //changement de base de toutes les grandeurs internes tensorielles stockées
138         // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gpB
139         // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
140         // gpH(i) = gamma(i,j) * gH(j)
141         virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
142
143         // procedure permettant de completer éventuellement les données particulières
144         // de la loi stockées
145         // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
146         // completer est appelé apres sa creation avec les donnees du bloc transmis
147         // peut etre appeler plusieurs fois
148         SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
149             ,const Loi_comp_abstraite* loi);
150
151         // ---- récupération d'information: spécifique à certaine classe dérivée
152         double Deformation_plastique() ;
153
154         // données protégées
155         // la liste des données protégées de chaque loi
156         list <SaveResul*> liste_des_SaveResul;
157         // la liste des contraintes initiales particulières pour chaque loi
158         list <TenseurHH* > l_sigoHH,l_sigoHH_t; // valeur courante, et valeur sauvegardée au pas
précédent
159         // la liste des énergies pour chaque loi
160         list <EnergieMeca > l_energ,l_energ_t; // valeur courante, et valeur sauvegardée au pas
précédent
161         // listes éventuelles des fonctions de pondération
162         list <double> f_ponder,f_ponder_t; // le résultat des fonctions de pondérations
163
164         // --- pour les plis ---
165         // éventuellement les directions principales des contraintes qui sont exprimées dans
166         // la base orthonormee ! donc qui n'évoluent pas avec la méthode: ChBase_des_grandeurs
167         Tableau <Coordonnee>* V_P_sig;
168         Tableau <Coordonnee>* V_P_sig_t;
169         Coordonnee2 eps_pli,eps_pli_t; // intensité des def de plis qui sont nulles si pas de pli,
c-a-d
170         // plis de membrane
171         // pour 1 plis: eps_pli(1) = eps22 en orthonormee
172         // pour 2 plis: eps_pli(2) = eps11 en orthonormee
173         // plis de biel
174         // eps_pli(1) = eps11 en orthonormee
175         // pour le critère pli, un pointeur sur une grandeur de travail
176         LoiContraintesPlanesDouble::SaveResul_LoiContraintesPlanesDouble * save_result_1DCP2;
177
178         Enum_Critere_Loi le_type_critere; // le type de critère de la loi
179         double niveau_declenchement_critere; // le niveau de déclenchement du critère
180
181         // -- indicateur de calcul de direction de plis --
182         // = 0 : pas encore de calcul effectué
183         // = -1 : pas de calcul de valeur propre possible en contrainte
184         // = 1 : pas de plis (pas de calcul de nouvelle direction )
185         // = -2 : pas de calcul de valeur propre de déformation
186         // = -3 : plis dans les deux sens, mais pas de calcul de direction propre valide

```

```

187 // = 2 : plis dans les deux sens, calcul des directions de plis
188 // = -4 : pas de calcul de vecteurs propres possible pour les contraintes
189 // = 3 : plis dans une seule directions, calcul ok
190 int cas_cal_plis,cas_cal_plis_t;
191
192 // --- gestion d'une map de grandeurs quelconques éventuelles ---
193 // une map de grandeurs quelconques particulière qui peut servir aux classes appelantes
194 // il s'agit ici d'une map interne qui a priori ne doit servir qu'aux class loi de comportement
195 // un exemple d'utilisation est une loi combinée qui a besoin de grandeurs spéciales définies
196 // -> n'est pas sauvegardé, car a priori il s'agit de grandeurs redondantes
197 map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >
map_type_quelconque;
198
199 // récupération des type quelconque sous forme d'un arbre pour faciliter la recherche
200 const map < EnumTypeQuelconque , TypeQuelconque, std::less < EnumTypeQuelconque> >*
Map_type_quelconque()
201     const {return &map_type_quelconque;};
202 private:
203     void Mise_a_jour_map_type_quelconque();
204
205 // ---- fin gestion d'une liste de grandeurs quelconques éventuelles ---
206
207
208 };
209
210 // def d'une instance de données spécifiques, et initialisation
211 SaveResul * New_et_Initialise() ;
212
213 friend class SaveResul_LoiCritere;
214
215 // Lecture des lois de comportement
216 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
217     ,LesFonctions_nD& lesFonctionsnD);
218
219 // affichage de la loi
220 void Affiche() const ;
221 // test si la loi est complete
222 // = 1 tout est ok, =0 loi incomplete
223 int TestComplet();
224
225 // calcul d'un module d'young équivalent à la loi, ceci pour un
226 // chargement nul
227 double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
228 // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
229 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
230 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
231     saveResul);
232
233 // récupération de la variation relative d'épaisseur calculée: h/h0
234 // cette variation n'est utile que pour des lois en contraintes planes
235 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
236 // - pour les lois 2D def planes: retour de 0
237 // les infos nécessaires à la récupération , sont stockées dans saveResul
238 // qui est le conteneur spécifique au point où a été calculé la loi
239 virtual double HsurH0(SaveResul * saveResul) const;
240
241 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
242 // exemple: mise en service des ddl de température aux noeuds
243 virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud,bool
244     dilatation,LesPtIntegMecaInterne& lesPtMecaInt);
245 // récupération des grandeurs particulière (hors ddl )
246 // correspondant à liTQ
247 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
248 virtual void Grandeur_particuliere
249     (bool absolue,List_io<TypeQuelconque>& liTQ,Loi_comp_abstraite::SaveResul * saveDon,list<int>&)
250     const;
251 // récupération de la liste de tous les grandeurs particulières
252 // ces grandeurs sont ajoutées à la liste passées en paramètres
253 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
254 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& liTQ) const;
255
256 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
257 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new LoiCritere(*this)); };
258
259 // indique le type Enum_comp_3D_CP_DP_1D correspondant à une loi de comportement
260 // la fonction est simple dans le cas d'une loi basique, par contre dans le cas
261 // d'une loi combinée, la réponse dépend des lois internes donc c'est redéfini
262 // dans les classes dérivées
263 virtual Enum_comp_3D_CP_DP_1D Comportement_3D_CP_DP_1D();
264
265 //----- lecture écriture de restart -----
266 // cas donne le niveau de la récupération
267 // = 1 : on récupère tout
268 // = 2 : on récupère uniquement les données variables (supposées comme telles)
269 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
270     lesCourbes1D
271     ,LesFonctions_nD& lesFonctionsnD);

```

```

268 // cas donne le niveau de sauvegarde
269 // = 1 : on sauvegarde tout
270 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
271 void Ecriture_base_info_loi(ofstream& sort,const int cas);
272
273 // affichage et definition interactive des commandes particulières à chaque lois
274 void Info_commande_LoisDeComp(UtilLecture& lec);
275
276 // activation du stockage de grandeurs quelconques qui pourront ensuite être récupéré
277 // via le conteneur SaveResul, si la grandeur n'existe pas ici, aucune action
278 virtual void Activation_stockage_grandeurs_quelconques(list <EnumTypeQuelconque >& listEnuQuelc);
279
280 // insertion des conteneurs ad hoc concernant le stockage de grandeurs quelconques
281 // passée en paramètre, dans le save result: ces conteneurs doivent être valides
282 // c-a-d faire partie de listdeTouslesQuelc_dispo_localement
283 virtual void Insertion_conteneur_dans_save_result(SaveResul * saveResul);
284
285 protected :
286
287 // def des différents critères
288 Enum_Critere_Loi type_critere; // le type de critère de la loi
289
290 // ----- controle de la sortie des informations
291 // -> maintenant définit dans LoiAbstraiteGeneral
292 // int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les erreurs
    et warning
293 // choix entre la première ou deuxième méthode pour le calcul des plis en membrane
294 int choix_methode_cal_plis_memb;
295
296 Tableau<int> ordre_criteres; // donne l'ordre d'application des différents critères
297 //%%% liste des paramètres associés à chaque critère %%%
298 //-----
299 // --- 1a) critère de plissement de membrane:
300 LoiContraintesPlanes * loi_2DCP_de_3D; // la loi de contrainte plane qui sert de support
301 // pour le plissement: il s'agit ici d'un pointeur sur la première loi de
    lois_interne
302 LoiContraintesPlanesDouble * loi_1DCP2_de_3D; // la loi de contrainte plane double qui sert de
    support
303 // il s'agit ici d'une nouvelle loi créée en interne qui sert pour ses méthodes
    internes
304 double niveau_declenchement_critere; // le niveau de déclenchement du critère
305 // contrôle éventuel du niveau d'apparition des plis
306 bool avecNiveauSigmaI_mini_pour_plis; // oui ou non on a des fonctions
307 // a) contrôle via une ou plusieurs grandeurs globales ou locales
308 Ponderation_TypeQuelconque* niveauF_fct_nD;
309 // b) via éventuellement un ddl étendu
310 Ponderation * niveauF_ddlEtendu;
311 // c) via éventuellement le temps
312 Courbe1D * niveauF_temps;
313 // d) contrôle via une ou plusieurs grandeurs consultables
314 Ponderation_Consultable* niveauF_grandeurConsultable;
315 // un indicateur pour gérer le cas relâchement complet pour le critère plis
316 // par défaut = 1 : on utilise l'épaisseur initiale -> vrai pour un comportement réversible sinon pas
    vrai !
317 // sera amélioré par la suite
318 int choix_calcul_epaisseur_si_relâchement_complet;
319 // gestion du recalcul des directions des plis
320 Fonction_nD* recalcul_dir_plis; // si nulle ==> recalcul tous les itérations
321
322 // --- 1b) critère de plissement 1D:
323 // la loi qui sert de support est la première loi la liste des lois_interne
324 // pour le plissement
325 // sinon on utilise les paramètres du la c-a-d des membranes pour le contrôle
326 //-----
327 // --- 2) critère de rupture en def ou contrainte unilatérale :
328 LoiContraintesPlanes * loi_2DCP_pour_rupture; // la loi de contrainte plane qui sert de support
329 // il s'agit ici d'une nouvelle loi créée en interne qui sert pour ses méthodes
    internes
330 LoiContraintesPlanesDouble * loi_1DCP2_pour_rupture; // la loi de contrainte plane double qui sert de
    support
331 // il s'agit ici d'une nouvelle loi créée en interne qui sert pour ses méthodes
    internes
332
333
334 // -- partie optionnelle au cas d'une somme pondérée par rapport à des
335 // grandeurs globales
336 bool avec_ponderation_grandeur_globale; // indique si oui ou non il y a des fonctions
337 // pour chaque loi il y a un élément Ponderation_GGglobal associé, qui contient lui-même m
    fonctions 1D dont le produit
338 // = la fonction finale de ponderation de la loi
339 list <Ponderation_GGglobal > list_ponderation_GGglob; // liste éventuellement vide des fonctions
    de ponderation_GGglob
340 list <double> fonc_ponder_GGglob; // le résultat des fonctions de pondérations
341
342
343 //-----
    // déclaration des variables internes nécessaires pour les passages 2D ou 3D --> 1D

```

```

344 // -- on définit des conteneurs pour le stockage des résultats des métriques, dimensionnés par défaut
      non vide
345 // on utilise des pointeurs pour dimensionner après les variables internes
346 Met_abstraite::Expli_t_tdt* expli_1D;
347 Met_abstraite::Impli* impli_1D;
348 Met_abstraite::Umat_cont* umat_cont_1D;
349
350 // -- variables nécessaires pour la création de expli_1D, impli_1D et umat_cont_1D
351 // certaines grandeurs sont associées à un pointeur qui peut soit être nul soit pointer sur le
      conteneur
352 // l'intérêt est que le fait d'avoir un pointeur nul est parfois utilisé pour éviter un calcul
353 BaseB giB_0_1D;
354 BaseH giH_0_1D;
355 BaseB giB_t_1D;
356 BaseH giH_t_1D;
357 BaseB giB_tdt_1D;
358 BaseH giH_tdt_1D;
359 Tenseur1BB gijBB_0_1D;
360 Tenseur1HH gijHH_0_1D;
361 Tenseur1BB gijBB_t_1D;
362 Tenseur1HH gijHH_t_1D;
363 Tenseur1BB gijBB_tdt_1D;
364 Tenseur1HH gijHH_tdt_1D;
365
366 TenseurBB * gradVmoyBB_t_1D_P;      Tenseur1BB gradVmoyBB_t_1D;
367 TenseurBB * gradVmoyBB_tdt_1D_P;    Tenseur1BB gradVmoyBB_tdt_1D;
368 TenseurBB * gradVBB_tdt_1D_P;      Tenseur1BB gradVBB_tdt_1D;
369 double jacobien_tdt_1D;double jacobien_t_1D;double jacobien_0_1D; // pour les jacobiens on considère
      qu'ils existent toujours
370 Vecteur d_jacobien_tdt_1D;
371 // pour tous les tableaux de pointeurs, on double le tableau en déclarant un vrai tableau en //
372 Tableau <BaseB> d_giB_tdt_1D;
373 Tableau <BaseH> d_giH_tdt_1D;
374 Tableau <TenseurBB * > d_gijBB_tdt_1D_P;      Tableau <Tenseur1BB > d_gijBB_tdt_1D;
375 Tableau2 <TenseurBB *> d2_gijBB_tdt_1D_P;    Tableau2 <Tenseur1BB > d2_gijBB_tdt_1D; // a priori
      ne sera pas affecté, car ne sert
376
377 lois de comportement
378 Tableau <TenseurHH * > d_gijHH_tdt_1D_P;      Tableau <Tenseur1HH > d_gijHH_tdt_1D;
379 Tableau <TenseurBB * >* d_gradVmoyBB_t_1D_P;  Tableau <Tenseur1BB > d_gradVmoyBB_t_1D;
380 Tableau <TenseurBB * >* d_gradVmoyBB_tdt_1D_P; Tableau <Tenseur1BB > d_gradVmoyBB_tdt_1D;
381 Tableau <TenseurBB * >* d_gradVBB_t_1D_P;      Tableau <Tenseur1BB > d_gradVBB_t_1D;
382 Tableau <TenseurBB * >* d_gradVBB_tdt_1D_P;    Tableau <Tenseur1BB > d_gradVBB_tdt_1D;
383
384 // -- on définit les conteneurs pour les passages d'appels entrant de la loi 1D : donc en 1D par
      défaut
385 Tenseur1HH sig_HH_t_1D, sig_HH_1D ;
386 Tenseur1BB Deps_BB_1D, eps_BB_1D, delta_eps_BB_1D;
387 Tableau <TenseurBB * > d_eps_BB_1D_P;      Tableau <Tenseur1BB > d_eps_BB_1D; // le tableau de
      pointeur puis les vrais grandeurs
388 Tableau <TenseurHH * > d_sig_HH_1D_P;      Tableau <Tenseur1HH > d_sig_HH_1D; // """"
389 TenseurHHHH* d_sigma_deps_1D_P;          Tenseur1HHHH d_sigma_deps_1D;
390
391 // -- on définit également certains conteneurs pour les passages d'appels entrant de la loi 2D :
      donc en 2D par défaut
392 Tenseur2BB eps_BB_2D_t,delta_eps_BB_2D;
393 Tenseur2HH eps_HH_2D_t; // tenseur de travail
394 // deux bases de travail qui servent dans Deuxieme_type_calcul_en_un_pli
395 BaseB ViB;
396 BaseH ViH;
397 // -- on définit les conteneurs pour les passages d'appels entrant de la loi 3D : donc en 3D par
      défaut
398 Tenseur3HH sig_HH_t_3D, sig_HH_3D ;
399 Tenseur3BB Deps_BB_3D, eps_BB_3D, delta_eps_BB_3D;
400
401 // cas d'un point d'intégration locale (méthode CalculGrandeurTravail par exemple)
402 PtIntegMecaInterne ptintmecca;
403 //----- la suite du type des lois membres -----
404
405 // un type énuméré pour faciliter la lecture
406 enum Enumcompletudecalcul { CONTRAINTE_ET_TANGENT =0, CONTRAINTE_UNIQUEMENT, TANGENT_UNIQUEMENT};
407 // donnees protegees
408 list <Loi_comp_abstraite * > lois_internes; // liste des lois constitutives
409 list <Enumcompletudecalcul> list_completude_calcul; // pour savoir si on utilise tout ou une
      partie
410
411 int type_calcul; // indique si l'on travail sur la contrainte ou l'incrément de contrainte
412 //-- partie optionnelle au cas d'une somme pondérée
413 bool avec_ponderation; // indique si oui ou non il y a des fonctions de ponderation
414 // pour chaque loi il y a un élément Ponderation associé, qui contient lui-même m fonctions 1D
      dont le produit
415 // = la fonction finale de ponderation de la loi
416 list <Ponderation > list_ponderation; // liste éventuellement vide des fonctions de ponderation
417 list <double> fonc_ponder; // le résultat des fonctions de pondérations
418
419 // ---- tableau de travail

```



```

420     Tableau <TenseurHH *> d_sigtotalHH;
421     // tenseur du 4ième ordre de travail
422     TenseurHHHH* d_sigma_deps_inter;
423     Tenseur3HHHH d_sig_deps_3D_HHHH;
424
425     // codage des METHODES VIRTUELLES protegees:
426     // calcul des contraintes a t+dt
427     // calcul des contraintes
428 void Calcul_SigmaHH (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
429     , TenseurBB & gijBB_t, TenseurHH & gijHH_t, BaseB & giB, BaseH & gi_H, TenseurBB & epsBB_
430     , TenseurBB & delta_epsBB_
431     , TenseurBB & gijBB_, TenseurHH & gijHH_, Tableau <TenseurBB *> & d_gijBB_
432     , double& jacobien_0, double& jacobien, TenseurHH & sigHH
433     , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
434     module_cisaillement
435     , const Met_abstraite::Expli_t_tdt& ex);
436
437     // calcul des contraintes et de ses variations a t+dt
438 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t, TenseurBB& DepsBB, DdlElement & tab_ddl
439     , BaseB & giB_t, TenseurBB & gijBB_t, TenseurHH & gijHH_t
440     , BaseB & giB_tdt, Tableau <BaseB> & d_giB_tdt, BaseH & giH_tdt, Tableau <BaseH> & d_giH_tdt
441     , TenseurBB & epsBB_tdt, Tableau <TenseurBB *> & d_epsBB
442     , TenseurBB & delta_epsBB, TenseurBB & gijBB_tdt, TenseurHH & gijHH_tdt
443     , Tableau <TenseurBB *> & d_gijBB_tdt
444     , Tableau <TenseurHH *> & d_gijHH_tdt, double& jacobien_0, double& jacobien
445     , Vecteur& d_jacobien_tdt, TenseurHH& sigHH, Tableau <TenseurHH *> & d_sigHH
446     , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
447     module_cisaillement
448     , const Met_abstraite::Impli & ex);
449
450     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
451     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
452     // le tenseur de déformation et son incrémentsont également en orthonormees
453     // si = false: les bases transmises sont utilisées
454     // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
455 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
456     , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
457     , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps
458     , EnergieMeca & energ, const EnergieMeca & energ_t, double& module_compressibilite, double&
459     module_cisaillement
460     , const Met_abstraite::Umat_cont& ex) ; // = 0;
461
462 // fonction surchargée dans les classes dérivée si besoin est
463 virtual void CalculGrandeurTravail
464     (const PtIntegMecaInterne& ptintmeca
465     , const Deformation & def, Enum_dure temps, const ThermoDonnee& dTP
466     , const Met_abstraite::Impli* ex_impli
467     , const Met_abstraite::Expli_t_tdt* ex_expli_tdt
468     , const Met_abstraite::Umat_cont* ex_umat
469     , const List_io<Ddl_etendu>* exclure_dd_etend
470     , const List_io<const TypeQuelconque *>* exclure_Q
471     );
472 // permet d'indiquer à la classe à quelle valeur de PtIntegMecaInterne il faut se référer
473 // en particulier est utilisé par les lois additives,
474 // par contre doit être utilisé avec prudence
475 virtual void IndiquePtIntegMecaInterne(const PtIntegMecaInterne * ptintmeca)
476 { list <Loi_comp_abstraite *>::iterator il, ilfin= lois_internes.end();
477   for ( il = lois_internes.begin(); il != ilfin; il++)
478     (*il)->IndiquePtIntegMecaInterne(ptintmeca);
479   // puis la classe mère
480   Loi_comp_abstraite::IndiquePtIntegMecaInterne(ptintmeca);
481 };
482 // fonction interne utilisée par les classes dérivées de Loi_comp_abstraite
483 // pour répercuter les modifications de la température
484 // ici utiliser pour modifier la température des lois élémentaires
485 // l'Enum_dure: indique quel est la température courante : 0 t ou tdt
486 void RepercuteChangeTemperature(Enum_dure temps);
487
488 // application d'un critère
489 // en retour:
490 // 0 : il y a eu un pb que l'on n'a pas pu résoudre, rien n'a été modifié
491 // 1 : le critère n'a rien modifié
492 // 2 : l'application du critère conduit à une contrainte et un opérateur tangent nul
493 // 3 : les données d'entrée ont été modifiées: contraintes, opérateur tangent, module, énergies
494 int Critere(bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
495     , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
496     , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps_inter
497     , EnergieMeca & energ, const EnergieMeca & energ_t, double&
498     module_compressibilite, double& module_cisaillement
499     , bool implicit, const Met_abstraite::Umat_cont& ex) ;
500 // fonction critère de plissement de membrane
501 // en entrée:
502 // implicit : si oui on est en implicite, sinon en explicite

```

```

503 // retour:
504 // = -1 : pas de calcul de valeur propre possible en contrainte
505 // = 1 : pas de plis (pas de calcul de nouvelle direction )
506 // = -2 : pas de calcul de valeur propre de déformation
507 // = -3 : plis dans les deux sens, mais pas de calcul de direction propre valide
508 // = 2 : plis dans les deux sens, calcul des directions de plis
509 // = -4 : pas de calcul de vecteurs propres possible pour les contraintes
510 // = 3 : plis dans une seule directions, calcul ok
511
512 int Critere_plis_membrane(bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
513 , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
514 , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps_inter
515 , EnergieMeca & energ, const EnergieMeca & energ_t, double&
516 module_compressibilite, double& module_cisaillement
517 , bool implicit, const Met_abstraite::Umat_cont& ex);
518
519 // fonction critère de plissement de biel
520 // en entrée:
521 // implicite : si oui on est en implicite, sinon en explicite
522 // retour:
523 // 0 : il y a eu un pb que l'on n'a pas pu résoudre, rien n'a été modifié
524 // 1 : le critère n'a rien modifié
525 // 2 : l'application du critère conduit à une contrainte et un opérateur tangent nul
526 int Critere_plis_biel(bool en_base_orthonormee, TenseurHH & sigHH_t, TenseurBB& DepsBB
527 , TenseurBB & epsBB_tdt, TenseurBB & delta_epsBB, double& jacobien_0, double& jacobien
528 , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps_inter
529 , EnergieMeca & energ, const EnergieMeca & energ_t, double&
530 module_compressibilite, double& module_cisaillement
531 , bool implicit, const Met_abstraite::Umat_cont& ex);
532
533 //--- méthodes internes
534 // création du conteneur UMAT a partir des vecteurs propres
535 void Creation_metrique_a_partir_vecteurs_propres_pour_Umat1D
536 (const Met_abstraite::Umat_cont& ex, const Mat_pleine& gamma);
537 // passage des grandeurs métriques de l'ordre 3 ou 2 à 1: cas implicite
538 void Passage_metrique_ordre_3_2_vers_1(const Met_abstraite::Umat_cont& ex, const Mat_pleine& gamma);
539 // passage des grandeurs métriques de l'ordre 3 ou 2 à 1: cas explicite
540 void Passage_metrique_ordre_3_2_vers_1(const Met_abstraite::Umat_cont& ex, Mat_pleine& gamma);
541
542 // chgt de repère et de dimension pour les variables de passage pour le calcul final de la loi en 1D
543 void Passage_3ou2_vers_1(const Mat_pleine& gamma, const TenseurHH & sigHH_t, const Mat_pleine& beta
544 , const TenseurBB& DepsBB
545 , const TenseurBB & epsBB_tdt, const TenseurBB & delta_epsBB
546 , const bool& deux_plis, Coordonnee2& eps_pli);
547 // passage inverse: chgt de repère et de dimension pour les variables de passage
548 // et stockage des infos pour le prochain appel
549 // en entrée: d_sigma_deps_1D : l'opérateur qui a été calculé
550 // en sortie: d_sigma_deps_inter
551 // l'umat c'est uniquement pour des vérifs
552 void Passage_1_vers_3ou2(const Mat_pleine& gamma, TenseurHH & sigHH
553 , const TenseurHHHH& d_sigma_deps_1D
554 , const Mat_pleine& beta
555 , TenseurHHHH& d_sigma_deps_inter, const Met_abstraite::Umat_cont& ex
556 , const Tableau <Coordonnee2H>& V_Pr_H
557 , const Tableau <Coordonnee>& V_P_sig);
558
559 // calcul d'une nouvelle direction de plis
560 // en entrée: force = true par défaut
561 // si == false -> pas de calcul de direction, et mise en place
562 // des indicateurs en cohérence avec le fait de ne pas faire de calcul
563 // en retour:
564 // = -1 : pas de calcul de valeur propre possible en contrainte
565 // = 1 : pas de plis (pas de calcul de nouvelle direction )
566 // = -2 : pas de calcul de valeur propre de déformation
567 // = -3 : plis dans les deux sens, mais pas de calcul de direction propre valide
568 // = 2 : plis dans les deux sens, calcul des directions de plis
569 // = -4 : pas de calcul de vecteurs propres possible pour les contraintes
570 // = 3 : plis dans une seule directions, calcul ok
571 // = 0 : erreur inconnue ??
572
573 int Calcul_directions_plis(const TenseurBB & epsBB_tdt, const TenseurHH& sigHH
574 , Coordonnee2& valPropreEps
575 , Tableau <Coordonnee2H>& V_Pr_H
576 , const Met_abstraite::Umat_cont& ex
577 , Coordonnee2& valPropreSig
578 , bool force = true
579 );
580
581 // premier type de calcul dans le cas d'un pli dans une seule direction
582 void Premie_type_calcul_en_un_pli(const TenseurBB & epsBB_tdt, const TenseurBB & delta_epsBB
583 , const TenseurHH & sigHH_t
584 , const double& jacobien_0, const double& jacobien
585 , EnergieMeca & energ, const EnergieMeca & energ_t
586 , const TenseurBB& DepsBB
587 , double& module_compressibilite, double& module_cisaillement
588 , const Tableau <Coordonnee2H>& V_Pr_H
589 , TenseurHH& sigHH, TenseurHHHH& d_sigma_deps_inter

```

```

588             ,const Coordonnee2& valPropreSig
589             ,const Met_abstraite::Umat_cont& ex);
590
591 // premier type de calcul dans le cas d'un pli dans une seule direction
592 void Deuxieme_type_calcul_en_un_pli(const TenseurBB & epsBB_tdt,const TenseurBB & delta_epsBB
593             ,const TenseurHH & sigHH_t
594             ,double& jacobien_0,double& jacobien
595             ,EnergieMeca & energ,const EnergieMeca & energ_t
596             ,const TenseurBB& DepsBB
597             ,double& module_compressibilite,double& module_cisaillement
598             ,const Tableau <Coordonnee2H>& V_Pr_H
599             ,TenseurHH& sigHH,TenseurHHH& d_sigma_deps_inter
600             ,const Coordonnee2& valPropreSig
601             ,const Met_abstraite::Umat_cont& ex);
602
603 // préparation à l'appel du comportement
604 //Par exemple dans le cas d'un critère pli (plutôt seconde méthode), l'incrément de déformation
605 // dépend de la situation précédente: avec pli ou pas
606 int Pre_Critere
607     (const TenseurBB& DepsBB,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double&
608     jacobien_0,double& jacobien,const Met_abstraite::Expli_t_tdt* ex_expli,const Met_abstraite::Impli*
609     ex_impli,const Met_abstraite::Umat_cont* ex_umat
610     );
611 // fonction pre_critère de plissement de membrane
612 void Pre_Critere_plis_membrane
613     (TenseurBB & epsBB_tdt_,TenseurBB & delta_epsBB
614     ,const Met_abstraite::Expli_t_tdt* ex_expli
615     ,const Met_abstraite::Impli* ex_impli
616     ,const Met_abstraite::Umat_cont* ex_umat
617     );
618 // passage des informations liées à la déformation et contrainte de 2 vers 3
619 void Passage_deformation_contrainte_ordre2_vers_3
620     (const TenseurBB& DepsBB,const TenseurBB & epsBB_tdt
621     ,const TenseurBB & delta_epsBB,const TenseurHH& sig_HH_t);
622 // passage des informations liées à la nouvelle contrainte de 2 vers 3
623 // et à l'opérateur tangent : méthode 2
624 // et mise à jour d'eps_pli
625 void Passage_contrainte_et_operateur_tangent_ordre2_vers_3
626     (TenseurHH& sig_HH_tdt,TenseurHHH& d_sigma_deps_inter
627     ,const bool& deux_plis,Coordonnee2& eps_pli);
628
629 };
630 /// @} // end of group
631
632 #endif
633
634
635

```

## 7.74 LoiDeformationsPlanes.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:      8/02/2012      *

```

```

33 *
34 * AUTEUR:      G RIO      (mailto:gerardrio56@free.fr)      $      *
35 *
36 * PROJET:      Herezh++      $      *
37 *
38 *****
39 * BUT: La loi est 2D_D et est associée à une loi 3D quelconque *
40 * L'objectif est de transformer une loi 3D en 2D déformations*
41 * planes.
42 *
43 * *****
44 * VERIFICATION:
45 *
46 * ! date ! auteur ! but !
47 * -----
48 * ! ! ! !
49 *
50 * *****
51 * MODIFICATIONS:
52 * ! date ! auteur ! but !
53 * -----
54 *
55 *****/
56
57 // FICHER : LoiDeformationsPlanes.h
58 // CLASSE : LoiDeformationsPlanes
59
60 #ifndef LOIDEFORMATIONSPLANES_H
61 #define LOIDEFORMATIONSPLANES_H
62
63 #include "Loi_comp_abstraite.h"
64
65
66 /// @addtogroup Les_lois_combinees
67 /// @{
68 ///
69
70
71 class LoiDeformationsPlanes : public Loi_comp_abstraite
72 {
73     public :
74
75         // CONSTRUCTEURS :
76
77         // Constructeur par défaut
78         LoiDeformationsPlanes ();
79
80         // Constructeur de copie
81         LoiDeformationsPlanes (const LoiDeformationsPlanes& loi) ;
82
83         // DESTRUCTEUR :
84         ~LoiDeformationsPlanes ();
85
86
87
88         // initialise les donnees particulieres a l'elements
89         // de matiere traite ( c-a-dire au pt calcule)
90         // Il y a creation d'une instance de SaveResul particuliere
91         // a la loi concernee
92         // la SaveResul classe est remplie par les instances heritantes
93         // le pointeur de SaveResul est sauvegarde au niveau de l'element
94         // c'a-d que les info particulieres au point considere sont stocke
95         // au niveau de l'element et non de la loi.
96         class SaveResul_LoiDeformationsPlanes: public SaveResul
97         { public :
98             SaveResul_LoiDeformationsPlanes(); // constructeur par défaut (a ne pas utiliser)
99             // le constructeur courant
100            SaveResul_LoiDeformationsPlanes(SaveResul* l_des_SaveResul);
101            // constructeur de copie
102            SaveResul_LoiDeformationsPlanes(const SaveResul_LoiDeformationsPlanes& sav );
103            // destructeur
104            ~SaveResul_LoiDeformationsPlanes();
105            // définition d'une nouvelle instance identique
106            // appelle du constructeur via new
107            SaveResul * Nevez_SaveResul() const {return (new
108            SaveResul_LoiDeformationsPlanes(*this));};
109            // affectation
110            virtual SaveResul & operator = ( const SaveResul & a);
111            //===== lecture écriture dans base info =====
112            // cas donne le niveau de la récupération
113            // = 1 : on récupère tout
114            // = 2 : on récupère uniquement les données variables (supposées comme telles)
115            void Lecture_base_info (ifstream& ent,const int cas);
116            // cas donne le niveau de sauvegarde
117            // = 1 : on sauvegarde tout
118            // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

```

```

118         void Ecriture_base_info(ofstream& sort,const int cas);
119
120         // mise à jour des informations transitoires en définitif s'il y a convergence
121         // par exemple (pour la plasticité par exemple)
122         void TdtversT() ;
123         void TversTdt() ;
124
125         // affichage à l'écran des infos
126         void Affiche() const;
127
128         //changement de base de toutes les grandeurs internes tensorielles stockées
129         // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
130         // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
131         // gpH(i) = gamma(i,j) * gH(j)
132         virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
133
134         // procedure permettant de completer éventuellement les données particulières
135         // de la loi stockées
136         // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
137         // completer est appelé apres sa creation avec les donnees du bloc transmis
138         // peut etre appelle plusieurs fois
139         SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
140             ,const Loi_comp_abstraite* loi);
141
142         // ---- récupération d'information: spécifique à certaine classe dérivée
143         double Deformation_plastique() ;
144
145         // données protégées
146         // les données protégées de la loi
147         SaveResul* le_SaveResul;
148         // les contraintes qui servent d'entrée au calcul de la loi associée
149         TenseurHH* l_sigoHH, * l_sigoHH_t; // valeur courante, et valeur sauvegardée au pas
précédent
150         Vecteur sigInvar,sigInvar_t; // on sauvegarde les invariants ordre 3 à l'instat t
151         // les énergies pour la loi
152         EnergieMeca l_energ,l_energ_t; // valeur courante, et valeur sauvegardée au pas précédent
153     };
154
155     // def d'une instance de données spécifiques, et initialisation
156     SaveResul * New_et_Initialise() ;
157
158     friend class SaveResul_LoiDeformationsPlanes;
159
160     // Lecture des donnees de la classe sur fichier
161     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
162         ,LesFonctions_nD& lesFonctionsnD);
163
164     // affichage de la loi
165     void Affiche() const ;
166     // test si la loi est complete
167     // = 1 tout est ok, =0 loi incomplete
168     int TestCompleter();
169
170     // calcul d'un module d'young équivalent à la loi, ceci pour un
171     // chargement nul
172     double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
173     // récupération d'un module de compressibilité équivalent à la loi, ceci pour un chargement nul
174     // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
175     double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
176         saveResul);
177
178     // récupération de la variation relative d'épaisseur calculée: h/h0
179     // cette variation n'est utile que pour des lois en contraintes planes
180     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
181     // - pour les lois 2D def planes: retour de 0
182     // les infos nécessaires à la récupération , sont stockées dans saveResul
183     // qui est le conteneur spécifique au point où a été calculé la loi
184     virtual double HsurH0(SaveResul * saveResul) const {return 0.};
185
186     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
187     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new LoiDeformationsPlanes(*this)); };
188
189     // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
190     // exemple: mise en service des ddl de température aux noeuds
191     virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud,bool
192         dilatation,LesPtIntegMecaInterne& lesPtMecaInt);
193     // récupération des grandeurs particulière (hors ddl )
194     // correspondant à liTQ
195     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
196     virtual void Grandeur_particuliere
197         (bool absolue,List_io<TypeQuelconque>& liTQ,Loi_comp_abstraite::SaveResul * saveDon,list<int>&
198         decal) const;
199     // récupération de la liste de tous les grandeurs particulières
200     // ces grandeurs sont ajoutées à la liste passées en paramètres
201     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
202     virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& liTQ) const;
203
204

```

```

201 //----- lecture écriture de restart -----
202 // cas donne le niveau de la récupération
203 // = 1 : on récupère tout
204 // = 2 : on récupère uniquement les données variables (supposées comme telles)
205 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbesID&
    lesCourbesID
206                                     ,LesFonctions_nD& lesFonctionsnD);
207
208 // cas donne le niveau de sauvegarde
209 // = 1 : on sauvegarde tout
210 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
211 void Ecriture_base_info_loi(ofstream& sort,const int cas);
212
213 // affichage et definition interactive des commandes particulières à chaque lois
214 void Info_commande_LoisDeComp(UtilLecture& lec);
215
216 protected :
217 // donnees protegees
218 Loi_comp_abstraite * lois_interne; // loi 3D correspondante
219 // tenseur du 4ième orde de travail
220 TenseurHHHH* d_sigma_deps_inter;
221
222
223 // codage des METHODES VIRTUELLES protegees:
224 // calcul des contraintes a t+dt
225 // calcul des contraintes
226 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
227 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB & giB,BaseH & gi_H, TenseurBB & epsBB_
228 ,TenseurBB & delta_epsBB_
229 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
230 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
231 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
232 ,const Met_abstraite::Expli_t_tdt& ex);
233
234 // calcul des contraintes et de ses variations a t+dt
235 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
236 ,BaseB & giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
237 ,BaseB & giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH & giH_tdt,Tableau <BaseH> & d_giH_tdt
238 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
239 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
240 ,Tableau <TenseurBB *>& d_gijBB_tdt
241 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
242 ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
243 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
244 ,const Met_abstraite::Impli& ex);
245
246 // calcul des contraintes et ses variations par rapport aux déformations a t+dt
247 // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
248 // le tenseur de déformation et son incrémentsont également en orthonormees
249 // si = false: les bases transmises sont utilisées
250 // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
251 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
252 ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
253 ,TenseurHH& sigHH,TenseurHHHH& d_sigma_deps
254 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
255 ,const Met_abstraite::Umat_cont& ex) ; // = 0;
256
257
258 // fonction surchargée dans les classes dérivée si besoin est
259 virtual void CalculGrandeurTravail
260 (const PtIntegMecaInterne& ptintmeca
261 ,const Deformation & def,Enum_dure temps,const ThermoDonnee& dTP
262 ,const Met_abstraite::Impli* ex_impli
263 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
264 ,const Met_abstraite::Umat_cont* ex_umat
265 ,const List_io<Ddl_etendu>* exclure_dd_etend
266 ,const List_io<const TypeQuelconque *>* exclure_Q
267 );
268
269 // permet d'indiquer à la classe à quelle valeur de PtIntegMecaInterne il faut se référer
270 // en particulier est utilisé par les lois additives,
271 // par contre doit être utilisé avec prudence
272 virtual void IndiquePtIntegMecaInterne(const PtIntegMecaInterne * ptintmeca)
273 { lois_interne->IndiquePtIntegMecaInterne(ptintmeca);
274 // puis la classe mère
275 Loi_comp_abstraite::IndiquePtIntegMecaInterne(ptintmeca);
276 };
277
278
279 // fonction interne utilisée par les classes dérivées de Loi_comp_abstraite
280 // pour répercuter les modifications de la température
281 // ici utiliser pour modifier la température des lois élémentaires
282 // l'Enum_dure indique quel est la température courante : 0 t ou tdt
283 void RepercuteChangeTemperature(Enum_dure temps);

```

```

284
285 private :
286 // déclaration des variables internes nécessaires pour les passages 2D - 3D
287 // -- on définit des conteneurs pour le stockage des résultats des métriques, dimensionnés par défaut
    non vide
288 // on utilise des pointeurs pour dimensionner après les variables internes
289 Met_abstraite::Expli_t_tdt* expli_3D;
290 Met_abstraite::Impli* impli_3D;
291 Met_abstraite::Umat_cont* umat_cont_3D;
292
293 // -- variables nécessaires pour la création de expli_3D, impli_3D et umat_cont_3D
294 // certaines grandeurs sont associées à un pointeur qui peut soit être nulle soit pointer sur le
    conteneur
295 // l'intérêt est que le fait d'avoir un pointeur nul est parfois utilisé pour éviter un calcul
296 BaseB giB_0_3D;
297 BaseH giH_0_3D;
298 BaseB giB_t_3D;
299 BaseH giH_t_3D;
300 BaseB giB_tdt_3D;
301 BaseH giH_tdt_3D;
302 Tenseur3BB gijBB_0_3D;
303 Tenseur3HH gijHH_0_3D;
304 Tenseur3BB gijBB_t_3D;
305 Tenseur3HH gijHH_t_3D;
306 Tenseur3BB gijBB_tdt_3D;
307 Tenseur3HH gijHH_tdt_3D;
308
309 TenseurBB * gradVmoyBB_t_3D_P;      Tenseur_ns3BB gradVmoyBB_t_3D;
310 TenseurBB * gradVmoyBB_tdt_3D_P;   Tenseur_ns3BB gradVmoyBB_tdt_3D;
311 TenseurBB * gradVBB_tdt_3D_P;      Tenseur_ns3BB gradVBB_tdt_3D;
312 double jacobien_tdt_3D;double jacobien_t_3D;double jacobien_0_3D; // pour les jacobiens on considère
    qu'ils existent toujours
313 Vecteur d_jacobien_tdt_3D;
314 // pour tous les tableaux de pointeurs, on double le tableau en déclarant un vrai tableau en //
315 Tableau <BaseB> d_giB_tdt_3D;
316 Tableau <BaseH> d_giH_tdt_3D;
317 Tableau <TenseurBB * > d_gijBB_tdt_3D_P;      Tableau <Tenseur3BB > d_gijBB_tdt_3D;
318 Tableau2 <TenseurBB * >* d2_gijBB_tdt_3D_P;   Tableau2 <Tenseur3BB > d2_gijBB_tdt_3D; // a priori ne
    sera pas affecté, car ne sert
319 // dans les
    lois de comportement
320 Tableau <TenseurHH * > d_gijHH_tdt_3D_P;      Tableau <Tenseur3HH > d_gijHH_tdt_3D;
321 Tableau <TenseurBB * >* d_gradVmoyBB_t_3D_P;   Tableau <Tenseur_ns3BB > d_gradVmoyBB_t_3D;
322 Tableau <TenseurBB * >* d_gradVmoyBB_tdt_3D_P;   Tableau <Tenseur_ns3BB > d_gradVmoyBB_tdt_3D;
323 Tableau <TenseurBB * >* d_gradVBB_t_3D_P;      Tableau <Tenseur_ns3BB > d_gradVBB_t_3D;
324 Tableau <TenseurBB * >* d_gradVBB_tdt_3D_P;   Tableau <Tenseur_ns3BB > d_gradVBB_tdt_3D;
325
326 // -- on définit les conteneurs pour les passages d'appels entrant de la loi 3D : donc en 3D par
    défaut
327 Tenseur3HH sig_HH_t_3D, sig_HH_3D ;
328 Tenseur3BB Deps_BB_3D, eps_BB_3D, delta_eps_BB_3D;
329 Tableau <TenseurBB * > d_eps_BB_3D_P;      Tableau <Tenseur3BB > d_eps_BB_3D; // le tableau de
    pointeur puis les vrais grandeurs
330 Tableau <TenseurHH * > d_sig_HH_3D_P;      Tableau <Tenseur3HH > d_sig_HH_3D; // """"
331 Tenseur3HHHH d_sigma_deps_3D;
332
333 // cas d'un point d'intégration locale (méthode CalculGrandeurTravail par exemple)
334 PtIntegMecaInterne ptintmeca;
335
336 //--- méthodes internes
337 // passage des grandeurs métriques de l'ordre 2 à 3: cas implicite
338 void Passage_metrrique_ordre2_vers_3(const Met_abstraite::Impli& ex);
339 // passage des grandeurs métriques de l'ordre 2 à 3: cas explicite
340 void Passage_metrrique_ordre2_vers_3(const Met_abstraite::Expli_t_tdt& ex);
341 // passage des informations liées à la déformation de 2 vers 3, et variation de volume éventuelle
342 // si le pointeur d_jacobien_tdt est non nul
343 // idem pour d_epsBB
344 void Passage_deformation_volume_ordre2_vers_3(TenseurBB& DepsBB
    ,TenseurBB & epsBB_tdt,Tableau <TenseurBB * >* d_epsBB
    ,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
    ,Vecteur* d_jacobien_tdt);
345
346
347
348
349
350 };
351 /// @} // end of group
352
353 #endif
354
355
356

```

## 7.75 LoiDesMelangesEnSigma.h

1  
2

```

3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           10/04/2004
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 * *****/
39 *   BUT: Définir une loi telle que la contrainte résultante soit la
40 *         somme pondérée : 1-alpha, et alpha , de contraintes
41 *         élémentaires, eux-même définies à partir de lois quelconques.*
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   !-----!-----!-----!
49 *   !           !           !           !
50 *   *****
51 *   MODIFICATIONS:
52 *
53 *   ! date !   auteur !           but
54 *   !-----!-----!-----!
55 *   *****/
56
57 // FICHER : LoiDesMelangesEnSigma.h
58 // CLASSE : LoiDesMelangesEnSigma
59
60 #ifndef LOIDESMELANGESENSIGMA_H
61 #define LOIDESMELANGESENSIGMA_H
62
63 #include "Loi_comp_abstraite.h"
64 #include "Ponderation.h"
65
66
67 /// @addtogroup Les_lois_combinees
68 /// @
69 ///
70
71 class LoiDesMelangesEnSigma : public Loi_comp_abstraite
72 {
73     public :
74
75         // CONSTRUCTEURS :
76
77         // Constructeur par défaut
78         LoiDesMelangesEnSigma ();
79
80         // Constructeur de copie
81         LoiDesMelangesEnSigma (const LoiDesMelangesEnSigma& loi) ;
82
83         // DESTRUCTEUR :
84         ~LoiDesMelangesEnSigma ();
85
86
87
88         // initialise les donnees particulieres a l'elements

```



```

89 // de matiere traite ( c-a-dire au pt calcule)
90 // Il y a creation d'une instance de SaveResul particuliere
91 // a la loi concernee
92 // la SaveResul classe est remplie par les instances heritantes
93 // le pointeur de SaveResul est sauvegarde au niveau de l'element
94 // c'a-d que les info particulieres au point considere sont stocke
95 // au niveau de l'element et non de la loi.
96 class SaveResul_LoiDesMelangesEnSigma: public SaveResul
97 { public :
98     SaveResul_LoiDesMelangesEnSigma(); // constructeur par défaut (a ne pas utiliser)
99     // le constructeur courant
100     SaveResul_LoiDesMelangesEnSigma(list <SaveResul*>& l_des_SaveResul
101                                     ,list <TenseurHH* >& l_siHH,list <TenseurHH* >& l_siHH_t
102                                     ,list <TenseurHH* >& J_siHH,list <TenseurHH* >& J_siHH_t
103                                     ,list <EnergieMeca >& l_energ,list <EnergieMeca >& l_energ_t
104                                     ,int type_evol_proportion);
105     // constructeur de copie
106     SaveResul_LoiDesMelangesEnSigma(const SaveResul_LoiDesMelangesEnSigma& sav );
107     // destructeur
108     ~SaveResul_LoiDesMelangesEnSigma();
109     // définition d'une nouvelle instance identique
110     // appelle du constructeur via new
111     SaveResul * Nevez_SaveResul() const {return (new
SaveResul_LoiDesMelangesEnSigma(*this));};
112     // affectation
113     virtual SaveResul & operator = ( const SaveResul & a);
114     //===== lecture écriture dans base info =====
115     // cas donne le niveau de la récupération
116     // = 1 : on récupère tout
117     // = 2 : on récupère uniquement les données variables (supposées comme telles)
118     void Lecture_base_info (ifstream& ent,const int cas);
119     // cas donne le niveau de sauvegarde
120     // = 1 : on sauvegarde tout
121     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
122     void Ecriture_base_info(ofstream& sort,const int cas);
123
124     // mise à jour des informations transitoires en définitif s'il y a convergence
125     // par exemple (pour la plasticité par exemple)
126     void TdtversI() ;
127     void TversTdt () ;
128
129     // affichage à l'écran des infos
130     void Affiche() const;
131
132     //changement de base de toutes les grandeurs internes tensorielles stockées
133     // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
134     // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
135     // gpH(i) = gamma(i,j) * gH(j)
136     virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma);
137
138     // procedure permettant de completer éventuellement les données particulières
139     // de la loi stockées
140     // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
141     // completer est appelé apres sa creation avec les donnees du bloc transmis
142     // peut etre appeler plusieurs fois
143     SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
144                                   ,const Loi_comp_abstraite* loi);
145
146     // ---- récupération d'information: spécifique à certaine classe dérivée
147     double Deformation_plastique() ;
148
149     // données protégées
150     // la liste des données protégées de chaque loi
151     list <SaveResul*> liste_des_SaveResul;
152     // la liste des contraintes initiales particulières pour chaque loi
153     // 1) les contraintes qui servent d'entrée au calcul des lois élémentaires (non
proportionnées)
154     list <TenseurHH* > l_sigoHH,l_sigoHH_t; // valeur courante, et valeur sauvegardée au pas
précédent
155     // 2) les contraintes proportionnées
156     list <TenseurHH* > J_sigoHH,J_sigoHH_t; // valeur courante, et valeur sauvegardée au pas
précédent
157     // la liste des énergies pour chaque loi
158     list <EnergieMeca > l_energ,l_energ_t; // valeur courante, et valeur sauvegardée au pas
précédent
159     // proportion: qui est a priori variable au cours de l'évolution
160     double proportion,proportion_t;
161     int type_evolution_proportion; // = 0 : aucune particularité
162     // = 1 : au premier incrément la proportion
163     // démarre à 1 et ensuite évolue de manière strictement
164     // décroissante
165     // = 2 : pour chaque incrément
166     // à chaque mise à jour tdt_t ou l'inverse: la proportion
167     // démarre à 1 et ensuite évolue de manière strictement
168     // décroissante
169     bool deja_actif_sur_iter1,deja_actif_sur_iter2;
170 };

```

```

171
172 // def d'une instance de données spécifiques, et initialisation
173 SaveResul * New_et_Initialise() ;
174
175 friend class SaveResul_LoiDesMelangesEnSigma;
176
177 // Lecture des donnees de la classe sur fichier
178 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D & lesCourbes1D
179                                     ,LesFonctions_nD & lesFonctionsnD);
180
181 // affichage de la loi
182 void Affiche() const ;
183 // test si la loi est complete
184 // = 1 tout est ok, =0 loi incomplete
185 int TestComplet();
186
187 // calcul d'un module d'young equivalent à la loi, ceci pour un
188 // chargement nul
189 double Module_young_equivalent(Enum_dure temps,const Deformation & def,SaveResul * saveResul);
190 // récupération d'un module de compressibilité equivalent à la loi, ceci pour un chargement nul
191 // il s'agit ici de la relation -pression = sigma_trace/3. = module de compressibilité * I_eps
192 double Module_compressibilite_equivalent(Enum_dure temps,const Deformation & def,SaveResul *
193                                     saveResul);
194
195 // récupération de la variation relative d'épaisseur calculée: h/h0
196 // cette variation n'est utile que pour des lois en contraintes planes
197 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
198 // - pour les lois 2D def planes: retour de 0
199 // les infos nécessaires à la récupération , sont stockées dans saveResul
200 // qui est le conteneur spécifique au point où a été calculé la loi
201 virtual double HsurH0(SaveResul * saveResul) const
202 { cout << "\n LoiDesMelangesEnSigma::HsurH0(.. , methode non implante pour l'instant ";
203   Sortie(1);
204 };
205
206 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
207 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new LoiDesMelangesEnSigma(*this)); };
208
209 // activation des données des noeuds et/ou elements nécessaires au fonctionnement de la loi
210 // exemple: mise en service des ddl de température aux noeuds
211 virtual void Activation_donnees(Tableau<Noeud *>& tabnoeud,bool
212                               dilatation,LesPtIntegMecaInterne& lesPtMecaInt);
213 // récupération des grandeurs particulière (hors ddl )
214 // correspondant à liTQ
215 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
216 virtual void Grandeur_particuliere
217 (bool absolue,List_io<TypeQuelconque>& liTQ,Loi_comp_abstraite::SaveResul * saveDon,list<int>&
218  decal) const;
219 // récupération de la liste de tous les grandeurs particulières
220 // ces grandeurs sont ajoutées à la liste passées en paramètres
221 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
222 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& liTQ) const;
223
224 // indique le type Enum_comp_3D_CP_DP_1D correspondant à une loi de comportement
225 // la fonction est simple dans le cas d'une loi basique, par contre dans le cas
226 // d'une loi combinée, la réponse dépend des lois internes donc c'est redéfini
227 // dans les classes dérivées
228 virtual Enum_comp_3D_CP_DP_1D Comportement_3D_CP_DP_1D();
229
230 //----- lecture écriture de restart -----
231 // cas donne le niveau de la récupération
232 // = 1 : on récupère tout
233 // = 2 : on récupère uniquement les données variables (supposées comme telles)
234 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
235                             lesCourbes1D
236                                     ,LesFonctions_nD & lesFonctionsnD);
237
238 // cas donne le niveau de sauvegarde
239 // = 1 : on sauvegarde tout
240 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
241 void Ecriture_base_info_loi(ofstream& sort,const int cas);
242
243 // affichage et definition interactive des commandes particulières à chaque lois
244 void Info_commande_LoisDeComp(UtilLecture& lec);
245
246 protected :
247
248 // donnees protegees
249 Loi_comp_abstraite * lois_internes1; // liste des lois constitutives
250 Loi_comp_abstraite * lois_internes2; // liste des lois constitutives
251 // gestion du calcul conditionnel si prop non nulle
252 Tableau<int> calcule_si_prop_non_nulle; // = 0 : pas actif, =1 : calcul uniquement si non nul
253 // = 2 : calcul à partir du moment où il a été une fois non nul
254 // = 3 : calcul à partir du moment où il a été une fois non nul, avec ré-init à chaque
255 // incrément
256 // tableau de travail
257 Tableau <TenseurHH *> d_sigtotalHH;
258 // tenseur du 4ième orde de travail

```

```

253     TenseurHHHH* d_sigma_deps_inter;
254     // grandeur qui sert de proportion
255     Ddl_enum_etendu type_grandeur;
256     // indique si la grandeur est calculée à partir des noeuds (valeur par défaut)
257     // ou directement à partir d'une valeur obtenue au point d'intégration
258     bool valeur_aux_noeuds;
259     double proportion,aA; // aA est le maxi de la somme des deux lois
260     CourbeId* c_proport; // courbe éventuelle permettant le calcul de la proportion en fonction de
    la grandeur
261                                     // de pilotage de la proportion
262     // a) contrôle via une fct nd avec une ou plusieurs grandeurs globales et autres grandeurs
263     Fonction_nD* niveauF_grandeurND;
264     // b) via éventuellement un ddl étendu
265     Ponderation * niveauF_ddlEtendu;
266     // c) via éventuellement le temps
267     Ponderation_temps * niveauF_temps;
268     // d) contrôle via une ou plusieurs grandeurs consultables
269     Ponderation_Consultable* niveauF_grandeurConsultable;
270
271     // type de loi des mélanges
272     int type_melange; // = 1 : type historique: le mélange se fait sur la contrainte totale (par
    défaut)
273     // = 2 : type historique: le mélange se fait sur l'accroissement de la contrainte
274     // = 3 : nouveau type, fonctions complexes, mélange sur la contrainte totale (par défaut)
275     // = 4 : nouveau type, fonctions complexes, mélange sur l'accroissement de la contrainte
276
277
278     // codage des METHODES VIRTUELLES protegees:
279     // calcul des contraintes a t+dt
280     // calcul des contraintes
281 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
282     ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
283     ,TenseurBB & delta_epsBB_
284     ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
285     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
286     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
287     ,const Met_abstraite::Expli_t_tdt& ex);
288
289     // calcul des contraintes et de ses variations a t+dt
290 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
291     ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
292     ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
293     ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
294     ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
295     ,Tableau <TenseurBB *>& d_gijBB_tdt
296     ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
297     ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
298     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
299     ,const Met_abstraite::Impli& ex);
300
301     // calcul des contraintes et ses variations par rapport aux déformations a t+dt
302     // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
303     // le tenseur de déformation et son incrémentsont également en orthonormees
304     // si = false: les bases transmises sont utilisées
305     // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
306 void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & sigHH_t,TenseurBB& DepsBB
307     ,TenseurBB & epsBB_tdt,TenseurBB & delta_epsBB,double& jacobien_0,double& jacobien
308     ,TenseurHH& sigHH,TenseurHHHH & d_sigma_deps
309     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
    module_cisaillement
310     ,const Met_abstraite::Umat_cont& ex) ; // = 0;
311
312
313     // fonction surchargée dans les classes dérivée si besoin est
314 virtual void CalculGrandeurTravail
315     (const PtIntegMecaInterne& ptintmeca
316     ,const Deformation & def,Enum_dure temps,const ThermoDonnee& dTP
317     ,const Met_abstraite::Impli* ex_impli
318     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
319     ,const Met_abstraite::Umat_cont* ex_umat
320     ,const List_io<Ddl_etendu>* exclure_dd_etend
321     ,const List_io<const TypeQuelconque *>* exclure_Q
322     );
323
324     // permet d'indiquer à la classe à quelle valeur de PtIntegMecaInterne il faut se référer
325     // en particulier est utilisé par les lois additives,
326     // par contre doit être utilisé avec prudence
327 virtual void IndiquePtIntegMecaInterne(const PtIntegMecaInterne * ptintmeca)
328     { lois_interne1->IndiquePtIntegMecaInterne(ptintmeca);
329     lois_interne2->IndiquePtIntegMecaInterne(ptintmeca);
330     // puis la classe mère
331     Loi_comp_abstraite::IndiquePtIntegMecaInterne(ptintmeca);
332     };
333
334

```

```

335 // fonction interne utilisée par les classes dérivées de Loi_comp_abstraite
336 // pour répercuter les modifications de la température
337 // ici utiliser pour modifier la température des lois élémentaires
338 // l'Enum_dure: indique quel est la température courante : 0 t ou tdt
339 void RepercuteChangeTemperature(Enum_dure temps);
340
341 };
342 /// @} // end of group
343
344 #endif
345
346
347

```

## 7.76 Loi\_rien1D.h

```

1 // FICHER : Loi_rien1D.h
2 // CLASSE : Loi_rien1D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33
34 /*****
35 *      DATE:          31/01/2006
36 *
37 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *      PROJET:        Herezh++
40 *
41 *
42 *      BUT:          La classe Loi_rien1D definit une loi 1D qui ne fait rien
43 *
44 *      *****
45 *
46 *      VERIFICATION:
47 *      ! date !   auteur !           but
48 *      -----
49 *      !           !           !
50 *
51 *      *****
52 *      MODIFICATIONS:
53 *      ! date !   auteur !           but
54 *      -----
55 *
56 *      *****/
57
58
59 #ifndef LOI_RIEN_1D_H
60 #define LOI_RIEN_1D_H
61
62
63 #include "Loi_comp_abstraite.h"
64
65
66 class Loi_rien1D : public Loi_comp_abstraite
67 {
68

```

```

69
70 public :
71
72
73     // CONSTRUCTEURS :
74
75     // Constructeur par défaut
76     Loi_rien1D ();
77
78
79     // Constructeur de copie
80     Loi_rien1D (const Loi_rien1D& loi) ;
81
82     // DESTRUCTEUR :
83
84     ~Loi_rien1D ();
85
86     // Lecture des donnees de la classe sur fichier
87     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
88                                     ,LesFonctions_nD& lesFonctionsnD);
89
90     // affichage de la loi
91     void Affiche() const ;
92     // test si la loi est complete
93     // = 1 tout est ok, =0 loi incomplete
94     int TestComplet();
95
96     //----- lecture écriture de restart -----
97     // cas donne le niveau de la récupération
98     // = 1 : on récupère tout
99     // = 2 : on récupère uniquement les données variables (supposées comme telles)
100    void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
101                             lesCourbes1D
102                             ,LesFonctions_nD& lesFonctionsnD);
103
104    // cas donne le niveau de sauvegarde
105    // = 1 : on sauvegarde tout
106    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
107    void Ecriture_base_info_loi(ofstream& sort,const int cas);
108
109    // calcul d'un module d'young équivalent la loi, ceci pour un
110    // chargement nul
111    double Module_young_equivalent(Enum_dure ,const Deformation & ,SaveResul * ) {return 0.;;};
112
113    // récupération de la variation relative d'épaisseur calculée: h/h0
114    // cette variation n'est utile que pour des lois en contraintes planes
115    // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
116    // - pour les lois 2D def planes: retour de 0
117    // les infos nécessaires à la récupération , sont stockées dans saveResul
118    // qui est le conteneur spécifique au point où a été calculé la loi
119    virtual double HsurH0(SaveResul * saveResul) const {return 1.;;};
120
121    // création d'une loi à l'identique et ramène un pointeur sur la loi créée
122    Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_rien1D(*this)); };
123
124    // affichage et definition interactive des commandes particulières à chaque lois
125    void Info_commande_LoisDeComp(UtilLecture& lec);
126
127 protected :
128     // donnée de la loi
129
130     // codage des METHODES VIRTUELLES protegees:
131     // calcul des contraintes a t+dt
132     void Calcul_SigmaHH (TenseurHH& ,TenseurBB& ,DdlElement & ,
133                         TenseurBB & ,TenseurHH & ,BaseB& ,BaseH& ,TenseurBB & ,
134                         TenseurBB & ,
135                         TenseurBB & ,TenseurHH & ,Tableau <TenseurBB *>& ,
136                         double& ,double& ,TenseurHH &
137                         ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
138                         ,const Met_abstraite::Expli_t_tdt& )
139     {};
140
141     // calcul des variations des contraintes a t+dt
142     void Calcul_DsigmaHH_tdt (TenseurHH& ,TenseurBB& ,DdlElement &
143                              ,BaseB& ,TenseurBB & ,TenseurHH & ,
144                              BaseB& ,Tableau <BaseB> & ,BaseH& ,Tableau <BaseH> & ,
145                              TenseurBB & ,Tableau <TenseurBB *>& ,
146                              TenseurBB & ,TenseurBB & ,TenseurHH & ,
147                              Tableau <TenseurBB *>& ,
148                              Tableau <TenseurHH *>& ,double& ,double& ,
149                              Vecteur& ,TenseurHH& ,Tableau <TenseurHH *>&
150                              ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
151                              ,const Met_abstraite::Impli_t_tdt& )
152     {};
153
154     // fonction surchargée dans les classes dérivée si besoin est
155     virtual void CalculGrandeurTravail
156         (const PtIntegMecaInterne& ,const Deformation &

```

```

155         ,Enum_dure,const ThermoDonnee&
156         ,const Met_abstraite::Impli* ex_impli
157         ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
158         ,const Met_abstraite::Umat_cont* ex_umat
159         ,const List_io<Ddl_etendu>* exclure_dd_etend
160         ,const List_io<const TypeQuelconque *>* exclure_Q
161     ) {};
162
163 };
164
165
166 #endif
167
168
169

```

## 7.77 Loi\_rien2D.h

```

1 // FICHER : Loi_rien2D.h
2 // CLASSE : Loi_rien2D
3
4 /*****
5 *   UNIVERSITE DE BRETAGNE SUD (UBS)   --- I.U.P/I.U.T. DE LORIENT   *
6 *****/
7 *   LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)   *
8 *   Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex   *
9 *   tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
10 *****/
11 *   DATE:           31/01/2006           $   *
12 *                                     *   *
13 *   AUTEUR:        G RIO   (mailto:gerard.rio@univ-ubs.fr)   *
14 *               Tel 0297874571   fax : 02.97.87.45.72   *
15 *                                     $   *
16 *   PROJET:        Herezh++           *
17 *                                     $   *
18 *****/
19 *   BUT:   La classe Loi_rien2D definit une loi 2D qui ne fait rien *
20 *                                     $   *
21 *   ***** *
22 *   *
23 *   VERIFICATION:           *
24 *   ! date !   auteur !           but           !           *
25 *   ----- *
26 *   !           !           !           !           *
27 *                                     $   *
28 *   ***** *
29 *   MODIFICATIONS:           *
30 *   ! date !   auteur !           but           !           *
31 *   ----- *
32 *                                     $   *
33 *****/
34
35
36 #ifndef LOI_RIEN_2D_H
37 #define LOI_RIEN_2D_H
38
39
40 #include "Loi_comp_abstraite.h"
41
42
43 class Loi_rien2D : public Loi_comp_abstraite
44 {
45
46
47     public :
48
49
50         // CONSTRUCTEURS :
51
52         // Constructeur par default
53         Loi_rien2D ();
54
55
56         // Constructeur de copie
57         Loi_rien2D (const Loi_rien2D& loi) ;
58
59         // DESTRUCTEUR :
60
61         ~Loi_rien2D ();
62
63         // Lecture des donnees de la classe sur fichier
64         void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D);
65         // affichage de la loi
66         void Affiche() const ;

```

```

67     // test si la loi est complete
68     // = 1 tout est ok, =0 loi incomplete
69     int TestComplet();
70
71     //----- lecture écriture de restart -----
72     // cas donne le niveau de la récupération
73     // = 1 : on récupère tout
74     // = 2 : on récupère uniquement les données variables (supposées comme telles)
75     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D);
76     // cas donne le niveau de sauvegarde
77     // = 1 : on sauvegarde tout
78     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
79     void Ecriture_base_info_loi(ofstream& sort,const int cas);
80
81     // calcul d'un module d'young équivalent la loi, ceci pour un
82     // chargement nul
83     double Module_young_equivalent(const Deformation & ) {return 0.};
84
85     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
86     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_rien2D(*this)); };
87
88     // affichage et definition interactive des commandes particulières à chaque lois
89     void Info_commande_LoisDeComp(UtilLecture& lec);
90
91
92 protected :
93     // donnée de la loi
94
95     // codage des METHODES VIRTUELLES protegees:
96     // calcul des contraintes a t+dt
97     void Calcul_SigmaHH (TenseurHH& ,TenseurBB& ,DdlElement & ,
98     TenseurBB & ,TenseurHH & ,BaseB& ,BaseH& ,TenseurBB & ,
99     TenseurBB & ,
100     TenseurBB & ,TenseurHH & ,Tableau <TenseurBB *>& ,
101     double& ,double& ,TenseurHH &
102     ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
103     ,const Met_abstraite::Expli_t_tdt& )
104     {};
105
106     // calcul des variations des contraintes a t+dt
107     void Calcul_DsigmaHH_tdt (TenseurHH& ,TenseurBB& ,DdlElement &
108     ,BaseB& ,TenseurBB & ,TenseurHH & ,
109     BaseB& ,Tableau <BaseB> & ,BaseH& ,Tableau <BaseH> & ,
110     TenseurBB & ,Tableau <TenseurBB *>& ,
111     TenseurBB & ,TenseurBB & ,TenseurHH & ,
112     Tableau <TenseurBB *>& ,
113     Tableau <TenseurHH *>& ,double& ,double& ,
114     Vecteur& ,TenseurHH& ,Tableau <TenseurHH *>&
115     ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
116     ,const Met_abstraite::Impli& )
117     {};
118
119     // fonction surchargée dans les classes dérivée si besoin est
120     virtual void CalculGrandeurTravail(const PtIntegMecaInterne& ,const Deformation &
,Enum_dure,const ThermoDonnee& ) {};
121
122 };
123
124
125 #endif
126
127
128

```

## 7.78 Loi\_rien2D\_C.h

```

1 // FICHER : Loi_rien2D.h
2 // CLASSE : Loi_rien2D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify

```

```

19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33
34 /*****
35 *      DATE:          31/01/2006
36 *
37 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *      PROJET:        Herezh++
40 *
41 *
42 *      BUT:           La classe Loi_rien2D_C definit une loi 2D qui ne fait rien*
43 *                    de type contraintes planes.
44 *
45 *      *****
46 *
47 *      VERIFICATION:
48 *      ! date !      auteur !      but
49 *      -----
50 *      !           !           !
51 *
52 *      *****
53 *      MODIFICATIONS:
54 *      ! date !      auteur !      but
55 *      -----
56 *
57 *      *****/
58
59
60 #ifndef LOI_RIEN_2D_C_H
61 #define LOI_RIEN_2D_C_H
62
63
64 #include "Loi_comp_abstraite.h"
65
66
67 class Loi_rien2D_C : public Loi_comp_abstraite
68 {
69
70
71     public :
72
73
74         // CONSTRUCTEURS :
75
76         // Constructeur par default
77         Loi_rien2D_C ();
78
79
80         // Constructeur de copie
81         Loi_rien2D_C (const Loi_rien2D_C & loi) ;
82
83         // DESTRUCTEUR :
84
85         ~Loi_rien2D_C ();
86
87         // Lecture des donnees de la classe sur fichier
88         void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D & lesCourbes1D
89                                           ,LesFonctions_nD & lesFonctionsnD);
90
91         // affichage de la loi
92         void Affiche() const ;
93         // test si la loi est complete
94         // = 1 tout est ok, =0 loi incomplete
95         int TestComplet();
96
97         //----- lecture écriture de restart -----
98         // cas donne le niveau de la récupération
99         // = 1 : on récupère tout
100        // = 2 : on récupère uniquement les données variables (supposées comme telles)
101        void Lecture_base_info_loi(ifstream & ent,const int cas,LesReferences & lesRef,LesCourbes1D &
102                                  lesCourbes1D
103                                  ,LesFonctions_nD & lesFonctionsnD);
104
105        // cas donne le niveau de sauvegarde
106        // = 1 : on sauvegarde tout

```



```

104 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
105 void Ecriture_base_info_loi(ofstream& sort,const int cas);
106
107 // calcul d'un module d'young équivalent la loi, ceci pour un
108 // chargement nul
109 double Module_young_equivalent(Enum_dure ,const Deformation & ,SaveResul * ) {return 0.;;};
110
111 // récupération de la variation relative d'épaisseur calculée: h/h0
112 // cette variation n'est utile que pour des lois en contraintes planes
113 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
114 // - pour les lois 2D def planes: retour de 0
115 // les infos nécessaires à la récupération , sont stockées dans saveResul
116 // qui est le conteneur spécifique au point où a été calculé la loi
117 virtual double HsurH0(SaveResul * saveResul) const {return 1.;;};
118
119 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
120 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_rien2D_C(*this)); };
121
122 // affichage et definition interactive des commandes particulières à chaque lois
123 void Info_commande_LoisDeComp(UtilLecture& lec);
124
125
126 protected :
127 // donnée de la loi
128
129 // codage des METHODES VIRTUELLES protegees:
130 // calcul des contraintes a t+dt
131 void Calcul_SigmaHH (TenseurHH& ,TenseurBB& ,DdlElement & ,
132     TenseurBB & ,TenseurHH & ,BaseB& ,BaseH & ,TenseurBB & ,
133     TenseurBB & ,
134     TenseurBB & ,TenseurHH & ,Tableau <TenseurBB *> & ,
135     double& ,double& ,TenseurHH &
136     ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
137     ,const Met_abstraite::Expli_t_tdt & )
138     {};
139
140 // calcul des variations des contraintes a t+dt
141 void Calcul_DsigmaHH_tdt (TenseurHH& ,TenseurBB& ,DdlElement &
142     ,BaseB& ,TenseurBB & ,TenseurHH & ,
143     BaseB& ,Tableau <BaseB> & ,BaseH & ,Tableau <BaseH> & ,
144     TenseurBB & ,Tableau <TenseurBB *>& ,
145     TenseurBB & ,TenseurBB & ,TenseurHH & ,
146     Tableau <TenseurBB *>& ,
147     Tableau <TenseurHH *>& ,double& ,double& ,
148     Vecteur& ,TenseurHH& ,Tableau <TenseurHH *>&
149     ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
150     ,const Met_abstraite::Impli& )
151     {};
152
153 // fonction surchargée dans les classes dérivée si besoin est
154 virtual void CalculGrandeurTravail
155     (const PtIntegMecaInterne& ,const Deformation &
156     ,Enum_dure,const ThermoDonnee&
157     ,const Met_abstraite::Impli* ex_impli
158     ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
159     ,const Met_abstraite::Umat_cont* ex_umat
160     ,const List_io<Ddl_etendu>* exclure_dd_etend
161     ,const List_io<const TypeQuelconque *>* exclure_Q
162     ) {};
163
164 };
165
166
167 #endif
168
169
170

```

## 7.79 Loi\_rien2D\_D.h

```

1 // FICHER : Loi_rien2D_D.h
2 // CLASSE : Loi_rien2D_D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio

```

```

16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32 //
33
34 /*****
35 *   DATE:           31/01/2006
36 *
37 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:        Herezh++
40 *
41 *
42 *   BUT:   La classe Loi_rien2D_D definit une loi 2D qui ne fait rien*
43 *          de type déformations planes.
44 *
45 *   *****
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !           but
50 *   -----
51 *   !           !           !
52 *   *****
53 *
54 *   MODIFICATIONS:
55 *
56 *   ! date !   auteur !           but
57 *   -----
58 *
59 *   *****/
60 #ifndef LOI_RIEN_2D_D_H
61 #define LOI_RIEN_2D_D_H
62
63
64 #include "Loi_comp_abstraite.h"
65
66
67 class Loi_rien2D_D : public Loi_comp_abstraite
68 {
69
70
71     public :
72
73
74         // CONSTRUCTEURS :
75
76         // Constructeur par défaut
77         Loi_rien2D_D ();
78
79
80         // Constructeur de copie
81         Loi_rien2D_D (const Loi_rien2D_D & loi) ;
82
83         // DESTRUCTEUR :
84
85         ~Loi_rien2D_D ();
86
87         // Lecture des donnees de la classe sur fichier
88         void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D & lesCourbes1D
89                                           ,LesFonctions_nD & lesFonctionsnD);
90
91         // affichage de la loi
92         void Affiche() const ;
93         // test si la loi est complete
94         // = 1 tout est ok, =0 loi incomplete
95         int TestCompleet();
96
97         //----- lecture écriture de restart -----
98         // cas donne le niveau de la récupération
99         // = 1 : on récupère tout
100        // = 2 : on récupère uniquement les données variables (supposées comme telles)
101        void Lecture_base_info_loi(ifstream & ent,const int cas,LesReferences & lesRef,LesCourbes1D &
102        lesCourbes1D

```

```

101                                     ,LesFonctions_nD& lesFonctionsnD);
102 // cas donne le niveau de sauvegarde
103 // = 1 : on sauvegarde tout
104 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
105 void Ecriture_base_info_loi(ofstream& sort,const int cas);
106
107 // calcul d'un module d'young équivalent la loi, ceci pour un
108 // chargement nul
109 double Module_young_equivalent(Enum_dure ,const Deformation & ,SaveResul * ) {return 0.;;};
110
111 // récupération de la variation relative d'épaisseur calculée: h/h0
112 // cette variation n'est utile que pour des lois en contraintes planes
113 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
114 // - pour les lois 2D def planes: retour de 0
115 // les infos nécessaires à la récupération , sont stockées dans saveResul
116 // qui est le conteneur spécifique au point où a été calculé la loi
117 virtual double HsurH0(SaveResul * saveResul) const {return 0.;;};
118
119 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
120 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_rien2D_D(*this)); };
121
122 // affichage et definition interactive des commandes particulières à chaque lois
123 void Info_commande_LoisDeComp(UtilLecture& lec);
124
125
126 protected :
127 // donnée de la loi
128
129 // codage des METHODES VIRTUELLES protegees:
130 // calcul des contraintes a t+dt
131 void Calcul_SigmaHH (TenseurHH& ,TenseurBB& ,DdlElement & ,
132 TenseurBB & ,TenseurHH & ,BaseB& ,BaseH& ,TenseurBB & ,
133 TenseurBB & ,
134 TenseurBB & ,TenseurHH & ,Tableau <TenseurBB *>& ,
135 double& ,double& ,TenseurHH &
136 ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
137 ,const Met_abstraite::Expli_t_tdt& )
138 {};
139
140 // calcul des variations des contraintes a t+dt
141 void Calcul_DsigmaHH_tdt (TenseurHH& ,TenseurBB& ,DdlElement &
142 ,BaseB& ,TenseurBB & ,TenseurHH & ,
143 BaseB& ,Tableau <BaseB> & ,BaseH& ,Tableau <BaseH> & ,
144 TenseurBB & ,Tableau <TenseurBB *>& ,
145 TenseurBB & ,TenseurBB & ,TenseurHH & ,
146 Tableau <TenseurBB *>& ,
147 Tableau <TenseurHH *>& ,double& ,double& ,
148 Vecteur& ,TenseurHH& ,Tableau <TenseurHH *>&
149 ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
150 ,const Met_abstraite::Impli& )
151 {};
152
153 // fonction surchargée dans les classes dérivée si besoin est
154 virtual void CalculGrandeurTravail
155 (const PtIntegMecaInterne& ,const Deformation &
156 ,Enum_dure,const ThermoDonnee&
157 ,const Met_abstraite::Impli* ex_impli
158 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
159 ,const Met_abstraite::Umat_cont* ex_umat
160 ,const List_io<Ddl_etendu>* excludre_dd_etend
161 ,const List_io<const TypeQuelconque *>* excludre_Q
162 ) {};
163
164 };
165
166
167 #endif
168
169
170

```

## 7.80 Loi\_rien3D.h

```

1 // FICHER : Loi_rien3D.h
2 // CLASSE : Loi_rien3D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.

```

```

13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32 //
33
34 /*****
35 *   DATE:           31/01/2006
36 *
37 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:        Herezh++
40 *
41 *
42 *   BUT:   La classe Loi_rien3D definit une loi 3D qui ne fait rien
43 *
44 *   *****
45 *
46 *   VERIFICATION:
47 *   ! date !   auteur !           but
48 *   -----
49 *   !           !           !
50 *
51 *   *****
52 *   MODIFICATIONS:
53 *   ! date !   auteur !           but
54 *   -----
55 *
56 *   *****/
57
58
59 #ifndef LOI_RIEN_3D_H
60 #define LOI_RIEN_3D_H
61
62
63 #include "Loi_comp_abstraite.h"
64
65
66 class Loi_rien3D : public Loi_comp_abstraite
67 {
68
69     public :
70
71
72
73         // CONSTRUCTEURS :
74
75         // Constructeur par défaut
76         Loi_rien3D ();
77
78
79         // Constructeur de copie
80         Loi_rien3D (const Loi_rien3D& loi) ;
81
82         // DESTRUCTEUR :
83
84         ~Loi_rien3D ();
85
86         // Lecture des donnees de la classe sur fichier
87         void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
88             ,LesFonctions_nD& lesFonctionsnD);
89
90         // affichage de la loi
91         void Affiche() const ;
92         // test si la loi est complete
93         // = 1 tout est ok, =0 loi incomplete
94         int TestComplet();
95
96         //----- lecture écriture de restart -----
97         // cas donne le niveau de la récupération
98         // = 1 : on récupère tout
99         // = 2 : on récupère uniquement les données variables (supposées comme telles)

```

```

99     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
100     lesCourbes1D
101                                     ,LesFonctions_nD& lesFonctionsnD);
102     // cas donne le niveau de sauvegarde
103     // = 1 : on sauvegarde tout
104     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
105     void Ecriture_base_info_loi(ofstream& sort,const int cas);
106
107     // calcul d'un module d'young équivalent la loi, ceci pour un
108     // chargement nul
109     double Module_young_equivalent(Enum_dure ,const Deformation & ,SaveResul * ) {return 0.;;};
110
111     // récupération de la variation relative d'épaisseur calculée: h/h0
112     // cette variation n'est utile que pour des lois en contraintes planes
113     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
114     // - pour les lois 2D def planes: retour de 0
115     // les infos nécessaires à la récupération , sont stockées dans saveResul
116     // qui est le conteneur spécifique au point où a été calculé la loi
117     virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;;};
118
119     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
120     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Loi_rien3D(*this)); };
121
122     // affichage et definition interactive des commandes particulières à chaque lois
123     void Info_commande_LoisDeComp(UtilLecture& lec);
124
125     protected :
126         // donnée de la loi
127
128         // codage des METHODES VIRTUELLES proteges:
129         // calcul des contraintes a t+dt
130         void Calcul_SigmaHH (TenseurHH& ,TenseurBB& ,DdlElement & ,
131             TenseurBB & ,TenseurHH & ,BaseB& ,BaseH& ,TenseurBB & ,
132             TenseurBB & ,
133             TenseurBB & ,TenseurHH & ,Tableau <TenseurBB *>& ,
134             double& ,double& ,TenseurHH &
135             ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
136             ,const Met_abstraite::Expli_t_tdt& )
137             {};
138
139         // calcul des variations des contraintes a t+dt
140         void Calcul_DsigmaHH_tdt (TenseurHH& ,TenseurBB& ,DdlElement &
141             ,BaseB& ,TenseurBB & ,TenseurHH & ,
142             BaseB& ,Tableau <BaseB> & ,BaseH& ,Tableau <BaseH> & ,
143             TenseurBB & ,Tableau <TenseurBB *>& ,
144             TenseurBB & ,TenseurBB & ,TenseurHH & ,
145             Tableau <TenseurBB *>& ,
146             Tableau <TenseurHH *>& ,double& ,double& ,
147             Vecteur& ,TenseurHH& ,Tableau <TenseurHH *>&
148             ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
149             ,const Met_abstraite::Impli& )
150             {};
151
152
153         // calcul des contraintes et ses variations par rapport aux déformations a t+dt
154         // en_base_orthonormee: le tenseur de contrainte en entrée est en orthonormee
155         // le tenseur de déformation et son incrémentsont également en orthonormees
156         // si = false: les bases transmises sont utilisées
157         // ex: contient les éléments de métrique relativement au paramétrage matériel = X_(0)^a
158         virtual void Calcul_dsigma_deps (bool en_base_orthonormee, TenseurHH & ,TenseurBB&
159             ,TenseurBB & ,TenseurBB & ,double& ,double&
160             ,TenseurHH& ,TenseurHHHH&
161             ,EnergieMeca & ,const EnergieMeca & ,double& ,double&
162             ,const Met_abstraite::Umat_cont& )
163             {};
164
165         // fonction surchargée dans les classes dérivée si besoin est
166         virtual void CalculGrandeurTravail
167             (const PtIntegMecaInterne& ,const Deformation &
168             ,Enum_dure,const ThermoDonnee&
169             ,const Met_abstraite::Impli* ex_impli
170             ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
171             ,const Met_abstraite::Umat_cont* ex_umat
172             ,const List_io<Ddl_etendu>* exclure_dd_etend
173             ,const List_io<const TypeQuelconque *>* exclure_Q
174             ) {};
175
176     };
177
178
179 #endif
180
181
182

```

## 7.81 Prandtl\_Reuss.h

```

1 // FICHER : Prandtl_Reuss.h
2 // CLASSE : Prandtl_Reuss
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:      19/01/2001
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *
41 *   BUT:      La classe Prandtl-Reuss permet de calculer la contrainte
42 *            et ses derivees pour une loi élasto-plastique de type
43 *            Prandtl-Reuss.
44 *
45 *            *****
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !           but
50 *   -----
51 *   !       !           !
52 *
53 *   MODIFICATIONS:
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *****/
58
59
60 #ifndef PRANDTL_REUSS_H
61 #define PRANDTL_REUSS_H
62
63
64 #include "Loi_comp_abstraite.h"
65 #include "CourbelD.h"
66 #include "TypeConsTens.h"
67
68 /** @defgroup Les_lois_de_plasticite
69 *
70 *   BUT:   groupe des lois de plasticité
71 *
72 *
73 * \author   Gérard Rio
74 * \version  1.0
75 * \date    19/01/2001
76 * \brief   Définition des lois de plasticité
77 *
78 */
79
80 /// @addtogroup Les_lois_de_plasticite
81 /// @{
82 ///
83
84

```

```

85 class Prandtl_Reuss : public Loi_comp_abstraite
86 {
87
88
89     public :
90
91
92         // CONSTRUCTEURS :
93
94         // Constructeur par défaut
95         Prandtl_Reuss ();
96
97
98         // Constructeur de copie
99         Prandtl_Reuss (const Prandtl_Reuss& loi) ;
100
101         // DESTRUCTEUR :
102
103         ~Prandtl_Reuss ();
104
105         // initialise les donnees particulieres a l'elements
106         // de matiere traite ( c-a-dire au pt calcule)
107         // Il y a creation d'une instance de SaveResul particuliere
108         // a la loi concernee
109         // la SaveResul classe est remplie par les instances heritantes
110         // le pointeur de SaveResul est sauvegarde au niveau de l'element
111         // c'a-d que les info particulieres au point considere sont stocke
112         // au niveau de l'element et non de la loi.
113         class SaveResulPrandtl_Reuss: public SaveResul
114         { public :
115             SaveResulPrandtl_Reuss() :
116                 epsilon_barre_t(0),def_plasBB(),def_plasBB_t()
117                 {};
118             SaveResulPrandtl_Reuss(const SaveResulPrandtl_Reuss& sav):
119                 epsilon_barre(sav.epsilon_barre),epsilon_barre_t(sav.epsilon_barre_t)
120                 ,def_plasBB(sav.def_plasBB),def_plasBB_t(sav.def_plasBB_t)
121                 {};
122             // définition d'une nouvelle instance identique
123             // appelle du constructeur via new
124             SaveResul * Nevez_SaveResul() const{return (new SaveResulPrandtl_Reuss(*this));};
125         // affectation
126         virtual SaveResul & operator = ( const SaveResul & a);
127         //===== lecture écriture dans base info =====
128         // cas donne le niveau de la récupération
129         // = 1 : on récupère tout
130         // = 2 : on récupère uniquement les données variables (supposées comme telles)
131         void Lecture_base_info (ifstream& ent,const int cas);
132         // cas donne le niveau de sauvegarde
133         // = 1 : on sauvegarde tout
134         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
135         void Ecriture_base_info(ofstream& sort,const int cas);
136
137         // mise à jour des informations transitoires en définitif s'il y a convergence
138         // par exemple (pour la plasticité par exemple)
139         void TdtversT() {epsilon_barre_t = epsilon_barre;def_plasBB_t = def_plasBB;};
140         void TversTdt() {epsilon_barre = epsilon_barre_t;def_plasBB = def_plasBB_t;};
141
142         // affichage à l'écran des infos
143         void Affiche() const;
144
145         //changement de base de toutes les grandeurs internes tensorielles stockées
146         // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
147         // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
148         // actuellement les tenseurs sont exprimés en absolu donc pas de chgt de base actif
149         // gpH(i) = gamma(i,j) * gH(j)
150         virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) {};
151
152         // procedure permettant de completer éventuellement les données particulières
153         // de la loi stockées
154         // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
155         // completer est appelé apres sa creation avec les donnees du bloc transmis
156         // peut etre appeler plusieurs fois
157         SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
158                                     ,const Loi_comp_abstraite* loi) {};
159
160         // ---- récupération d'information: spécifique à certaine classe dérivée
161         double Deformation_plastique() { return epsilon_barre_t;};
162
163         // données protégées
164         double epsilon_barre; // déformation plastique cumulée
165         Tenseur3BB def_plasBB; // déformation plastique
166         double epsilon_barre_t; // la dernière enregistrée
167         Tenseur3BB def_plasBB_t; // déformation plastique, la dernière enregistrée
168         // les tenseurs def_plasBB sont exprimés en absolu
169     };
170
171     SaveResul * New_et_Initialise() { return (new SaveResulPrandtl_Reuss());};

```

```

172
173 // Lecture des donnees de la classe sur fichier
174 void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
175                                     ,LesFonctions_nD& lesFonctionsnD);
176 // affichage de la loi
177 void Affiche() const ;
178 // test si la loi est complete
179 // = 1 tout est ok, =0 loi incomplete
180 int TestCompleet();
181
182 // calcul d'un module d'young equivalent à la loi, ceci pour un
183 // chargement nul
184 double Module_young_equivalent(Enum_dure ,const Deformation & ,SaveResul * )
185 { return E; };
186
187 // récupération de la variation relative d'épaisseur calculée: h/h0
188 // cette variation n'est utile que pour des lois en contraintes planes
189 // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
190 // - pour les lois 2D def planes: retour de 0
191 // les infos nécessaires à la récupération , sont stockées dans saveResul
192 // qui est le conteneur spécifique au point où a été calculé la loi
193 virtual double HsurH0(SaveResul * saveResul) const {return ConstMath::tresgrand;};
194
195 // création d'une loi à l'identique et ramène un pointeur sur la loi créée
196 Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Prandtl_Reuss(*this)); };
197
198 // affichage et definition interactive des commandes particulières à chaque lois
199 void Info_commande_LoisDeComp(UtilLecture& lec);
200
201 //----- lecture écriture de restart -----
202 // cas donne le niveau de la récupération
203 // = 1 : on récupère tout
204 // = 2 : on récupère uniquement les données variables (supposées comme telles)
205 void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
lesCourbes1D
206                                     ,LesFonctions_nD& lesFonctionsnD);
207
208 // cas donne le niveau de sauvegarde
209 // = 1 : on sauvegarde tout
210 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
211 void Ecriture_base_info_loi(ofstream& sort,const int cas);
212
213 protected :
214
215 // donnees protegees
216 double E,nu; // paramètres de la partie élastique de la loi
217 CourbelD* f_écrouissage; // courbe d'écrouissage
218 // paramètres de l'algorithme
219 double tolerance_plas; // tolérance sur la résolution de la plasticité
220 int nb_boucle_maxi; // le maximum d'itération de plasticité permis
221
222 // codage des METHODES VIRTUELLES protegees:
223 // calcul des contraintes a t+dt
224 // calcul des contraintes
225 void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
226 ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& giH, TenseurBB & epsBB_
227 ,TenseurBB & delta_epsBB_
228 ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
229 ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
230 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
231 ,const Met_abstraite::Expli_t_tdt& ex);
232
233 // calcul des contraintes et de ses variations a t+dt
234 void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
235 ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
236 ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
237 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
238 ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
239 ,Tableau <TenseurBB *>& d_gijBB_tdt
240 ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
241 ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
242 ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
243 ,const Met_abstraite::Impli& ex);
244
245
246 // fonction surchargée dans les classes dérivée si besoin est
247 virtual void CalculGrandeurTravail
248 (const PtIntegMecaInterne& ,const Deformation &
249 ,Enum_dure,const ThermoDonnee&
250 ,const Met_abstraite::Impli* ex_impli
251 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
252 ,const Met_abstraite::Umat_cont* ex_umat
253 ,const List_io<Ddl_etendu>* exclure_dd_etend
254 ,const List_io<const TypeQuelconque *>* exclure_Q
255 ) {};

```



```

256
257 };
258 /// @} // end of group
259
260
261 #endif
262
263
264

```

## 7.82 Prandtl\_Reuss1D.h

```

1 // FICHER : Prandtl_Reuss1D.h
2 // CLASSE : Prandtl_Reuss1D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *      DATE:      19/01/2001
35 *
36 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *
41 *      BUT:      La classe Prandtl-Reuss1D permet de calculer la contrainte*
42 *              et ses derivees pour une loi élasto-plastique de type
43 *              Prandtl-Reuss en dimension 1.
44 *
45 *              *****
46 *
47 *      VERIFICATION:
48 *      ! date ! auteur ! but
49 *      -----
50 *      ! ! !
51 *
52 *      *****
53 *      MODIFICATIONS:
54 *      ! date ! auteur ! but
55 *      -----
56 *
57 *      *****/
58
59
60 #ifndef PRANDTL_REUSS1D_H
61 #define PRANDTL_REUSS1D_H
62
63
64 #include "Loi_comp_abstraite.h"
65 #include "Courbe1D.h"
66 #include "TypeConsTens.h"
67
68 /// @addtogroup Les_lois_de_plasticite
69 /// @{
70 ///
71
72

```

```

73 class Prandtl_Reuss1D : public Loi_comp_abstraite
74 {
75
76
77     public :
78
79
80         // CONSTRUCTEURS :
81
82         // Constructeur par défaut
83         Prandtl_Reuss1D ();
84
85
86         // Constructeur de copie
87         Prandtl_Reuss1D (const Prandtl_Reuss1D& loi) ;
88
89         // DESTRUCTEUR :
90
91         ~Prandtl_Reuss1D ();
92
93         // initialise les donnees particulieres a l'elements
94         // de matiere traite ( c-a-dire au pt calcule)
95         // Il y a creation d'une instance de SaveResul particuliere
96         // a la loi concernee
97         // la SaveResul classe est remplie par les instances heritantes
98         // le pointeur de SaveResul est sauvegarde au niveau de l'element
99         // c'a-d que les info particulieres au point considere sont stocke
100        // au niveau de l'element et non de la loi.
101        class SaveResulPrandtl_Reuss1D: public SaveResul
102        { public :
103            SaveResulPrandtl_Reuss1D() :
104                epsilon_barre(0),epsilon_barre_t(0),def_plasBB(),def_plasBB_t()
105                {};
106            SaveResulPrandtl_Reuss1D(const SaveResulPrandtl_Reuss1D& sav):
107                epsilon_barre(sav.epsilon_barre),epsilon_barre_t(sav.epsilon_barre_t)
108                ,def_plasBB(sav.def_plasBB),def_plasBB_t(sav.def_plasBB_t)
109                {};
110            // définition d'une nouvelle instance identique
111            // appelle du constructeur via new
112            SaveResul * Nevez_SaveResul() const{return (new SaveResulPrandtl_Reuss1D(*this));};
113
114        // affectation
115        virtual SaveResul & operator = ( const SaveResul & a);
116        //===== lecture écriture dans base info =====
117        // cas donne le niveau de la récupération
118        // = 1 : on récupère tout
119        // = 2 : on récupère uniquement les données variables (supposées comme telles)
120        void Lecture_base_info (ifstream& ent,const int cas);
121        // cas donne le niveau de sauvegarde
122        // = 1 : on sauvegarde tout
123        // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
124        void Ecriture_base_info(ofstream& sort,const int cas);
125
126        // mise à jour des informations transitoires en définitif s'il y a convergence
127        // par exemple (pour la plasticité par exemple)
128        void TdtversT() {epsilon_barre_t = epsilon_barre;def_plasBB_t = def_plasBB;};
129        void TversTdt() {epsilon_barre = epsilon_barre_t;def_plasBB = def_plasBB_t;};
130
131        // affichage à l'écran des infos
132        void Affiche() const;
133
134        // changement de base de toutes les grandeurs internes tensorielles stockées
135        // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
136        // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
137        // actuellement les tenseurs sont exprimés en absolu donc pas de chgt de base actif
138        // gpH(i) = gamma(i,j) * gH(j)
139        virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) {};
140
141        // procedure permettant de completer éventuellement les données particulières
142        // de la loi stockées
143        // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
144        // completer est appelé apres sa creation avec les donnees du bloc transmis
145        // peut etre appeler plusieurs fois
146        SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
147            ,const Loi_comp_abstraite* loi) {};
148
149        // ---- récupération d'information: spécifique à certaine classe dérivée
150        double Deformation_plastique() { return epsilon_barre_t;};
151
152        // données protégées
153        double epsilon_barre; // déformation plastique cumulée
154        Tenseur1BB def_plasBB; // déformation plastique
155        double epsilon_barre_t; // la dernière enregistrée
156        Tenseur1BB def_plasBB_t; // déformation plastique, la dernière enregistrée
157        // les tenseurs def_plasBB sont exprimés en absolu
158
159 };

```

```

160     SaveResul * New_et_Initialise()
161     { SaveResulPrandtl_Reuss1D * pt = new SaveResulPrandtl_Reuss1D();
162       return pt;};
163
164     // Lecture des donnees de la classe sur fichier
165     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
166                                       ,LesFonctions_nD& lesFonctionsnD);
167
168     // affichage de la loi
169     void Affiche() const ;
170     // test si la loi est complete
171     // = 1 tout est ok, =0 loi incomplete
172     int TestComplet();
173
174     // calcul d'un module d'young equivalent à la loi, ceci pour un
175     // chargement nul
176     double Module_young_equivalent(Enum_dure ,const Deformation & ,SaveResul * )
177     { return E; };
178
179     // récupération de la variation relative d'épaisseur calculée: h/h0
180     // cette variation n'est utile que pour des lois en contraintes planes
181     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
182     // - pour les lois 2D def planes: retour de 0
183     // les infos nécessaires à la récupération , sont stockées dans saveResul
184     // qui est le conteneur spécifique au point où a été calculé la loi
185     virtual double HsurH0(SaveResul * saveResul) const
186     { cout << "\n Prandtl_Reuss1D::HsurH0(.. , methode non implante pour l'instant ";
187       Sortie(1);
188     };
189
190     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
191     Loi_comp_abstraite* Nouvelle_loi_identique() const { return (new Prandtl_Reuss1D(*this)); };
192
193     //----- lecture écriture de restart -----
194     // cas donne le niveau de la récupération
195     // = 1 : on récupère tout
196     // = 2 : on récupère uniquement les données variables (supposées comme telles)
197     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
198                               lesCourbes1D
199                               ,LesFonctions_nD& lesFonctionsnD);
200
201     // cas donne le niveau de sauvegarde
202     // = 1 : on sauvegarde tout
203     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
204     void Ecriture_base_info_loi(ofstream& sort,const int cas);
205
206     // affichage et definition interactive des commandes particulières à chaque lois
207     void Info_commande_LoisDeComp(UtilLecture& lec);
208
209     protected :
210
211     // donnees protegees
212     double E,nu; // paramètres de la partie élastique de la loi
213     Courbe1D* f_écrouissage; // courbe d'écrouissage
214     // paramètres de l'algorithme
215     double tolerance_plas; // tolérance sur la résolution de la plasticité
216     int nb_boucle_maxi; // le maximum d'itération de plasticité permis
217     int nb_sous_increment; // le maxi de sous incrément prévu
218
219     // codage des METHODES VIRTUELLES protegees:
220     // calcul des contraintes a t+dt
221     // calcul des contraintes
222     void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
223                        ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
224                        ,TenseurBB & delta_epsBB_
225                        ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
226                        ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
227                        ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
228                        module_cisaillement
229                        ,const Met_abstraite::Expli_t_tdt& ex);
230
231     // calcul des contraintes et de ses variations a t+dt
232     void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
233                               ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
234                               ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
235                               ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
236                               ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
237                               ,Tableau <TenseurBB *>& d_gijBB_tdt
238                               ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
239                               ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
240                               ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
241                               module_cisaillement
242                               ,const Met_abstraite::Impli& ex);
243
244     // fonction surchargée dans les classes dérivée si besoin est
245     virtual void CalculGrandeurTravail
246     (const PtIntegMecaInterne& ,const Deformation &
247     ,Enum_dure,const ThermoDonnee&

```

```

244         ,const Met_abstraite::Impli* ex_impli
245         ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
246         ,const Met_abstraite::Umat_cont* ex_umat
247         ,const List_io<Ddl_etendu>* exclure_dd_etend
248         ,const List_io<const TypeQuelconque *>* exclure_Q
249     ) {};
250
251 };
252 /// @} // end of group
253
254
255 #endif
256
257
258

```

## 7.83 Prandtl\_Reuss2D\_D.h

```

1 // FICHER : Prandtl_Reuss2D_D.h
2 // CLASSE : Prandtl_Reuss2D_D
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:           19/01/2001
35 *
36 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:         Herezh++
39 *
40 *
41 *   BUT:            La classe Prandtl-Reuss 2D_D permet de calculer la
42 *                  contrainte et ses derivees pour une loi élasto-plastique de type
43 *                  Prandtl-Reuss en déformation plane.
44 *
45 *   *****/
46 *
47 *   VERIFICATION:
48 *   ! date ! auteur ! but
49 *   -----
50 *   ! ! !
51 *
52 *   *****/
53 *
54 *   MODIFICATIONS:
55 *   ! date ! auteur ! but
56 *   -----
57 *
58 *   *****/
59
60 #ifndef PRANDTL_REUSS2D_D_H
61 #define PRANDTL_REUSS2D_D_H
62
63
64 #include "Loi_comp_abstraite.h"
65 #include "CourbelD.h"
66 #include "TypeConsTens.h"

```

```

67
68 /// @addtogroup Les_lois_de_plasticite
69 /// @{
70 ///
71
72
73 class Prandtl_Reuss2D_D : public Loi_comp_abstraite
74 {
75
76
77     public :
78
79         // CONSTRUCTEURS :
80
81         // Constructeur par défaut
82         Prandtl_Reuss2D_D ();
83
84
85         // Constructeur de copie
86         Prandtl_Reuss2D_D (const Prandtl_Reuss2D_D& loi) ;
87
88
89         // DESTRUCTEUR :
90
91         ~Prandtl_Reuss2D_D ();
92
93         // initialise les donnees particulieres a l'elements
94         // de matiere traite ( c-a-dire au pt calcule)
95         // Il y a creation d'une instance de SaveResul particuliere
96         // a la loi concernee
97         // la SaveResul classe est remplie par les instances heritantes
98         // le pointeur de SaveResul est sauvegarde au niveau de l'element
99         // c'a-d que les info particulieres au point considere sont stocke
100        // au niveau de l'element et non de la loi.
101        class SaveResulPrandtl_Reuss2D_D: public SaveResul
102        { public :
103            SaveResulPrandtl_Reuss2D_D() :
104                epsilon_barre(0), epsilon_barre_t(0), def_plasBB(), def_plasBB_t()
105            {};
106            SaveResulPrandtl_Reuss2D_D(const SaveResulPrandtl_Reuss2D_D& sav):
107                epsilon_barre(sav.epsilon_barre), epsilon_barre_t(sav.epsilon_barre_t),
108                def_plasBB(sav.def_plasBB), def_plasBB_t(sav.def_plasBB_t)
109            {};
110            // définition d'une nouvelle instance identique
111            // appelle du constructeur via new
112            SaveResul * Nevez_SaveResul() const{return (new SaveResulPrandtl_Reuss2D_D(*this));};
113
114        // affectation
115        virtual SaveResul & operator = ( const SaveResul & a);
116        //===== lecture écriture dans base info =====
117        // cas donne le niveau de la récupération
118        // = 1 : on récupère tout
119        // = 2 : on récupère uniquement les données variables (supposées comme telles)
120        void Lecture_base_info (ifstream& ent,const int cas);
121        // cas donne le niveau de sauvegarde
122        // = 1 : on sauvegarde tout
123        // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
124        void Ecriture_base_info(ofstream& sort,const int cas);
125
126        // mise à jour des informations transitoires en définitif s'il y a convergence
127        // par exemple (pour la plasticité par exemple)
128        void TdtversT() {epsilon_barre_t = epsilon_barre;def_plasBB_t = def_plasBB;};
129        void TversTdt() {epsilon_barre = epsilon_barre_t;def_plasBB = def_plasBB_t;};
130
131        // affichage à l'écran des infos
132        void Affiche() const;
133
134        //changement de base de toutes les grandeurs internes tensorielles stockées
135        // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
136        // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
137        // actuellement les tenseurs sont exprimés en absolu donc pas de chgt de base actif
138        // gpH(i) = gamma(i,j) * gH(j)
139        virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma) {};
140
141        // procedure permettant de completer éventuellement les données particulières
142        // de la loi stockées
143        // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
144        // completer est appelé apres sa creation avec les donnees du bloc transmis
145        // peut etre appeler plusieurs fois
146        SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
147            ,const Loi_comp_abstraite* loi) {};
148
149        // ---- récupération d'information: spécifique à certaine classe dérivée
150        double Deformation_plastique() { return epsilon_barre_t;};
151
152        // données protégées
153        double epsilon_barre; // déformation plastique cumulée
154        Tenseur2BB def_plasBB; // déformation plastique

```

```

154         double epsilon_barre_t; // la dernière enregistrée
155         Tenseur2BB def_plasBB_t; // déformation plastique, la dernière enregistrée
156         // les tenseurs def_plasBB sont exprimés en absolu
157
158     };
159
160     SaveResul * New_et_Initialise() { return ( new SaveResulPrandtl_Reuss2D_D());};
161
162     // Lecture des donnees de la classe sur fichier
163     void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
164                                     ,LesFonctions_nD& lesFonctionsnD);
165
166     // affichage de la loi
167     void Affiche() const ;
168     // test si la loi est complete
169     // = 1 tout est ok, =0 loi incomplete
170     int TestComplet();
171
172     // calcul d'un module d'young équivalent à la loi, ceci pour un
173     // chargement nul
174     double Module_young_equivalent(Enum_dure ,const Deformation &,SaveResul * )
175     { return E; };
176
177     // récupération de la variation relative d'épaisseur calculée: h/h0
178     // cette variation n'est utile que pour des lois en contraintes planes
179     // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
180     // - pour les lois 2D def planes: retour de 0
181     // les infos nécessaires à la récupération , sont stockées dans saveResul
182     // qui est le conteneur spécifique au point où a été calculé la loi
183     virtual double HsurH0(SaveResul * saveResul) const {return 0.;};
184
185     // création d'une loi à l'identique et ramène un pointeur sur la loi créée
186     Loi_comp_abstraite* Nouvelle_loi_identique() const { return new Prandtl_Reuss2D_D(*this); };
187
188     //----- lecture écriture de restart -----
189     // cas donne le niveau de la récupération
190     // = 1 : on récupère tout
191     // = 2 : on récupère uniquement les données variables (supposées comme telles)
192     void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
193                             lesCourbes1D
194                             ,LesFonctions_nD& lesFonctionsnD);
195
196     // cas donne le niveau de sauvegarde
197     // = 1 : on sauvegarde tout
198     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
199     void Ecriture_base_info_loi(ofstream& sort,const int cas);
200
201     // affichage et definition interactive des commandes particulières à chaque lois
202     void Info_commande_LoisDeComp(UtilLecture& lec);
203
204     protected :
205
206     // donnees protegees
207     double E,nu; // paramètres de la partie élastique de la loi
208     Courbe1D* f_écrouissage; // courbe d'écrouissage
209     // paramètres de l'algorithme
210     double tolerance_plas; // tolérance sur la résolution de la plasticité
211     int nb_boucle_maxi; // le maximum d'itération de plasticité permis
212
213     // codage des METHODES VIRTUELLES protegees:
214     // calcul des contraintes a t+dt
215     // calcul des contraintes
216     void Calcul_SigmaHH (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
217                       ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H, TenseurBB & epsBB_
218                       ,TenseurBB & delta_epsBB_
219                       ,TenseurBB & gijBB_,TenseurHH & gijHH_,Tableau <TenseurBB *>& d_gijBB_
220                       ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
221                       ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
222                       module_cisaillement
223                       ,const Met_abstraite::Expli_t_tdt& ex);
224
225     // calcul des contraintes et de ses variations a t+dt
226     void Calcul_DsigmaHH_tdt (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
227                              ,BaseB& giB_t,TenseurBB & gijBB_t,TenseurHH & gijHH_t
228                              ,BaseB& giB_tdt,Tableau <BaseB> & d_giB_tdt,BaseH& giH_tdt,Tableau <BaseH> & d_giH_tdt
229                              ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *>& d_epsBB
230                              ,TenseurBB & delta_epsBB,TenseurBB & gijBB_tdt,TenseurHH & gijHH_tdt
231                              ,Tableau <TenseurBB *>& d_gijBB_tdt
232                              ,Tableau <TenseurHH *>& d_gijHH_tdt,double& jacobien_0,double& jacobien
233                              ,Vecteur& d_jacobien_tdt,TenseurHH& sigHH,Tableau <TenseurHH *>& d_sigHH
234                              ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
235                              module_cisaillement
236                              ,const Met_abstraite::Impli& ex);
237
238     // fonction surchargée dans les classes dérivée si besoin est
239     virtual void CalculGrandeurTravail
240             (const PtIntegMecaInterne& ,const Deformation &

```

```

238         ,Enum_dure,const ThermoDonnee&
239         ,const Met_abstraite::Impli* ex_impli
240         ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
241         ,const Met_abstraite::Umat_cont* ex_umat
242         ,const List_io<Ddl_etendu>* exclure_dd_etend
243         ,const List_io<const TypeQuelconque *>* exclure_Q
244     ) {};
245
246 };
247 /// @} // end of group
248
249
250 #endif
251
252
253

```

## 7.84 Loi\_de\_Tait.h

```

1 // FICHER : Loi_de_Tait.h
2 // CLASSE : Loi_de_Tait
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *      DATE:      16/10/2004
35 *
36 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *****/
41 *      BUT:      Définition de la loi de Tait en considérant un comportement*
42 *               isotrope thermique.
43 *               La loi est utilisable quelque soit la dimension.
44 *               *****
45 *
46 *      VERIFICATION:
47 *      ! date ! auteur ! but
48 *      -----
49 *      ! ! !
50 *
51 *      *****
52 *      MODIFICATIONS:
53 *      ! date ! auteur ! but
54 *      -----
55 *
56 *****/
57
58
59
60 #ifndef LOI_DE_TAIT_H
61 #define LOI_DE_TAIT_H
62
63
64 #include "CompThermoPhysiqueAbstraite.h"
65 #include "CristaliniteAbstraite.h"

```

```

66 #include "Bloc.h"
67
68 /*
69 prise en compte de la cristallinité
70 1) un indicateur dans la loi, qui donne les différents cas: =0 , pas de cristallinité, =1 cristallinité
   calculé et stockée,
71 mais pas de modification des autres paramètres et calculs de la loi de tait, =2 forme particulière de la
   loi de tait qui
72 utilise le taux de cristallinité
73 Pour le stockage: 2 classes save_result : celle actuelle et une autre qui dérive et qui contient les
   éléments du taux
74 taux précédent, taux en cours ?? a voir
75 méthode: retour dans la loi thermoPhysique générale d'une grandeur: on utilise le conteneur ThermoDonnee
76
77 */
78
79 /// @addtogroup Les_lois_concernant_thermique
80 /// @{
81 ///
82
83 class Loi_de_Tait : public CompThermoPhysiqueAbstraite
84 {
85     public :
86         // CONSTRUCTEURS :
87
88         // Constructeur par défaut
89         Loi_de_Tait ();
90         // Constructeur de copie
91         Loi_de_Tait (const Loi_de_Tait& loi) ;
92
93         // DESTRUCTEUR :
94         ~Loi_de_Tait ();
95
96         // Lecture des donnees de la classe sur fichier
97         void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
98             ,LesFonctions_nD& lesFonctionsnD);
99         // affichage de la loi
100        void Affiche() const ;
101        // test si la loi est complete
102        // = 1 tout est ok, =0 loi incomplete
103        int TestComplet();
104
105        //----- lecture écriture de restart -----
106        // cas donne le niveau de la récupération
107        // = 1 : on récupère tout
108        // = 2 : on récupère uniquement les données variables (supposées comme telles)
109        void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
110            lesCourbes1D
111            ,LesFonctions_nD& lesFonctionsnD);
112        // cas donne le niveau de sauvegarde
113        // = 1 : on sauvegarde tout
114        // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
115        void Ecriture_base_info_loi(ofstream& sort,const int cas);
116
117        // création d'une loi à l'identique et ramène un pointeur sur la loi créée
118        CompThermoPhysiqueAbstraite* Nouvelle_loi_identique() const { return (new Loi_de_Tait(*this)); };
119
120        // affichage et definition interactive des commandes particulières à chaque lois
121        void Info_commande_LoisDeComp(UtilLecture& lec);
122
123        // initialise les donnees particulieres a l'elements
124        // de matiere traite ( c-a-dire au pt calcule)
125        // Il y a creation d'une instance de SaveResul particuliere
126        // a la loi concerne
127        // la SaveResul classe est remplie par les instances heritantes
128        // le pointeur de SaveResul est sauvegarde au niveau de l'element
129        // c'a-d que les info particulieres au point considere sont stocke
130        // au niveau de l'element et non de la loi.
131        class SaveResul_Loi_de_Tait: public SaveResul
132        { public :
133            // le constructeur par défaut
134            SaveResul_Loi_de_Tait();
135            // le constructeur courant
136            // sCrista s'il est NULL, on en tient pas compte
137            // s'il est non NULL, on cré l'instances saveCrista par défaut
138            // idem pour stock
139            SaveResul_Loi_de_Tait(CompThermoPhysiqueAbstraite::StockParaInt* stock
140                ,CristaliniteAbstraite::SaveCrista* sCrista);
141            // constructeur de copie
142            SaveResul_Loi_de_Tait(const SaveResul_Loi_de_Tait& sav );
143            // destructeur
144            ~SaveResul_Loi_de_Tait();
145            // définition d'une nouvelle instance identique
146            // appelle du constructeur via new
147            SaveResul * Nevez_SaveResul() const {return (new SaveResul_Loi_de_Tait(*this));};
148            // affectation
149            virtual SaveResul & operator = ( const SaveResul & a);

```



```

149 //===== lecture écriture dans base info =====
150 // cas donne le niveau de la récupération
151 // = 1 : on récupère tout
152 // = 2 : on récupère uniquement les données variables (supposées comme telles)
153 void Lecture_base_info (ifstream& ent,const int cas);
154 // cas donne le niveau de sauvegarde
155 // = 1 : on sauvegarde tout
156 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
157 void Ecriture_base_info(ofstream& sort,const int cas);
158
159 // affichage des infos
160 void Affiche();
161
162 //changement de base de toutes les grandeurs internes tensorielles stockées
163 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
164 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
165 // ici il n'y a pas de données tensorielles donc rien n'a faire
166 // gpH(i) = gamma(i,j) * gH(j)
167 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma){};
168
169 // procedure permettant de completer éventuellement les données particulières
170 // de la loi stockées
171 // au niveau du point d'intégration par exemple: exemple: un repère d'anisotropie
172 // completer est appelé apres sa creation avec les donnees du bloc transmis
173 // peut etre appeler plusieurs fois
174 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
175 ,const CompThermoPhysiqueAbstraite* loi) {};
176
177 // idem sur un ofstream
178 void Affiche(ofstream& sort);
179
180 // mise à jour des informations transitoires en définitif s'il y a convergence
181 // par exemple (pour la plasticité par exemple)
182 void TdtversT();
183 void TversTdt();
184
185 // données protégées
186 // la liste des données protégées de chaque loi
187 // IMPORTANT: a priori, la classe n'a pas à sauvegarder la pression et la température
188 // ce n'est pas son boulot, mais il est plus facile et économique de sauvegarder ces grandeurs et
189 // d'ensuite calculer les grandeurs spécifiques de la classe Loi_de_Tait, plutôt que de sauve
190 // garder toutes les grandeurs spécifiques de la classe Loi_de_Tait
191 CompThermoPhysiqueAbstraite::StockParaInt * stockParaInt;
192 // pointeurs non nulles que s'il y a de la cristallinité
193 CristaliniteAbstraite::SaveCrista* saveCrista;
194 };
195
196 // def d'une instance de données spécifiques, et initialisation
197 SaveResul * New_et_Initialise();
198
199 // affichage des donnees particulieres a l'elements
200 // de matiere traite ( c-a-dire au pt calcule)
201 virtual void AfficheDataSpecif(ofstream& sort,SaveResul * a) const
202 { ((SaveResul_Loi_de_Tait*) a)->Affiche(sort);};
203
204
205 // récupération des grandeurs particulière (hors ddl )
206 // correspondant à liTQ
207 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
208 virtual void Grandeur_particuliere(bool absolue,List_io<TypeQuelconque>&
209 ,CompThermoPhysiqueAbstraite::SaveResul * ,list<int>&);
210 // récupération de la liste de tous les grandeurs particulières
211 // ces grandeurs sont ajoutées à la liste passées en paramètres
212 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
213 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) ;
214
215 //----- dérivées de virtuelle pures -----
216 // ramène les données thermiques définies au point
217 // P: la pression à l'énuméré temps, et P_t la pression au temps t
218 void Cal_donnees_thermiques(const double& P_t,CompThermoPhysiqueAbstraite::SaveResul * saveTP
219 ,const Deformation & def,const double& P,Enum_dure temps,ThermoDonnee&
220 donneeThermique);
221
222 protected :
223 // donnée de la loi
224 double alphaT; // coefficient de dilatation linéaire
225 double compressibilite; // compressibilité = inverse du coefficient de compressibilité
226 double lambda ; // conductivité
227 CourbelD* lambda_temperature; // courbe éventuelle d'évolution de lambda en fonction de la
228 température
229 double cp; // capacité calorifique
230 CourbelD* cp_temperature; // courbe éventuelle d'évolution de cp en fonction de la température
231
232 // définition des différents coefficients
233 double b1s,b2s,b3s,b4s;
234 double b1m,b2m,b3m,b4m;
235 double b5,b6,b7,b8,b9;

```

```

233
234     int type_de_calcul; // =0: le plus simple et pas de cristallinité
235     // =1: calcul du taux de cristallinité, mais pas de dépendance des variables de
ThermoDonnee
236     // avec la cristallinité
237     // =2: calcul du taux de cristallinité, avec dépendance des variables de
ThermoDonnee
238
239     // cristallinité
240     CristaliniteAbstraite* crista; // pointeur éventuellement sur le calcul de la cristallinité
241
242     // indicateur
243     bool sortie_post; // indique si oui ou non on réserve des infos pour le post-traitement
244
245     // méthode interne
246     // remontée aux différentes variables
247     // taux_cris n'est calculé que si crista est non NULL
248     void Calcul_diff_valeurs(const double& pression, double& temp_trans, double & vol_spec);
249     protected :
250
251 // 3) METHODES VIRTUELLES PURES protegees:
252 // calcul des contraintes à un instant t+deltat
253 // les indices t se rapporte au pas précédent, sans indice au temps actuel
254 // virtual void Calcul_SigmaHH
255 //     (TenseurHH & sigHH_t,TenseurBB& DepsBB,DdlElement & tab_ddl
256 //     ,TenseurBB & gijBB_t,TenseurHH & gijHH_t,BaseB& giB,BaseH& gi_H,TenseurBB & epsBB
257 //     ,TenseurBB & delta_epsBB,TenseurBB & gijBB,TenseurHH & gijHH,Tableau <TenseurBB *>& d_gijBB
258 //     ,double& jacobien_0,double& jacobien,TenseurHH & sigHH
259 //     ,EnergieMeca & energ,const EnergieMeca & energ_t,double& module_compressibilite,double&
module_cisaillement
260 //     ,const Met_abstraite::Expli_t_tdt& ex) = 0;
261
262 // calcul du flux et ses variations par rapport aux ddl a t+dt: stockage dans ptIntegThermi et dans
d_flux
263 // calcul également des paramètres thermiques dTP ainsi que des énergies mises en jeux
264 // calcul des énergies thermiques
265 // en entrée: température, gradient de temp, et grandeurs associées, métrique
266 virtual void Calcul_DfluxH_tdt
267 (const double & P_t,PtIntegThermiInterne& ptIntegThermi, const double & P,DdlElement & tab_ddl
268 ,const Deformation & def // prévue pour servir pour l'interpolation
269 , Tableau <CoordonneeB >& d_gradTB,Tableau <CoordonneeH >& d_flux,ThermoDonnee& dTP
270 ,EnergieThermi & energ,const EnergieThermi & energ_t,const Met_abstraite::Impli& ex);
271 };
272 /// @} // end of group
273
274
275 #endif
276
277
278

```

## 7.85 Loi\_iso\_thermo.h

```

1 // FICHER : Loi_iso_thermo.h
2 // CLASSE : Loi_iso_thermo
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32

```

```

33 /*****
34 *   DATE:      14/04/2004
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 * *****/
41 *   BUT:  Définition d'une loi isotrope thermique simple.
42 *         La loi est utilisable quelque soit la dimension.
43 *   *****/
44 *
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   -----
49 *
50 *   *****/
51 *
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *   *****/
58
59 #ifndef LOI_ISO_THERMO_H
60 #define LOI_ISO_THERMO_H
61
62
63 #include "CompThermoPhysiqueAbstraite.h"
64 #include "CristaliniteAbstraite.h"
65
66 /** @defgroup Les_lois_concernant_thermique
67 *
68 *   BUT:  groupe des lois concernant la thermique
69 *
70 *
71 *   \author   Gérard Rio
72 *   \version  1.0
73 *   \date     14/04/2004
74 *   \brief    Définition des lois concernant la thermique
75 *
76 */
77
78 /// @addtogroup Les_lois_concernant_thermique
79 /// @{
80 ///
81
82
83 class Loi_iso_thermo : public CompThermoPhysiqueAbstraite
84 {
85     public :
86         // CONSTRUCTEURS :
87
88         // Constructeur par défaut
89         Loi_iso_thermo ();
90         // Constructeur de copie
91         Loi_iso_thermo (const Loi_iso_thermo& loi) ;
92
93         // DESTRUCTEUR :
94         ~Loi_iso_thermo ();
95
96         // Lecture des donnees de la classe sur fichier
97         void LectureDonneesParticulieres (UtilLecture * ,LesCourbes1D& lesCourbes1D
98                                           ,LesFonctions_nD& lesFonctionsnD);
99
100        // affichage de la loi
101        void Affiche() const ;
102        // test si la loi est complete
103        // = 1 tout est ok, =0 loi incomplete
104        int TestCompleet();
105
106        //----- lecture écriture de restart -----
107        // cas donne le niveau de la récupération
108        // = 1 : on récupère tout
109        // = 2 : on récupère uniquement les données variables (supposées comme telles)
110        void Lecture_base_info_loi(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
111                                  lesCourbes1D
112                                  ,LesFonctions_nD& lesFonctionsnD);
113
114        // cas donne le niveau de sauvegarde
115        // = 1 : on sauvegarde tout
116        // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
117        void Ecriture_base_info_loi(ofstream& sort,const int cas);
118
119        // création d'une loi à l'identique et ramène un pointeur sur la loi créée
120        CompThermoPhysiqueAbstraite* Nouvelle_loi_identique() const { return (new Loi_iso_thermo(*this));}

```

```

};

118
119 // affichage et definition interactive des commandes particulieres a chaque lois
120 void Info_commande_LoisDeComp(UtilLecture& lec);
121
122 // initialise les donnees particulieres a l'elements
123 // de matiere traite ( c-a-dire au pt calcule)
124 // Il y a creation d'une instance de SaveResul particuliere
125 // a la loi concernee
126 // la SaveResul classe est remplie par les instances heritantes
127 // le pointeur de SaveResul est sauvegarde au niveau de l'element
128 // c'a-d que les info particulieres au point considere sont stocke
129 // au niveau de l'element et non de la loi.
130 class SaveResul_Loi_iso_thermo: public SaveResul
131 {public :
132 SaveResul_Loi_iso_thermo(); // constructeur par defaut (a ne pas utiliser)
133 // le constructeur courant
134 SaveResul_Loi_iso_thermo(CompThermoPhysiqueAbstraite::StockParaInt* stock
135 ,CristaliniteAbstraite::SaveCrista* sCrista);
136 // constructeur de copie
137 SaveResul_Loi_iso_thermo(const SaveResul_Loi_iso_thermo& sav );
138 // destructeur
139 ~SaveResul_Loi_iso_thermo();
140 // definition d'une nouvelle instance identique
141 // appelle du constructeur via new
142 SaveResul * Nevez_SaveResul() const {return (new SaveResul_Loi_iso_thermo(*this));};

143 // affectation
144 virtual SaveResul & operator = ( const SaveResul & a);
145 //===== lecture ecriture dans base info =====
146 // cas donne le niveau de la recuperation
147 // = 1 : on recupere tout
148 // = 2 : on recupere uniquement les donnees variables (supposees comme telles)
149 void Lecture_base_info (ifstream& ent,const int cas);
150 // cas donne le niveau de sauvegarde
151 // = 1 : on sauvegarde tout
152 // = 2 : on sauvegarde uniquement les donnees variables (supposees comme telles)
153 void Ecriture_base_info(ofstream& sort,const int cas);
154
155 // affichage des infos
156 void Affiche();
157
158 //changement de base de toutes les grandeurs internes tensorielles stockees
159 // beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
160 // gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
161 // ici il n'y a pas de donnees tensorielles donc rien n'a faire
162 // gpH(i) = gamma(i,j) * gH(j)
163 virtual void ChBase_des_grandeurs(const Mat_pleine& beta,const Mat_pleine& gamma){};
164
165 // procedure permettant de completer eventuellement les donnees particulieres
166 // de la loi stockees
167 // au niveau du point d'integration par exemple: exemple: un repere d'anisotropie
168 // completer est appele apres sa creation avec les donnees du bloc transmis
169 // peut etre appele plusieurs fois
170 SaveResul* Complete_SaveResul(const BlocGen & bloc, const Tableau <Coordonnee>& tab_coor
171 ,const CompThermoPhysiqueAbstraite* loi) {};
172
173 // idem sur un ofstream
174 void Affiche(ofstream& sort);
175
176 // mise a jour des informations transitoires en definitif s'il y a convergence
177 // par exemple (pour la plasticite par exemple)
178 void TdtversT();
179 void TversTdt ();
180
181 // donnees protegees
182 // la liste des donnees protegees de chaque loi
183 // IMPORTANT: a priori, la classe n'a pas a sauvegarder la pression et la temperature
184 // ce n'est pas son boulot, mais il est plus facile et economique de sauvegarder ces
grandeurs et
185 // d'ensuite calculer les grandeurs specifiques de la classe Loi_iso_thermo, plutot que de
sauve
186 // garder toutes les grandeurs specifiques de la classe Loi_iso_thermo
187 CompThermoPhysiqueAbstraite::StockParaInt * stockParaInt;
188 // pointeurs non nulles que s'il y a de la cristallinite
189 CristaliniteAbstraite::SaveCrista* saveCrista;
190 };
191
192 // def d'une instance de donnees specifiques, et initialisation
193 SaveResul * New_et_Initialise();
194
195 // affichage des donnees particulieres a l'elements
196 // de matiere traite ( c-a-dire au pt calcule)
197 virtual void AfficheDataSpecif(ofstream& sort,SaveResul * a) const
198 { ((SaveResul_Loi_iso_thermo*) a)->Affiche(sort);};
199
200

```

```

201 // récupération des grandeurs particulière (hors ddl )
202 // correspondant à liTQ
203 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
204 virtual void Grandeur_particuliere(bool absolue,List_io<TypeQuelconque>&
,CompThermoPhysiqueAbstraite::SaveResul * ,list<int>&);
205 // récupération de la liste de tous les grandeurs particulières
206 // ces grandeurs sont ajoutées à la liste passées en paramètres
207 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
208 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) ;
209
210
211 //----- dérivées de virtuelle pures -----
212 // ramène les données thermiques définies au point
213 // P: la pression à l'énuméré temps, et P_t la pression au temps t
214 void Cal_donnees_thermiques(const double& P_t,CompThermoPhysiqueAbstraite::SaveResul * saveTP
215 ,const Deformation & def,const double& P,Enum_dure temps,ThermoDonnee&
donneeThermique);
216
217
218 protected :
219
220 // calcul du flux et ses variations par rapport aux ddl a t+dt: stockage dans ptIntegThermi et
dans d_flux
221 // calcul des énergies thermiques
222 // en entrée: température, gradient de temp, et grandeurs associées, métrique
223 virtual void Calcul_DfluxH_tdt
224 (const double & P_t,PtIntegThermiInterne& ptIntegThermi, const double & P,DdlElement &
tab_ddl
225 ,const Deformation & def // prévue pour servir pour l'interpolation
226 , Tableau <CoordonneeB >& d_gradTB,Tableau <CoordonneeH >& d_flux,ThermoDonnee& dTP
227 ,EnergieThermi & energy,const EnergieThermi & energ_t,const Met_abstraite::Impli& ex);
228
229
230
231 protected :
232 // donnée de la loi
233 double alphaT; // coefficient de dilatation
234 CourbeID* alphaT_temperature; // courbe éventuelle d'évolution de alphaT en fonction de la
température
235 double compressibilite; // compressibilité
236 CourbeID* compressibilite_temperature; // courbe éventuelle d'évolution de la compressibilité
237 //en fonction de la température
238 double lambda ; // conductivité
239 CourbeID* lambda_temperature; // courbe éventuelle d'évolution de lambda en fonction de la
température
240 double cp; // capacité calorifique
241 CourbeID* cp_temperature; // courbe éventuelle d'évolution de cp en fonction de la température
242
243 int type_de_calcul; // =0: le plus simple et pas de cristallinité
244 // =1: calcul du taux de cristallinité, mais pas de dépendance des variables
de ThermoDonnee
245 // avec la cristallinité
246 // =2: calcul du taux de cristallinité, avec dépendance des variables de
ThermoDonnee
247
248 // cristallinité
249 CristaliniteAbstraite* crista; // pointeur éventuellement sur le calcul de la cristallinité
250
251 // indicateur
252 bool sortie_post; // indique si oui ou non on réserve des infos pour le post-traitement
253
254 private :
255
256
257
258
259 };
260 /// @} // end of group
261
262
263 #endif
264
265
266

```

## 7.86 CristaliniteAbstraite.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.

```

```

7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:          04/03/2008
33 *
34 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:        Herezh++
37 *
38 *
39 *      BUT:           Interface pour les types de calcul de cristallinité.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date !      auteur !          but
45 *      -----
46 *      !           !          !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !      auteur !          but
51 *      -----
52 *
53 *****/
54 // FICHER : CristaliniteAbstraite.h
55 // CLASSE : CristaliniteAbstraite
56
57 /// pour les calculs de cristallinité
58
59
60 #ifndef CRISTALINITEABSTRAITE_H
61 #define CRISTALINITEABSTRAITE_H
62
63
64 #include "Enum_crista.h"
65 #include "Tableau_T.h"
66 #include "LesCourbes1D.h"
67 #include "Enum_dure.h"
68 #include "LesFonctions_nD.h"
69
70 /// @addtogroup Les_lois_concernant_thermique
71 /// @{
72 ///
73
74
75 class CristaliniteAbstraite
76 {
77
78     public :
79
80     // CONSTRUCTEURS :
81
82     // Constructeur par défaut
83     CristaliniteAbstraite () : id_crista(CRISTA_PAS_DEFINI),saveCrista(NULL) {};
84
85     // Constructeur utile si l'identificateur du nom est connu
86     CristaliniteAbstraite (Enum_crista id_crita) : id_crista(id_crita),saveCrista(NULL) {};
87
88     // Constructeur utile si le nom est connu
89     CristaliniteAbstraite (char* nom) : id_crista(Id_nom_Enum_crista(nom)),saveCrista(NULL) {};
90
91     // Constructeur de copie
92     CristaliniteAbstraite (const CristaliniteAbstraite & a )

```

```

93     : id_crista(a.id_crista),saveCrista(a.saveCrista->Nevez_SaveCrista()) {};
94
95     // DESTRUCTEUR VIRTUEL :
96
97     virtual ~CristaliniteAbstraite () {};
98
99
100 // 2) METHODES VIRTUELLES public:
101
102 // ----- données spécifiques éventuelles de la loi -----
103     // initialise les donnees particulieres a l'elements
104     // de matiere traite ( c-a-dire au pt calcule)
105     // Il y a creation d'une instance de SaveCrista particuliere
106     // a la loi concerne
107     // la SaveCrista classe est remplie par les instances heritantes
108     // le pointeur de SaveCrista est sauvegarde au niveau de l'element
109     // c'a-d que les info particulieres au point considere sont stocke
110     // au niveau de l'element et non de la loi.
111     class SaveCrista
112     { public :
113         // destructeur virtuelle
114         virtual ~SaveCrista() {};
115         // définition d'une nouvelle instance identique
116         // appelle du constructeur via new
117         virtual SaveCrista * Nevez_SaveCrista() const =0;
118         //===== lecture écriture dans base info =====
119         // cas donne le niveau de la récupération
120         // = 1 : on récupère tout
121         // = 2 : on récupère uniquement les données variables (supposées comme telles)
122         virtual void Lecture_base_info (ifstream& ent,const int cas) = 0;
123         // cas donne le niveau de sauvegarde
124         // = 1 : on sauvegarde tout
125         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
126         virtual void Ecriture_base_info(ofstream& sort,const int cas) = 0;
127
128         // mise à jour des informations transitoires en définitif s'il y a convergence
129         // par exemple (pour la plasticité par exemple)
130         virtual void TdtversT(=0);
131         virtual void TversTdt(=0);
132         // affichage des infos
133         virtual void Affiche() = 0;
134         // idem sur ofstream
135         virtual void Affiche(ofstream& sort)=0;
136         // Surcharge de l'operateur =
137         virtual SaveCrista& operator= (const SaveCrista& a) = 0;
138     };
139
140     // création d'une instance du même type
141     virtual SaveCrista * New_et_Initialise() { return NULL;};
142     // création d'une instance identique
143     virtual SaveCrista * New_et_Initialise(const SaveCrista& ) { return NULL;};
144
145     // affichage des donnees particulieres a l'elements
146     // de matiere traite ( c-a-dire au pt calcule)
147     virtual void AfficheDataSpecif(ofstream& ,SaveCrista * ) const {};
148
149 // ----- fin données spécifiques éventuelles de la loi -----
150
151     // affichage de la loi
152     virtual void Affiche() const = 0;
153
154     // Lecture des donnees de la classe sur fichier
155     virtual void LectureDonneesLoiCrista(UtilLecture * entreePrinc,LesCourbes1D& ,LesFonctions_nD& ) =
156     0;
157
158     // affichage et definition interactive des paramètres
159     virtual void Info_commande_LoisCrista(UtilLecture& entreePrinc) = 0;
160
161     // calcul de la fonction K(T,P)
162     virtual double fct_KT(const double& P, const double& T ) const = 0;
163
164     // calcul du taux de cristalinité à tdt à partir de sa valeur à t
165     // P_t, P : pression à t et, t+deltat (= actuelle)
166     // T_t, T : température à t et, t+deltat (=actuelle)
167     virtual double Cristalinite
168     (const double& P_t, const double& T_t,CristaliniteAbstraite::SaveCrista * saveTP
169     ,const double& P, const double& T,Enum_dure temps) = 0 ;
170
171     // calcul du taux de cristalinité à tdt à partir des grandeurs à tdt
172     // P : pression à t+deltat (= actuelle)
173     // T : température à t+deltat (=actuelle)
174     virtual double Cristalinite
175     (CristaliniteAbstraite::SaveCrista * saveTP,const double& P, const double& T) = 0 ;
176
177     //----- lecture écriture de restart spécifique aux données de la classe -----
178     // cas donne le niveau de la récupération
179     // = 1 : on récupère tout

```

```

179 // = 2 : on récupère uniquement les données variables (supposées comme telles)
180 virtual void Lecture_don_base_info(ifstream& ,const int ,LesCourbesID& ,LesFonctions_nD& ) = 0;
181 // cas donne le niveau de sauvegarde
182 // = 1 : on sauvegarde tout
183 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
184 virtual void Ecriture_don_base_info(ofstream& ,const int ) const = 0;
185
186 //----- non virtuelle -----
187 // le type
188 Enum_crista Type_Crista() const {return id_crista;};
189 // pointeur de travail utilise par les classes derivantes
190 SaveCrista * saveCrista;
191
192 // ----- static -----
193
194 // ramène un pointeur correspondant au type de crista passé en paramètre
195 // IMPORTANT : il y a création d'une instance (utilisation d'un new)
196 static CristaliniteAbstraite* New_Cristalinite(Enum_crista typeCrista);
197 // ramène un pointeur sur une instance copie de celle passée en paramètre
198 // IMPORTANT : il y a création d'une instance (utilisation d'un new)
199 static CristaliniteAbstraite* New_Cristalinite(const CristaliniteAbstraite& Co);
200
201 // VARIABLES PROTEGEES :
202
203 protected:
204 // l'identificateur de la classe calculant la cristallinité
205 Enum_crista id_crista;
206 };
207 /// @} // end of group
208
209
210 #endif

```

## 7.87 Hoffman1.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:          24/01/2008
33 *
34 *      AUTEUR:        G RIO      (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:        Herezh++
37 *
38 *
39 *      BUT: Calcul de la constante cinétique à l'aide de la loi
40 *          d'Hoffman. Cette constante est ensuite utilisée pour
41 *          le calcul de la cristallinité à l'aide de la forme de Nakamura.*
42 *      la loi de Hoffman1 permet le calcul de K(T,P), puis du taux c
43 *      la loi nécessite 11 coefficients
44 *          n,G0,N01,N02,Ustar,Tinf,Kg,Tm,alphaP,betaP,X_inf
45 *      le calcul de K(T,P) s'effectue alors suivant la formule:
46 *      K(T,P) = G0*((4.0/3.0*PI*N0)**(1/3))
47 *              *exp(- Ustar/(R*(Tcorr - Tinf)))*exp(- Kg/(Tcorr*(Tm - Tcorr)))*
48 *      avec les relations:
49 *          DTcorr_Pres = P*(alphaP + betaP*P);
50 *          Tcorr      = T - DTcorr_Pres + 273.;

```



```

51 *      NO      = exp(N01*(Tm - Tcorr) + N02);      *
52 *      et: P la pression relative, T la temperature celsius      *
53 *      R la constantes des gaz parfaits      *
54 *      $      *
55 *      *****      *
56 *      VERIFICATION:      *
57 *      *
58 *      ! date ! auteur ! but !      *
59 *      -----      *
60 *      ! ! ! ! !      *
61 *      $      *
62 *      *****      *
63 *      MODIFICATIONS:      *
64 *      ! date ! auteur ! but !      *
65 *      -----      *
66 *      $      *
67 *      *****/
68 #ifndef HOFFMAN1_H
69 #define HOFFMAN1_H
70
71 #include "UtilLecture.h"
72 #include "LesCourbes1D.h"
73 #include "CristaliniteAbstraite.h"
74 #include "Algo_Integ1D.h"
75
76 /// @addtogroup Les_lois_concernant_thermique
77 /// @{
78 ///
79
80
81 class Hoffman1 : public CristaliniteAbstraite
82 {
83 public :
84 // CONSTRUCTEURS :
85 // constructeur par défaut
86 Hoffman1();
87 // constructeur de copie
88 Hoffman1(const Hoffman1& co );
89
90 // DESTRUCTEUR :
91 ~Hoffman1() {};
92
93 // METHODES PUBLIQUES :
94
95 // 2) METHODES VIRTUELLES public:
96
97 // ----- données spécifiques éventuelles de la loi -----
98 // initialise les donnees particulieres a l'elements
99 // de matiere traite ( c-a-dire au pt calcule)
100 // Il y a creation d'une instance de SaveCrista particuliere
101 // a la loi concerne
102 // la SaveCrista classe est remplie par les instances heritantes
103 // le pointeur de SaveCrista est sauvegarde au niveau de l'element
104 // c'a-d que les info particulieres au point considere sont stocke
105 // au niveau de l'element et non de la loi.
106 class SaveCrista_Hoffman1: public SaveCrista
107 { public :
108 SaveCrista_Hoffman1();
109 // le constructeur courant
110 SaveCrista_Hoffman1(const double & I_Kcinetique_t,const double & I_Kcinetique);
111 // constructeur de copie
112 SaveCrista_Hoffman1(const SaveCrista_Hoffman1& sav );
113 // destructeur
114 ~SaveCrista_Hoffman1() {};
115 // définition d'une nouvelle instance identique
116 // appelle du constructeur via new
117 SaveCrista * Nevez_SaveCrista() const {return (new SaveCrista_Hoffman1(*this));};
118 //===== lecture écriture dans base info =====
119 // cas donne le niveau de la récupération
120 // = 1 : on récupère tout
121 // = 2 : on récupère uniquement les données variables (supposées comme telles)
122 void Lecture_base_info (ifstream& ent,const int cas);
123 // cas donne le niveau de sauvegarde
124 // = 1 : on sauvegarde tout
125 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
126 void Ecriture_base_info(ofstream& sort,const int cas);
127
128 // affichage des infos
129 void Affiche();
130 // idem sur un ofstream
131 void Affiche(ofstream& sort);
132
133 // mise à jour des informations transitoires en définitif s'il y a convergence
134 // par exemple (pour la plasticité par exemple)
135 void TdtversI() {I_Kcinetique_t = I_Kcinetique;};
136 void TversTdt() {I_Kcinetique = I_Kcinetique_t;};
137 // Surcharge de l'operateur =

```

```

138         SaveCrista& operator= (const SaveCrista& a);
139
140         // données protégées
141         // c'est l'intégrale de la constante cinétique que l'on sauvegarde
142         double I_Kcinetique_t,I_Kcinetique;
143     };
144
145     // def d'une instance de données spécifiques, et initialisation
146     SaveCrista * New_et_Initialise() {return (new SaveCrista_Hoffman1());};
147
148 // ----- fin données spécifiques éventuelles de la loi -----
149
150 // Lecture des donnees de la classe sur fichier
151 void LectureDonneesLoiCrista (UtilLecture * entreePrinc
152     ,LesCourbes1D& ,LesFonctions_nD& lesFonctionsnD);
153
154 // affichage de la loi
155 void Affiche() const;
156
157 // affichage et definition interactive des paramètres
158 void Info_commande_LoisCrista(UtilLecture& entreePrinc);
159
160 // calcul de la fonction K(T,P)
161 double fct_KT(const double& P, const double& T ) const;
162 // calcul du taux de cristallinité
163 // P_t, P : pression à t et, t+deltat (= actuelle)
164 // T_t, T : température à t et, t+deltat (=actuelle)
165 double Cristallinite(const double& P_t, const double& T_t
166     ,CristalliniteAbstraite::SaveCrista * saveTP
167     ,const double& P, const double& T,Enum_dure temps);
168
169 // calcul du taux de cristallinité à tdt à partir des grandeurs à tdt
170 // P : pression à t+deltat (= actuelle)
171 // T : température à t+deltat (=actuelle)
172 double Cristallinite
173     (CristalliniteAbstraite::SaveCrista * saveTP,const double& P, const double& T);
174
175 //----- lecture écriture de restart spécifique aux données de la classe -----
176 // cas donne le niveau de la récupération
177 // = 1 : on récupère tout
178 // = 2 : on récupère uniquement les données variables (supposées comme telles)
179 void Lecture_don_base_info(ifstream& ,const int cas ,LesCourbes1D& ,LesFonctions_nD& ) ;
180 // cas donne le niveau de sauvegarde
181 // = 1 : on sauvegarde tout
182 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
183 void Ecriture_don_base_info(ofstream& sort,const int cas) const ;
184
185 private :
186     // VARIABLES PROTEGEES :
187
188     // coefficients de la loi
189     double n,G0,N01,N02,Ustar,Tinf,Kg,Tm,alphaP,betaP,X_inf;
190
191     // classe pour l'intégration
192     Algo_Integ1D* algo_integ;
193     // variable pour passage à la fonction d'intégration fct_KT_
194     double Te_t,Te_tdt,Pr_t,Pr_tdt;
195
196     // METHODES PROTEGEES :
197     // calcul de la fonction K(t) en fonction du temps et des variables
198     // définit par ailleurs: Te_t,Te_tdt,Pr_t,Pr_tdt
199     double fct_KT_(const double& t );
200
201 };
202 /// @} // end of group
203
204 #endif

```

## 7.88 Hoffman2.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //

```

```

16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           24/01/2008
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 *   *****/
39 *   BUT: Calcul de la constante cinétique à l'aide de la loi
40 *   d'Hoffman. Cette constante est ensuite utilisée pour
41 *   le calcul de la cristalinité à l'aide de la forme de Nakamura.
42 *   la loi de Hoffman2 permet le calcul de K(T,P), puis du taux c
43 *   la loi necessite 9 coefficients
44 *   n, invt012, Ustar, Tinf, Kg, Tm, alphaP, betaP, X_inf
45 *   le calcul de K(T,P) s'effectue alors suivant la formule:
46 *    $K(T,P) = (\log(2))^{**}(1/n) * \text{invt012} * \exp(-Ustar/(R*(Tcorr - Tinf)))$ 
47 *   *  $\exp(-Kg/(Tcorr * (Tm - Tcorr) * f))$ 
48 *   avec  $f=2*Tcorr/(Tcorr + Tm)$  et
49 *    $Tcorr = T$  (en Kelvin) -  $P*(alphaP + betaP*P) + 273$ 
50 *   avec les relations:
51 *    $DTcorr\_Pres = P*(alphaP + betaP*P);$ 
52 *    $Tcorr = T - DTcorr\_Pres + 273.;$ 
53 *    $N0 = \exp(N01*(Tm - Tcorr) + N02);$ 
54 *   et: P la pression relative, T la temperature celsius
55 *   R la constante des gaz parfaits
56 *
57 *   *****
58 *   VERIFICATION:
59 *
60 *   ! date !   auteur !   but
61 *   -----
62 *   !     !     !
63 *
64 *   *****
65 *   MODIFICATIONS:
66 *   ! date !   auteur !   but
67 *   -----
68 *
69 *****/
70 #ifndef HOFFMAN2_H
71 #define HOFFMAN2_H
72
73 #include "UtilLecture.h"
74 #include "LesCourbes1D.h"
75 #include "CristaliniteAbstraite.h"
76 #include "Algo_Integ1D.h"
77
78 /// @addtogroup Les_lois_concernant_thermique
79 /// @{
80 ///
81
82
83 class Hoffman2 : public CristaliniteAbstraite
84 {
85 public :
86     // CONSTRUCTEURS :
87     // constructeur par défaut
88     Hoffman2();
89     // constructeur de copie
90     Hoffman2(const Hoffman2& co );
91
92     // DESTRUCTEUR :
93     ~Hoffman2() {};
94
95     // METHODES PUBLIQUES :
96
97 // 2) METHODES VIRTUELLES public:
98
99 // ----- données spécifiques éventuelles de la loi -----
100 // initialise les donnees particulieres a l'elements
101 // de matiere traite ( c-a-dire au pt calcule)
102 // Il y a creation d'une instance de SaveCrista particuliere

```

```

103 // a la loi concernee
104 // la SaveCrista classe est remplie par les instances heritantes
105 // le pointeur de SaveCrista est sauvegarde au niveau de l'element
106 // c'a-d que les info particulieres au point considere sont stocke
107 // au niveau de l'element et non de la loi.
108 class SaveCrista_Hoffman2: public SaveCrista
109 { public :
110     SaveCrista_Hoffman2();
111     // le constructeur courant
112     SaveCrista_Hoffman2(const double & I_Kcinetique_t,const double & I_Kcinetique);
113     // constructeur de copie
114     SaveCrista_Hoffman2(const SaveCrista_Hoffman2& sav );
115     // destructeur
116     ~SaveCrista_Hoffman2() {};
117     // definition d'une nouvelle instance identique
118     // appelle du constructeur via new
119     SaveCrista * Nevez_SaveCrista() const {return (new SaveCrista_Hoffman2(*this));};
120     //===== lecture écriture dans base info =====
121     // cas donne le niveau de la récupération
122     // = 1 : on récupère tout
123     // = 2 : on récupère uniquement les données variables (supposées comme telles)
124     void Lecture_base_info (ifstream& ent,const int cas);
125     // cas donne le niveau de sauvegarde
126     // = 1 : on sauvegarde tout
127     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
128     void Ecriture_base_info(ofstream& sort,const int cas);
129
130     // affichage des infos
131     void Affiche();
132     // idem sur un ofstream
133     void Affiche(ofstream& sort);
134
135     // mise à jour des informations transitoires en définitif s'il y a convergence
136     // par exemple (pour la plasticité par exemple)
137     void TdtversT() {I_Kcinetique_t = I_Kcinetique;};
138     void TversTdt() {I_Kcinetique = I_Kcinetique_t;};
139     // Surcharge de l'operateur =
140     SaveCrista& operator= (const SaveCrista& a);
141
142     // données protégées
143     // c'est l'intégrale de la constante cinétique que l'on sauvegarde
144     double I_Kcinetique_t,I_Kcinetique;
145 };
146
147 // def d'une instance de données spécifiques, et initialisation
148 SaveCrista * New_et_Initialise() {return (new SaveCrista_Hoffman2());};
149
150 // ----- fin données spécifiques éventuelles de la loi -----
151
152 // Lecture des donnees de la classe sur fichier
153 void LectureDonneesLoiCrista (UtilLecture * entreePrinc,LesCourbes1D& ,LesFonctions_nD&);
154
155 // affichage de la loi
156 void Affiche() const;
157
158 // affichage et definition interactive des paramètres
159 void Info_commande_LoisCrista(UtilLecture& entreePrinc);
160
161 // calcul de la fonction K(T,P)
162 double fct_KT(const double& P, const double& T ) const;
163 // calcul du taux de cristallinité
164 // P_t, P : pression à t et, t+deltat (= actuelle)
165 // T_t, T : température à t et, t+deltat (=actuelle)
166 double Cristallinite(const double& P_t, const double& T_t
167     ,CristalliniteAbstraite::SaveCrista * saveTP
168     ,const double& P, const double& T,Enum_dure temps);
169
170 // calcul du taux de cristallinité à tdt à partir des grandeurs à tdt
171 // P : pression à t+deltat (= actuelle)
172 // T : température à t+deltat (=actuelle)
173 double Cristallinite
174     (CristalliniteAbstraite::SaveCrista * saveTP,const double& P, const double& T);
175
176 //----- lecture écriture de restart spécifique aux données de la classe -----
177 // cas donne le niveau de la récupération
178 // = 1 : on récupère tout
179 // = 2 : on récupère uniquement les données variables (supposées comme telles)
180 void Lecture_don_base_info(ifstream& ,const int cas ,LesCourbes1D& ,LesFonctions_nD& ) ;
181 // cas donne le niveau de sauvegarde
182 // = 1 : on sauvegarde tout
183 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
184 void Ecriture_don_base_info(ofstream& sort,const int cas) const ;
185
186 private :
187     // VARIABLES PROTEGEES :
188
189     // coefficients de la loi

```

```

190     double n, invt012, Ustar, Tinf, Kg, Tm, alphaP, betaP, X_inf;
191
192     // classe pour l'intégration
193     Algo_IntegID* algo_integ;
194     // variable pour passage à la fonction d'intégration fct_KT_
195     double Te_t, Te_tdt, Pr_t, Pr_tdt;
196
197     // METHODES PROTEGEES :
198     // calcul de la fonction K(t) en fonction du temps et des variables
199     // définit par ailleurs: Te_t, Te_tdt, Pr_t, Pr_tdt
200     double fct_KT_(const double& t );
201
202 };
203 /// @} // end of group
204
205 #endif

```

## 7.89 ThermoDonnee.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:      13/04/2004
33 *
34 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:    Herezh++
37 *
38 *
39 *      BUT: Conteneur pour stocker des données pour la thermique.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date !   auteur !           but
45 *      -----
46 *      !           !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      -----
52 *
53 *
54 #ifndef THERMODONNEE_H
55 #define THERMODONNEE_H
56
57 #include <iostream>
58 #include <fstream>
59 #include <stdlib.h>
60 #include "Sortie.h"
61
62
63 /// @addtogroup Les_conteneurs_energies
64 /// @{
65 ///

```

```

66
67 class ThermoDonnee
68 {
69     // surcharge de l'operator de lecture avec le type
70     friend istream & operator » (istream &, ThermoDonnee &);
71     // surcharge de l'operator d'écriture
72     friend ostream & operator « (ostream &, const ThermoDonnee &);
73
74 public :
75     // CONSTRUCTEURS :
76     // par défaut
77     ThermoDonnee() :
78         alphaT(0.),active_dilatation(false),lambda(0.),cp(0.),compressibilite(0.)
79         ,taux_crista(NULL) {};
80
81     // de copie
82     ThermoDonnee(const ThermoDonnee& co) :
83         alphaT(co.alphaT),active_dilatation(co.active_dilatation),lambda(co.lambda)
84         ,cp(co.cp),compressibilite(co.compressibilite)
85         ,taux_crista(new double)
86         {*taux_crista = *(co.taux_crista)};
87
88
89     // DESTRUCTEUR :
90     ~ThermoDonnee() { if (taux_crista != NULL) delete taux_crista;};
91
92     // METHODES PUBLIQUES :
93     // retourne le coefficient de dilatation
94     const double& Dilatation() const {return alphaT;};
95     // retourne la conductivité
96     const double& Conductivite() const {return lambda;};
97     // retourne la capacité calorifique
98     const double & CapaciteCalorifique() const {return cp;};
99     // retourne la compressibilité
100    const double & Compressibilite() const {return compressibilite;};
101    // indique si oui ou non la dilatation est active
102    bool ActiveDilatation() const {return active_dilatation;};
103    // retourne un pointeur sur taux de cristallinité. Si le pointeur est nul
104    // indique qu'il n'y a pas de taux de cristallinité
105    const double * TauxCrista() const {return taux_crista;};
106
107    // changement du coefficient de dilatation
108    void ChangeDilatation(double dila) { alphaT = dila;};
109    // changement de: alphaT, lambda, cp
110    void ChangeAlphaTLambdaCp(const double& alph, const double & lamb, const double & ccc)
111        { alphaT=alph; lambda=lamb; cp=ccc;};
112    void ChangeCompressibilite(const double compress)
113        { compressibilite = compress;};
114    void ChangeTauxCrista(const double taux_cris)
115        { if (taux_crista == NULL) {taux_crista = new double;};
116          *taux_crista = taux_cris;
117        };
118
119 private :
120     // VARIABLES PROTEGEES :
121     double alphaT; // coefficient de dilatation
122     bool active_dilatation;
123     double lambda ; // conductivité
124     double cp; // capacité calorifique
125     double compressibilite; // compressibilité
126     double* taux_crista; // taux de cristallinité: pointeur null si ne sert pas
127
128
129     // METHODES PROTEGEES :
130
131 };
132 /// @} // end of group
133
134 #endif

```

## 7.90 Cercle.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr

```

```

14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      19/01/2007
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:      Def géométrie d'un cercle: un centre , un rayon
39 *             et une normale au plan du cercle dans le cas 3D
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   -----
47 *   !           !           !           !
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *****/
54 #ifndef CERCLEE_H
55 #define CERCLEE_H
56
57 #include "Droite.h"
58 #include "Coordonnee.h"
59
60 /// @addtogroup Groupe_sur_les_contacts
61 /// @{
62 ///
63
64
65 class Cercle
66 {
67     // surcharge de l'operator de lecture
68     friend istream & operator » (istream &, Cercle &);
69     // surcharge de l'operator d'écriture
70     friend ostream & operator « (ostream &, const Cercle &);
71
72 public :
73     // CONSTRUCTEURS :
74     // par défaut
75     Cercle ();
76     // avec les datas
77     Cercle ( const Coordonnee& B, const double& r, const Coordonnee* N);
78     // avec la dimension
79     Cercle (int dim);
80     // de copie
81     Cercle ( const Cercle& a);
82     // DESTRUCTEUR :
83     ~Cercle ();
84     // surcharge des operator
85     Cercle& operator = ( const Cercle & P);
86
87     // METHODES PUBLIQUES :
88     // retourne le centre du Cercle
89     inline const Coordonnee& CentreCercle() const { return centre;};
90     // retourne le rayon du Cercle
91     inline double RayonCercle() const { return rayon;};
92     // retourne la normale au cercle en 3D sinon NULL
93     inline const Coordonnee* NormaleCercle() const { return N;};
94
95     // change le centre du Cercle
96     void Change_centre( const Coordonnee& B);
97     // change le rayon du Cercle
98     void Change_rayon( const double & r);
99     // change la normale au Cercle en 3D, sinon c'est inutile, warning
100    void Change_Normale( const Coordonnee& N);

```

```

100 // change toutes les donnees
101 // N n'est utilisé qu'en 3D, en 2D on met NULL
102 void change_donnees( const Coordonnee& B, const double& r, const Coordonnee* N);
103
104 // calcul les deux interceptions M1 et M2 d'une droite avec le Cercle,
105 // ramene 0 s'il n'y a pas d'intersection, ramene -1 si l'intercection
106 // ne peut pas etre calculee
107 // et 1 s'il y a deux points d'intercection
108 int Intersection( const Droite & D, Coordonnee& M1, Coordonnee& M2);
109
110 // calcul la distance d'un point au Cercle
111 double Distance_au_Cercle(const Coordonnee& M) const;
112
113 // calcul du projeté d'un point sur le plan
114 Coordonnee Projete(const Coordonnee& M) const;
115
116 // indique si oui ou non le projeté du point est dans le cercle
117 bool ProjeteDedans(const Coordonnee& M) const
118 { return ((Projete(M)-centre).Carre() < rayon*rayon);};
119
120 // fonction valable uniquement en 2D, sinon erreur
121 // ramène true si le point est à l'intérieur du cercle, false sinon
122 bool Dedans(const Coordonnee& M) const ;
123
124
125
126 protected :
127 // VARIABLES PROTEGEES :
128 Coordonnee centre; // centre du Cercle
129 double rayon; // rayon du Cercle
130 Coordonnee* N; // normal au cercle : utilisé qu'en 3D
131 // la grandeur stockée est normée
132 // METHODES PROTEGEES :
133
134 };
135 /// @} // end of group
136
137 #endif

```

## 7.91 Cylindre.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      19/01/2007
32 *
33 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:      Def géométrie d'un cylindre: un cercle 3D, donc avec une
39 *             normale. Le cylindre n'est pas limité en hauteur.
40 *
41 *   *****
42 *   VERIFICATION:
43 *

```



```

44 *      ! date !   auteur !           but           !      *
45 *      -----
46 *      !           !           !           !           !      *
47 *      $           $           $           $           $      *
48 *      ***** *
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but           !      *
51 *      -----
52 *      $           $           $           $           $      *
53 *****/
54 #ifndef CYLINDREE_H
55 #define CYLINDREE_H
56
57 #include "Droite.h"
58 #include "Coordonnee.h"
59 #include "Cercle.h"
60
61 /// @addtogroup Groupe_sur_les_contacts
62 /// @{
63 ///
64
65
66 class Cylindre
67 { // surcharge de l'operator de lecture
68     friend istream & operator » (istream &, Cylindre &);
69     // surcharge de l'operator d'écriture
70     friend ostream & operator « (ostream &, const Cylindre &);
71
72 public :
73     // CONSTRUCTEURS :
74     // par défaut
75     Cylindre ();
76     // avec les datas
77     Cylindre ( const Cercle& B);
78     // avec la dimension
79     Cylindre (int dim);
80     // de copie
81     Cylindre ( const Cylindre& a);
82     // DESTRUCTEUR :
83     ~Cylindre ();
84     // surcharge des operator
85     Cylindre& operator = ( const Cylindre & P);
86
87     // METHODES PUBLIQUES :
88     // retourne le cercle d'embase du Cylindre
89     inline const Cercle& CercleCylindre() const { return cercle;};
90
91     // change le centre du Cylindre
92     void Change_cercle( const Cercle& B);
93     // change toutes les donnees
94     void change_donnees( const Cercle& cer);
95
96     // calcul les deux intercections M1 et M2 d'une droite avec le Cylindre,
97     // ramene 0 s'il n'y a pas d'intersection, ramene -1 si l'intercection
98     // ne peut pas etre calculee
99     // et 1 s'il y a deux points d'intercection
100    int Intersection( const Droite & D,Coordonnee& M1, Coordonnee& M2);
101
102    // calcul la distance d'un point au Cylindre
103    double Distance_au_Cylindre(const Coordonnee& M) const;
104
105    // ramène true si le point est à l'intérieur du cylindre, false sinon
106    bool Dedans(const Coordonnee& M)const ;
107
108    // projection d'un point M sur la parois du cylindre
109    // dans le cas où M appartient à l'axe du cylindre, la projection n'est pas
110    // possible, dans ce cas projection_ok = false en retour, sinon true
111    Coordonnee Projete(const Coordonnee& M,bool& projection_ok) const;
112
113 protected :
114     // VARIABLES PROTEGEES :
115     Cercle cercle; // cercle du Cylindre
116     // METHODES PROTEGEES :
117
118 };
119 /// @} // end of group
120
121 #endif

```

## 7.92 Droite.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field

```

```

5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:  Def de la geometrie d'une droite: un point et un vecteur
39 *         directeur.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   -----
47 *   !           !           !
48 *   *****
49 *
50 *   MODIFICATIONS:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *
55 *   *****/
54 #ifndef DROITE_H
55 #define DROITE_H
56
57 #include "Coordonnee.h"
58
59 /// @addtogroup Groupe_sur_les_contacts
60 /// @{
61 ///
62
63
64 class Droite
65 {
66     // surcharge de l'operator de lecture
67     friend istream & operator » (istream &, Droite &);
68     // surcharge de l'operator d'écriture
69     friend ostream & operator « (ostream &, const Droite &);
70
71     public :
72     // CONSTRUCTEURS :
73     // par défaut définit une droite // à x et passant par 0, et de dimension celle de l'espace de travail
74     Droite ();
75     // avec les datas
76     // vec est quelconque, non norme
77     Droite ( const Coordonnee& B, const Coordonnee& vec);
78     // avec la dimension: la droite générée par défaut est une droite qui passe par 0 et est // à l'axe
79     // des x
80     Droite (int dim);
81     // de copie
82     Droite ( const Droite& a);
83     // DESTRUCTEUR :
84     ~Droite ();
85     // surcharge des operator
86     Droite& operator = ( const Droite & P);
87
88     // METHODES PUBLIQUES :
89     // retour le point de ref la droite
90     inline const Coordonnee& PointDroite() const { return A;};
91     // retourne le vecteur directeur de la droite (de norme = 1 )

```

```

90     inline const Coordonnee& VecDroite() const { return U;};
91
92     // change la dimension de la droite
93     void Change_dim(int dima) {A.Change_dim(dima);U.Change_dim(dima);};
94     // change le point de ref de la droite
95     void Change_ptref( const Coordonnee& B);
96     // change le vecteur de la droite
97     void Change_vect( const Coordonnee& vec);
98     // change toutes les donnees
99     void change_donnees( const Coordonnee& B, const Coordonnee& vec);
100
101     // calcul l'intercection M de la droite avec une autre droite, ramene 0 s'il n'y
102     // a pas d'Intersection, ramene -1 si l'Intersection ne peut pas etre calculee
103     // et 1 s'il y a un point d'Intersection
104     int Intersection( const Droite & D,Coordonnee& M);
105
106     // calcul si un point donné appartient ou non à la droite (à la précision donnée)
107     bool Appartient_a_la_droite(const Coordonnee& B);
108
109     // ramene une normale, en 2D il y a une solution, en 3D a chaque appel on ramene
110     // une nouvelle normale calculée aleatoirement
111     Coordonnee UneNormale();
112
113     // calcul la distance d'un point à la droite
114     double Distance_a_la_droite(const Coordonnee& M) const;
115
116     // calcul du projeté d'un point sur la droite
117     Coordonnee Projete(const Coordonnee& M) const
118         {return (A+((M-A)*U) * U);};
119
120     // fonction valable uniquement en 2D !!!, sinon erreur
121     // ramène true si les deux points sont du même coté de la droite, false sinon
122     bool DuMemeCote(const Coordonnee& M1, const Coordonnee& M2) const;
123
124
125     protected :
126         // VARIABLES PROTEGEES :
127         Coordonnee A; // un point de la droite
128         Coordonnee U; // vecteur directeur de la droite
129         static int alea; // une variables aleatoire qui sert pour la sortie d'une normale
130         // aleatoire en 3D
131         // METHODES PROTEGEES :
132     };
133     /// @} // end of group
134
135 #endif

```

## 7.93 EIContact.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *

```

```

37 *****
38 *      BUT: Element de contact. *
39 * * * * * $ *
40 * * * * * *
41 *      VERIFICATION: *
42 * * * * * *
43 *      ! date ! auteur ! but ! *
44 *      ----- *
45 *      ! ! ! ! *
46 * * * * * $ *
47 * * * * * *
48 *      MODIFICATIONS: *
49 *      ! date ! auteur ! but ! *
50 *      ----- *
51 * * * * * $ *
52 *****/
53 #ifndef ELCONTACT_H
54 #define ELCONTACT_H
55
56 #include "Front.h"
57 #include "Noeud.h"
58 #include "Condilinaire.h"
59 #include "LesFonctions_nD.h"
60
61
62 /** @defgroup Groupe_sur_les_contacts
63 *
64 *      BUT: groupe relatif aux contacts
65 *
66 *
67 * \author Gérard Rio
68 * \version 1.0
69 * \date 23/01/97
70 * \brief groupe relatif aux contacts
71 *
72 */
73
74 /// @addtogroup Groupe_sur_les_contacts
75 /// @{
76 ///
77
78 class ElContact
79 {
80 public :
81 // une classe structure de transfert pour simplifier le passage de paramètres
82 class Fct_nD_contact
83 { public:
84 Fct_nD_contact();
85 Fct_nD_contact(const Fct_nD_contact& a);
86 ~Fct_nD_contact();
87 Fct_nD_contact& operator= (const Fct_nD_contact& a);
88
89 // utilisation de fct nD
90 Fonction_nD * fct_nD_penalisationPenetration; // fct nD dans le cas d'une valeur pilotée
91 Fonction_nD * fct_nD_penetration_contact_maxi; // fct nD dans le cas d'une valeur pilotée
92 Fonction_nD * fct_nD_penetration_borne_regularisation; // fct nD dans le cas d'une valeur pilotée
93 Fonction_nD * fct_nD_force_contact_noeud_maxi; // fct nD dans le cas d'une valeur pilotée
94 Fonction_nD * fct_nD_penalisationTangentielle;
95 Fonction_nD * fct_nD_tangentielle_contact_maxi; // fct nD dans le cas d'une valeur pilotée
96 Fonction_nD * fct_nD_tangentielle_borne_regularisation; // fct nD dans le cas d'une valeur pilotée
97 Fonction_nD * fct_nD_force_tangentielle_noeud_maxi; // fct nD dans le cas d'une valeur pilotée
98 };
99
100 // CONSTRUCTEURS :
101 // par défaut
102 ElContact();
103
104 // la version avec fonction de pilotage nD
105 ElContact(Fct_nD_contact & fct_contact);
106
107 // fonction d'un pointeur d'element frontiere et d'un pointeur de noeud
108 // du fait éventuel qu'il peut-être collant ou pas
109 ElContact ( const Front * elfront, const Noeud * noeud, Fct_nD_contact & fct_contact_, int collant =
110 0);
111 // de copie
112 ElContact ( const ElContact & a);
113 // DESTRUCTEUR :
114 ~ElContact ();
115
116 // METHODES PUBLIQUES :
117
118 // init d'une fonction nD pour le pilotage du type de contact 4
119 static void Init_fct_pilotage_contact4(Fonction_nD * fct_pilo_contact4)
120 {fct_pilotage_contact4 = fct_pilo_contact4;};
121 // méthode statique de modification éventuelle du type de contact utilisé localement

```

```

122 // par exemple pour le type 4 en fonction des itérations
123 static int Recup_et_mise_a_jour_type_contact() ;
124
125 // affichage à l'écran des informations liées au contact
126 void Affiche() const ;
127
128 // calcul d'un contact eventuel entre le noeud esclave et la frontiere
129 // ramene true s'il y a contact
130 // si init = false, on recherche le contact a partir du precedent point sauvegarde
131 // sinon on commence a l'aide d'element de reference,
132 // et on calcule et sauvegarde les coordonnées
133 // initiale locales theta^i du point de contact
134
135 // si le contact existe et si l'algo le demande (cf. ParaAlgoControle) :
136 // le noeud pourrait-être ramené sur la surface mais:
137 // on ne fait pas de projection, sinon on ne peut pas tester plusieurs contacts pour
138 // choisir le meilleur, puisque les choses changent entre avant et après le test de contact
139 // donc ici la position du noeud esclave n'est pas modifiée
140 bool Contact( bool init = true);
141 // juste après l'utilisation de la méthode Contact(), ramène le point en contact
142 const Coordonnee& Point_intersection() const { return Mtdt; };
143
144 // un stockage utilisé par les méthodes appelantes
145 int& Num_zone_contact() {return num_zone_contact;};
146
147 // calcul de la trajectoire a prendre en compte pour le contact
148 // ---- a) cas ou à t il n'y avait pas de contact, ou que l'on n'a pas fait de projection sur la
149 // surface (cas du contact cinématique)
150 // 4 cas : 1) le noeud bouge, dans ce cas la trajectoire est determinee
151 // par la variation de la position du noeud
152 // 2) le noeud est immobile, mais la frontiere bouge, la trajectoire est determinee
153 // par une parallele a la variation moyenne de la frontiere (variation de G)
154 // 4) est une variante du 2), cas où l'on a une rotation autour de G, dans ce cas on prend comme
155 // trajectoire
156 // le maxi des déplacements de noeud
157 // 3) rien ne bouge, on utilise la normale au point de reference de l'element de frontiere
158 // pour calculer la trajectoire .
159 // dans le cas 3 la variable test = 0 , sinon elle vaut 1 pour le cas 1 et 2 pour le cas 2 , 4 pour le
160 // cas 4
161 // ---- b) cas ou à t on était déjà en contact avec projection sur la surface
162 // la trajectoire est alors systématiquement la direction de la dernière normale,
163 // retour : test=5
164 Coordonnee Trajectoire(int & test);
165
166 // calcul de l'Intersection de la trajectoire du noeud definit par le vecteur V
167 // avec l'element frontiere
168 // ramene les coordonnees du noeud projete
169 // dans le cas où il n'y a pas d'intersection, on ramène un point de dimension nulle
170 Coordonnee Intersection( const Coordonnee& V,bool init);
171
172 // construction de la condition lineaire de contact
173 // nb_assemb : indique le numéro d'assemblage correspondant
174 Condilinaire ConditionLi(int nb_assemb);
175
176 // ramene le tableau de tous les noeuds, le premier est celui esclave
177 Tableau <Noeud*> TabNoeud() {return tabNoeud;};
178 const Tableau <Noeud*> Const_TabNoeud() const {return tabNoeud;};
179 // ramene le tableau pour assemblage: depend du Cas_solide()
180 // est cohérent avec TableauDdlCont
181 Tableau <Noeud*> TabNoeud_pour_assemblage() {return tabNoeud_pour_assemblage;};
182
183 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
184 // qui sont actifs au moment de la demande
185 // Tableau de DdlElement pour l'assemblage uniquement
186 const DdlElement & TableauDdlCont() const {return *ddlElement_assemblage;};
187 // ramene le noeud esclave
188 inline Noeud* Esclave() { return noeud;};
189 // ramene la force de contact pour consultation
190 const Coordonnee& Force_contact() const {return force_contact;};
191 // ramène les maxis concernant le noeud esclave
192 double F_N_MAX() const {return F_N_max;};
193 double F_T_MAX() const {return F_T_max;};
194 // ramène la pénalisation actuelle éventuelle
195 const double& Penalisation() const {return penalisation;};
196 const double& Penalisation_tangentielle() const {return penalisation_tangentielle;};
197
198 // ramène le déplacement tangentiel actuel
199 const Coordonnee& Dep_tangentiel() const {return dep_tangentiel;};
200 // ramène la normale actuelle
201 const Coordonnee& Normale_actuelle() const {return N;};
202
203 // idem pour les réactions sur les noeuds de la facette
204 const Tableau <Coordonnee*> TabForce_cont() const {return tabForce_cont;};
205
206 // actualisation de la projection du noeud esclave en fonction de la position de l'element
207 // maitre frontiere. Lorsque le noeud change d'element fontiere, on change l'element
208 // frontiere de l'element de contact en consequence

```

```

206 // dans le cas ou on ne trouve pas d'intersection, cas d'un noeud qui sort d'une zone de
207 // contact, on retourne false, sinon retourne true
208 // en fonction de la méthode de contact, le noeud est ramené éventuellement sur la frontière
209 bool Actualisation();
210
211 // ramene un pointeur sur l'element frontiere
212 inline Front * Elfront() const { return elfront;};
213 // permet de changer le deplacement maxi de tous les noeuds, qui sert
214 // pour éviter de considérer des contact trop éloignés
215 static void Change_dep_max(const double & deplac_max)
216     {dep_max = deplac_max * ParaGlob::param->ParaAlgoControleActifs().FacPourRayonAccostage();};
217
218 // test et met à jour le compteur de décollage du noeud
219 // si le noeud decolle ou non en fonction de la force de reaction
220 // ramene 1: s'il decolle
221 //      0: s'il ne decolle pas
222 bool Decol();
223 // change force: permet de changer la valeur de la force
224 // utile quand la force est calculée en dehors de l'élément de contact
225 void Change_force(const Coordonnee& force);
226
227 // idem pour les forces réparties sur la facette
228 void Change_TabForce_cont(const Tableau <Coordonnee>& tab) {tabForce_cont=tab;};
229
230 // gestion de l'activité
231 int Actif() const {return actif;}; // ramène l'activité du contact
232 void Met_Inactif() { actif = 0;nb_decol_tdt=0;}; // met en inactif
233 void Met_actif() { actif++;nb_decol_tdt=0;}; // met en actif une fois de plus
234
235 // ramène le nombre actuel de décolement
236 int Nb_decol() const {return nb_decol_tdt;};
237 // ramène le nombre de pénétration actuel
238 int Nb_pene() const {return nb_pene_tdt;};
239
240 // --- calcul des puissances virtuelles développées par les efforts de contact -----
241 // et eventuellement calcul de la raideur associé
242 // -> explicite à tdt
243 virtual Vecteur* SM_charge_contact();
244 // -> implicite,
245 virtual Element::ResRaid SM_K_charge_contact();
246 // récupération des énergies intégrées sur l'éléments, résultants d'un précédent calcul
247 // explicite, ou implicite:
248 // 1- il s'agit ici de l'énergie développée par le frottement glissant ou pas
249 const EnergieMeca& EnergieFrottement() const {return energie_frottement;};
250 // 2- énergie de pénalisation (élastique a priori)
251 const double& EnergiePenalisation() const {return energie_penalisation;};
252
253 // cas d'une méthode avec pénalisation: calcul éventuel d'un pas de temps idéal,
254 // permettant de limiter la pénétration
255 // si oui retour de la valeur delta_t proposé
256 // sinon dans tous les autres cas retour de 0.
257 // le calcul se fait en fonction du pas de temps courant et de la pénétration
258 // donc nécessite que le contact ait déjà été étudié
259 double Pas_de_temps_ideal()const;
260
261 // ramène l'info sur le fait que le contact est avec un solide ou pas
262 // retour = 0 : contact bi déformable,
263 //      = 1 le noeud est libre et la frontière est bloqué (solide)
264 //      = 2 le noeud est bloqué (solide) la frontière est libre
265 //      = 3: tout est bloqué (solide)
266 int Cas_solide() const {return cas_solide;};
267
268 // permet de modifier le contact entre collant ou non suivant "change"
269 void Change_contact_collant(bool change);
270 // récup de l'information concernant le contact collant ou pas
271 int Collant() const {return cas_collant;};
272
273 // récup des gaps calculés
274 const double & Gapttdt() const {return gap_tdt;};
275 const double & Dep_T_tdt() const {return dep_T_tdt;};
276
277
278 // mise à jour du niveau de commentaire
279 static void Mise_a_jour_niveau_commentaire(int niveau)
280     {niveau_commentaire = niveau;};
281
282 // récupération des ddl ou des grandeurs actives de tdt vers t
283 void TdtversT();
284 // actualisation des ddl et des grandeurs actives de t vers tdt
285 void TversTdt();
286
287 //----- lecture écriture de restart -----
288 void Lec_base_info_ElContact(ifstream& ent);
289 void Ecri_base_info_ElContact(ofstream& sort);
290
291
292 protected :

```

```

293 // VARIABLES PROTEGEES :
294 int actif; // un indicateur, disant si le contact est actif ou pas
295 // conteneur plutôt utilisé par les classes appelantes
296 // =1 -> premier contact, > 1 contact qui suit un contact
297 Front* elfront; // un element frontiere
298 Noeud * noeud; // un pointeur de noeud
299 int num_zone_contact; // un stockage uniquement utilisé par les méthodes appelantes
300 // pour éviter de le construire à chaque demande on définit un tableau de tous les noeuds
301 Tableau <Noeud> tabNoeud; // tableau de tous les noeud, le premier est noeud
302 // le tableau des positions successives du noeud esclave, ceci pour effectuer une moyenne glissante
303 Tableau <Coordonnee> tab_posi_esclave;
304 // le nombre courant de positions de noeuds esclaves actuellement stockées
305 int nb_posi_esclave_stocker_t,nb_posi_esclave_stocker_tdt;
306 // indice dans tab_posi_esclave, du prochain stockage
307 int indice_stockage_glissant_t,indice_stockage_glissant_tdt;
308
309 Coordonnee Mtdt,Mt; // sauvegarde du point d'interception s'il est recevable
310 Coordonnee M_noeud_tdt_avant_projection; // dans le cas d'une projection du noeud sur la surface
    maître
311 // sauvegarde des coordonnées du noeuds esclave: utilisation avec le type de contact 4
312 // les fonctions d'interpolation au premier point en contact: sert pour le contact collant
313 Vecteur phi_theta_0 ;
314 EnergieMeca energie_frottement; // énergie développée par le frottement glissant ou pas
315 double energie_penaliation; // énergie due à la pénalisation (élastique a priori)
316 // cas_solide permet de simplifier le contact dans le cas ou le maître ou l'esclave est solide
317 // sert pour diminuer la taille de la raideur uniquement
318 int cas_solide; // =0 contact bi déformable, =1 le noeud est libre et la frontière est bloqué
    (solide)
319 // = 2 le noeud est bloqué (solide) la frontière est libre
320 // = 3 tout est solide
321 int cas_collant; // prise en compte éventuelle d'un contact collant, si 0: contact normal
322 // = 1 : contact collant
323 Vecteur * residu; // residu local
324 Mat_pleine * raideur; // raideur locale
325 DdlElement * ddlElement_assemblage; // le ddlElement qui correspond
326 Tableau <Noeud> tabNoeud_pour_assemblage; // tableau des noeuds qui servent pour l'assemblage
327 Coordonnee force_contact,force_contact_t; // force de contact sur le noeud principal
328 double F_N_max,F_T_max,F_N_max_t,F_T_max_t; // les maxis constatés
329 Tableau <Coordonnee> tabForce_cont,tabForce_cont_t; // le tableau des forces sur les noeuds de la
    facette
330 Coordonnee N; // la dernière normale calculée
331 Coordonnee dep_tangentiel; // le dernier déplacement tangentiel calculé
332
333 int nb_decol_t,nb_decol_tdt; // nombre de fois où le noeud décolle de manière consécutive
334 double gap_t,gap_tdt; // les pénétrations d'un pas à l'autre
335 double dep_T_t, dep_T_tdt; // les valeurs absolue des déplacements tangentiels d'un pas à l'autre
336 double nb_pene_t,nb_pene_tdt; // le nombre de penetration positives
337
338 // cas TypeCalculPenaliationPenetration() = 4 ou -4
339 // -> cas d'un facteur multiplicatif évolutif, on fait une moyenne glissante de 2
340 // on mémorise donc les facteurs multiplicatifs successifs
341 double mult_pene_t,mult_pene_tdt;
342 // cas TypeCalculPenaliationTangentielle() = 4 ou -4
343 // -> cas d'un facteur multiplicatif évolutif, on fait une moyenne glissante de 2
344 // on mémorise donc les facteurs multiplicatifs successifs
345 double mult_tang_t,mult_tang_tdt;
346
347 double penalisation,penaliation_tangentielle; // on sauvegarde la pénalisation
348 // l'intérêt est de pouvoir la visualiser
349 double penalisation_t,penaliation_tangentielle_t; // les grandeurs à t
350
351 // utilisation de fct nD
352 Fct_nD_contact fct_nD_contact;
353
354 // pour le contact 4, pour le calcul de la pénalisation avec une moyenne glissante
355 Tableau <double > val_penal; // tableau de stockage intermédiaire
356 int pt_dans_val_penal; // indice du prochain elem du tableau a remplir
357
358 // stocke le dernier type de trajectoire du noeud par rapport à la facette
359 // c-a-d la valeur de la variable test dans la fonction Trajectoire(int & test)
360 int type_trajectoire_t,type_trajectoire_tdt;
361
362 // concernant les enum de ddl associées aux éléments de contact, on définit un tableau global
363 // qui est utilisé par tous les éléments
364 static list <DdlElement> list_Ddl_global; // liste de tous les DdlElements des éléments de contact
365 static list <Vecteur> list_SM; // list de seconds membres locaux: sert pour tous les éléments de
    contact
366 static list <Mat_pleine> list_raideur; // list de raideurs locales: " " " "
367 // stockage du maximum de distance tolérée entre noeud à tdt et le projeté, sert pour éliminer les
    contacts aberrants
368 static double dep_max;
369 static int niveau_commentaire;
370 // stockage d'une fonction nD pour le pilotage du type de contact 4
371 static Fonction_nD * fct_pilotage_contact4;
372
373 // METHODES PROTEGEES :
374 // calcul la normal en fonction de differente conditions

```

```

375 Coordonnee Calcul_Normale (int dim, Plan & pl, Droite & dr,int indic);
376 void Libere();
377 // construction du tableau de tous les noeuds, le premier est celui esclave
378 // et mise à jour de ddlElement et de list_Ddl_global éventuellement
379 void Construction_TabNoeud();
380 // récupération d'informations des classes internes pour le calcul du résidu
381 // N: le vecteur normal
382 // M_impact: le point d'impact sur la surface (ou ligne)
383 // phii : les fonctions d'interpolation au point d'impact
384 // si avec_var est vrai: il y a retour du tableau de variation de la normale
385 Tableau <Coordonnee >* RecupInfo(Coordonnee& N,Coordonnee& M_impact,Vecteur& phii,bool avec_var );
386
387 // mise à jour de cas_solide et donc de ddlElement en fonction de l'activité des ddl
388 // mise à jour du tableau de noeud pour l'assemblage tabNoeud_pour_assemblage
389 void Mise_a_jour_ddlelement_cas_solide_assemblage();
390 // récup d'une place pour le résidu local et mise à jour de list_SM éventuellement
391 void RecupPlaceResidu(int nbddl);
392 // récup d'une place pour la raideur locale et mise à jour de list_raideur éventuellement
393 void RecupPlaceRaideur(int nbddl);
394 // calcul du facteur de pénalisation en pénétration, en fonction de la géométrie
395 // du module de compressibilité et des différents possibles
396 // éventuellement, calcul de la dérivée: d_beta_gapdu, du facteur par rapport au gap
397 // la sensibilité dépend du type de calcul du facteur de pénalisation
398 double CalFactPenal(const double& gap,double & d_beta_gap,int contact_type);
399 // calcul du facteur de pénalisation tangentielle, en fonction de la géométrie
400 // du module de compressibilité et des différents cas possibles
401 // éventuellement, calcul de la dérivée: d_beta_gapdu, du facteur par rapport au gap
402 // la sensibilité dépend du type de calcul du facteur de pénalisation
403 double CalFactPenalTangentiel(const double& gap,double & d_beta_gap);
404 // limitation éventuelle au niveau de la pénétration maxi
405 //void Limitation_penetration_maxi(
406 // calcul éventuel de la moyenne glissante des positions successive du noeud esclave
407 void ChangeEnMoyGlissante(Coordonnee& Noe_atdt);
408
409 // calcul de la moyenne glissante de la pénalisation
410 void Moyenne_glissante_penalisation(double& penalisation, double& essai_penalisation);
411
412 // calcul d'une fonction nD relative à des données de contact
413 double Valeur_fct_nD(Fonction_nD * fct_nD) const;
414
415 };
416 /// @} // end of group
417
418 #endif

```

## 7.94 LesContacts.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *   ****
38 *   BUT:      Creation et gestion des contacts.
39 *
40 *****/

```



```

39 *                                     $ *
40 *          ***** *
41 *          VERIFICATION: *
42 *          * *
43 *          ! date ! auteur ! but ! *
44 *          ----- *
45 *          ! ! ! ! ! *
46 *          * *
47 *          ***** *
48 *          MODIFICATIONS: *
49 *          ! date ! auteur ! but ! *
50 *          ----- *
51 *          * *
52 *          *****/
53 #ifndef LESCONTACTS_H
54 #define LESCONTACTS_H
55
56 #include "Front.h"
57 #include <list>
58 #include "List_io.h"
59 #include "ElContact.h"
60 #include "Condilineaire.h"
61 #include "Nb_assemb.h"
62 #include "LesMaillages.h"
63 #include "LesReferences.h"
64 #include "Basiques.h"
65 #include "TypeQuelconque.h"
66 #include "Temps_CPU_HZpp.h"
67
68
69 /// @addtogroup Groupe_sur_les_contacts
70 /// @{
71 ///
72
73
74 class LesContacts
75 {
76 public :
77 // CONSTRUCTEURS :
78 // constructeur par défaut
79 LesContacts ();
80 // constructeur de copie
81 LesContacts (const LesContacts& a);
82 // DESTRUCTEUR :
83 ~LesContacts ();
84
85 // METHODES PUBLIQUES :
86
87 // ramène la liste des éléments de contact
88 LaLIST <ElContact>& LesElementsDeContact() {return listContact;};
89
90 // initialisation des zones de contacts éventuelles à partir des éléments de frontières et des noeuds
91 // esclaves
92 // sauf les frontières qui sont les mêmes pendant tout le calcul
93 // --- en entrée:
94 // les maillages, les ref et les fonctions nD
95 // -- en sortie:
96 // cas du contact type 4 : on renseigne éventuellement une fonction de pilotage -->
97 // fct_pilotage_contact4
98 // récup de: nb_mail_Esclave, nbmailMaitre,
99 // récup du tableau "indice" : = la liste pour chaque noeud, des éléments qui contient ce noeud
100 // création des conteneurs internes : tescttotal, tesN_collant, t_listFront, tesN_encontact
101 void Init_contact(LesMaillages& lesMail
102                 ,const LesReferences& lesRef
103                 ,LesFonctions_nD* lesFonctionsnD);
104 // mise à jour du tableau "indice": là on utilise les numéros de noeuds qui peuvent changer
105 // via une renumérotation. Le tableau indice, est un tableau utilisé pour les recherches
106 // mais le numéro de noeud n'est pas utilisé pour les stockages divers (on utilise un pointeur de
107 // noeud)
108 // du coup le tableau indice peut évoluer ex: après un remaillage avec chg de num de noeud
109 // il doit donc être remis à jour avant l'étude de nouveau contact
110 // la liste pour chaque noeud, des éléments qui contient ce noeud
111 // indice(i)(j) : = la liste des éléments qui contiennent le noeud j, pour le maillage i
112 void Mise_a_jour_indice(const Tableau < const Tableau <List_io <Element* > > *) ind)
113 { indice = ind;};
114
115 // indique si l'initialisation a déjà été effectuée ou pas, car celle-ci ne doit-êre faite
116 // qu'une fois
117 bool Init_contact_pas_effectue() const { return (tescttotal.Taille() == 0);};
118
119 // verification qu'il n'y a pas de contact avant le premier increment de charge
120 void Verification();
121 // definition des elements de contact eventuels

```

```

122 // - imposition des conditions de non penetration
123 // dep_max : déplacement maxi des noeuds du maillage
124 //           , sert pour def des boites d'encombrement maxi des frontières
125 // ramène true s'il y a effectivement création d'élément de contact
126 bool DefElemCont(double dep_max);
127
128 // reexamen du contact pour voir s'il n'y a pas de nouveau element de
129 // contact
130 // ramene false s'il n'y a pas de nouveau element de contact
131 // true sinon
132 // dep_max : déplacement maxi des noeuds du maillage
133 //           , sert pour def des boites d'encombrement maxi des frontières
134 bool Nouveau(double dep_max);
135
136 // suppression définitive, si le contact à disparu, des éléments inactifs
137 // ramène false si aucun élément n'est finalement supprimé
138 bool SuppressionDefinitiveElemInactif();
139
140 // relachement des noeuds collés
141 // ramène true s'il y a des noeuds qui ont été relachés
142 bool RelachementNoeudcolle();
143
144 // definition de conditions lineaires de contact
145 // marquage des ddl correspondant aux directions bloquees s'il s'agit d'un contact de type 1
146 // casAssemb : donne le cas d'assemblage en cours
147 //
148 const Tableau <Condilinaire>& ConditionLin(const Nb_assemb& casAssemb);
149
150 // création d'un tableau de condition linéaire, correspondant à tous les éléments de contact en cours
151 // qu'ils soient actifs ou pas (a priori cette méthode est conçu pour donner des infos relativement à
152 // la largeur
153 // de bandes en noeuds due aux CLL)
154 // chacune des condition ne contient "d'exploitable" que le tableau de noeuds associés à la CLL,
155 const Tableau <Condilinaire>& ConnectionCLL();
156
157 // effacement du marquage de ddl bloque du au conditions lineaire de contact
158 void EffMarque();
159
160 // indique si les surfaces des maillages maitres ont des déplacements fixés
161 // c-a-d sont de type solide imposé
162 // retourne true si les déplacements des maitres sont imposés
163 bool Maitres_avec_deplacement_imposer() const;
164
165 // def de la largeur de bande des elements contacts
166 // casAssemb : donne le cas d'assemblage a prendre en compte
167 // les condi linéaires ne donnent pas des largeurs de bande sup et inf égales !!!
168 // demi = la demi largeur de bande maxi ,
169 // total = le maxi = la largeur sup + la largeur inf +1
170 // cumule = la somme des maxis, ce qui donnera la largeur finale, due à des multiples
171 // multiplications: une par conditions linéaires
172 // ceci dans le cas de la prise en compte par rotation (et uniquement dans ce cas)
173 // en retour, ramène un booleen qui :
174 // = true : si la largeur de bande en noeud est supérieure à 1 (cas d'un contact avec une surface
175 // déformable)
176 // = false : si non, ce qui signifie dans ce cas qu'il n'y a pas d'augmentation de la largeur
177 // en noeud (cas d'un contact avec une surface rigide)
178 bool Largeur_Bande(int& demi,int& total,const Nb_assemb& casAssemb,int& cumule);
179
180 // actualisation du contact, on examine que les elements de contact, dont on
181 // actualise la projection du noeud esclave en fonction de la position de l'element
182 // maitre frontiere (mais la position finale du noeud n'est pas forcément changée, cela dépend
183 // du
184 // modèle de contact (cinématique, pénalisation etc.)
185 // dans le cas où la réaction est négative, en fonction de l'algorithme l'élément de contact est
186 // inactivé, cependant tous les éléments de contact sont passés en revue (actif ou pas)
187 // ramène true si quelque chose à changé, false sinon
188 bool Actualisation();
189
190 // ramène une liste de noeuds dont la position a été perturbé par le contact
191 // (dépend du type de contact : ex cas contact = 4)
192 // la liste passée en paramètre est supprimée et remplacée par la liste résultat
193 void Liste_noeuds_position_changer(list <Noeud * >& li_noe);
194
195 // calcul des reactions de contact et stockage des valeurs
196 // solution : le vecteur residu
197 // test d'un decollement eventuelle, pour un noeud en contact
198 // ramene true s'il y a decollement, sinon false
199 // casAssemb : donne le cas d'assemblage en cours
200 void CalculReaction(Vecteur& residu,bool& decol,const Nb_assemb& casAssemb
201 // ,bool affiche);
202
203 // récupération via les éléments de contact des forces maxis
204 // un : le maxi en effort normal, deux: le maxi en effort tangentiel
205 DeuxDoubles Forces_contact_maxi(bool affiche) const;
206
207 // récupération via les éléments de contact des gaps maxis

```

```

205 // un : le maxi en gap normal, deux: le maxi en gap tangentiel
206 DeuxDoubles Gap_contact_maxi(bool affiche) const;
207
208 // cas d'une méthode avec pénalisation: calcul éventuel d'un pas de temps idéal,
209 // si oui retour de la valeur delta_t proposé
210 // sinon dans tous les autres cas retour de 0.
211 // le calcul se fait en fonction du pas de temps courant, des forces de réaction et de la
    pénétration
212 // donc nécessite que le contact ait déjà été étudié et que les efforts de contact ait été
    calculé
213 double Pas_de_temps_ideal()const;
214
215 // calcul d'une estimation du pas de temps critique du aux éléments de contact
216
217
218 // affichage des reactions de contact sur la sortie
219 void Affiche(ofstream& sort) const ;
220
221 // affichage à l'écran des informations liées au contact
222 void Affiche() const ;
223
224 // affichage et definition interactive des commandes
225 void Info_commande_LesContacts(UtilLecture & entreePrinc);
226
227 // lecture éventuelle des zones où le contact doit être recherché, à l'exclusion de tout
228 // autre zone, ainsi que la définition de l'auto-contact éventuel
229 // ainsi que des contacts solide-deformables éventuel
230 void Lecture_zone_contact(UtilLecture & entreePrinc,const LesReferences& lesRef);
231
232 // récupération des ddl ou des grandeurs actives de tdt vers t
233 void TdtversT();
234 // actualisation des ddl et des grandeurs actives de t vers tdt
235 void TversTdt();
236
237 //----- temps cpu -----
238 // retourne temps cumulé pour imposer les CL imposées
239 const Temps_CPU_HZpp& Temps_cpu_Contact() const {return tempsContact;};
240
241 //----- lecture écriture de restart -----
242 // cas donne le niveau de sauvegarde
243 void Ecri_base_info_LesContacts(ofstream& sort);
244 // on utilise deux pointeurs de fonctions qui permettent de récupérer le pointeur de noeud esclave
245 // idem au niveau de l'élément
246 template <class T>
247 void Lec_base_info_LesContacts(ifstream& ent
    ,T& instance // l'instance qui permet d'appeler les pointeurs de fonctions
    ,Noeud& (T::*RecupNoeud)(int i, int j) const
    ,Element& (T::*RecupElement_LesMaille) (int i, int j) const);
251
252 // méthode générale: récupération des grandeurs particulière (hors ddl )
253 // correspondant à liTQ
254 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
255 // ==> n'est pas utilisée dans le cas du contact, c'est la méthode spécifique
256 // Mise_a_jour_Pour_Grandeur_particuliere qui la remplace (qui n'est pas en const car elle
257 // modifie les conteneurs des noeuds et éventuellement éléments)
258 void Grandeur_particuliere
    (bool absolue,List_io<TypeQuelconque>& ,Loi_comp_abstraite::SaveResul * ,list<int>& decal) const
260     {};
261
262 // Il s'agit ici de mettre à jour les conteneurs stockés aux noeuds et/ou aux éléments
263 // qui servent à récupérer les infos liés aux contact correspondant à liTQ
264 // actuellement les conteneurs passés en paramètre ne servent que pour
265 // les énumérés, et les informations résultantes sont stockées au niveau des noeuds
266 // constituant les éléments de contact
267 //--> important : les conteneurs sont supposés initialisés avec l'appel
268 void Mise_a_jour_Pour_Grandeur_particuliere(
    List_io < TypeQuelconque >& li_restreinte_TQ
    );
271
272 // récupération de la liste de tous les grandeurs particulières disponibles avec le contact
273 // cette liste est identique quelque soit le maillage: il n'y a donc pas de tableau indicé du num de
    maillage
274 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
275 List_io<TypeQuelconque> ListeGrandeurs_particulieres(bool absolue) const;
276
277 // concernant les grandeurs gérées par le contact:
278 // ramène une liste d'iterator correspondant aux List_io<TypeQuelconque> passé en paramètre
279 // idem pour une List_io < Ddl _enum_etendu >
280 void List_reduite_aux_contact(const List_io<TypeQuelconque>& liTQ
    ,List_io < TypeQuelconque >& li_restreinte_TQ
    );
282
283
284 // initialisation des listes de grandeurs qu'ils faudra transférer aux niveaux des noeuds des
    éléments
285 // de contact, on définit si besoin, les conteneurs ad hoc au niveau des noeuds
286 // ok, mais à revoir sans doute cf. pense bete 14 oct
287 void Init_Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& l1l);

```

```

288
289 // ne sert plus (à virer car problématique !! )
290 // mise à jour du stockage inter, pour prendre en
291 // compte une nouvelle numérotation des noeuds
292 // void Prise_en_compte_nouvelle_numerotation_noeud();
293
294 // initialisation de la liste de grandeurs qui sont effectivement gérées par le contact
295 // ok, mais à revoir sans doute cf. pense bete 14 oct
296 // List_io<TypeQuelconque> Init_list_grandeur_contact_a_sortir(const Tableau <List_io < TypeQuelconque
> >& lil);
297
298 class ReactCont
299 { // surcharge de l'operator de lecture
300     friend istream & operator » (istream &, ReactCont &);
301     // surcharge de l'operator d'écriture
302     friend ostream & operator « (ostream &, const ReactCont &);
303
304     public :
305     Noeud* noe; // noeud esclave
306     Coordonnee force ; // force de reaction au noeud esclave
307     Tableau <Noeud *> tabNoeud; // les noeuds de la frontiere maitre
308     Tableau <Coordonnee> tabForce; // les reac "" ""
309     // constructeur par default
310     ReactCont () ;
311     // constructeur en fonction des datas du noeud esclave seul
312     ReactCont(Noeud* no,const Coordonnee& forc) ;
313     // constructeur en fonction des datas de tous les noeuds
314     ReactCont(Noeud* no,const Coordonnee& forc,Tableau <Noeud *> tN,const Tableau <Coordonnee>&
tFor);
315     // constructeur de copie
316     ReactCont(const ReactCont& a);
317     // affectation
318     ReactCont& operator = (const ReactCont& a);
319     // test
320     bool operator == (const ReactCont& a);
321     bool operator != (const ReactCont& a);
322 };
323
-----
324 private :
325 // VARIABLES PROTEGEES de la classe LesContacts:
326 LesFonctions_nD* sauve_lesFonctionsN ; // sauvegarde à l'initialisation (méthode Init_contact)
327 ElContact::Fct_nD_contact fct_nD_contact; // fonctions nD de pilotage: peuvent ne pas exister
328
329 LaLIST <ElContact> listContact; // la liste des elements en contact
330 LaLIST <LaLIST <ElContact>::iterator> listContact_nouveau_tatdt; // la liste des nouveaux contacts
qui sont apparus sur l'incrément
331 LaLIST <ElContact> listContact_efface_tatdt; // la liste des contacts effacés sur l'incrément
332 int nb_contact_actif; // nombre de contact actif courant
333 // list <TroisEntiers> numtesN; // .un : maillage, .deux: num zone de contact, .trois: num propre du
noeud
334 // // la liste des numéros dans tesN des noeuds en contact
335 Tableau <ReactCont> tabReactCont; // les reactions
336 Tableau <Condilinaire> tabCoLin; // tableau des conditions lineaires
337
338 static MotCle motCle; // liste des mots clés
339 // la liste des éléments qui contiennent des frontières, est reconstitué au démarrage avec
Init_contact(..
340 list <Element *> liste_lemens_front;
341 // la liste pour chaque noeud, des éléments qui contient ce noeud : construite avec Init_contact
342 // indice(i)(j) : = la liste des éléments qui contiennent le noeud j, pour le maillage i
343 Tableau < const Tableau <List_io < Element* > > * > indice;
344
345 //----- les tableaux de gestions pour la recherche de contact -----
346 list <quatre_string_un_entier> nom_ref_zone_contact; // liste des noms de références des zones de
contact éventuelle
347 // cette liste peut être vide, dans ce cas cela signifie que toutes les frontières des pièces sont
348 // suceptible de rentrer en contact. Dans le cas où la liste n'est pas vide, seules les grandeurs
349 // référencées sont suceptibles de rentrer en contact
350 // nom1 -> nom_mail_ref_zone_contact pour les noeuds
351 // nom2 -> le nom de la référence de noeuds
352 // nom3 -> nom_mail_ref_zone_contact pour les frontières
353 // nom4 -> le nom de la référence de frontière
354 // n -> l'entier = 0 : pas d'utilisation
355 // = 1 : indique qu'il faut considérer un contact collant (glue)
356
357
358 // la liste des éléments frontières suceptibles d'entrer en contact
359 // ces Front (qui contiennent que des pointeurs sauf une boîte d'encombrement)
360 // sont différents de ceux des maillages, et sont donc stocké en interne
361 Tableau < Tableau < LaLIST_io <Front> > > t_listFront;
362 // t_listFront(i)(j)(k) : maillage maître (i)
363 // zone de contact (j)
364 // (K) = kième frontière (dans l'ordre de la liste)
365 // ** pour le tableaux t_listFront, le numéros dit de maillage, n'est pas le numéro
366 // ** intrinsèque de maillage (telle que ceux associés aux noeuds et éléments)

```

```

367         // ** mais uniquement un numéro locale d'ordre
368         // ** mais on a: les éléments de frontière de t_listFront(i) font partie du maillage
369         // i + (nb_mail_Esclave-nbmailautocontact)
370
371         // les noeuds esclaves potentiels
372         Tableau < Tableau < Tableau < Noeud* > > > tescttotal;
373         // tescttotal(i)(j)(k) : maillage esclave (i), zone de contact (j)
374         // noeud esclave (k) : k= le numéro d'ordre dans tescttotal(i)(j),
375         //                               ==> ce n'est pas le numéro du noeud dans le maillage !
376
377
378         // indicateur de noeuds collants
379         Tableau < Tableau < Tableau < int > > > tesN_collant;
380         // tesN_collant(i)(j)(k): maillage esclave (i), zone de contact (j)
381         // noeud esclave (k):k= le numéro d'ordre dans tescttotal(i)(j),
382         //                               ==> ce n'est pas le numéro du noeud dans le maillage !
383         // indique pour si le noeud esclave doit être
384         // considéré en contact collant (=1) ou pas (=0)
385
386         // tesN_encontact: globalise tous les contacts sur un noeud (indépendamment des zones de contact)
387         // tesN_encontact(i)(num_noe) : nombre de contact avec le noeud
388         // ancien stockage: Tableau < Tableau < LaLIST < LaLIST<ElContact>::iterator > > > tesN_encontact;
389         // ancien stockage // tesN_encontact(i)(num_noe): maillage (i), noeud -> num_noe:
390         // indique pour le noeud esclave s'il est en contact ou non via la taille de la liste associée
391         // utilisation: pour chaque Noeud* = tescttotal(i)(k) -> la liste des éléments en contact
392         // contenant le noeud k
393
394         // on remplace par une map: l'avantage c'est que l'on utilise plus les numéros de noeuds pour
395         // retrouver la liste, mais on utilise à la place la valeur pointeur de noeud esclave qui
396         // doit être unique, du coup c'est indépendant de la numérotation des noeuds -> permet de la
397         // renumérotation
398         // sans changer la map
399         Tableau < std::map<Noeud*,LaLIST < LaLIST<ElContact>::iterator > > > tesN_encontact;
400         // tesN_encontact(numMail_esclave)[*pt_noeud] -> la liste des iterators d'élément en contact
401         // avec le noeud
402
403         //----- fin tableaux de gestions pour la recherche de contact -----
404
405         int nb_mail_Esclave; // indique le nombre de maillages esclaves
406         int nbmailautocontact; // indique le nombre de maillages esclaves en auto contact, donc qui joue
407         // le rôle esclave et maître, ou maillages mixte: dépend de la manière dont les zones de contact ont été
408         // définies
409         int nbmailMaitre; // indique le nombre de maillages maitres
410         // = nombre total de maillage - (nb_mail_Esclave-nbmailautocontact)
411         // si l'on considère l'ensemble des maillages du calcul on a successivement:
412         // les nb_mail_Esclave-nbmailautocontact premier maillages: purement esclave
413         // puis les nbmailautocontact qui sont en auto contact: ils jouent le rôle d'esclaves et
414         // maîtres en même temps
415         // puis nbmailMaitre-nbmailautocontact : purement maître
416
417         list < Deux_String > cont_solide_defor; // liste des noms de maillages définissant les contacts
418         // solide-déformable: (*ie).nom1 -> le maillage solide , (*ie).nom2 -> le maillage
419         // déformable
420
421         // ---- pour le post traitement ---
422         List_io<TypeQuelconque> liQ_en_sortie; // liste de grandeurs quelconque qu'il faut sortir
423
424         // ----- une variable de travail pour la méthode LesContacts::ConnectionCLL()---
425         Tableau <Condilinaire> t_connectionCLL;
426
427         //----- temps cpu -----
428         // retourne temps cumulé pour imposer les CL imposées
429         Temps_CPU_HZpp tempsContact;
430
431         // METHODES PROTEGEES :
432         // mise à jour des boites d'encombrement pour les éléments qui contiennent une frontière
433         void Mise_a_jour_boite_encombrement_element_contenant_front();
434
435         // suppression du gap de contact pour les noeuds "collant avec suppression de gap"
436         void Suppression_gap_pour_noeud_collant();
437
438         // récupération de la zone de contact d'un élément de contact existant
439         // c'est un peu lourd, mais l'avantage c'est que cela s'adapte à une situation qui
440         // a par exemple changé
441         // si en retour le numéro de zone = 0, cela signifie que le contact ne peut plus exister
442         int Recup_ref( ElContact& al) ;
443
444         // récupère le nombre de contact actif et met à jour ce nombre
445         // normalement devrait toujours être correct mais ?? il y a quelque chose d'incorrecte quelque part
446         int Recalcul_Nb_contact_actif();
447
448         // création du conteneur Fct_nD_contact
449         void Creation_Fct_nD_contact();
450
451     };
452     /// @} // end of group

```

```

449
450
451 // pour faire de l'inline: nécessaire avec les templates
452 // on n'inclut que les méthodes templates
453 #include "LesContacts_2.cc"
454 #define LesContacts_2_deja_inclus
455
456 #endif

```

## 7.95 Plan.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *      $
38 *      *****
39 *      BUT:      Def de la geometrie d'un plan: un point et une normale.
40 *      $
41 *      *****
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      -----
47 *      !           !           !
48 *      $
49 *      *****
50 *      MODIFICATIONS:
51 *
52 *      ! date !   auteur !           but
53 *      -----
54 *      $
55 *      *****/
56 #ifndef PLAN_H
57 #define PLAN_H
58
59 #include "Droite.h"
60 #include "Coordonnee.h"
61
62 /// @addtogroup Groupe_sur_les_contacts
63 /// @
64
65 class Plan
66 {
67     // surcharge de l'operator de lecture
68     friend istream & operator » (istream &, Plan &);
69     // surcharge de l'operator d'écriture
70     friend ostream & operator « (ostream &, const Plan &);
71
72 public :
73     // CONSTRUCTEURS :
74     // par défaut
75     Plan ();

```

```

74 // avec les datas
75 Plan ( const Coordonnee& B, const Coordonnee& vec);
76 // avec la dimension
77 Plan (int dim);
78 // de copie
79 Plan ( const Plan& a);
80 // DESTRUCTEUR :
81 ~Plan ();
82 // surcharge des operator
83 Plan& operator = ( const Plan & P);
84
85 // METHODES PUBLIQUES :
86 // retourne le point de ref du plan
87 inline const Coordonnee& PointPlan() const { return A;};
88 // retourne la normale au plan
89 inline const Coordonnee& Vecplan() const { return N;};
90
91 // change la dimension de la droite
92 void Change_dim(int dima) {A.Change_dim(dima);N.Change_dim(dima);};
93 // change le point de ref du plan
94 void Change_ptref( const Coordonnee& B);
95 // change le vecteur normal du plan
96 void Change_normal( const Coordonnee& vec);
97 // change toutes les donnees
98 void change_donnees( const Coordonnee& B, const Coordonnee& vec);
99
100 // calcul l'intercection M d'une droite avec le plan, ramene 0 s'il n'y
101 // a pas d'interseccion, ramene -1 si l'interceccion ne peut pas etre calculee
102 // et 1 s'il y a un point d'interceccion
103 int Intersection( const Droite & D,Coordonnee& M)const;
104
105 // calcul la distance d'un point à la droite
106 double Distance_au_plan(const Coordonnee& M) const;
107
108 // calcul du projeté d'un point sur le plan
109 Coordonnee Projete(const Coordonnee& M) const
110 {return (M - ((M-A)*N)*N);};
111
112 // ramène true si les deux points sont du même coté du plan, false sinon
113 bool DuMemeCote(const Coordonnee& M1, const Coordonnee& M2) const;
114
115 protected :
116 // VARIABLES PROTEGEES :
117 Coordonnee A; // un point du plan
118 Coordonnee N; // normale au plan
119 // METHODES PROTEGEES :
120
121 };
122 /// @} // end of group
123
124 #endif

```

## 7.96 Sphere.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      19/01/2007
32 *

```

```

32 *                                     $ *
33 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)   $ *
34 *                                     $ *
35 *   PROJET:      Herezh++                               $ *
36 *                                     $ *
37 *****
38 *   BUT:   Def géométrie d'une sphere: un centre et un rayon   $ *
39 *                                     $ *
40 *   ////////////////////////////////////////////////// *
41 *                                     *
42 *   VERIFICATION:                                           *
43 *   ! date !   auteur !           but                       ! *
44 *   -----
45 *   !           !           !                               ! *
46 *                                     $ *
47 *   ////////////////////////////////////////////////// *
48 *   MODIFICATIONS:                                           *
49 *   ! date !   auteur !           but                       ! *
50 *   -----
51 *                                     $ *
52 *****/
53 #ifndef SPHEREE_H
54 #define SPHEREE_H
55
56 #include "Droite.h"
57 #include "Coordonnee.h"
58
59 /// @addtogroup Groupe_sur_les_contacts
60 /// @{
61 ///
62
63
64 class Sphere
65 { // surcharge de l'operator de lecture
66   friend istream & operator » (istream &, Sphere &);
67   // surcharge de l'operator d'écriture
68   friend ostream & operator « (ostream &, const Sphere &);
69
70 public :
71   // CONSTRUCTEURS :
72   // par défaut
73   Sphere ();
74   // avec les datas
75   Sphere ( const Coordonnee& B, const double& r);
76   // avec la dimension
77   Sphere (int dim);
78   // de copie
79   Sphere ( const Sphere& a);
80   // DESTRUCTEUR :
81   ~Sphere ();
82   // surcharge des operator
83   Sphere& operator = ( const Sphere & P);
84
85   // METHODES PUBLIQUES :
86   // retourne le centre de la Sphere
87   inline const Coordonnee& CentreSphere() const { return centre;};
88   // retourne le rayon de la sphere
89   inline double RayonSphere() const { return rayon;};
90
91   // change le centre de la sphere
92   void Change_centre( const Coordonnee& B);
93   // change le rayon de la sphere
94   void Change_rayon( const double & r);
95   // change toutes les donnees
96   void change_donnees( const Coordonnee& B, const double& r);
97
98   // calcul les deux intercections M1 et M2 d'une droite avec la sphere,
99   // ramene 0 s'il n'y a pas d'intersection, ramene -1 si l'intercection
100  // ne peut pas etre calculee
101  // et 1 s'il y a deux points d'intercection
102  int Intersection( const Droite & D, Coordonnee& M1, Coordonnee& M2);
103
104  // calcul la distance d'un point à la sphere
105  double Distance_a_sphere(const Coordonnee& M) const;
106
107  // ramène true si le point est à l'intérieur de la sphère, false sinon
108  bool Dedans(const Coordonnee& M) const
109  { return ((centre - M).Norme() < rayon);};
110
111  // projection d'un point M sur la parois de la sphère
112  // dans le cas où M = le centre de la sphère, la projection n'est pas
113  // possible, dans ce cas projection_ok = false en retour, sinon true
114  Coordonnee Projete(const Coordonnee& M, bool& projection_ok) const;
115
116
117 protected :

```



```

118 // VARIABLES PROTEGEES :
119 Coordonnee centre; // centre de la sphere
120 double rayon; // rayon de la sphere
121 // METHODES PROTEGEES :
122
123 };
124 /// @} // end of group
125
126 #endif

```

## 7.97 ElemGeomC0.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           23/01/97
32 *
33 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:        Herezh++
36 *
37 * *****/
38 *   BUT:  Element geometrique generique pour une discrétisation C0.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !           but
45 *   !-----!-----!-----!-----!
46 *   !           !           !           !
47 *   *****
48 *   MODIFICATIONS:
49 *
50 *   ! date !   auteur !           but
51 *   !-----!-----!-----!-----!
52 *   $
53 * *****/
54 #ifndef ELEMGEOMCO_H
55 #define ELEMGEOMCO_H
56 // #include <bool.h>
57 #include "Mat_pleine.h"
58 #include "Tableau_T.h"
59 #include "Vecteur.h"
60 #include "Coordonnee.h"
61 #include "Enum_interpol.h"
62 #include "Enum_geom.h"
63 #include "Enum_type_pt_integ.h"
64
65 /** @defgroup Les_Elements_de_geometrie
66 *
67 *   BUT:  groupe concernant les éléments de géométrie 1D, 2D, 3D
68 *
69 *   \author   Gérard Rio
70 *   \version  1.0
71 *   \date     23/01/97
72 *   \brief    groupe concernant les éléments de géométrie 1D, 2D, 3D

```

```

73 *
74 */
75
76
77 /// @addtogroup Les_Elements_de_geometrie
78 /// @{
79 ///
80
81 class ElemGeomC0
82 {
83 public :
84     // CONSTRUCTEURS :
85     ElemGeomC0(); // pardefaut
86     // cas ou l'on connait la dimension = dim,
87     // le nombre de point d'integration nbi et le nombre
88     // de noeud de l'element nbne
89     // le nombre de face nbfe, le nombre de segment nbse
90
91     // IMPORTANT : par defaut le nombre de pt d'integ pour les faces ou segents
92     // s'il y en a, est calcule a partir du nombre total de point d'integ
93     // se referer a la description des elements
94     // par defaut le type de point d'integration est de gauss
95
96     ElemGeomC0(int dim,int nbi,int nbne,int nbfe,int nbse
97         ,Enum_geom geom, Enum_interpol interpol
98         ,Enum_type_pt_integ type_pti = PTI_GAUSS);
99     // de copie
100    ElemGeomC0(const ElemGeomC0& a);
101    // DESTRUCTEUR :
102    virtual ~ElemGeomC0();
103    // METHODES PUBLIQUES :
104    // retourne la dimension de l'element
105    inline int Dimension() const { return dimension;};
106    // retourne le nombre total de point d'integration
107    inline int Nbi() const { return tabPhi.Taille();};
108    // retourne le nombre total de noeud
109    inline int Nbne() const { return NBNE;};
110    // retourne le nombre de face
111    inline int NbFe() const { return NBFE;};
112    // retourne le nombre de segment
113    inline int NbSe() const { return NBSE;};
114    // retourne les coordonnees du point d'integration i
115    inline Coordonnee const & CoorPtInteg(int i) const { return ptInteg(i);};
116    // retourne les fonctions d'interpolation au point d'integration i
117    inline Vecteur const & Phi(int i) { return tabPhi(i);};
118    // retourne les tableau de fonctions d'interpolation
119    inline Tableau <Vecteur> const & TaPhi() const { return tabPhi;};
120    // retourne les derivees des fonctions d'interpolation au point d'integration i
121    inline Mat_pleine const& Dphi(int i) { return tabDPhi(i);};
122    // retourne le tableau des derivees des fonctions d'interpolation
123    inline Tableau < Mat_pleine > const& TaDphi() { return tabDPhi;};
124    // retourne les poids d'integration du point d'integration i
125    inline double Wi(int i) const { return WI(i);};
126    // retourne le vecteur des poids d'integration
127    inline Vecteur const & TaWi() const { return WI;};
128    // retourne l'"element face de référence" correspondant a la face i
129    inline ElemGeomC0 const & ElemFace(int i) const { return *face(i);};
130    // retourne l'"element segment de référence" correspondant a l'arrête i
131    inline ElemGeomC0 const & ElemSeg(int i) const { return *seg(i);};
132    // retourne true si l'element est complet sinon false
133    bool Complet() const ;
134    // retourne la connection des noeuds des faces par rapport a ceux de l'element
135    inline Tableau<Tableau<int> > const & Nonf() const { return NONE;};
136    // retourne la connection des noeuds des arêtes par rapport a ceux de l'element
137    inline Tableau<Tableau<int> > const & NonS() const { return NONS;};
138    // retourne les coordonnées des noeuds locaux
139    inline Tableau <Coordonnee > const & PtelemRef() const {return ptelem;};
140    // retourne le type de géométrie
141    inline Enum_geom TypeGeometrie() const {return id_geom;};
142    // retourne le type d'interpolation
143    inline Enum_interpol TypeInterpolation() const {return id_interpol;};
144    // retourne le type de points d'integration
145    inline Enum_type_pt_integ TypePointIntegration() const {return id_type_pt_integ;};
146
147    // cas de l'extrapolation de grandeur des points d'integrations aux noeuds
148    // ramène un tableau de pondération tab(i,j) qu'il faut appliquer
149    // aux noeuds pour avoir la valeur aux noeuds
150    // val_au_noeud(i) = somme_(de j=indir(i)(1) à indir(i)(taille(indir(i))) {tab(i)(j) *
151    // val_pt_integ(j) }
152    // cas = 1: la valeur au noeud = la valeur au pt d'integ le plus près ou une moyenne des
153    // pt les plus près (si le nb de pt d'integ < nb noeud)
154    class ConteneurExtrapolation
155    { // surcharge de l'operator de lecture avec le type
156      friend istream & operator » (istream &, ConteneurExtrapolation &);
157      // surcharge de l'operator d'écriture
158      friend ostream & operator « (ostream &, const ConteneurExtrapolation &);
159    public:

```

```

159     ConteneurExtrapolation() : tab(),indir() {};
160     ConteneurExtrapolation(const ConteneurExtrapolation &a) :
161         tab(a.tab),indir(a.indir) {};
162     ConteneurExtrapolation& operator= ( const ConteneurExtrapolation& a)
163     {tab=a.tab;indir=a.indir; return *this;};
164     // les tableaux publiques
165     Tableau<Tableau<double> > tab;
166     Tableau<Tableau<int> > indir;
167 };
168 inline ConteneurExtrapolation const & ExtrapolationNoeud(int cas) const { return extrapol(cas);};
169
170 // création d'éléments identiques : cette fonction est analogue à la fonction new
171 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
172 // dérivée
173 // pt est le pointeur qui est affecté par la fonction
174 virtual ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) = 0;
175
176 // creation d'une numérotation correspondant à un élément d'orientation inverse
177 // la numérotation est donnée par rapport à la numérotation normale de l'élément (de 1 à NBNE)
178 inline Tableau<int> const & InvConnec() const { return INVCONNEC;};
179
180 // ramène le tableau des tranches de numérotation
181 // c-a-d: ex: un quadrangle quadratique à 9 noeuds à 3 tranches: 4, 4, 1
182 // - la première tranche correspond aux 4 noeuds sommets, la numérotation peut être cyclique
(permutation circulaire)
183 // parmi ses 4 noeuds
184 // - la second tranche correspond aux 4 noeuds intermédiaires, là aussi, la numérotation peut être
une
185 // permutation circulaire parmi ses 4 noeuds
186 // - la troisième tranche correspond au noeud central
187 // NB: la somme de toutes les tranches = le nombre total de noeud de l'élément
188 inline Tableau<int> const & Ind() const { return IND;};
189
190 // ramène les tableaux de permutations permettant de calculer le tableau
191 // de connection permuté, ceci par rapport à la numérotation normale
192 // le second tableau correspond aux noeuds sommets (angle par exemple), les autres tableaux sont
193 // les noeuds internes aux arrêtes ou à l'élément
194 // chaque tableau contient le double du nombre de noeud associé,
195 // Utilisation: soit un noeud de numéro local nl, soit nl <= permut(2).Taille()= t1, dans ce cas il
s'agit du
196 // premier tableau : la numérotation sera pour i=1 à t1: permut(2)(nl+i-1)
197 // pour les numéros au-dessus de t1, on décale de permut(1)(1) = d1, c-a-d si permut(3).Taille()=t2
198 // la numérotation qui suit est : pour j=1 à t2 : permut(3)(
199 // en chantier inline Tableau<Tableau<int> > const & Permutation() const { return permut;};
200
201 //----- cas de coordonnees locales quelconques -----
202 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
203 virtual const Vecteur& Phi(const Coordonnee& M) = 0;
204 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
205 virtual const Mat_pleine& Dphi(const Coordonnee& M) = 0;
206 // en fonction de coordonnees locales, retourne true si le point est a l'interieur
207 // de l'element, false sinon
208 virtual bool Interieur(const Coordonnee& M) = 0;
209 // en fonction de coordonnees locales, retourne le point local P, maximum interieur à l'élément,
donc sur la frontière
210 // dont les coordonnées sont sur la droite GM: c-a-d GP = alpha GM, avec apha maxi et P appartenant
à la frontière
211 // de l'élément, G étant le centre de gravité, sauf si GGM est nul, dans ce cas retour de M
212 virtual Coordonnee Maxi_Coor_dans_directionGM(const Coordonnee& M)
213 { cout << "\n Pas implante pour l'instant, Maxi_Coor_dans_direction..."; Sortie(1);
214   Coordonnee P=M;
215   return P;}; // =0; pour l'instant essaie
216
217 //----- triangulation des faces lorsqu'elles existent -----
218 // le retour est la liste de la connection par rapport à la numérotation locale
219 // de tous les facettes triangulaire linéaire qui composent les faces de l'élément
220 // l'indice de premier niveau indique le numéro de la face
221 // l'indice de deuxième niveau correspond à la numérotation du triangle
222 // le troisième indice qui varie de 1 à 3 indique la connection
223 // tab(i)(j)(k) : pour la face i, le triangle j, le numéro du noeud dans la
224 // numérotation locale des noeuds de l'élément
225 const Tableau<Tableau<Tableau<int> > >& Trian_lin() const { return NONFt; };
226
227 //----- segmentation des arêtes -----
228 // le retour est la liste de la connection par rapport à la numérotation locale
229 // de tous les segments linéaire qui composent les arêtes de l'élément
230 // l'indice de premier niveau indique le numéro de l'arête
231 // l'indice de deuxième niveau correspond à la numérotation du segment
232 // le troisième indice qui varie de 1 à 2 indique la connection
233 // tab(i)(j)(k) : pour l'arête i, le segment j, le numéro du noeud dans la
234 // numérotation locale des noeuds de l'élément
235 const Tableau<Tableau<Tableau<int> > >& Trian_seg() const { return NONSs; };
236
237 // fonctions utilitaires spécifiques à l'interpolation linéaire:
238 // méthode utilisée pour extrapoler une grandeur à partir:
239 // soit de 2 points -> ex: pour une extrapolation linéaire (métrique en 1D)
240 // soit de 3 points -> ex: pour une extrapolation linéaire (métrique en 2D )

```

```

241 // soit de 4 points (non coplanaires) -> ex: pour une extrapolation linéaire (métrique en 3D )
242 // ces nipt points sont définis par le tableau tab_M,
243 //et en sortie on récupère les dim vecteurs de la base naturelle et les dim vecteurs de la base
duale
244 // dans le cas où le calcul n'est pas possible (points trop près) on ramène false, sinon true
245 static bool Bases_naturelles_duales(const Tableau <Coordonnee >& tab_M
246 ,Tableau <Coordonnee> & giB,Tableau <Coordonnee> & giH );
247 // on donne les vecteurs de la base naturelle, et les vecteurs de la base duale
248 // l'origine de la base O,
249 // et un point A,
250 // en sortie: les coordonnées locale de A dans la base naturelle, et les fonctions d'interpolation
251 // linéaire (1,2 ou 3D ce qui correspond à ligne, triangle, tetraedre) au point A
252 static void Coor_phi(const Coordonnee& O
253 ,const Tableau <Coordonnee> & giH_, const Coordonnee& A
254 ,Vecteur& phi, Coordonnee& theta) ;
255
256 protected :
257 // VARIABLES PROTEGEES :
258 int dimension; // dimension de l'element
259 int NBNE; // nombre de noeuds de l'element
260 int NBFE; // nombre de face par element
261 int NBSE; // nombre de segment par element
262 Tableau<Tableau<int> > NONE; // connexion des noeuds des faces par rapport a ceux des elements
263 // NONF(j) (i) noeud i de la face j = noeud de l'element
264 Tableau<Tableau<int> > NONS; // connexion des noeuds des segments par rapport a ceux des elements
265 // NONS(j) (i) noeud i du segment j = noeud de l'element
266 Tableau<int> INVCONNEX; // conection d'un élément d'orientation inverse du sens normal,
267 // la numérotation est donnée par rapport à la numérotation normale (de 1 à NBNE)
268 Tableau <int> IND; //tableau donnant les tranches de numérotation qui ont une numérotation
cyclique
269 // c-a-d: ex: un quadrangle quadratique à 9 noeuds à 3 tranches: 4, 4, 1
270 // - la première tranche correspond aux 4 noeuds sommets, la numérotation peut être cyclique
(permutation circulaire)
271 // parmi ses 4 noeuds
272 // - la second tranche correspond aux 4 noeuds intermédiaires, là aussi, la numérotation peut être
une
273 // permutation circulaire parmi ses 4 noeuds
274 // - la troisième tranche correspond au noeud central
275 // NB: la somme de toutes les tranches = le nombre total de noeud de l'élément
276
277
278 // Tableau<Tableau<int> > permut; // définit les tableaux de permutations permettant de calculer le
tableau
279 // de connection permuté, ceci par rapport à la numérotation normale
280 // le premier tableau correspond aux noeuds sommets (angle par exemple), les autres tableaux sont
281 // les noeuds internes aux arrêtes ou à l'élément
282 // en chantier !!!!!
283 // ramène les tableaux de permutations permettant de calculer le tableau
284 // de connection permuté, ceci par rapport à la numérotation normale
285 // le second tableau correspond aux noeuds sommets (angle par exemple), les autres tableaux sont
286 // les noeuds internes aux arrêtes ou à l'élément
287 // chaque tableau contient le double du nombre de noeud associé,
288 // Utilisation: soit un noeud de numéro local nl, soit nl <= permut(2).Taille()= t1, dans ce cas il
s'agit du
289 // premier tableau : la numérotation sera pour i=1 à t1: permut(2)(nl+i-1)
290 // pour les numéros au-dessus de t1, on décale de permut(1)(1) = d1, c-a-d si permut(3).Taille()=t2
291 // la numérotation qui suit est : pour j=1 à t2 : permut(3)(
292
293 // cas de l'extrapolation de grandeur des points d'intégrations aux noeuds
294 // def du tableau de pondération tab(i)(j) qu'il faut appliquer
295 // aux noeuds pour avoir la valeur aux noeuds
296 // val_au_noeud(i) = somme_(de j=borne_inf(i) à borne_sup(i)) {tab(i)(j) * val_pt_integ(j) }
297 // cas = 1: la valeur au noeud = la valeur au pt d'integ le plus près ou une moyenne des
298 // pt les plus près (si le nb de pt d'integ < nb noeud)
299 Tableau <ConteneurExtrapolation > extrapol;
300
301 // pour l'element
302 Tableau <Coordonnee > ptelem ; // coordonnees des points de l'élément
303 Tableau <Coordonnee > ptInteg ; // coordonnees des points d'integration
304 Tableau <Vecteur> tabPhi ; // tabPhi(ni) = phi = fonctions d'interpolation
305 // au point d'interpolation ni, phi a la dimension de nbne
306 Tableau < Mat_pleine > tabDPhi; // tabDPhi(ni) = Dphi tel que, Dphi(i,r) =
307 // valeur de la derivee de la fonction phi(r) par rapport a la coordonnee`
308 // locale i (= 1 ou 2 ou 3, ceci dependant de la dimension de l'element)
309 Vecteur WI; // poids d'integration
310 // pour les faces
311 Tableau < ElemGeomC0 * > face ; // pointe sur les elements faces
312 // pour les segments
313 Tableau < ElemGeomC0 * > seg ; // pointe sur les elements segments
314 // type de géométrie
315 Enum_geom id_geom;
316 // type d'interpolation
317 Enum_interpol id_interpol;
318 // type de point d'intégration
319 Enum_type_pt_integ id_type_pt_integ;
320
321 // pour la triangulation linéaires des faces lorsqu'elles existent

```

```

322 // connexion des noeuds des faces par rapport a ceux des elements
323 // l'indice de premier niveau indique le numéro de la face
324 // l'indice de deuxième niveau correspond à la numérotation du triangle
325 // le troisième indice qui varie de 1 à 3 indique la connexion
326 Tableau<Tableau<Tableau<int> > > NONFt;
327
328 // pour la segmentation linéaires des arêtes lorsqu'elles existent
329 // connexion des noeuds des arêtes par rapport a ceux des elements
330 // l'indice de premier niveau indique le numéro de l'arête
331 // l'indice de deuxième niveau correspond à la numérotation du segment
332 // le troisième indice qui varie de 1 à 2 indique la connexion
333 Tableau<Tableau<Tableau<int> > > NONSs;
334
335 // METHODES PROTEGEES :
336
337 // -- méthode identique à Bases_naturelles_duales sauf que l'on utilise une numérotation
338 // indirecte ce qui permet d'éviter de construire le tableau de coordonnées lorsque celui-ci
339 // existe globalement
340
341 // méthode utilisée pour extrapoler une grandeur à partir:
342 // soit de 2 points -> ex: pour une extrapolation linéaire (métrique en 1D)
343 // soit de 3 points -> ex: pour une extrapolation linéaire (métrique en 2D )
344 // soit de 4 points cas = 2 : extrapolation bi-linéaire (métrique en 2D)
345 // soit de 4 points (non coplanaires) cas = 1 -> ex: pour une extrapolation linéaire (métrique en 3D
)
346 // soit de 8 points cas = 1 : pour une extrapolation bi-linéaire (métrique en 3D)
347 // ces nipt points d'intégration, dont les numéros dans ptInteg sont défini par le tableau
indirec(i),
348 // i=1 à nipt, avec ptInteg(indirec(i)) le point à considérer, et en sortie on récupère les dim
vecteurs
349 // de la base naturelle et les dim vecteurs de la base duale
350 // dans le cas où le calcul n'est pas possible (points trop près) on ramène false, sinon true
351 bool Bases_naturel_duales(const Tableau <int>& indirec
352 ,Tableau <Coordonnee> & giB,Tableau <Coordonnee> & giH_
353 , int cas = 1) const;
354 };
355 /// @} // end of group
356
357 #endif

```

## 7.98 ElemGeomC1.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 * DATE: 23/01/97 *
32 * * $ *
33 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
34 * * $ *
35 * PROJET: Herezh++ *
36 * * $ *
37 *****/
38 * BUT: Element geometrique generique pour une discrétisation C1 *
39 * et semi C1. *
40 * Ici le nombre de fonction d'interpolation est supérieur au *
41 * nombre de noeud. *
42 * * $ *

```

```

43 *      *
44 *      VERIFICATION:
45 *
46 *      ! date ! auteur ! but
47 *      -----
48 *      ! ! ! !
49 *      $
50 *      *
51 *      MODIFICATIONS:
52 *      ! date ! auteur ! but
53 *      -----
54 *      $
55 *      *****/
56 #ifndef ELEMGEOMC1_H
57 #define ELEMGEOMC1_H
58
59 #include <bool.h>
60 #include "Mat_pleine.h"
61 #include "Tableau_T.h"
62 #include "Vecteur.h"
63 #include "Coordonnee.h"
64 #include "ElemGeomC0.h"
65
66 /// @addtogroup Les_Elements_de_geometrie
67 /// @{
68 ///
69
70
71 class ElemGeomC1 : public ElemGeomC0
72 {
73 public :
74 // CONSTRUCTEURS :
75 ElemGeomC1(); // pardefaut
76
77 ElemGeomC1(int dim,int nbi,int nbne,int nbfe,int nbse,Tableau<int> nddNoeud);
78 // cas ou l'on connait la dimension = dim,
79 // le nombre de point d'integration nbi et le nombre
80 // de noeud de l'element nbne
81 // le nombre de face nbfe, le nombre de segment nbse
82
83 // IMPORTANT : par defaut le nombre de pt d'integ pour les faces ou segents
84 // s'il y en a, est calcule a partir du nombre total de point d'integ
85 // se referer a la description des elements
86
87 // nddNoeud : nombre de ddl par noeud, qui sont utilisé pour
88 // définir la géométrie
89
90
91 ElemGeomC1(ElemGeomC1& a); // de copie
92 // DESTRUCTEUR :
93 ~ElemGeomC1();
94 // METHODES PUBLIQUES :
95
96 protected :
97 // VARIABLES PROTEGEES en plus de celles définies dans ElemGeomC0:
98 Tableau<int> nddNoeud; // nombre de ddl par noeud, qui sont utilisé pour
99 // définir la géométrie
100 // pour l'element
101 //1) les tableaux définis dans ElemGeomC0
102 // ---> Tableau <Vecteur> tabPhi ; // tabPhi(ni) = phi = fonctions d'interpolation
103 // au point d'interpolation ni, phi a la dimension de la somme des valeurs de
104 // nddNoeud, et non le nombre de noeud comme dans le cas de ElemGeomC0
105 // ---> Tableau < Mat_pleine > tabDPhi; // tabDPhi(ni) = Dphi tel que, Dphi(i,r) =
106 // valeur de la derivee de la fonction phi(r) par rapport a la coordonnee`
107 // locale i (= 1 ou 2 ou 3, ceci dependant de la dimension de l'element)
108 // ici r varie de 1 à somme des valeurs de nddNoeud
109
110 // METHODES PROTEGEES :
111 };
112 /// @} // end of group
113
114 #endif

```

## 7.99 GeomSeg.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.

```

```

10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29 //
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:   Définir La geometrie d'un segment .
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date ! auteur ! but
45 *   -----!
46 *   ! ! ! !
47 *   *****
48 *   MODIFICATIONS:
49 *
50 *   ! date ! auteur ! but
51 *   -----!
52 *   *****/
53 #ifndef SEGMENT_H
54 #define SEGMENT_H
55
56 #include"ElemGeomC0.h"
57
58 /*
59 // lieaire -> 2 noeuds
60 //
61 //      -1      0      1      KSI
62 //      ---*-----|-----*----->
63 //      (1)      (2)
64 //
65 // quadratique -> 3 noeuds
66 //
67 //      -1      0      1      KSI
68 //      ---*-----|-----*----->
69 //      (1)      (2)      (3)
70 //
71 // cubique -> 4 noeuds
72 //
73 //      -1      0      1      KSI
74 //      ---*-----*-----*-----*----->
75 //      (1)      (2)      (3)      (4)
76 //
77 // points d'intégration de Gauss: code = nbi
78 // 1 pt d'integ -> KSI = 0;
79 // 2 pt d'integ -> KSI = -1/sqrt(3), 1/sqrt(3);
80 // 3 pt d'integ -> KSI = -sqrt(3./5.), 0, sqrt(3./5.);
81 // 4 pt d'integ -> KSI = -T1, -T, T, T1 , T1 et T environ 0.86 et 0.33
82 // avec T = sqrt( (3.-S)/7) et T1 =sqrt((3.+2.*sqrt(6./5.))/7.);
83 // ce qu'il faut retenir c'est que les points sont a suivre
84 // points d'intégration de Gauss-Lobatto: code = 2000+nbi
85 // de 1 à 16 points, qui sont connus uniquement sous forme numérique
86 */
87
88 /// @addtogroup Les_Elements_de_geometrie
89 /// @{
90 ///
91
92
93 class GeomSeg : public ElemGeomC0
94 {
95 public :

```

```

96 // CONSTRUCTEURS :
97 // par défaut on considere un segment un point d'integ et a deux noeuds
98 // par défaut ce sont des points de gauss, cependant, il est possible d'utiliser
99 // également une intégration de Gauss-lobatto
100 GeomSeg(int code_nbi =1, int nbe = 2, Enum_type_pt_integ type_pti = PTI_GAUSS);
101 // de copie
102 GeomSeg(const GeomSeg& a);
103 // DESTRUCTEUR :
104 ~GeomSeg() {};
105
106 // retourne le tableau des derivees secondes des fonctions d'interpolation
107 inline Tableau < Mat_pleine >& taD2phi() { return tabD2Phi;};
108
109 // création d'éléments identiques : cette fonction est analogue à la fonction new
110 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
111 // dérivée
112 // pt est le pointeur qui est affecté par la fonction
113 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
114
115 //----- cas de coordonnees locales quelconques -----
116 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
117 const Vecteur& Phi(const Coordonnee& M);
118 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
119 const Mat_pleine& Dphi(const Coordonnee& M);
120 // en fonction de coordonnees locales, retourne true si le point est a l'interieur
121 // de l'element, false sinon
122 bool Interieur(const Coordonnee& M);
123 // en fonction de coordonnees locales, retourne le point local P, maximum interieur à l'élément,
124 // donc sur la frontière
125 // dont les coordonnées sont sur la droite GM: c-a-d GP = alpha GM, avec apha maxi et P appartenant
126 // à la frontière
127 // de l'élément, G étant le centre de gravité, sauf si GM est nul, dans ce cas retour de M
128 Coordonnee Maxi_Coor_dans_directionGM(const Coordonnee& M);
129
130 // ----- methodes particuliere-----
131 // fourni la coordonnees ksi du point d'integ i
132 inline double& KSI(int i) { return ptInteg(i)(1);};
133
134 protected :
135 // VARIABLES PROTEGEES :
136 Tableau < Mat_pleine > tabD2Phi; // tabD2Phi(ni) = D2phi pour le point d'intégration ni
137 // tel que, D2phi(i,r) =
138 // valeur de la derivee seconde de la fonction phi(r) par rapport a la coordonnee `
139 // locale i = 1
140
141 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
142 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
143 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
144
145 // METHODES PROTEGEES :
146 // constitution du tableau Extrapol
147 void Calcul_extrapol(int nbi);
148
149 // concernant l'intégration à l'aide des points et poids de Gauss_Lobatto
150 static Tableau <Tableau <double > > ksi_gl;
151 static Tableau <Tableau <double > > wi_gl;
152 class Construire_Gauss_Lobatto {public: Construire_Gauss_Lobatto();};
153 static Construire_Gauss_Lobatto construire_gauss_lobatto;
154
155 };
156 /// @} // end of group
157 #endif

```

## 7.100 GeomPoint.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.

```



```

19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      06/01/00
32 *
33 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:   Définir La geometrie d'un point .
39 *         L'interpolation est constant.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *   *****
48 *
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *   *****/
53
54 #ifndef POINT_H
55 #define POINT_H
56
57 #include "ElemGeomC0.h"
58
59
60 /// @addtogroup Les_Elements_de_geometrie
61 /// @{
62 ///
63
64
65 class GeomPoint : public ElemGeomC0
66 {
67 public :
68     // CONSTRUCTEURS :
69     // par défaut on considere un point d'integ et un noeud
70     GeomPoint();
71     // de copie
72     GeomPoint(const GeomPoint& a);
73     // DESTRUCTEUR :
74     ~GeomPoint() {};
75
76     // retourne le tableau des derivees secondes des fonctions d'interpolation
77     inline Tableau < Mat_pleine >& taD2phi() { return tabD2Phi;};
78
79     // création d'élément identiques : cette fonction est analogue à la fonction new
80     // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
81     // dérivée
82     // pt est le pointeur qui est affecté par la fonction
83     ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
84
85     //----- cas de coordonnees locales quelconques -----
86     // retourne les fonctions d'interpolation au point M (en coordonnees locales)
87     const Vecteur& Phi(const Coordonnee& M);
88     // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
89     const Mat_pleine& Dphi(const Coordonnee& M);
90     // en fonction de coordonnees locales, retourne true si le point est a l'interieur
91     // de l'element, false sinon
92     bool Interieur(const Coordonnee& M);
93     // en fonction de coordonnees locales, retourne le point local P, maximum interieur à l'élément, donc
94     // sur la frontière
95     // dont les coordonnées sont sur la droite GM: c-a-d GP = alpha GM, avec apha maxi et P appartenant à
96     // la frontière
97     // de l'élément, G étant le centre de gravité, sauf si GM est nul, dans ce cas retour de M
98     Coordonnee Maxi_Coor_dans_directionGM(const Coordonnee& M);
99
100     // ----- methodes particuliere-----
101     // fourni la coordonnees ksi du point d'integ i
102     inline double& KSI(int i) { return ptInteg(i)(1);};
103
104 protected :

```

```

103
104 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
105 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
106 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
107
108 // VARIABLES PROTEGEES :
109 Tableau < Mat_pleine > tabD2Phi; // tabD2Phi(ni) = D2phi pour le point d'integration ni
110 // tel que, D2phi(i,r) =
111 // valeur de la derivee seconde de la fonction phi(r) par rapport a la coordonnee `
112 // locale i = 1
113 // METHODES PROTEGEES :
114
115
116 };
117 /// @} // end of group
118
119 #endif

```

## 7.101 GeomQuadrangle.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:  Définir La geometrie du quadrangle .
39 *           les points sur la surface sont calcules par
40 *           la methode produit.
41 *
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      !           !           !
47 *
48 *
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *
52 *
53 *****/
54 #ifndef GEOMQUADRANGLE_H
55 #define GEOMQUADRANGLE_H
56
57 #include "ElemGeomCO.h"
58
59 /*
60 // description de la numerotation employee
61 //
62 // cas des noeuds:
63 //
64 //      lineaire          quadratique          lineaire/quadratique          cubique complet

```

```

65 //
66 // (4) (3) (4) (7) (3) (4) (6) (3)
67 // *-----* *-----* *-----* (4)--(10)--(9)--(3)
68 // | | | | | | | | | | | | | | | |
69 // | | | | | | | | | | | | | | | |
70 // | | | | | (8)* (9) * (6) | | | | | |
71 // | | | | | | | | | | | | | | | |
72 // | | | | | | | | | | | | | | | |
73 // *-----* *-----* *-----* (12)-(13)--(14)-(7)
74 // (1) (2) (1) (5) (2) (1) (5) (2) (1)---(5)---(6)---(2)
75 //
76 // dans le cas d'un quadratique incomplet, nbne = 8, le noeud (9) est supprime
77 //
78 // face 1 : noeuds de l'élément
79 // on attribue le même nombre de points d'integration pour la face que pour l'élément
80 //
81 // pour les aretes on suis le fichier Elmail, 4 aretes
82 // 1) pour le quadrangle bilinéaire
83 // 1 2 2 3 3 4 4 1
84 // 2) pour le quadrangle quadratique complet ou incomplet
85 // 1 5 2 2 6 3 3 7 4 4 8 1
86 // 3) pour le quadrangle cubique complet
87 // 1 5 6 2 2 7 8 3 3 9 10 4 4 11 12 1
88 //
89 // on attribue la racine carré du nombre de point d'intégration de l'élément pour l'arrête
90 // d'ou en général : 1 point d'integration par arete pour les bilinéaires et 2 points pour
91 // les quadratiques
92 //
93 //
94 // cas des points d'integration
95 // ^ y ^ y
96 // | | | | | |
97 // | | | | | |
98 // 1 point 4 points
99 // *-----* *-----*
100 // | | | | | | | | | |
101 // | | | | | | | | | |
102 // | (1) | | | | | | | | | |
103 // | | | | | | | | | |
104 // | | | | | | | | | |
105 // *-----* *-----*
106 //
107 // 9 points 16 points
108 // *-----* *-----*
109 // | (7) (8) (9) | | (13) (14) (15) (16) |
110 // | | | | | | | | | |
111 // | (4) (5) (6) | --> x | (9) (10) (11) (12) | --> x
112 // | | | | | | | | | |
113 // | (1) (2) (3) | | | | | | | | | |
114 // *-----* *-----*
115 // | | | | | | | | | |
116 // | (1) (2) (3) (4) |
117 // *-----*
118 // concernant la triangulation linéaire de l'élément :
119 // - pour l'élément linéaire : décomposé en 2 éléments triangulaires de connexion :
120 // 1e : 1 2 3, 2e : 1 3 4
121 // - pour l'élément quadratique incomplet : décomposé en 6 éléments triangulaires de connexion :
122 // 1e : 1 5 8, 2e : 8 5 6, 3e : 5 2 6, 4e : 6 3 7, 5e : 8 6 7, 6e : 8 7 4
123 // - pour l'élément quadratique complet : décomposé en 8 éléments triangulaires de connexion :
124 // 1e : 8 5 9, 2e : 5 6 9, 3e : 9 6 7, 4e : 8 9 7, 5e : 1 5 8, 6e : 5 2 6, 7e : 6 3 7, 8e : 8 7
125 // 4
126 // - pour l'élément linéaire/quadratique : décomposé en 4 éléments triangulaires de connexion :
127 // 1e : 1 6 4, 2e : 1 5 6, 3e : 5 3 6, 4e : 5 2 3
128 // - pour l'élément cubique complet : décomposé en 18 éléments triangulaires de connexion :
129 // 1e : 1 5 12, 2e : 5 13 12, 3e : 5 6 13, 4e : 6 14 13
130 // 5e : 6 2 14, 6e : 2 7 14, 7e : 12 13 11, 8e : 13 16 11
131 // 9e : 13 14 16, 10e : 14 15 16, 11e : 14 7 15, 12e : 7 8 15
132 // 13e : 11 16 4, 14e : 16 10 4, 15e : 16 15 10, 16e : 15 9 10
133 // 17e : 15 8 9, 18e : 8 3 9
134 // */
135 //
136 //
137 // @addtogroup Les_Elements_de_geometrie
138 // @{
139 //
140 //
141 //
142 class GeomQuadrangle : public ElemGeomC0
143 {
144 public :
145 // CONSTRUCTEURS :
146 // par défaut on a 4 pt d'integ et 4 noeuds
147 // sans_extrapole : cas particulier pour lequel on ne construit pas le
148 // tableau d'extrapolation: nécessaire pour justement définir ce tableau pour

```

```

149 // des éléments particuliers (ex: hexaèdre) en évitant une boucle récursive
150 // via l'utilisation de sous-éléments particuliers (quadrangle, hexaèdre etc.)
151 GeomQuadrangle(int nbi = 4, int nbne = 4, int sans_extrapole = 0);
152 // de copie
153 GeomQuadrangle(const GeomQuadrangle& a);
154 // DESTRUCTEUR :
155 ~GeomQuadrangle();
156
157 // création d'élément identiques : cette fonction est analogue à la fonction new
158 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
159 // dérivée
160 // pt est le pointeur qui est affecté par la fonction
161 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
162
163 //----- cas de coordonnees locales quelconques -----
164 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
165 const Vecteur& Phi(const Coordonnee& M);
166 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
167 const Mat_pleine& Dphi(const Coordonnee& M);
168 // en fonction de coordonnees locales, retourne true si le point est a l'interieur
169 // de l'element, false sinon
170 bool Interieur(const Coordonnee& M);
171 // en fonction de coordonnees locales, retourne le point local P, maximum interieur à l'élément,
172 // donc sur la frontière
173 // dont les coordonnées sont sur la droite GM: c-a-d GP = alpha GM, avec apha maxi et P appartenant
174 // à la frontière
175 // de l'élément, G étant le centre de gravité, sauf si GM est nul, dans ce cas retour de M
176 Coordonnee Maxi_Coor_dans_directionGM(const Coordonnee& M);
177
178 protected :
179 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
180 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
181 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
182
183 // METHODES PROTEGEES :
184 // constitution du tableau Extrapol
185 void Calcul_extrapol(int nbi);
186 };
187 /// @} // end of group
188 #endif

```

## 7.102 GeomTriangle.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *   *****
38 *   BUT:      Définir les info qui sont communs aux elements
39 *             triangle.
40 *

```

```

41 *      *
42 *      VERIFICATION:
43 *
44 *      ! date !   auteur !           but
45 *      -----
46 *      !           !           !           !
47 *
48 *      *
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      -----
52 *      $
53 *      *****/
54 #ifndef GEOMTRIANGLE_H
55 #define GEOMTRIANGLE_H
56
57 #include "ElemGeomC0.h"
58
59 /*
60 // les fonctions d'interpolation sont definis pour lineaire quadratique cubique
61 //
62 // lineaire:                quadratique :                (3)                cubique et cubique incomplet
63 //      (3) | \                (3) | \                | \                = 9 premiers
64 //      noeuds                | \                | \                | \
65 //      | \                (6) | \ (5)                (8) | \ (7)
66 //      (1) | \ (2)                | \ (2)                (9) | (10) \ (6)
67 //
68 //
69 //
70 //      concernant les aretes
71 //      elles sont lineaire quadratique ou cubique comme l'element (4) (5)
72 //      numerotation des arretes :
73 //      en lineaire :      1 : 1 2,      2 : 2 3,      3 : 3 1
74 //      en quadratique :   1 : 1 4 2,    2 : 2 5 3,    3 : 3 6 1
75 //      en cubique :      1 : 1 4 5 2   2 : 2 6 7 3   3 : 3 8 9 1
76 //      leur nombre de point d'integration nbil depend du nombre nbi de l'element
77 //      nbi = 1 -> nbil = 1
78 //      nbi = 3 ou 4 -> nbil = 2
79 //      concernant la triangulation lineaire de l'element :
80 //      - pour l'element lineaire : decompose en lui meme
81 //      - pour l'element quadratique : decompose en 4 elements triangulaires de connexion :
82 //      1e : 6 4 5, 2e : 1 4 6, 3e : 4 2 5, 4e : 6 5 3
83 //      - pour l'element cubique : decompose en 9 elements triangulaires de connexion
84 //      1e : 1 4 9 2e : 10 9 4 3e : 4 5 10 4e : 6 10 5
85 //      5e : 5 2 6 6e : 9 10 8 7e : 7 8 10 8e : 10 6 7 9e : 8 7 3
86 //
87 //      concernant les points d'integrations
88 //
89 //      un point -> au centre de gravite
90 //
91 //      3 points                3 points (code 1003)                4 points                6 points
92 //
93 //      | \                | \                | \                | \
94 //      | \                | \                | \                | \
95 //      | \                | 3 \                | 4 \                | 6 \
96 //      | \                | \                | \                | \
97 //      2      1                | \                | \                | \
98 //      | \                | \                | 1 \                | 2 1
99 //      | \                | \                | \                | \
100 //      | \                | \                | \                | \
101 //      | \                | 1 2 \                | 2 3 \                | 4 3 5 \
102 //      | \                | \                | \                | \
103 //      | \                | \                | \                | \
104 //      | \                | \                | \                | \
105 //      | \                | \                | \                | \
106 //      | 7 \                | 3 \                | 12 11 \                | 6
107 //      | \                | \                | \                | \
108 //      | \                | \                | \                | \
109 //      | \                | \                | \                | \
110 //      | \                | \                | \                | \
111 //      | 3 \                | \                | \                | \
112 //      | \                | \                | \                | \
113 //      | \                | \                | \                | \
114 //      | 5 \                | \                | \                | \
115 //      | \                | \                | \                | \
116 //      | \                | \                | \                | \

```

```

117 */
118
119
120
121
122
123
124 /// @addtogroup Les_Elements_de_geometrie
125 /// @{
126 ///
127
128
129 class GeomTriangle : public ElemGeomC0
130 {
131 public :
132 // CONSTRUCTEURS :
133 // par default on suppose un point d'integration et
134 // un triangle lineaire a 3 noeuds
135 GeomTriangle( int nbi = 1, int nbne = 3);
136 // de copie
137 GeomTriangle(const GeomTriangle& a);
138 // DESTRUCTEUR :
139 ~GeomTriangle();
140 // methodes particuliere
141
142 // création d'élément identiques : cette fonction est analogue à la fonction new
143 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
144 // dérivée
145 // pt est le pointeur qui est affecté par la fonction
146 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
147
148 //----- cas de coordonnees locales quelconques -----
149 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
150 const Vecteur& Phi(const Coordonnee& M);
151 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
152 const Mat_pleine& Dphi(const Coordonnee& M);
153 // en fonction de coordonnees locales, retourne true si le point est a l'interieur
154 // de l'element, false sinon
155 bool Interieur(const Coordonnee& M);
156 // en fonction de coordonnees locales, retourne le point local P, maximum interieur à l'élément,
157 // donc sur la frontière
158 // dont les coordonnées sont sur la droite GM: c-a-d GP = alpha GM, avec apha maxi et P appartenant
159 // à la frontière
160 // de l'élément, G étant le centre de gravité, sauf si GM est nul, dans ce cas retour de M
161 Coordonnee Maxi_Coor_dans_directionGM(const Coordonnee& M);
162
163 protected :
164
165 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
166 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
167 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
168
169 // METHODES PROTEGEES :
170 // fourni la coordonnees ksi du point d'integ i
171 inline double& KSI(int i) { return ptInteg(i)(1);};
172 inline double& ETA(int i) { return ptInteg(i)(2);};
173 // constitution du tableau Extrapol
174 void Calcul_extrapol(int nbi);
175
176 };
177 /// @} // end of group
178
179 #endif

```

## 7.103 GeomHexaCom.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //

```

```

20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      19/12/99
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 * *****/
38 *   BUT:  Définir Les éléments geometrique commun aux hexaedres.
39 *         Fonction d'interpolation, points d'integration etc
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   -----
47 *   !       !           !
48 *   *****
49 *
50 *   MODIFICATIONS:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   $
55 * *****/
54 #ifndef GEOMHEXACOM_H
55 #define GEOMHEXACOM_H
56
57 #include"ElemGeomC0.h"
58
59 /*
60 // pour l'élément linéaire
61 // -----
62 //
63 //   ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
64 //
65 //   Nicolas COUTY   04/12/95
66 //   Source : Dhatt et Touzot p 133, 134, 293
67 //   + donnee suivante des valeurs des fonctions
68 //   d'interpolation aux points d'integration
69 //   (qui a permis la verification)
70 // -----
71 //
72 //
73 //
74 //
75 //
76 //
77 //
78 //
79 //
80 //
81 //
82 //
83 //
84 //
85 //
86 //
87 //
88 //
89 //
90 //
91 //
92 //
93 //
94 //
95 // Points d'integration par défaut
96 // a=1/racine(3)
97 // Pt1 (a,a,a) ; Pt2 (a,a,-a) ; Pt3 (a,-a,a) ; Pt4 (a,-a,-a)
98 // Pt5 (-a,a,a) ; Pt6 (-a,a,-a) ; Pt7 (-a,-a,a) ; Pt8 (-a,-a,-a)
99 //
100 //
101 // face 1 : noeud 1 4 3 2, face 2 : noeud 1 5 8 4,
102 // face 3 : noeud 1 2 6 5, face 4 : noeud 5 6 7 8,
103 // face 5 : noeud 2 3 7 6, face 6 : noeud 3 4 8 7,
104 // les normales sortent des faces des elements
105 // on attribue 4 points d'integration par face

```

```

106 //
107 // pour les aretes on suis le fichier Elmail, 12 aretes
108 //   1 2   2 3   3 4   4 1
109 //   1 5   2 6   3 7   4 8
110 //   5 6   6 7   7 8   8 5
111 //
112 //   on attribu 1 point d'integration par arete
113 //
114 //
115 // *****
116 //
117 // pour l'élément quadratique incomplet
118 //
119 //-----
120 //
121 //   ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
122 //
123 //   Nicolas COUTY   04/12/95
124 //   Source : Pour l'element : Modulef Guide No 2, p 139
125 //   Source : Pour les fonctions d'interpolation :
126 //             Dhatt et Touzot p 135
127 //             + donnee suivante des valeurs des fonctions
128 //             d'interpolation aux points d'integration
129 //             (qui a permis la verification)
130 //
131 //-----*
132 //
133 //
134 //
135 //
136 //
137 //
138 //
139 //
140 //
141 //
142 //
143 //
144 //
145 //
146 //
147 //
148 //
149 //
150 //
151 //
152 //
153 //
154 //
155 // Points d'integration 8 par défaut
156 // a=1/racine(3)
157 // Pt1 (a,a,a) ; Pt2 (a,a,-a) ; Pt3 (a,-a,a) ; Pt4 (a,-a,-a)
158 // Pt5 (-a,a,a) ; Pt6 (-a,a,-a) ; Pt7 (-a,-a,a) ; Pt8 (-a,-a,-a)
159 //
160 // sinon on utilise les points d'intégrations calculés à partir du segment
161 // et on a 1,2x2x2, 3x3x3, 4x4x4 etc.
162 //
163 // face 1 : noeud 1 4 3 2 12 11 10 9, face 2 : noeud 1 5 8 4 13 20 16 12,
164 // face 3 : noeud 1 2 6 5 9 14 17 13, face 4 : noeud 5 6 7 8 17 18 19 20,
165 // face 5 : noeud 2 3 7 6 10 15 18 14, face 6 : noeud 3 4 8 7 11 16 19 15,
166 // les normales sortent des faces des elements
167 // on attribue 4 points d'integration par face
168 //
169 // pour les aretes on suis le fichier Elmail, 12 aretes
170 //   1 9 2   2 10 3   3 11 4   4 12 1
171 //   1 13 5   2 14 6   3 15 7   4 16 8
172 //   5 17 6   6 18 7   7 19 8   8 20 5
173 //
174 //
175 //   on attribue 2 point d'integration par arete par défaut
176 //
177 //   concernant la triangulation de chaque face elle est réalisée à l'aide
178 //   de la triangulation implantée sur l'élément de référence de la face
179 //
180 //
181 //-----
182 //
183 // dans le cas où l'on sort des points d'intégrations par défaut on se sert
184 // d'une combinaison de segment pour recréer l'hexaèdre ce qui permet
185 // d'avoir 1x1x1, ou 2x2x2, ou 3x3x3, ou 4x4x4 etc. pt d'integ
186 // pour l'élément quadratique incomplet
187 //
188 //-----
189 //
190 // pour l'élément quadratique complet
191 //
192 //

```



```

193 //-----*
194 // *
195 //   ELEMENT DE REFERENCE , POINTS D'INTEGRATION: *
196 // *
197 //   construction à partir des deux précédents éléments *
198 // *
199 //-----*
200 //
201 //
202 //
203 //
204 //
205 //
206 //
207 //
208 //
209 //
210 //
211 //
212 //
213 //
214 //
215 //
216 //
217 //
218 //
219 //
220 //
221 //
222 //
223 // par rapport au quadratique incomplet, 21 est au centre de la face 1,
224 // 22 sur la face 3, 23 sur la face 5, 24 sur la face 6
225 // 25 sur la face 2, 26 sur la face 4, 27 au centre de l'élément
226 // Points d'integration 8 par défaut
227 // a=1/racine(3)
228 // Pt1 (a,a,a) ; Pt2 (a,a,-a) ; Pt3 (a,-a,a) ; Pt4 (a,-a,-a)
229 // Pt5 (-a,a,a) ; Pt6 (-a,a,-a) ; Pt7 (-a,-a,a) ; Pt8 (-a,-a,-a)
230 //
231 // sinon on utilise les points d'intégrations calculés à partir du segment
232 // et on a 1,2x2x2, 3x3x3, 4x4x4 etc.
233 //
234 // face 1 : noeud 1 4 3 2 12 11 10 9 21, face 2 : noeud 1 5 8 4 13 20 16 12 25,
235 // face 3 : noeud 1 2 6 5 9 14 17 13 22, face 4 : noeud 5 6 7 8 17 18 19 20 26,
236 // face 5 : noeud 2 3 7 6 10 15 18 14 23, face 6 : noeud 3 4 8 7 11 16 19 15 24,
237 // les normales sortent des faces des elements
238 // on attribue 4 points d'integration par face
239 //
240 // pour les aretes on suis le fichier Elmail, 12 aretes
241 //   1 9 2   2 10 3   3 11 4   4 12 1
242 //   1 13 5   2 14 6   3 15 7   4 16 8
243 //   5 17 6   6 18 7   7 19 8   8 20 5
244 //
245 //
246 //   on attribue 2 point d'integration par arete par défaut
247 //
248 // concernant la triangulation de chaque face elle est réalisée à l'aide
249 // de la triangulation implantée sur l'élément de référence de la face
250 //
251 //
252 // *****
253 */
254
255 // dans le cas où l'on sort des points d'intégrations par défaut on se sert
256 // d'une combinaison de segment pour recréer l'hexaèdre ce qui permet
257 // d'avoir 1x1x1, ou 2x2x2, ou 3x3x3, ou 4x4x4 etc. pt d'integ
258
259 // dans le cas où on utilise 27 pti, la numérotation est la suivante
260 // ( ici on ne représente pas le contour de l'élément)
261 //
262 //
263 //
264 //
265 //
266 //
267 //
268 //
269 //
270 //
271 //
272 //
273 //
274 //
275 //
276 //
277 //
278 //
279 //

```

```

280 //
281 //
282 // dans le cas où on utilise 64 pti, la numérotation suit la même logique
283 // on va indiquer les numéros par couche
284 //
285 // couche 1) 27pti
286 // *-----*
287 // | (7) (8) (9) |
288 // | | |
289 // | (4) (5) (6) | --> xi
290 // | | |
291 // | (1) (2) (3) |
292 // *-----*
293 //
294 //
295 // couche 2) 27pti
296 // *-----*
297 // | (16) (17) (18) |
298 // | | |
299 // | (13) (14) (15) | --> xi
300 // | | |
301 // | (10) (11) (12) |
302 // *-----*
303 //
304 //
305 // couche 3) 27pti
306 // *-----*
307 // | (25) (26) (27) |
308 // | | |
309 // | (22) (23) (24) | --> xi
310 // | | |
311 // | (19) (20) (21) |
312 // *-----*
313 //
314 //
315 // couche 4)
316 //
317 //
318 //
319 //
320 //
321 //
322 //
323 //
324 //
325 //
326 // pour ne pas surcharger la figure, on indique les pti de la base
327 // puis uniquement sur les arêtes
328 // à noter qu'ici on n'indique pas le cube d'interpolation des noeuds
329 // qui englobe les pti !
330 //
331 // |zeta
332 // 49-----53-----|57-----61
333 // | \ | | | | |
334 // | 50 | | | | | 62
335 // | 33 | | | | | 45
336 // | 51 | | | | | 63
337 // | | | | | |
338 // | 52 |-----56-----|60-----64
339 // | 17 | | | | | 29
340 // | | | | | |
341 // | 36 |-----48-----|eta
342 // | 1 |-----5-----9-----|13 17 |
343 // | 2 | | 6 | 10 | | 14 32
344 // | 20 | | | | | |
345 // | 3 | | 7 | 11 | | 15 |
346 // | | | | | |
347 // | 4 |-----8-----12-----|16
348 // | | | | | |
349 // | | | | | |
350 // | | | | | |
351 // | | | | | |
352 //
353 /// @addtogroup Les_Elements_de_geometrie
354 /// @{
355 ///
356 ///
357 ///
358 class GeomHexaCom : public ElemGeomC0
359 {
360 public :
361 // CONSTRUCTEURS :
362 // le constructeur par défaut ne doit pas être utilisé
363 GeomHexaCom();
364
365 // constructeur en fonction du nombre de noeud et du nombre de point d'intégration
366 // et du type d'interpolation

```

```

367     GeomHexaCom(int nbi, int nbe, Enum_interpol interpol);
368     // de copie
369     GeomHexaCom(const GeomHexaCom& a);
370     // DESTRUCTEUR :
371     ~GeomHexaCom();
372
373     //----- cas de coordonnees locales quelconques -----
374     // en fonction de coordonnees locales, retourne true si le point est a l'interieur
375     // de l'element, false sinon
376     bool Interieur(const Coordonnee& M);
377     // en fonction de coordonnees locales, retourne le point local P, maximum interieur a l'element,
378     // donc sur la frontiere
379     // dont les coordonnees sont sur la droite GM: c-a-d GP = alpha GM, avec apha maxi et P appartenant
380     // a la frontiere
381     // de l'element, G etant le centre de gravite, sauf si GM est nul, dans ce cas retour de M
382     Coordonnee Maxi_Coor_dans_directionGM(const Coordonnee& M);
383
384     protected :
385     // METHODES PROTEGEES :
386     // constitution du tableau Extrapol
387     void Calcul_extrapol(int nbi);
388
389 };
390 /// @} // end of group
391 #endif

```

## 7.104 GeomHexaCubique.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          10/02/2012
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *
38 *      BUT:  Définir La geometrie de l'hexaedre cubique complet.
39 *      Fonction d'interpolation, points d'integration etc
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date ! auteur ! but
45 *      -----
46 *      ! ! !
47 *      $
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date ! auteur ! but
51 *      -----
52 *      $
53 *      *****/
54 #ifndef GEOMHEXACUBIQUE_H
55 #define GEOMHEXACUBIQUE_H

```

```

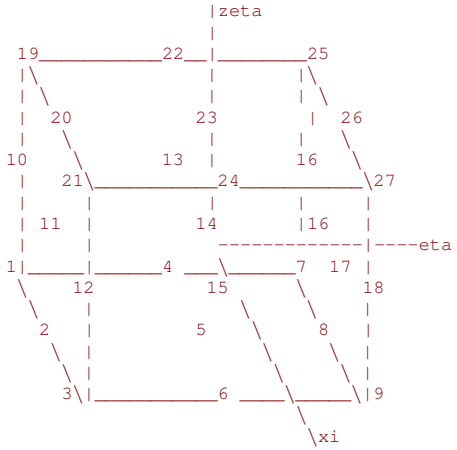
56
57 #include"GeomHexaCom.h"
58
59 // l'élément cubique complet
60
61 /*
62 //*****
63 //
64 //      ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
65 //
66 //
67 //-----*
68 //
69 //              |zeta(z)
70 //              |
71 //      5-----32-----|_31_8
72 //      |\          |          |
73 //              |          |          |          |          |
74 //      | 25          |          |          |          |          |
75 //      18 \ 26          |          |          |          |          |
76 //      | \          |          |          |          |          |
77 //      |          |          |          |          |          |
78 //      |          |          |          |          |          |
79 //      17          |          |          |          |          |
80 //      |          |          |          |          |          |
81 //      |          |          |          |          |          |
82 //      |          |          |          |          |          |
83 //      |          |          |          |          |          |
84 //      |          |          |          |          |          |
85 //      |          |          |          |          |          |
86 //      |          |          |          |          |          |
87 //      |          |          |          |          |          |
88 //      |          |          |          |          |          |
89 //      |          |          |          |          |          |
90 //      |          |          |          |          |          |
91 //      |          |          |          |          |          |
92 //      |          |          |          |          |          |
93 //
94 //      (1)---(16)---(15)---(4)          (5)---(25)---(26)---(6)          (5)---(32)---(31)---(8)
95 //
96 //      | | | |          | | | |          | | | |
97 //      (9)---(33)---(36)---(14)          (18)---(44)---(43)---(20)          (25)---(45)---(48)---(30)
98 //      | | | | y          | | | | x          | | | | y
99 //      (10)---(34)---(35)---(13)          (17)---(41)---(42)---(19)          (26)---(46)---(47)---(29)
100 //      | | | |          | | | |          | | | |
101 //      (2)---(11)---(12)---(3)          (1)---(9)---(10)---(2)          (6)---(27)---(28)---(7)
102 //
103 //      x          x
104 //
105 //      face 2          face 5          face 6
106 //      (5)---(32)---(31)---(8)          (6)---(27)---(28)---(7)          (7)---(29)---(30)---(8)
107 //      | | | |          | | | |          | | | |
108 //      (18)---(38)---(39)---(24)          (20)---(52)---(51)---(22)          (22)---(56)---(55)---(24)
109 //      | | | | y          | | | | y x |          | | | |
110 //      (17)---(37)---(40)---(23)          (19)---(49)---(50)---(21)          (21)---(53)---(54)---(23)
111 //      | | | |          | | | |          | | | |
112 //      (1)---(16)---(15)---(4)          (2)---(11)---(12)---(3)          (3)---(13)---(14)---(4)
113 //
114 //
115 //
116 //
117 // Points d'integration 8, 27, 64 : par exemple pour 8 pti:
118 // a=1/racine(3)
119 // Pt1 (a,a,a) ; Pt2 (a,a,-a) ; Pt3 (a,-a,a) ; Pt4 (a,-a,-a)
120 // Pt5 (-a,a,a) ; Pt6 (-a,a,-a) ; Pt7 (-a,-a,a) ; Pt8 (-a,-a,-a)
121 //

```

```

122 // sinon on utilise les points d'integrations calculés à partir du segment
123 // et on a 1,2x2x2, 3x3x3, 4x4x4 etc.
124 //
125 // face 1 : noeud 1 4 3 2 16 15 14 13 12 11 10 9 33 36 35 34, face 2 : noeud 1 5 8 4 17 18 32 31 24 23
15 16 37 38 39 40,
126 // face 3 : noeud 1 2 6 5 9 10 19 20 26 25 18 17 41 42 43 44, face 4 : noeud 5 6 7 8 25 26 27 28 29 30
31 32 45 46 47 48,
127 // face 5 : noeud 2 3 7 6 11 12 21 22 28 27 20 19 49 50 51 52, face 6 : noeud 3 4 8 7 13 14 23 24 30 29
22 21 53 54 55 56,
128 // les normales sortent des faces des elements
129 //
130 // pour les aretes, 12 aretes
131 // 1 9 10 2 2 11 12 3 3 13 14 4 4 15 16 1
132 // 1 17 18 5 2 19 20 6 3 21 22 7 4 23 24 8
133 // 5 25 26 6 6 27 28 7 7 29 30 8 8 31 32 5
134 //
135 //
136 //
137 // concernant la triangulation de chaque face elle est réalisée à l'aide
138 // de la triangulation implantée sur l'élément de référence de la face
139 //
140 //
141 // *****
142 */
143
144 // dans le cas où l'on sort des points d'integrations par défaut on se sert
145 // d'une combinaison de segment pour recréer l'hexaèdre ce qui permet
146 // d'avoir 1x1x1, ou 2x2x2, ou 3x3x3, ou 4x4x4 etc. pt d'integ
147
148 // dans le cas où on utilise 27 pti, la numérotation est la suivante
149 // ( ici on ne représente pas le contour de l'élément)
150 //
151 //
152 //
153 //
154 //
155 //
156 //
157 //
158 //
159 //
160 //
161 //
162 //
163 //
164 //
165 //
166 //
167 //
168 //
169 //
170 //
171 // dans le cas où on utilise 64 pti, la numérotation suit la même logique
172 // on va indiquer les numéros par couche
173 //
174 // couche 1) 27pti
175 //
176 //
177 //
178 //
179 //
180 //
181 //
182 //
183 //
184 // couche 2) 27pti
185 //
186 //
187 //
188 //
189 //
190 //
191 //
192 //
193 //
194 // couche 3) 27pti
195 //
196 //
197 //
198 //
199 //
200 //
201 //
202 //
203 //
204 // couche 4)
205 //

```



```

206 // | (61) (62) (63) (64) |
207 // | | | | |
208 // | (57) (58) (59) (60) |
209 // | | | | | --> xi
210 // | (53) (54) (55) (56) |
211 // | | | | |
212 // | (49) (50) (51) (52) |
213 // *-----*
214
215 // pour ne pas surcharger la figure, on indique les pti de la base
216 // puis uniquement sur les arêtes
217 // à noter qu'ici on n'indique pas le cube d'interpolation des noeuds
218 // qui englobe les pti !
219 // |zeta
220 //
221 // 49-----53-----|57-----61
222 // | \ | | | | \ |
223 // |50 | | | | | |62
224 // |33 | \ | | | | \ |
225 // | | 51 | | | | | |63
226 // | | | | | | | |
227 // | | 52 |-----56-----|60-----64
228 // |17 | | | | | 29 |
229 // | | | | | | | |
230 // | | 36 |-----48-----|eta
231 // |1 |-----5-----9-----|13 17 |
232 // | \ | | | | | \ |
233 // | 2 | 6 10 | | 14 32
234 // | \ | 20 | | | | | \ |
235 // | 3 | 7 11 | | 15 |
236 // | \ | | | | | \ |
237 // | 4 | 8 12 |-----16
238 // | | | | |
239 // |xi
240 //
241
242 /// @addtogroup Les_Elements_de_geometrie
243 /// @{
244 ///
245
246 class GeomHexaCubique : public GeomHexaCom
247 {
248 public :
249 // CONSTRUCTEURS :
250 // il y a 8 points d'integration par défaut et 27 noeuds
251 GeomHexaCubique(int nbi = 8);
252 // de copie
253 GeomHexaCubique(const GeomHexaCubique& a);
254 // DESTRUCTEUR :
255 ~GeomHexaCubique();
256
257 // création d'éléments identiques : cette fonction est analogue à la fonction new
258 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
259 // dérivée
260 // pt est le pointeur qui est affecté par la fonction
261 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
262
263 //----- cas de coordonnees locales quelconques -----
264 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
265 const Vecteur& Phi(const Coordonnee& M);
266 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
267 const Mat_pleine& Dphi(const Coordonnee& M);
268
269 protected :
270
271 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
272 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
273 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
274
275 // METHODES PROTEGEES :
276 inline double& DPHI(int i,int j,int k) { return tabDPhi(k)(i,j);};
277 inline double& PHI(int i,int j) {return tabPhi(j)(i); };
278 // because les routine de calcul de phi et dphi aux pt d'integ sont trop grandes
279 // on en fait des routines
280 void Phiphi();
281 void DphiDphi();
282 // constitution du tableau Extrapol
283 void Calcul_extrapol(int nbi);
284
285 };
286 /// @} // end of group
287
288 #endif

```

## 7.105 GeomHexalin.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           23/01/97
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *****/
38 *   BUT: Définir La geometrie de l'hexaedre lineaire.
39 *   Fonction d'interpolation, points d'integration etc
40 *
41 *   *****
42 *   VERIFICATION:
43 *
44 *   ! date !  auteur !          but
45 *   -----
46 *   !           !          !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !  auteur !          but
51 *   -----
52 *
53 *****/
54 #ifndef GEOMHEXALIN_H
55 #define GEOMHEXALIN_H
56
57 #include "GeomHexaCom.h"
58
59 /*
60 // *****
61 //
62 //   ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
63 //
64 //   Nicolas COUTY   04/12/95
65 //   Source : Dhatt et Touzot p 133, 134, 293
66 //           + donnee suivante des valeurs des fonctions
67 //           d'interpolation aux points d'integration
68 //           (qui a permis la verification)
69 // -----
70 //
71 //
72 //
73 //
74 //
75 //
76 //
77 //
78 //
79 //
80 //
81 //
82 //
83 //
84 //

```

```

85 //
86 //
87 //
88 //
89 //
90 //
91 //
92 //
93 //
94 // Points d'integration
95 // a=1/racine(3)
96 // Pt1 (a,a,a) ; Pt2 (a,a,-a) ; Pt3 (a,-a,a) ; Pt4 (a,-a,-a)
97 // Pt5 (-a,a,a) ; Pt6 (-a,a,-a) ; Pt7 (-a,-a,a) ; Pt8 (-a,-a,-a)
98 //
99 //
100 // face 1 : noeud 1 4 3 2, face 2 : noeud 1 5 8 4,
101 // face 3 : noeud 1 2 6 5, face 4 : noeud 5 6 7 8,
102 // face 5 : noeud 2 3 7 6, face 6 : noeud 3 4 8 7,
103 // les normales sortent des faces des elements
104 // on attribue 4 points d'integration par face
105 //
106 // pour les aretes on suis le fichier Elmail, 12 aretes
107 // 1 2 2 3 3 4 4 1
108 // 1 5 2 6 3 7 4 8
109 // 5 6 6 7 7 8 8 5
110 //
111 // on attribue 1 point d'integration par arete
112 //
113 // concernant la triangulation de chaque face elle est réalisée à l'aide
114 // de la triangulation implantée sur l'élément de référence de la face
115 //
116 // *****
117 //
118 // dans le cas où on utilise 27 pti, la numérotation est la suivante
119 // ( ici on ne représente pas le contour de l'élément)
120 //
121 //
122 //
123 //
124 //
125 //
126 //
127 //
128 //
129 //
130 //
131 //
132 //
133 //
134 //
135 //
136 //
137 //
138 //
139 //
140 //
141 // dans le cas où on utilise 64 pti, la numérotation suit la même logique
142 // on va indiquer les numéros par couche
143 //
144 //
145 //
146 //
147 //
148 //
149 //
150 //
151 //
152 //
153 //
154 //
155 //
156 //
157 //
158 //
159 //
160 //
161 //
162 //
163 //
164 //
165 //
166 //
167 //
168 //
169 //
170 //
171 //

```



```

172 // | (33) (34) (35) (36) |
173 // *-----*
174 //   couche 4)           64 pti
175 // *-----*
176 // | (61) (62) (63) (64) |
177 // | | | | |
178 // | (57) (58) (59) (60) |
179 // | | | | | --> xi
180 // | (53) (54) (55) (56) |
181 // | | | | |
182 // | (49) (50) (51) (52) |
183 // *-----*
184
185 // pour ne pas surcharger la figure, on indique les pti de la base
186 // puis uniquement sur les arêtes
187 // à noter qu'ici on n'indique pas le cube d'interpolation des noeuds
188 // qui englobe les pti !
189 // |zeta
190 // |
191 // 49-----53-----|57-----61
192 // | \ | | | | | | | | |
193 // | 150 \ | | | | | | | | |
194 // | 33 \ | | | | | | | | |
195 // | | 51 | | | | | | | | |
196 // | | | | | | | | | |
197 // | | 52 \ | | | | | | | | |
198 // | | 17 | | | | | | | | |
199 // | | | | | | | | | |
200 // | | | | | | | | | |
201 // | 1 | | | | | | | | |
202 // | 2 | | | | | | | | |
203 // | | 20 | | | | | | | | |
204 // | | | | | | | | | |
205 // | | 3 | | | | | | | | |
206 // | | | | | | | | | |
207 // | | 4 | | | | | | | | |
208 // | | | | | | | | | |
209 // | | | | | | | | | |
210 // | | | | | | | | | |
211 // | | | | | | | | | |
212 // | | | | | | | | | |
213 // | | | | | | | | | |
214 /// @addtogroup Les_Elements_de_geometrie
215 /// @{
216 ///
217
218 class GeomHexalin : public GeomHexaCom
219 {
220 public :
221 // CONSTRUCTEURS :
222 // il y a 8 points d'integration par défaut et 8 noeuds
223 GeomHexalin(int nbi = 8);
224 // de copie
225 GeomHexalin(const GeomHexalin& a);
226 // DESTRUCTEUR :
227 ~GeomHexalin();
228
229 // création d'éléments identiques : cette fonction est analogue à la fonction new
230 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
231 // dérivée
232 // pt est le pointeur qui est affecté par la fonction
233 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
234
235 //----- cas de coordonnees locales quelconques -----
236 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
237 const Vecteur& Phi(const Coordonnee& M);
238 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
239 const Mat_pleine& Dphi(const Coordonnee& M);
240
241 protected :
242
243 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
244 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
245 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
246
247 // METHODES PROTEGEES :
248 inline double& DPHI(int i,int j,int k) { return tabDPhi(k)(i,j);};
249 inline double& PHI(int i,int j) {return tabPhi(j)(i); };
250 // because les routine de calcul de phi et dphi aux pt d'integ sont trop grandes
251 // on en fait des routines
252 void Phiphi();
253 void DphiDphi();
254 // constitution du tableau Extrapol
255 void Calcul_extrapol(int nbi);
256
257 };
258 /// @} // end of group

```

```

259
260 #endif

```

## 7.106 GeomHexaQuad.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          23/01/97
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *
38 *      BUT:   Définir La geometrie de l'hexaedre quadratique incomplet.
39 *             Fonction d'interpolation, points d'integration etc
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date !   auteur !           but
45 *      -----
46 *      !           !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      -----
52 *
53 *****/
54 #ifndef GEOMHEXAQUAD_H
55 #define GEOMHEXAQUAD_H
56
57 #include "GeomHexaCom.h"
58
59 /*
60 //-----*
61 //
62 //      ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
63 //
64 //      Nicolas COUTY   04/12/95
65 //      Source : Pour l'element : Modulef Guide No 2, p 139
66 //      Source : Pour les fonctions d'interpolation :
67 //              Dhatt et Touzot p 135
68 //              + donnee suivante des valeurs des fonctions
69 //              d'interpolation aux points d'integration
70 //              (qui a permis la verification)
71 //-----*
72 //
73 //
74 //
75 //
76 //      5-----20-----8
77 //      | \      |      | \
78 //      |  \     |      |  \
79 //      |   \    |      |   \

```

```

80 //
81 //      |      \      |      |      \
82 //      13      6      18      16      7
83 //
84 //      |      |      |      |      |
85 //      |      |      |      |      |-----eta
86 //      1-----12-----4-----15
87 //      |      |      |      |      |
88 //      |      |      |      |      |
89 //      9      |      |      |      |
90 //      |      |      |      |      |
91 //      |      |      |      |      |
92 //      2-----10-----13
93 //
94 //      |      \      |
95 //      |      \      |
96 // Points d'integration 8 par défaut
97 // a=1/racine(3)
98 // Pt1 (a,a,a) ; Pt2 (a,a,-a) ; Pt3 (a,-a,a) ; Pt4 (a,-a,-a)
99 // Pt5 (-a,a,a) ; Pt6 (-a,a,-a) ; Pt7 (-a,-a,a) ; Pt8 (-a,-a,-a)
100 //
101 // sinon on utilise les points d'integrations calculés à partir du segment
102 // et on a 1,2x2x2, 3x3x3, 4x4x4 etc.
103 //
104 // face 1 : noeud 1 4 3 2 12 11 10 9, face 2 : noeud 1 5 8 4 13 20 16 12,
105 // face 3 : noeud 1 2 6 5 9 14 17 13, face 4 : noeud 5 6 7 8 17 18 19 20,
106 // face 5 : noeud 2 3 7 6 10 15 18 14, face 6 : noeud 3 4 8 7 11 16 19 15,
107 // les normales sortent des faces des elements
108 // on attribue 4 points d'integration par face
109 //
110 // pour les aretes on suis le fichier Elmail, 12 aretes
111 // 1 9 2 2 10 3 3 11 4 4 12 1
112 // 1 13 5 2 14 6 3 15 7 4 16 8
113 // 5 17 6 6 18 7 7 19 8 8 20 5
114 //
115 //
116 // on attribue 2 point d'integration par arete par défaut
117 //
118 // concernant la triangulation de chaque face elle est réalisée à l'aide
119 // de la triangulation implantée sur l'élément de référence de la face
120 //
121 //
122 // *****
123 */
124 // dans le cas où on utilise 27 pti, la numérotation est la suivante
125 // ( ici on ne représente pas le contour de l'élément)
126 //
127 //      |
128 //      |zeta
129 //      |
130 //      |
131 //      |
132 //      |
133 //      |
134 //      |
135 //      |
136 //      |
137 //      |-----eta
138 //      |
139 //      |
140 //      |
141 //      |
142 //      |
143 //      |
144 //      |
145 //      |
146 //      |
147 //      |
148 //      |
149 //      |
150 //      |
151 //      |
152 //      |
153 //      |
154 //      |
155 //      |
156 //      |
157 //      |
158 //      |
159 //      |
160 //      |
161 //      |
162 //      |
163 //      |
164 //      |
165 //      |
166 //      |
167 //      |
168 //      |
169 //      |
170 //      |
171 //      |
172 //      |
173 //      |
174 //      |
175 //      |
176 //      |
177 //      |
178 //      |
179 //      |
180 //      |
181 //      |
182 //      |
183 //      |
184 //      |
185 //      |
186 //      |
187 //      |
188 //      |
189 //      |
190 //      |
191 //      |
192 //      |
193 //      |
194 //      |
195 //      |
196 //      |
197 //      |
198 //      |
199 //      |
200 //      |
201 //      |
202 //      |
203 //      |
204 //      |
205 //      |
206 //      |
207 //      |
208 //      |
209 //      |
210 //      |
211 //      |
212 //      |
213 //      |
214 //      |
215 //      |
216 //      |
217 //      |
218 //      |
219 //      |
220 //      |
221 //      |
222 //      |
223 //      |
224 //      |
225 //      |
226 //      |
227 //      |
228 //      |
229 //      |
230 //      |
231 //      |
232 //      |
233 //      |
234 //      |
235 //      |
236 //      |
237 //      |
238 //      |
239 //      |
240 //      |
241 //      |
242 //      |
243 //      |
244 //      |
245 //      |
246 //      |
247 //      |
248 //      |
249 //      |
250 //      |
251 //      |
252 //      |
253 //      |
254 //      |
255 //      |
256 //      |
257 //      |
258 //      |
259 //      |
260 //      |
261 //      |
262 //      |
263 //      |
264 //      |
265 //      |
266 //      |
267 //      |
268 //      |
269 //      |
270 //      |
271 //      |
272 //      |
273 //      |
274 //      |
275 //      |
276 //      |
277 //      |
278 //      |
279 //      |
280 //      |
281 //      |
282 //      |
283 //      |
284 //      |
285 //      |
286 //      |
287 //      |
288 //      |
289 //      |
290 //      |
291 //      |
292 //      |
293 //      |
294 //      |
295 //      |
296 //      |
297 //      |
298 //      |
299 //      |
300 //      |
301 //      |
302 //      |
303 //      |
304 //      |
305 //      |
306 //      |
307 //      |
308 //      |
309 //      |
310 //      |
311 //      |
312 //      |
313 //      |
314 //      |
315 //      |
316 //      |
317 //      |
318 //      |
319 //      |
320 //      |
321 //      |
322 //      |
323 //      |
324 //      |
325 //      |
326 //      |
327 //      |
328 //      |
329 //      |
330 //      |
331 //      |
332 //      |
333 //      |
334 //      |
335 //      |
336 //      |
337 //      |
338 //      |
339 //      |
340 //      |
341 //      |
342 //      |
343 //      |
344 //      |
345 //      |
346 //      |
347 //      |
348 //      |
349 //      |
350 //      |
351 //      |
352 //      |
353 //      |
354 //      |
355 //      |
356 //      |
357 //      |
358 //      |
359 //      |
360 //      |
361 //      |
362 //      |
363 //      |
364 //      |
365 //      |
366 //      |
367 //      |
368 //      |
369 //      |
370 //      |
371 //      |
372 //      |
373 //      |
374 //      |
375 //      |
376 //      |
377 //      |
378 //      |
379 //      |
380 //      |
381 //      |
382 //      |
383 //      |
384 //      |
385 //      |
386 //      |
387 //      |
388 //      |
389 //      |
390 //      |
391 //      |
392 //      |
393 //      |
394 //      |
395 //      |
396 //      |
397 //      |
398 //      |
399 //      |
400 //      |
401 //      |
402 //      |
403 //      |
404 //      |
405 //      |
406 //      |
407 //      |
408 //      |
409 //      |
410 //      |
411 //      |
412 //      |
413 //      |
414 //      |
415 //      |
416 //      |
417 //      |
418 //      |
419 //      |
420 //      |
421 //      |
422 //      |
423 //      |
424 //      |
425 //      |
426 //      |
427 //      |
428 //      |
429 //      |
430 //      |
431 //      |
432 //      |
433 //      |
434 //      |
435 //      |
436 //      |
437 //      |
438 //      |
439 //      |
440 //      |
441 //      |
442 //      |
443 //      |
444 //      |
445 //      |
446 //      |
447 //      |
448 //      |
449 //      |
450 //      |
451 //      |
452 //      |
453 //      |
454 //      |
455 //      |
456 //      |
457 //      |
458 //      |
459 //      |
460 //      |
461 //      |
462 //      |
463 //      |
464 //      |
465 //      |
466 //      |
467 //      |
468 //      |
469 //      |
470 //      |
471 //      |
472 //      |
473 //      |
474 //      |
475 //      |
476 //      |
477 //      |
478 //      |
479 //      |
480 //      |
481 //      |
482 //      |
483 //      |
484 //      |
485 //      |
486 //      |
487 //      |
488 //      |
489 //      |
490 //      |
491 //      |
492 //      |
493 //      |
494 //      |
495 //      |
496 //      |
497 //      |
498 //      |
499 //      |
500 //      |
501 //      |
502 //      |
503 //      |
504 //      |
505 //      |
506 //      |
507 //      |
508 //      |
509 //      |
510 //      |
511 //      |
512 //      |
513 //      |
514 //      |
515 //      |
516 //      |
517 //      |
518 //      |
519 //      |
520 //      |
521 //      |
522 //      |
523 //      |
524 //      |
525 //      |
526 //      |
527 //      |
528 //      |
529 //      |
530 //      |
531 //      |
532 //      |
533 //      |
534 //      |
535 //      |
536 //      |
537 //      |
538 //      |
539 //      |
540 //      |
541 //      |
542 //      |
543 //      |
544 //      |
545 //      |
546 //      |
547 //      |
548 //      |
549 //      |
550 //      |
551 //      |
552 //      |
553 //      |
554 //      |
555 //      |
556 //      |
557 //      |
558 //      |
559 //      |
560 //      |
561 //      |
562 //      |
563 //      |
564 //      |
565 //      |
566 //      |
567 //      |
568 //      |
569 //      |
570 //      |
571 //      |
572 //      |
573 //      |
574 //      |
575 //      |
576 //      |
577 //      |
578 //      |
579 //      |
580 //      |
581 //      |
582 //      |
583 //      |
584 //      |
585 //      |
586 //      |
587 //      |
588 //      |
589 //      |
590 //      |
591 //      |
592 //      |
593 //      |
594 //      |
595 //      |
596 //      |
597 //      |
598 //      |
599 //      |
600 //      |
601 //      |
602 //      |
603 //      |
604 //      |
605 //      |
606 //      |
607 //      |
608 //      |
609 //      |
610 //      |
611 //      |
612 //      |
613 //      |
614 //      |
615 //      |
616 //      |
617 //      |
618 //      |
619 //      |
620 //      |
621 //      |
622 //      |
623 //      |
624 //      |
625 //      |
626 //      |
627 //      |
628 //      |
629 //      |
630 //      |
631 //      |
632 //      |
633 //      |
634 //      |
635 //      |
636 //      |
637 //      |
638 //      |
639 //      |
640 //      |
641 //      |
642 //      |
643 //      |
644 //      |
645 //      |
646 //      |
647 //      |
648 //      |
649 //      |
650 //      |
651 //      |
652 //      |
653 //      |
654 //      |
655 //      |
656 //      |
657 //      |
658 //      |
659 //      |
660 //      |
661 //      |
662 //      |
663 //      |
664 //      |
665 //      |
666 //      |
667 //      |
668 //      |
669 //      |
670 //      |
671 //      |
672 //      |
673 //      |
674 //      |
675 //      |
676 //      |
677 //      |
678 //      |
679 //      |
680 //      |
681 //      |
682 //      |
683 //      |
684 //      |
685 //      |
686 //      |
687 //      |
688 //      |
689 //      |
690 //      |
691 //      |
692 //      |
693 //      |
694 //      |
695 //      |
696 //      |
697 //      |
698 //      |
699 //      |
700 //      |
701 //      |
702 //      |
703 //      |
704 //      |
705 //      |
706 //      |
707 //      |
708 //      |
709 //      |
710 //      |
711 //      |
712 //      |
713 //      |
714 //      |
715 //      |
716 //      |
717 //      |
718 //      |
719 //      |
720 //      |
721 //      |
722 //      |
723 //      |
724 //      |
725 //      |
726 //      |
727 //      |
728 //      |
729 //      |
730 //      |
731 //      |
732 //      |
733 //      |
734 //      |
735 //      |
736 //      |
737 //      |
738 //      |
739 //      |
740 //      |
741 //      |
742 //      |
743 //      |
744 //      |
745 //      |
746 //      |
747 //      |
748 //      |
749 //      |
750 //      |
751 //      |
752 //      |
753 //      |
754 //      |
755 //      |
756 //      |
757 //      |
758 //      |
759 //      |
760 //      |
761 //      |
762 //      |
763 //      |
764 //      |
765 //      |
766 //      |
767 //      |
768 //      |
769 //      |
770 //      |
771 //      |
772 //      |
773 //      |
774 //      |
775 //      |
776 //      |
777 //      |
778 //      |
779 //      |
780 //      |
781 //      |
782 //      |
783 //      |
784 //      |
785 //      |
786 //      |
787 //      |
788 //      |
789 //      |
790 //      |
791 //      |
792 //      |
793 //      |
794 //      |
795 //      |
796 //      |
797 //      |
798 //      |
799 //      |
800 //      |
801 //      |
802 //      |
803 //      |
804 //      |
805 //      |
806 //      |
807 //      |
808 //      |
809 //      |
810 //      |
811 //      |
812 //      |
813 //      |
814 //      |
815 //      |
816 //      |
817 //      |
818 //      |
819 //      |
820 //      |
821 //      |
822 //      |
823 //      |
824 //      |
825 //      |
826 //      |
827 //      |
828 //      |
829 //      |
830 //      |
831 //      |
832 //      |
833 //      |
834 //      |
835 //      |
836 //      |
837 //      |
838 //      |
839 //      |
840 //      |
841 //      |
842 //      |
843 //      |
844 //      |
845 //      |
846 //      |
847 //      |
848 //      |
849 //      |
850 //      |
851 //      |
852 //      |
853 //      |
854 //      |
855 //      |
856 //      |
857 //      |
858 //      |
859 //      |
860 //      |
861 //      |
862 //      |
863 //      |
864 //      |
865 //      |
866 //      |
867 //      |
868 //      |
869 //      |
870 //      |
871 //      |
872 //      |
873 //      |
874 //      |
875 //      |
876 //      |
877 //      |
878 //      |
879 //      |
880 //      |
881 //      |
882 //      |
883 //      |
884 //      |
885 //      |
886 //      |
887 //      |
888 //      |
889 //      |
890 //      |
891 //      |
892 //      |
893 //      |
894 //      |
895 //      |
896 //      |
897 //      |
898 //      |
899 //      |
900 //      |
901 //      |
902 //      |
903 //      |
904 //      |
905 //      |
906 //      |
907 //      |
908 //      |
909 //      |
910 //      |
911 //      |
912 //      |
913 //      |
914 //      |
915 //      |
916 //      |
917 //      |
918 //      |
919 //      |
920 //      |
921 //      |
922 //      |
923 //      |
924 //      |
925 //      |
926 //      |
927 //      |
928 //      |
929 //      |
930 //      |
931 //      |
932 //      |
933 //      |
934 //      |
935 //      |
936 //      |
937 //      |
938 //      |
939 //      |
940 //      |
941 //      |
942 //      |
943 //      |
944 //      |
945 //      |
946 //      |
947 //      |
948 //      |
949 //      |
950 //      |
951 //      |
952 //      |
953 //      |
954 //      |
955 //      |
956 //      |
957 //      |
958 //      |
959 //      |
960 //      |
961 //      |
962 //      |
963 //      |
964 //      |
965 //      |
966 //      |
967 //      |
968 //      |
969 //      |
970 //      |
971 //      |
972 //      |
973 //      |
974 //      |
975 //      |
976 //      |
977 //      |
978 //      |
979 //      |
980 //      |
981 //      |
982 //      |
983 //      |
984 //      |
985 //      |
986 //      |
987 //      |
988 //      |
989 //      |
990 //      |
991 //      |
992 //      |
993 //      |
994 //      |
995 //      |
996 //      |
997 //      |
998 //      |
999 //      |
1000 //      |

```

```

167 // *-----* | | | |
168 // | (17) (18) (19) (20) |
169 // *-----*
170 // couche 3) 27pti | | | | 64 pti
171 // *-----* | | | | *-----*
172 // | (25) (26) (27) | | (45) (46) (47) (48) |
173 // | | | | | | | |
174 // | (22) (23) (24) | --> xi | (41) (42) (43) (44) |
175 // | | | | | | | | --> xi
176 // | (19) (20) (21) | | (37) (38) (39) (40) |
177 // *-----* | | | | *-----*
178 // | (33) (34) (35) (36) |
179 // *-----*
180 // couche 4) | | | | 64 pti
181 // *-----* | | | | *-----*
182 // | (61) (62) (63) (64) |
183 // | | | | | | | |
184 // | (57) (58) (59) (60) |
185 // | | | | | | | | --> xi
186 // | (53) (54) (55) (56) |
187 // | | | | | | | |
188 // | (49) (50) (51) (52) |
189 // *-----*
190
191 // pour ne pas surcharger la figure, on indique les pti de la base
192 // puis uniquement sur les arêtes
193 // à noter qu'ici on n'indique pas le cube d'interpolation des noeuds
194 // qui englobe les pti !
195 // |zeta
196 //
197 // 49-----53-----57-----61
198 // | \ | \ | \ | \
199 // |50 \ | \ | \ | \ 62
200 // |33 \ | \ | \ | \ 45 \ 63
201 // | | \ | \ | \ | \
202 // | | \ | \ | \ | \
203 // | | \ 52 \ 56 \ 60 \ 64
204 // | | \ | \ | \ | \ 29
205 // | | \ | \ | \ | \
206 // | | \ 36 \ | \ | \ | \ 48-----eta
207 // | | \ 1 \ 5 \ 9 \ 13 \ 17
208 // | | \ 2 \ 6 \ 10 \ 14 \ 32
209 // | | \ 3 \ 7 \ 11 \ 15
210 // | | \ 4 \ 8 \ 12 \ 16
211 // | | \ | \ | \ | \
212 // | | \ | \ | \ | \
213 // | | \ 4 \ 8 \ 12 \ 16
214 // | | \ | \ | \ | \
215 // | | \ | \ | \ | \ xi
216 //
217
218 /// @addtogroup Les_Elements_de_geometrie
219 /// @{
220 ///
221
222
223 class GeomHexaQuad : public GeomHexaCom
224 {
225 public :
226 // CONSTRUCTEURS :
227 // il y a 8 points d'integration par défaut et 20 noeuds
228 GeomHexaQuad(int nbi = 8);
229 // de copie
230 GeomHexaQuad(const GeomHexaQuad& a);
231 // DESTRUCTEUR :
232 ~GeomHexaQuad();
233
234 // création d'éléments identiques : cette fonction est analogue à la fonction new
235 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
236 // dérivée
237 // pt est le pointeur qui est affecté par la fonction
238 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
239
240 //----- cas de coordonnees locales quelconques -----
241 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
242 const Vecteur& Phi(const Coordonnee& M);
243 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
244 const Mat_pleine& Dphi(const Coordonnee& M);
245
246 protected :
247
248 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
249 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
250 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
251
252 // METHODES PROTEGEES :
253 inline double& DPHI(int i,int j,int k) { return tabDPhi(k)(i,j);};

```

```

254     inline double& PHI(int i,int j) {return tabPhi(j)(i); };
255     // because les routine de calcul de phi et dphi aux pt d'integ sont trop grandes
256     // on en fait des routines
257     void Phiphi();
258     void DphiDphi();
259     // constitution du tableau Extrapol
260     void Calcul_extrapol(int nbi);
261
262 };
263 /// @} // end of group
264
265 #endif

```

## 7.107 GeomHexaQuadComp.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           17/03/2003
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *   *****
38 *   BUT:  Définir La geometrie de l'hexaedre quadratique complet.
39 *   Fonction d'interpolation, points d'integration etc
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date ! auteur ! but
45 *   -----
46 *   ! ! !
47 *   $
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date ! auteur ! but
51 *   -----
52 *   $
53 *   *****/
54 #ifndef GEOMHEXAQUADCOMP_H
55 #define GEOMHEXAQUADCOMP_H
56
57 #include "GeomHexaCom.h"
58
59 // l'élément quadratique complet
60
61
62 /*
63 //*****
64 //
65 //   ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
66 //
67 //   construction à partir de l'éléments quadratique incomplet
68 //
69 //-----

```

```

70 //
71 //
72 //
73 //
74 //
75 //
76 //
77 //
78 //
79 //
80 //
81 //
82 //
83 //
84 //
85 //
86 //
87 //
88 //
89 //
90 //
91 //
92 //
93 // par rapport au quadratique incomplet, 21 est au centre de la face 1,
94 // 22 sur la face 3, 23 sur la face 5, 24 sur la face 6
95 // 25 sur la face 2, 26 sur la face 4, 27 au centre de l'élément
96 // Points d'integration 8 par défaut
97 // a=1/racine(3)
98 // Pt1 (a,a,a) ; Pt2 (a,a,-a) ; Pt3 (a,-a,a) ; Pt4 (a,-a,-a)
99 // Pt5 (-a,a,a) ; Pt6 (-a,a,-a) ; Pt7 (-a,-a,a) ; Pt8 (-a,-a,-a)
100 //
101 // sinon on utilise les points d'intégrations calculés à partir du segment
102 // et on a 1,2x2x2, 3x3x3, 4x4x4 etc.
103 //
104 // face 1 : noeud 1 4 3 2 12 11 10 9 21, face 2 : noeud 1 5 8 4 13 20 16 12 25,
105 // face 3 : noeud 1 2 6 5 9 14 17 13 22, face 4 : noeud 5 6 7 8 17 18 19 20 26,
106 // face 5 : noeud 2 3 7 6 10 15 18 14 23, face 6 : noeud 3 4 8 7 11 16 19 15 24,
107 // les normales sortent des faces des elements
108 // on attribue 4 points d'integration par face
109 //
110 // pour les aretes on suis le fichier Elmail, 12 aretes
111 // 1 9 2 2 10 3 3 11 4 4 12 1
112 // 1 13 5 2 14 6 3 15 7 4 16 8
113 // 5 17 6 6 18 7 7 19 8 8 20 5
114 //
115 //
116 // on attribue 2 point d'integration par arete par défaut
117 //
118 // concernant la triangulation de chaque face elle est réalisée à l'aide
119 // de la triangulation implantée sur l'élément de référence de la face
120 //
121 //
122 // *****
123 */
124
125 // dans le cas où l'on sort des points d'intégrations par défaut on se sert
126 // d'une combinaison de segment pour recréer l'hexaèdre ce qui permet
127 // d'avoir 1x1x1, ou 2x2x2, ou 3x3x3, ou 4x4x4 etc. pt d'integ
128
129 // dans le cas où on utilise 27 pti, la numérotation est la suivante
130 // ( ici on ne représente pas le contour de l'élément)
131 //
132 //
133 //
134 //
135 //
136 //
137 //
138 //
139 //
140 //
141 //
142 //
143 //
144 //
145 //
146 //
147 //
148 //
149 //
150 //
151 //
152 // dans le cas où on utilise 64 pti, la numérotation suit la même logique
153 // on va indiquer les numéros par couche
154 //
155 // couche 1) 27pti
156 //

```



```

          |zeta
          |
5-----20-----|-----8
 | \          |          | \
 |  \17       |          |  \19
 |   \        |          |   \
13----6-----|-----18----7
 |  \          |          |  \
 |   \22       |          |   \24-----eta
1-----12-----|-----4-----15
 |  \          |          |  \
 |   \9        |          |   \11
 |    \        |          |    \
2-----10-----|-----13
          |          |
          |xi

```

```

          |zeta
          |
19-----22-----|-----25
 | \          |          | \
 |  \20       |          |  \26
 |   \        |          |   \
10----13-----|-----16----17
 |  \          |          |  \
 |   \21       |          |   \24-----eta
1-----4-----|-----7-----18
 |  \          |          |  \
 |   \12       |          |   \8
 |    \        |          |    \
2-----5-----|-----9
          |          |
          |xi

```

```

157 // | (7) (8) (9) | | (13) (14) (15) (16) |
158 // | | | | | | |
159 // | (4) (5) (6) | --> xi | (9) (10) (11) (12) |
160 // | | | | | | | --> xi
161 // | (1) (2) (3) | | (5) (6) (7) (8) |
162 // *-----* | | | | |
163 // | (1) (2) (3) (4) |
164 // *-----*
165 // couche 2) 27pti | 64 pti
166 // *-----* | *-----*
167 // | (16) (17) (18) | | (29) (30) (31) (32) |
168 // | | | | | | |
169 // | (13) (14) (15) | --> xi | (25) (26) (27) (28) |
170 // | | | | | | | --> xi
171 // | (10) (11) (12) | | (21) (22) (23) (24) |
172 // *-----* | | | | |
173 // | (17) (18) (19) (20) |
174 // *-----*
175 // couche 3) 27pti | 64 pti
176 // *-----* | *-----*
177 // | (25) (26) (27) | | (45) (46) (47) (48) |
178 // | | | | | | |
179 // | (22) (23) (24) | --> xi | (41) (42) (43) (44) |
180 // | | | | | | | --> xi
181 // | (19) (20) (21) | | (37) (38) (39) (40) |
182 // *-----* | | | | |
183 // | (33) (34) (35) (36) |
184 // *-----*
185 // couche 4) | 64 pti
186 // *-----* | *-----*
187 // | (61) (62) (63) (64) |
188 // | | | | | | |
189 // | (57) (58) (59) (60) |
190 // | | | | | | | --> xi
191 // | (53) (54) (55) (56) |
192 // | | | | | | |
193 // | (49) (50) (51) (52) |
194 // *-----*
195
196 // pour ne pas surcharger la figure, on indique les pti de la base
197 // puis uniquement sur les arêtes
198 // à noter qu'ici on n'indique pas le cube d'interpolation des noeuds
199 // qui englobe les pti !
200 // |zeta
201 //
202 // 49-----53-----57-----61
203 // | \ | \ | \ | \
204 // |50 | 51 | 52 | 56 | 60 | 64
205 // |33 | 51 | 52 | 56 | 60 | 64
206 // | | | | | | |
207 // | | | | | | |
208 // | | | | | | |
209 // |17 | | | | | |
210 // | | | | | | |
211 // | | | | | | |
212 // |1 | | | | | |
213 // | | | | | | |
214 // | 2 | 6 | 10 | 14 | 32
215 // | | | | | | |
216 // | 3 | 7 | 11 | 15 |
217 // | | | | | | |
218 // | 4 | | 8 | 12 | 16
219 // | | | | | | |
220 // | | | | | | |
221 // | | | | | | |
222 //
223 /// @addtogroup Les_Elements_de_geometrie
224 /// @{
225 ///
226
227 class GeomHexaQuadComp : public GeomHexaCom
228 {
229 public :
230 // CONSTRUCTEURS :
231 // il y a 8 points d'integration par défaut et 27 noeuds
232 GeomHexaQuadComp(int nbi = 8);
233 // de copie
234 GeomHexaQuadComp(const GeomHexaQuadComp& a);
235 // DESTRUCTEUR :
236 ~GeomHexaQuadComp();
237
238 // création d'élément identiques : cette fonction est analogue à la fonction new
239 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
240 // dérivée
241 // pt est le pointeur qui est affecté par la fonction
242 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
243

```

```

244 //----- cas de coordonnees locales quelconques -----
245 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
246 const Vecteur& Phi(const Coordonnee& M);
247 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
248 const Mat_pleine& Dphi(const Coordonnee& M);
249
250 protected :
251
252 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
253 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
254 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
255
256 // METHODES PROTEGEES :
257 inline double& DPHI(int i,int j,int k) { return tabDPhi(k)(i,j);};
258 inline double& PHI(int i,int j) {return tabPhi(j)(i); };
259 // because les routine de calcul de phi et dphi aux pt d'integ sont trop grandes
260 // on en fait des routines
261 void Phiphi();
262 void DphiDphi();
263 // constitution du tableau Extrapol
264 void Calcul_extrapol(int nbi);
265
266 };
267 /// @} // end of group
268
269 #endif

```

## 7.108 GeomPentaCom.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      19/12/99
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *      *****
38 *      BUT:      Définir Les éléments communs aux géométrie pentaédrique.
39 *      Poids et points d'integration etc
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date ! auteur ! but
45 *      -----
46 *      ! ! !
47 *      $
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date ! auteur ! but
51 *      -----
52 *      $
53 *      *****/
54 #ifndef GEOMPENTACOM_H
55 #define GEOMPENTACOM_H

```



```

56
57 #include"ElemGeomC0.h"
58
59 /*
60 // *****
61 //
62 //     ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
63 //
64 // -----*
65 //
66 //
67 //          |zeta
68 //          |
69 //          4-----6
70 //         // | * |
71 //         // | * |
72 //         // | * |----- eta
73 //         // | * |
74 //         // | * |
75 //         // | * |
76 //         // | * |-----3
77 //         // | * |
78 //         // | * |
79 //         // | * |
80 //         // | * |
81 //         // | * |
82 //         // | * |
83 //         // | * |
84 //         // | * |
85 //         // | * |
86 //         // | * |
87 //
88 //          pentaèdre trilinéaire
89 //
90 //
91 //
92 // Points d'integration (voir triangle et segment)
93 //
94 // cas du trilinéaire -> description des faces , puis des arêtes
95 // face 1 : noeud 1 3 2, face 2 : noeud 1 4 6 3,
96 // face 3 : noeud 1 2 5 4, face 4 : noeud 4 5 6,
97 // face 5 : noeud 2 3 6 5
98 // les normales sortent des faces des elements
99 //
100 // par défaut: on attribue un point d'intégration dans le plan // aux triangles
101 // et 2 points dans l'épaisseur
102 //
103 // pour les aretes on suis le fichier Elmail, 9 aretes
104 // 1-> 1 2 2->2 3 3->3 1
105 // 4-> 1 4 5->2 5 6->3 6
106 // 7-> 4 5 8->5 6 9->6 4
107 //
108 // cas du triquadratique -> description des faces
109 // face 1 : noeud 9 3 8 2 7 1, face 2 : noeud 9 1 10 4 15 6 12 3,
110 // face 3 : noeud 7 2 11 5 13 4 10 1, face 4 : noeud 15 6 14 5 13 4,
111 // face 5 : noeud 8 3 12 6 14 5 11 2,
112 // les normales sortent des faces des elements, puis des arêtes
113 //
114 // pour les aretes on suit le fichier Elmail, 9 aretes
115 // 1->1 7 2 2->2 8 3 3->3 9 1
116 // 4->1 10 4 5->2 11 5 6->3 12 6
117 // 7->4 13 5 8->5 14 6 9->6 15 4
118 //
119 //
120 // on attribue 3 points d'intégration suivant les plans // aux triangles
121 // et 2 points dans l'épaisseur
122 //
123 // *****
124 //
125 //     ELEMENT DE REFERENCE en quadratique , POINTS D'INTEGRATION:
126 //
127 // -----*
128 //
129 //          |zeta
130 //          |
131 //          4---15----6
132 //         // | * |
133 //         // | * |
134 //         // | * |----- eta
135 //         // | * |-----12
136 //         // | * |
137 //         // | * |
138 //         // | * |-----3
139 //         // | * |
140 //         // | * |
141 //         // | * |
142 //         // | * |

```

```

143 //          11  7  8
144 //          // / *
145 //          xi | / *
146 //          | / *
147 //          2
148 //
149 //
150 //
151 //          pentaèdre triquadratique incomplet
152 //
153 //
154 // Points d'integration (voir triangle et segment)
155 //
156 // cas du triquadratique -> description des faces
157 // face 1 : noeud 9 3 8 2 7 1, face 2 : noeud 9 1 10 4 15 6 12 3,
158 // face 3 : noeud 7 2 11 5 13 4 10 1, face 4 : noeud 15 6 14 5 13 4,
159 // face 5 : noeud 8 3 12 6 14 5 11 2,
160 // les normales sortent des faces des elements, puis des arêtes
161 //
162 // pour les aretes on suit le fichier Elmail, 9 aretes
163 //1->1 7 2 2->2 8 3 3->3 9 1
164 //4->1 10 4 5->2 11 5 6->3 12 6
165 //7->4 13 5 8->5 14 6 9->6 15 4
166 //
167 //
168 // on attribue 3 points d'integration suivant les plans // aux triangles
169 // et 2 points dans l'épaisseur -> valeurs par défaut
170 //
171 //
172 // concernant la triangulation de chaque face elle est réalisée à l'aide
173 // de la triangulation implantée sur l'élément de référence de la face
174 //
175 //
176 // *****
177 //
178 //          ELEMENT DE REFERENCE 18 noeuds, POINTS D'INTEGRATION:
179 //
180 // -----*
181 //
182 //          |zeta
183 //          |
184 //          4----15---6
185 //          // | * |
186 //          // | * |
187 //          // | * |
188 //          13 10-14-18---12----- eta
189 //          // / * |
190 //          // * | |
191 //          // * | | 17
192 //          5 16 1-----9-----3
193 //          | // / *
194 //          | // / *
195 //          | // / *
196 //          11  7  8
197 //          // / *
198 //          xi | / *
199 //          | / *
200 //          2
201 //
202 //
203 //
204 //          pentaèdre triquadratique complet
205 //
206 //
207 // Points d'integration (voir triangle et segment)
208 //
209 // cas du triquadratique -> description des faces
210 // face 1 : noeud 1 3 2 9 8 7, face 2 : noeud 1 4 6 3 10 15 12 9 18,
211 // face 3 : noeud 1 2 5 4 7 11 13 10 16, face 4 : noeud 4 5 6 13 14 15,
212 // face 5 : noeud 2 3 6 5 8 12 14 11 17,
213 // les normales sortent des faces des elements, puis des arêtes
214 //
215 // pour les aretes on suit le fichier Elmail, 9 aretes
216 //1->1 7 2 2->2 8 3 3->3 9 1
217 //4->1 10 4 5->2 11 5 6->3 12 6
218 //7->4 13 5 8->5 14 6 9->6 15 4
219 //
220 //
221 // on attribue 3 points d'integration suivant les plans // aux triangles
222 // et 2 points dans l'épaisseur -> valeurs par défaut
223 //
224 //
225 // concernant la triangulation de chaque face elle est réalisée à l'aide
226 // de la triangulation implantée sur l'élément de référence de la face
227 //
228 //
229 // *****

```

```

230 */
231
232 /// @addtogroup Les_Elements_de_geometrie
233 /// @{
234 ///
235
236
237 class GeomPentaCom : public ElemGeomC0
238 {
239 public :
240     // CONSTRUCTEURS :
241     // le constructeur par défaut ne doit pas être utilisé
242     GeomPentaCom();
243
244     // il y a nbi points d'integration et nbn noeuds
245     // l'interpolation est donné par les classes dérivées
246     GeomPentaCom(int nbi , int nbn, Enum_interpol interpol);
247
248     // de copie
249     GeomPentaCom(const GeomPentaCom& a);
250     // DESTRUCTEUR :
251     ~GeomPentaCom();
252
253     //----- cas de coordonnees locales quelconques -----
254     // en fonction de coordonnees locales, retourne true si le point est a l'interieur
255     // de l'element, false sinon
256     bool Interieur(const Coordonnee& M);
257     // en fonction de coordonnees locales, retourne le point local P, maximum interieur à l'élément,
258     // donc sur la frontière
259     // dont les coordonnées sont sur la droite GM: c-a-d GP = alpha GM, avec apha maxi et P appartenant
260     // à la frontière
261     // de l'élément, G étant le centre de gravité, sauf si GM est nul, dans ce cas retour de M
262     Coordonnee Maxi_Coor_dans_directionGM(const Coordonnee& M);
263
264 protected :
265     // METHODES PROTEGEES :
266     // constitution du tableau Extrapol
267     void Calcul_extrapol(int nbi);
268 };
269 /// @} // end of group
270 #endif

```

## 7.109 GeomPentaL.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:          16/11/99
32 *
33 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:        Herezh++
36 *
37 *
38 *   BUT:   Définir Les éléments communs aux géométrie pentaédrique
39 *          linéaires :
40 *          Fonction d'interpolation, points d'integration etc

```

```

41 *                                     $ *
42 *      ***** *
43 *      *
44 *      VERIFICATION: *
45 *      ! date ! auteur ! but ! *
46 *      ----- *
47 *      ! ! ! ! *
48 *      * * * * $ *
49 *      ***** *
50 *      MODIFICATIONS: *
51 *      ! date ! auteur ! but ! *
52 *      ----- *
53 *      * * * * $ *
54 *      *****/
55 #ifndef GEOMPENTAL_H
56 #define GEOMPENTAL_H
57
58 #include"GeomPentaCom.h"
59
60 /*
61 // *****
62 //
63 //      ELEMENT DE REFERENCE , POINTS D'INTEGRATION: *
64 // *
65 // -----*
66 //
67 //
68 //      |zeta
69 //      4-----6
70 //      // | * |
71 //      // | * |
72 //      // | * |
73 //      // |----- eta
74 //      // |
75 //      // | * |
76 //      // | * |
77 //      // |-----3
78 //      // | * |
79 //      // | * |
80 //      // | * |
81 //      // | * |
82 //      // | * |
83 //      xi | * |
84 //      // | * |
85 //      // | * |
86 //      // |
87 //
88 //
89 //      pentaèdre trilinéaire
90 //
91 //
92 //
93 // Points d'integration (voir triangle et segment)
94 //
95 // cas du trilinéaire -> description des faces , puis des arêtes
96 // face 1 : noeud 1 3 2, face 2 : noeud 1 4 6 3,
97 // face 3 : noeud 1 2 5 4, face 4 : noeud 4 5 6,
98 // face 5 : noeud 2 3 6 5
99 // les normales sortent des faces des elements
100 //
101 // on attribue un point d'intégration dans le plan // aux triangles
102 // et 2 points dans l'épaisseur
103 //
104 // pour les aretes on suis le fichier Elmail, 9 aretes
105 // 1-> 1 2 2->2 3 3->3 1
106 // 4-> 1 4 5->2 5 6->3 6
107 // 7-> 4 5 8->5 6 9->6 4
108 //
109 //
110 // concernant la triangulation de chaque face elle est réalisée à l'aide
111 // de la triangulation implantée sur l'élément de référence de la face
112 //
113 //
114 // *****
115 */
116
117 /// @addtogroup Les_Elements_de_geometrie
118 /// @{
119 ///
120
121
122 class GeomPentaL : public GeomPentaCom
123
124 {
125 public :
126 // CONSTRUCTEURS :

```

```

127 // il y a 2 points d'integration par défaut
128 GeomPental(int nbi = 2);
129 // de copie
130 GeomPental(const GeomPental& a);
131 // DESTRUCTEUR :
132 ~GeomPental();
133
134 // création d'éléments identiques : cette fonction est analogue à la fonction new
135 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
136 // dérivée
137 // pt est le pointeur qui est affecté par la fonction
138 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
139
140 //----- cas de coordonnees locales -----
141 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
142 const Vecteur& Phi(const Coordonnee& M);
143 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
144 const Mat_pleine& Dphi(const Coordonnee& M);
145
146 protected :
147
148 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
149 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
150 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
151
152 // METHODES PROTEGEES :
153 // constitution du tableau Extrapol
154 void Calcul_extrapol(int nbi);
155
156 };
157 /// @} // end of group
158
159 #endif

```

## 7.110 GeomPentaQ.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          4/12/99
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *****/
38 *      BUT:  Définir Les éléments communs aux géométrie pentaédrique
39 *            quadratique incomplet :
40 *            Fonction d'interpolation, points d'integration etc
41 *
42 *      *****
43 *
44 *      VERIFICATION:
45 *
46 *      ! date !   auteur !           but
47 *      -----
48 *      !       !           !

```

```

49 *          *****
50 *          MODIFICATIONS:
51 *          ! date ! auteur ! but
52 *          -----
53 *          $
54 *          *****/
55 #ifndef GEOMPENQAQ_H
56 #define GEOMPENQAQ_H
57
58 #include "GeomPentaCom.h"
59
60 /*
61 // *****
62 //
63 //          ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
64 //
65 //          -----
66 //
67 //
68 //          |zeta
69 //          |
70 //          4---15---6
71 //          // | * |
72 //          // | * |
73 //          // | * |
74 //          13 10-14-----12----- eta
75 //          // | * |
76 //          // | * |
77 //          // | * |
78 //          5 // 1---9---3
79 //          // | * |
80 //          // | * |
81 //          11 7 8
82 //          // | * |
83 //          xi | / *
84 //          // *
85 //          2
86 //
87 //
88 //
89 //          pentaèdre triquadratique incomplet
90 //
91 //
92 // Points d'integration (voir triangle et segment)
93 //
94 // cas du triquadratique -> description des faces
95 // face 1 : noeud 1 3 2 9 8 7, face 2 : noeud 1 4 6 3 10 15 12 9,
96 // face 3 : noeud 1 2 5 4 7 11 13 10, face 4 : noeud 4 5 6 13 14 15,
97 // face 5 : noeud 2 3 6 5 8 12 14 11,
98 // les normales sortent des faces des elements, puis des arêtes
99 //
100 // pour les aretes on suis le fichier Elmail, 9 aretes
101 //1->1 7 2 2->2 8 3 3->3 9 1
102 //4->1 10 4 5->2 11 5 6->3 12 6
103 //7->4 13 5 8->5 14 6 9->6 15 4
104 //
105 //
106 // on attribue 3 points d'integration suivant les plans // aux triangles
107 // et 2 points dans l'épaisseur -> valeurs par défaut
108 //
109 //
110 // concernant la triangulation de chaque face elle est réalisée à l'aide
111 // de la triangulation implantée sur l'élément de référence de la face
112 //
113 //
114 // *****
115 */
116
117 /// @addtogroup Les_Elements_de_geometrie
118 /// @{
119 ///
120
121
122 class GeomPentaQ : public GeomPentaCom
123 {
124 {
125 public :
126 // CONSTRUCTEURS :
127 // il y a 2 points d'integration par défaut
128 GeomPentaQ(int nbi = 6);
129 // de copie
130 GeomPentaQ(const GeomPentaQ& a);
131 // DESTRUCTEUR :
132 ~GeomPentaQ();
133
134 // création d'élément identiques : cette fonction est analogue à la fonction new
135 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe

```

```

136 // dérivée
137 // pt est le pointeur qui est affecté par la fonction
138 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
139
140 //----- cas de coordonnees locales -----
141 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
142 const Vecteur& Phi(const Coordonnee& M);
143 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
144 const Mat_pleine& Dphi(const Coordonnee& M);
145
146 protected :
147
148 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
149 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
150 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
151
152 // METHODES PROTEGEES :
153 inline double& DPHI(int i,int j,int k) { return tabDPhi(k)(i,j);};
154 // constitution du tableau Extrapol
155 void Calcul_extrapol(int nbi);
156
157 };
158 /// @} // end of group
159
160 #endif

```

## 7.111 GeomPentaQComp.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          14/03/2003
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *
38 *      BUT:  Définir Les éléments communs aux géométrie pentaédrique
39 *           quadratique complet :
40 *           Fonction d'interpolation, points d'integration etc
41 *
42 *      *****
43 *
44 *      VERIFICATION:
45 *
46 *      ! date !   auteur !           but
47 *      !-----!-----!-----!
48 *      !           !           !           !
49 *      *****
50 *      MODIFICATIONS:
51 *      ! date !   auteur !           but
52 *      !-----!-----!-----!
53 *
54 *      *****/
55 #ifndef GEOMPENTAQCOMP_H
56 #define GEOMPENTAQCOMP_H

```

```

57
58 #include"GeomPentaCom.h"
59
60 /*
61 // *****
62 //
63 //     ELEMENT DE REFERENCE 18 noeuds, POINTS D'INTEGRATION:
64 //
65 // -----*
66 //
67 //
68 //          |zeta
69 //          |
70 //          4----15---6
71 //         // | * |
72 //         // | * |
73 //         // | * |
74 //          13 10-14-18---12---- eta
75 //         // * |
76 //         // * | 17
77 //         // * | 16 1----9----3
78 //         // | *
79 //         // | *
80 //         // | *
81 //         11 7 8
82 //         // | / *
83 //         xi | / *
84 //         // | / *
85 //         2
86 //
87 //
88 //
89 //          pentaèdre triquadratique complet
90 //
91 //
92 // Points d'integration (voir triangle et segment)
93 //
94 // cas du triquadratique -> description des faces
95 // face 1 : noeud 1 3 2 9 8 7, face 2 : noeud 1 4 6 3 10 15 12 9 18,
96 // face 3 : noeud 1 2 5 4 7 11 13 10 16, face 4 : noeud 4 5 6 13 14 15,
97 // face 5 : noeud 2 3 6 5 8 12 14 11 17,
98 // par rapport au quadratique incomplet, le noeud 16 est sur la face 3, 17 sur la face 5
99 // et 18 sur la face 2
100 // les normales sortent des faces des elements, puis des arêtes
101 //
102 // pour les aretes on suis le fichier Elmail, 9 aretes
103 //1->1 7 2 2->2 8 3 3->3 9 1
104 //4->1 10 4 5->2 11 5 6->3 12 6
105 //7->4 13 5 8->5 14 6 9->6 15 4
106 //
107 //
108 // on attribue 3 points d'integration suivant les plans // aux triangles
109 // et 2 points dans l'épaisseur -> valeurs par défaut
110 //
111 //
112 // concernant la triangulation de chaque face elle est réalisée à l'aide
113 // de la triangulation implantée sur l'élément de référence de la face
114 //
115 //
116 // *****
117 // */
118
119 /// @addtogroup Les_Elements_de_geometrie
120 /// @{}
121 ///
122
123
124 class GeomPentaQComp : public GeomPentaCom
125 {
126 public :
127 // CONSTRUCTEURS :
128 // il y a 2 points d'integration par défaut
129 GeomPentaQComp(int nbi = 6);
130 // de copie
131 GeomPentaQComp(const GeomPentaQComp& a);
132 // DESTRUCTEUR :
133 ~GeomPentaQComp();
134
135 // création d'élément identiques : cette fonction est analogue à la fonction new
136 // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
137 // dérivée
138 // pt est le pointeur qui est affecté par la fonction
139 ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
140
141 //----- cas de coordonnees locales quelconques -----
142 // retourne les fonctions d'interpolation au point M (en coordonnees locales)

```



```

144     const Vecteur& Phi(const Coordonnee& M);
145     // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
146     const Mat_pleine& Dphi(const Coordonnee& M);
147
148     protected :
149
150     // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
151     Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
152     Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
153
154     // METHODES PROTEGEES :
155     inline double& DPHI(int i,int j,int k) { return tabDPHI(k)(i,j);};
156     // constitution du tableau Extrapol
157     void Calcul_extrapol(int nbi);
158
159 };
160 /// @} // end of group
161
162 #endif

```

## 7.112 GeomTetra.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:  Définir La geometrie des tétraèdres.
39 *      Fonction d'interpolation, points d'integration etc
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date ! auteur ! but
45 *      -----
46 *      ! ! !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date ! auteur ! but
51 *      -----
52 *
53 *****/
54 #ifndef GEOMETETRA_H
55 #define GEOMETETRA_H
56
57 #include"ElemGeomC0.h"
58
59 // *****
60 //
61 //      ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
62 //

```

```

63 //      Source : Dhatt et Touzot p 130, 131, 132 pour les fonctions      *
64 //      d'interpolation. Pour la numérotation : Modulef                *
65 //      -----*
66 //
67 //
68 //          ^
69 //          |zeta
70 //          4
71 //         / \
72 //        /   \
73 //       /     \
74 //      /       \
75 //     /         \
76 //    /           \
77 //   /             \
78 //  /               \
79 // /               \
80 // /               \
81 // xi      tetraèdre linéaire
82 //          ^
83 //          |zeta
84 //          4
85 //         / \
86 //        /   \
87 //       /     \
88 //      /       \
89 //     /         \
90 //    /           \
91 //   /             \
92 //  /               \
93 // /               \
94 // /               \
95 // xi      tetraèdre quadratique
96 //
97 //
98 // Points d'integration
99 // a=1/racine(3)
100 // Pt1 (a,a,a) ; Pt2 (a,a,-a) ; Pt3 (a,-a,a) ; Pt4 (a,-a,-a)
101 // Pt5 (-a,a,a) ; Pt6 (-a,a,-a) ; Pt7 (-a,-a,a) ; Pt8 (-a,-a,-a)
102 //
103 //-----
104 // pour le tetraèdre linéaire :
105 //-----
106 // face 1 : noeud 1 3 2 , face 2 : noeud 1 4 3,
107 // face 3 : noeud 1 2 4, face 4 : noeud 2 3 4,
108 // les normales sortent des faces des elements
109 // on attribue 1 points d'integration par face
110 //
111 // pour les aretes on suit le fichier Elmail, 6 aretes
112 // A1-> 1 2 A2-> 2 3 A3-> 3 1
113 // A4-> 1 4 A5-> 2 4 A6-> 3 4
114 //
115 // on attribue 1 point d'integration par arete
116 //
117 //-----
118 // pour le tetraèdre quadratique :
119 //-----
120 // face 1 : noeud 7 3 6 2 5 1 , face 2 : noeud 5 2 9 4 8 1,
121 // face 3 : noeud 7 1 8 4 10 3 , face 4 : noeud 6 3 10 4 9 2,
122 // les normales sortent des faces des elements
123 // on attribue 3 points d'integration par face
124 //
125 // pour les aretes on suit le fichier Elmail, 6 aretes
126 // A1-> 1 5 2 A2-> 2 6 3 A3-> 3 7 1
127 // A4-> 1 8 4 A5-> 2 9 4 A6-> 3 10 4
128 //
129 // on attribue 2 point d'integration par arete
130 //
131 //
132 // *****
133
134 /// @addtogroup Les_Elements_de_geometrie
135 /// @{
136 ///
137
138
139 class GeomHexalin : public ElemGeomC0
140 {
141 public :
142 // CONSTRUCTEURS :
143 // il y a 8 points d'integration et 8 noeuds
144 GeomHexalin();
145 // de copie
146 GeomHexalin(const GeomHexalin& a);
147 // DESTRUCTEUR :
148 ~GeomHexalin();
149

```

```

150 //----- cas de coordonnees locales quelconques -----
151 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
152 const Vecteur& Phi(const Coordonnee& M);
153 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
154 const Mat_pleine& Dphi(const Coordonnee& M);
155 // en fonction de coordonnees locales, retourne true si le point est a l'interieur
156 // de l'element, false sinon
157 bool Interieur(const Coordonnee& M);
158
159 protected :
160
161 // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
162 Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
163 Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
164
165 // METHODES PROTEGEES :
166 inline double& DPHI(int i,int j,int k) { return tabDPHI(k)(i,j);};
167 inline double& PHI(int i,int j) {return tabPhi(j)(i); };
168 // because les routine de calcul de phi et dphi aux pt d'integ sont trop grandes
169 // on en fait des routines
170 void Phiphi();
171 void DphiDphi();
172
173 };
174 /// @} // end of group
175
176 #endif

```

## 7.113 GeomTetraCom.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *      *****
38 *      BUT:      Definition des grandeurs communes à tous les tétraèdres.
39 *      Points d'integration
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      -----
47 *      !           !           !
48 *      *****
49 *      MODIFICATIONS:
50 *
51 *      ! date !   auteur !           but
52 *      -----
53 *      !           !           !
54 *      *****/
55 #ifndef GEOMETETRACOM_H

```

```

55 #define GEOMTETRACOM_H
56
57 #include"ElemGeomCO.h"
58
59 /*
60 // *****
61 //
62 //      ELEMENT DE REFERENCE :
63 //
64 //      Source : Dhatt et Touzot p 130, 131, 132 pour les fonctions
65 //      d'interpolation. Pour la numérotation : Modulef
66 // -----
67 //
68 //
69 //      ^
70 //      |zeta
71 //      |
72 //      4
73 //      / \
74 //      |  |
75 //      |  |
76 //      |  |
77 //      |  | 1-----3 - - - -> eta
78 //      / \
79 //      2
80 //
81 // xi      tetraèdre linéaire
82 //
83 //
84 //
85 //      ^
86 //      |zeta
87 //      |
88 //      4
89 //      / \
90 //      |  |
91 //      9  8  10
92 //      |  |
93 //      |  |
94 //      |  | 1- 7- 3 - - - -> eta
95 //      / \ 5'  6'
96 //      2
97 //
98 // xi      tetraèdre quadratique
99 //
100 //
101 //-----
102 // Points d'integration
103 //-----
104 // 1 point : (ordre 1)
105 //      Pt1 (1/4,1/4,1/4)
106 // 4 points : (ordre 2)  a = (5. - sqrt(5))/20., b = (5+3.*sqrt(5))/20.
107 //      Pt1 (a,a,a) ; Pt2 (a,a,b) ; Pt3 (a,b,a) ; Pt4 (b,a,a)
108 //
109 // 5 points : (ordre 3)  a = 1/4, b=1/6, c=1/2,
110 //      Pt1 (a,a,a) ; Pt2 (b,b,b) ; Pt3 (b,b,c) ; Pt4 (b,c,b) ; Pt4 (c,b,b);
111 //
112 // 15 points : (ordre 5)  a = 1/4, b1=(7+sqrt(15))/34, b2=(7-sqrt(15))/34,
113 //      c1=(13+3sqrt(15))/34, c2=(13-3sqrt(15))/34,
114 //      d=(5-sqrt(15))/20, e=(5+sqrt(15))/20,
115 //      Pt1 (a,a,a) ; Pt2 (b1,b1,b1) ; Pt3 (b2,b2,b2) ; Pt4 (b1,b1,c1)
116 //      Pt5 (b2,b2,c2) ; Pt6 (b1,c1,b1) ; Pt7 (b2,c2,b2) ; Pt8 (c1,b1,b1)
117 //      Pt9 (c2,b2,b2) ; Pt10 (d,d,e) ; Pt11 (d,e,d) ; Pt12 (e,d,d)
118 //      Pt13 (d,e,e) ; Pt14 (e,d,e) ; Pt15 (e,e,d) ;
119 //
120 //-----
121 // pour le tetraèdre linéaire :
122 //-----
123 // face 1 : noeud 1 3 2 , face 2 : noeud 1 4 3,
124 // face 3 : noeud 1 2 4, face 4 : noeud 2 3 4,
125 // les normales sortent des faces des elements
126 // on attribue 1 points d'integration par face
127 //
128 // pour les aretes on suit le fichier Elmail, 6 aretes
129 // A1-> 1 2 A2-> 2 3 A3-> 3 1
130 // A4-> 1 4 A5-> 2 4 A6-> 3 4
131 //
132 // on attribue 1 point d'integration par arete
133 //
134 //
135 //-----
136 // pour le tetraèdre quadratique :
137 //-----
138 // face 1 : noeud 7 3 6 2 5 1 , face 2 : noeud 5 2 9 4 8 1,
139 // face 3 : noeud 7 1 8 4 10 3 , face 4 : noeud 6 3 10 4 9 2,
140 // les normales sortent des faces des elements
141 // on attribue 3 points d'integration par face

```

```

142 //
143 // pour les aretes on suit le fichier Elmail, 6 aretes
144 // A1-> 1 5 2  A2-> 2 6 3  A3-> 3 7 1
145 // A4-> 1 8 4  A5-> 2 9 4  A6-> 3 10 4
146 //
147 //   on attribue 2 point d'integration par arete
148 //
149 // concernant la triangulation de chaque face elle est réalisée à l'aide
150 // de la triangulation implantée sur l'élément de référence de la face
151 //
152 //
153 // *****
154 */
155
156 /// @addtogroup Les_Elements_de_geometrie
157 /// @{
158 ///
159
160
161 class GeomTetraCom : public ElemGeomC0
162 {
163 public :
164     // CONSTRUCTEURS :
165     // le constructeur par défaut ne doit pas être utilisé
166     GeomTetraCom();
167     // il y a 1 points d'integration et 4 noeuds par défaut
168     // nbi : le nombre de point d'intégration voulu
169     // interpol indique le type d'interpolation
170     GeomTetraCom(int nbi = 1, int nbe = 4, Enum_interpol interpol = RIEN_INTERPOL);
171     // de copie
172     GeomTetraCom(const GeomTetraCom& a);
173     // DESTRUCTEUR :
174     ~GeomTetraCom();
175
176     //----- cas de coordonnees locales quelconques -----
177     // en fonction de coordonnees locales, retourne true si le point est a l'interieur
178     // de l'element, false sinon
179     bool Interieur(const Coordonnee& M);
180     // en fonction de coordonnees locales, retourne le point local P, maximum interieur à l'élément,
181     // donc sur la frontière
182     // dont les coordonnées sont sur la droite GM: c-a-d GP = alpha GM, avec apha maxi et P appartenant
183     // à la frontière
184     // de l'élément, G étant le centre de gravité, sauf si GM est nul, dans ce cas retour de M
185     Coordonnee Maxi_Coor_dans_directionGM(const Coordonnee& M);
186
187 protected :
188     // METHODES PROTEGEES :
189     // constitution du tableau Extrapol
190     void Calcul_extrapol(int nbi);
191 };
192 /// @} // end of group
193
194 #endif

```

## 7.114 GeomTetraL.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29

```

```

30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:  Définir La geometrie des tétraèdres linéaires.
39 *         Fonction d'interpolation, points d'integration etc
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   !           !           !
47 *   !           !           !
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   !           !           !
52 *   !           !           !
53 *****/
54 #ifndef GEOMTETRAL_H
55 #define GEOMTETRAL_H
56
57 #include "GeomTetraCom.h"
58
59 /*
60 // *****
61 //
62 //   ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
63 //
64 //   Source : Dhatt et Touzot p 130, 131, 132 pour les fonctions
65 //           d'interpolation. Pour la numérotation : Modulef
66 // -----*
67 //
68 //
69 //           ^
70 //           |zeta
71 //           |
72 //           |4
73 //           |
74 //           |
75 //           |
76 //           |
77 //           |1-----3 - - - > eta
78 //           |
79 //           |
80 //           |
81 //           |
82 //           |
83 //           |
84 // -----*
85 // Points d'integration
86 // -----*
87 // 1 point : (ordre 1)
88 //           Pt1 (1/4,1/4,1/4)
89 // 4 points : (ordre 2)  a = (5. - sqrt(5))/20., b = (5+3.*sqrt(5))/20.
90 //           Pt1 (a,a,a) ; Pt2 (a,a,b) ; Pt3 (a,b,a) ; Pt4 (b,a,a)
91 //
92 // 5 points : (ordre 3)  a = 1/4, b=1/6, c=1/2,
93 //           Pt1 (a,a,a) ; Pt2 (b,b,b) ; Pt3 (b,b,c) ; Pt4 ( ; Pt4 (c,b,b)
94 //
95 // 15 points : (ordre 5)  a = 1/4, b1=(7+sqrt(15))/34,  b2=(7-sqrt(15))/34,
96 //                       c1=(13+3sqrt(15))/34, c2=(13-3sqrt(15))/34,
97 //                       d=(5-sqrt(15))/20, e=(5+sqrt(15))/20,
98 //           Pt1 (a,a,a) ; Pt2 (b1,b1,b1) ; Pt3 (b2,b2,b2) ; Pt4 (b1,b1,c1)
99 //           Pt5 (b2,b2,c2) ; Pt6 (b1,c1,b1) ; Pt7 (b2,c2,b2) ; Pt8 (c1,b1,b1)
100 //           Pt9 (c2,b2,b2) ; Pt10 (d,d,e) ; Pt11 (d,e,d) ; Pt12 (e,d,d)
101 //           Pt13 (d,e,e) ; Pt14 (e,d,e) ; Pt15 (e,e,d) ;
102 //
103 // -----*
104 // pour le tetraèdre linéaire :
105 // -----*
106 // face 1 : noeud 1 3 2 , face 2 : noeud 1 4 3,
107 // face 3 : noeud 1 2 4, face 4 : noeud 2 3 4,
108 // les normales sortent des faces des elements
109 // on attribue 1 points d'integration par face
110 //
111 // pour les aretes on suit le fichier Elmail, 6 aretes
112 // A1-> 1 2  A2-> 2 3  A3-> 3 1
113 // A4-> 1 4  A5-> 2 4  A6-> 3 4
114 //
115 // on attribue 1 point d'integration par arete

```

```

116 //
117 // concernant la triangulation de chaque face elle est réalisée à l'aide
118 // de la triangulation implantée sur l'élément de référence de la face
119 //
120 //
121 // *****
122 */
123
124 /// @addtogroup Les_Elements_de_geometrie
125 /// @{
126 ///
127
128
129 class GeomTetraL : public GeomTetraCom
130 {
131 public :
132     // CONSTRUCTEURS :
133     // il y a 1 points d'integration par défaut et 4 noeuds
134     GeomTetraL(int nbi = 1);
135     // de copie
136     GeomTetraL(const GeomTetraL& a);
137     // DESTRUCTEUR :
138     ~GeomTetraL();
139
140     // création d'élément identiques : cette fonction est analogue à la fonction new
141     // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
142     // dérivée
143     // pt est le pointeur qui est affecté par la fonction
144     ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
145
146     //----- cas de coordonnees locales quelconques -----
147     // retourne les fonctions d'interpolation au point M (en coordonnees locales)
148     const Vecteur& Phi(const Coordonnee& M);
149     // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
150     const Mat_pleine& Dphi(const Coordonnee& M);
151     // en fonction de coordonnees locales, retourne true si le point est a l'interieur
152     // de l'element, false sinon
153     bool Interieur(const Coordonnee& M);
154
155 protected :
156
157     // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
158     Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
159     Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
160
161     // METHODES PROTEGEES :
162     // constitution du tableau Extrapol
163     void Calcul_extrapol(int nbi);
164
165 };
166 /// @} // end of group
167
168 #endif

```

## 7.115 GeomTetraQ.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29

```

```

30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:  Définir La geometrie des tétraèdres quadratiques.
39 *         Fonction d'interpolation, points d'integration etc
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   !           !           !
47 *   !           !           !
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   !           !           !
52 *   !           !           !
53 *****/
54 #ifndef GEOMTETRAQ_H
55 #define GEOMTETRAQ_H
56
57 #include "GeomTetraCom.h"
58
59 /*
60 // *****/
61 //
62 //   ELEMENT DE REFERENCE , POINTS D'INTEGRATION:
63 //
64 //   Source : Dhatt et Touzot p 130, 131, 132 pour les fonctions
65 //           d'interpolation. Pour la numérotation : Modulef
66 // -----*
67 //
68 //
69 //
70 //
71 //
72 //
73 //
74 //
75 //
76 //
77 //
78 //
79 //
80 //
81 //
82 //   xi      tetraèdre quadratique
83 //
84 //-----
85 // Points d'integration
86 //-----
87 // 1 point : (ordre 1)
88 //           Pt1 (1/4,1/4,1/4)
89 // 4 points : (ordre 2)  a = (5. - sqrt(5))/20., b = (5+3.*sqrt(5))/20.
90 //           Pt1 (a,a,a) ; Pt2 (a,a,b) ; Pt3 (a,b,a) ; Pt4 (b,a,a)
91 //
92 // 5 points : (ordre 3)  a = 1/4, b=1/6, c=1/2,
93 //           Pt1 (a,a,a) ; Pt2 (b,b,b) ; Pt3 (b,b,c) ; Pt4 ( ; Pt4 (c,b,b)
94 //
95 // 15 points : (ordre 5)  a = 1/4, b1=(7+sqrt(15))/34, b2=(7-sqrt(15))/34,
96 //                       c1=(13+3sqrt(15))/34, c2=(13-3sqrt(15))/34,
97 //                       d=(5-sqrt(15))/20, e=(5+sqrt(15))/20,
98 //           Pt1 (a,a,a) ; Pt2 (b1,b1,b1) ; Pt3 (b2,b2,b2) ; Pt4 (b1,b1,c1)
99 //           Pt5 (b2,b2,c2) ; Pt6 (b1,c1,b1) ; Pt7 (b2,c2,b2) ; Pt8 (c1,b1,b1)
100 //           Pt9 (c2,b2,b2) ; Pt10 (d,d,e) ; Pt11 (d,e,d) ; Pt12 (e,d,d)
101 //           Pt13 (d,e,e) ; Pt14 (e,d,e) ; Pt15 (e,e,d) ;
102 //
103 //
104 //-----
105 // pour le tetraèdre quadratique :
106 //-----
107 // face 1 : noeud 7 3 6 2 5 1 , face 2 : noeud 5 2 9 4 8 1,
108 // face 3 : noeud 7 1 8 4 10 3 , face 4 : noeud 6 3 10 4 9 2,
109 // les normales sortent des faces des elements
110 // on attribue 3 points d'integration par face
111 //
112 // pour les aretes on suit le fichier Elmail, 6 aretes
113 // A1-> 1 5 2 A2-> 2 6 3 A3-> 3 7 1
114 // A4-> 1 8 4 A5-> 2 9 4 A6-> 3 10 4
115 //

```



```

116 //   on attribue 2 point d'integration par arete
117 //
118 //   concernant la triangulation de chaque face elle est réalisée à l'aide
119 //   de la triangulation implantée sur l'élément de référence de la face
120 //
121 //
122 //
123 // *****
124 */
125
126 /// @addtogroup Les_Elements_de_geometrie
127 /// @{
128 ///
129
130
131 class GeomTetraQ : public GeomTetraCom
132 {
133     public :
134         // CONSTRUCTEURS :
135         // il y a 4 points d'integration et 10 noeuds
136         GeomTetraQ(int nbi = 4);
137         // de copie
138         GeomTetraQ(const GeomTetraQ& a);
139         // DESTRUCTEUR :
140         ~GeomTetraQ();
141
142         // création d'élément identiques : cette fonction est analogue à la fonction new
143         // elle y fait d'ailleurs appel. l'implantation est spécifique dans chaque classe
144         // dérivée
145         // pt est le pointeur qui est affecté par la fonction
146         ElemGeomC0 * newElemGeomC0(ElemGeomC0 * pt) ;
147
148         //----- cas de coordonnees locales quelconques -----
149         // retourne les fonctions d'interpolation au point M (en coordonnees locales)
150         const Vecteur& Phi(const Coordonnee& M);
151         // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
152         const Mat_pleine& Dphi(const Coordonnee& M);
153         // en fonction de coordonnees locales, retourne true si le point est a l'interieur
154         // de l'element, false sinon
155         bool Interieur(const Coordonnee& M);
156
157     protected :
158
159         // variables de stockage transitoire, locales pour éviter de les reconstruire à chaque appel
160         Vecteur phi_M; // le tableau phi au point M(en coordonnees locales)
161         Mat_pleine dphi_M; //les derivees des fonctions d'interpolation au point M(en coordonnees locales)
162
163         // METHODES PROTEGEES :
164         inline double& DPHI(int i,int j,int k) { return tabDPhi(k)(i,j);};
165         inline double& PHI(int i,int j) {return tabPhi(j)(i); };
166         // because les routine de calcul de phi et dphi aux pt d'integ sont trop grandes
167         // on en fait des routines
168         void Phiphi();
169         void DphiDphi();
170         // constitution du tableau Extrapol
171         void Calcul_extrapol(int nbi);
172
173 };
174 /// @} // end of group
175
176 #endif

```

## 7.116 ElFrontiere.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty

```

```

22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *
38 *   BUT:      Classe generique des elements frontieres
39 *
40 *   ****
41 *
42 *   VERIFICATION:
43 *   ! date !   auteur !       but
44 *   -----
45 *   !       !       !
46 *
47 *   ****
48 *   MODIFICATIONS:
49 *   ! date !   auteur !       but
50 *   -----
51 *
52 *   ****
53 #ifndef ELFRONTIERE_H
54 #define ELFRONTIERE_H
55
56 #include "Tableau_T.h"
57 #include "Noeud.h"
58 #include "string.h"
59 // #include "bool.h"
60 #include "DdlElement.h"
61 #include "Plan.h"
62 #include "ElemGeomC0.h"
63 #include "Met_abstraite.h"
64 #include "Enum_type_geom.h"
65 #include "Enum_dure.h"
66
67 /** @defgroup Les_Elements_de_frontiere
68 *
69 *   BUT:   groupe concernant les éléments géométriques définissant les frontières 1D, 2D, 3D
70 *
71 * \author   Gérard Rio
72 * \version  1.0
73 * \date    23/01/97
74 * \brief   groupe concernant les éléments géométriques définissant les frontières 1D, 2D, 3D
75 *
76 */
77
78
79 /// @addtogroup Les_Elements_de_frontiere
80 /// @{
81 ///
82
83
84 class ElFrontiere
85 {
86 public :
87     // CONSTRUCTEURS :
88     // par défaut
89     ElFrontiere ();
90     // fonction du tableau des noeuds sommet, des ddlElements
91     ElFrontiere (const Tableau <Noeud *>& tab,const DdlElement& ddlElem);
92     // nbnoeud permet de faire les verifications de tailles de tableau en debug
93     // par les classe derivees
94     ElFrontiere (const Tableau <Noeud *>& tab,const DdlElement& ddlElem,int nbnoeud);
95     ElFrontiere(const ElFrontiere& a); // de copie
96     // DESTRUCTEUR : défini dans les classes dérivées
97     virtual ~ElFrontiere ();
98
99     // METHODES PUBLIQUES :
100    // surcharge de l'affectation
101    virtual ElFrontiere& operator = (const ElFrontiere& a);
102    // retourne le type de l'element frontiere
103    virtual string TypeFrontiere() const = 0;
104    // retourne le type de géométrie de l'élément frontiere: POINT_G, LIGNE ou SURFACE
105    Enum_type_geom Type_geom_front() const
106    {return Type_geom_generique(ElementGeometrique().TypeGeometrie());};
107    // retourne l'élément géométrique attaché à l'élément frontiere

```

```

108 virtual ElemGeomC0 const & ElementGeometrique() const = 0;
109 // surcharge des tests
110 // =====
111 // IMPORTANT !!!
112 // le test d'egalite ne concerne que les noeuds , pas
113 // les donnees particulieres des elements derivees, qui en faites sont
114 // sujet a variation, donc pas discriminant"
115 // =====
116 virtual bool operator == (const ElFrontiere& a) const;
117 bool operator != (const ElFrontiere& a) const
118 { if (*this == a) return false; else return true; };
119
120 // retourne le tableau de noeud en lecture seulement
121 inline const Tableau<Noeud *>& TabNoeud_const() const { return tabNoeud; };
122 // retourne le tableau de noeud en lecture /écriture
123 inline Tableau<Noeud *>& TabNoeud() { return tabNoeud; };
124 // retourne les ddl elements en lecture / écriture
125 inline DdlElement& DdlElem() { return ddlElem; };
126 // retourne les ddl elements en lecture seulement
127 inline const DdlElement& DdlElem_const() const { return ddlElem; };
128 // retourne le tableau d'element frontiere en lecture / écriture
129 inline Tableau<ElFrontiere*>& TabFront() { return tabfront; };
130 // retourne le tableau d'element frontiere en lecture uniquement
131 inline const Tableau<ElFrontiere*>& TabFront_const() const { return tabfront; };
132 // creation d'un nouvelle element frontiere identique
133 virtual ElFrontiere * NevezElemFront() const = 0;
134 // creation d'un nouvelle element frontiere de meme type mais
135 // avec des donnees differentes
136 virtual ElFrontiere * NevezElemFront
137 ( const Tableau<Noeud *>& tab, const DdlElement& ddlElem) const = 0;
138 // cree et retourne un element frontiere ayant une orientation opposee
139 virtual ElFrontiere * Oppose() const;
140
141 // ramene et calcul les coordonnees du point de reference de l'element
142 virtual Coordonnee Ref() = 0;
143 // ramene un plan tangent ou une droite tangente au point de reference
144 // si indic = 1 -> une droite, =2 -> un plan
145 // ces infos sont stocke et sauvegardees dans l'element de frontiere
146 virtual void TangentRef(Droite& dr, Plan& pl, int& indic) = 0;
147 // M est un point appartenant au dernier plan tangent, sauvegarde dans l'element frontiere
148 // - calcul du point M1 correspondant sur la surface, M1 est stocke
149 // _ calcul et retour du plan tangent (ou une droite tangente) au point M1
150 // si indic = 1 -> une droite, =2 -> un plan
151 // ces infos sont stocke et sauvegardees dans l'element de frontiere
152 virtual void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic) = 0;
153 // ramene un autre plan tangent ou une droite tangente genere de maniere pseudo aleatoire
154 // si indic = 1 -> une droite, =2 -> un plan
155 // ces infos sont stocke et sauvegardees dans l'element de frontiere
156 virtual void AutreTangent(Droite& dr, Plan& pl, int& indic) = 0;
157 // ramene true si le dernier point M1 est dans l'element , sinon false
158 // le calcul est fait à eps relatif près
159 virtual bool InSurf(const double& eps) const = 0;
160 // actualise , puis ramene le dernier plan tangent (ou droite tangente) calcule
161 // dans le cas 1D, ramene un point
162 // en sortie : si indic =0 -> un point, indic = 1 -> une droite, indic = 2 -> un plan
163 // ramene éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
164 // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
165 // sinon ramène NULL
166 virtual Tableau<Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false)
167 = 0;
168 // affichage des infos de l'elements frontiere
169 virtual void Affiche(Enum_dure temp = TEMPS_tdt) const = 0;
170 // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
171 // si le point est sur la surface, ramène false
172 // ramene true si hors matiere, sinon false
173 // le test sur a est executer uniquement dans les cas suivants :
174 // dimension 3D et frontiere 2D
175 // dimension 3D axi et frontiere 1D
176 // dimension 2D et frontiere 1D
177 // ->> dimension 3D et frontiere 1D, pas de verif
178 // ->> autre cas ne doivent pas arriver normalement !!
179 // retour de r = distance du point à la surface, ligne
180 virtual bool BonCote_t( const Coordonnee& a, double& r) const = 0; // cas ou on utilise la frontiere
181 a t
182 virtual bool BonCote_tdt( const Coordonnee& a, double& r) const = 0; // cas ou on utilise la
183 frontiere a tdt
184 // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
185 virtual const Vecteur& Phi() = 0;
186
187 // creation et ramene des pointeurs sur les frontieres de l'element frontiere
188 // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
189 // à moins qu'il soit effacé
190 virtual Tableau<ElFrontiere*>& Frontiere() = 0;
191 // effacement de la place memoire des frontieres de l'element frontiere
192 virtual void EffaceFrontiere();
193
194 // ramène la métrique associée à l'élément

```

```

191     virtual Met_abstraite * Metrique() =0;
192     // cas d'un élément frontière surface:
193     // ramène une surface approximative de l'élément (toujours > 0) : calculée à l'aide
194     // d'une triangulation, puis des produits vectoriels
195     // ramène une valeur nulle, s'il n'y a pas de surface
196     double SurfaceApprox();
197     // cas d'un élément frontière ligne:
198     // ramène, une longueur approximative de l'élément (toujours > 0) : calculée à l'aide
199     // de la ligne représentée par une suite de segments rejoignant les noeuds
200     // ramène une valeur nulle, s'il n'y a pas de ligne
201     virtual double LongueurApprox() { return 0; };
202     // ramène la distance maxi entre deux noeuds de l'élément à tdt
203     double MaxDiagonale_tdt();
204
205     // ramène l'encombrement de l'élément sous forme du point ayant les coordonnées mini
206     // et le point ayant les coordonnées les maxi
207     const Coordonnee& Encom_mini() {return encomb_min;};
208     const Coordonnee& Encom_maxi() {return encomb_max;};
209     // met à jour la boîte d'encombrement
210     void AjourBoiteEncombrement();
211
212     // calcul de la projection normale d'un point sur la frontière
213     // ramène true si la projection est effective, si elle est hors de l'élément
214     // ramène false
215     // P : retour du point projeté (s'il existe)
216     bool Projection_normale(const Coordonnee& M, Coordonnee& P);
217
218     //----- lecture écriture de restart -----
219     // ceci concerne uniquement les informations de la classe générique
220     void Lecture_base_info_ElFrontiere(istream& ent);
221     void Ecriture_base_info_ElFrontiere(ofstream& sort);
222     // ceci concerne uniquement les informations spécifiques des classes dérivées
223     // dans le cas de l'utilisation de la frontière pour la projection d'un point sur la frontière
224     // il s'agit donc d'un cas particulier
225     virtual void Lecture_base_info_ElFrontiere_pour_projection(istream& ent) = 0;
226     virtual void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) = 0;
227
228 protected :
229     // VARIABLES PROTEGEES :
230     Tableau <Noeud *> tabNoeud; // le tableau de noeud associe a l'element frontiere
231     DdlElement ddlElem; // les ddl spécifiques aux noeuds de l'element frontiere
232     Tableau <ElFrontiere*> tabfront; // frontiere de l'element frontiere
233     Coordonnee encomb_min, encomb_max; // boîte d'encombrement des noeuds de l'élément
234
235     static unsigned int nrand; // pour la generation de nombre aleatoire
236
237     // METHODES PROTEGEES :
238     // test si le point passé en argument appartient à la boîte d'encombrement de la frontière
239     // fonction interne pour faciliter les tests, mais c'est celle de Front qui est à utiliser
240     bool In_boite_emcombrement(const Coordonnee& M) const;
241
242 };
243 /// @} // end of group
244
245 #endif

```

## 7.117 Front.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //

```

```

28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *
38 *   BUT:   definir un element de stockage de frontiere relatif à un
39 *          éléments finis.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *   *****/
54 #ifndef FRONT_H
55 #define FRONT_H
56
57 #include "Element.h"
58 #include "ElemMeca.h"
59 #include "ElFrontiere.h"
60
61 /// @addtogroup Les_Elements_de_frontiere
62 /// @{
63 ///
64
65
66 // la classe Front stock un pointeur d'element frontiere, le nb de
67 // l'element finis auquel il se rapporte
68 class Front
69 { public :
70     // constructeur
71     //par default
72     Front();
73     // normal
74     Front ( const ElFrontiere& el, Element * pt, int num_front );
75     // de copie
76     Front ( const Front& a);
77     // destructeur
78     ~Front();
79
80 // ===== METHODES =====
81
82 // affichage à l'écran des informations liées au contact
83 void Affiche() const ;
84
85 // operator
86 // affectation de toute les donnees
87 Front& operator = ( const Front& a);
88
89 // test d'egalite total
90 inline bool operator == ( const Front& a) const
91 { if (*(this->elem) == *(a.elem))
92     && (this->ptEl == a.ptEl) && (this->tabmitoyen == a.tabmitoyen)
93     && (num_frontiere == a.num_frontiere) )
94     return true;
95     else
96     return false;
97 };
98 // test d'egalite sur les éléments originaux :
99 // l'élément elem, ptEl, et num_frontiere, mais qui ne comprend pas tabmitoyen
100 inline bool MemeOrigine( const Front& a) const
101 { if ( *(this->elem) == *(a.elem) && (this->ptEl == a.ptEl)
102     && (num_frontiere == a.num_frontiere))
103     return true;
104     else
105     return false;
106 };
107
108 inline bool operator != ( const Front& a) const
109 { if (*this == a) return false; else return true;};
110
111 // definition des elements mitoyens
112 void DefMitoyen(Tableau <Front*>& tabmitoyen);
113

```

```

114 // retourne les donnees
115 inline const Tableau <Front*>* TabMitoyen() const { return tabmitoyen;};
116
117 // ramène l'encombrement de l'élément frontière sous forme du point
118 //ayant les coordonnées mini et le point ayant les coordonnées les maxi
119 const Coordonnee& Encom_mini_FR() {return boite_Front.Premier();};
120 const Coordonnee& Encom_maxi_FR() {return boite_Front.Second();};
121 // test si le point passé en argument appartient à la boîte d'encombrement de la frontière
122 // tous les points sont supposées avoir la même dimension
123 bool In_boite_emcombrement_front(const Coordonnee& M) const;
124 // test si le point passé en argument appartient à la boîte d'encombrement de la frontière
125 // tous les points sont supposées avoir la même dimension
126 // ici la boîte est augmentée de extra dans tous les sens
127 bool In_boite_emcombrement_front(const Coordonnee& M,double extra) const;
128
129 // mise à jour de la boîte d'encombrement de la frontière
130 // dep_max : déplacement maxi des noeuds du maillage
131 // dans le cas d'un dep_max != 0., on agrandit la boîte (défavorable !!)
132 // , sert pour def des boites d'encombrement maxi des frontières
133 void Boite_encombrement_frontiere(Enum_dure temps,double dep_max=0.);
134
135
136 //----- lecture écriture de restart -----
137 // définition d'un conteneur signature permettant de définir les grandeurs internes de front
138 class Signature_Front { public: int numelem; int numMail;};
139 // la lecture s'effectue uniquement au niveau de la signature
140 // 1) lecture et retour de la signature
141 Signature_Front Lecture_base_info_Signature_Front(ifstream& ent);
142 // ici la lecture n'est pas complète il faut ensuite mettre à jour l'élément
143 // frontière en fonction de son numéro qui est déjà stocké dans l'élément, et le pointeur de
144 // l'élément
145 // -> utilisation de : Change_elem_frontiere, Change_PtEI
146 void Lecture_base_info_front(ifstream& ent);
147 void Ecriture_base_info_front(ofstream& sort);
148
149 // changement d'élément frontière, il faut également le numéro de l'élément fini associé
150 void Change_elem_frontiere(const ElFrontiere& el, int num_front)
151 { if (elem != NULL) {*elem = el;} else { elem = el.NevezElemFront();}; num_frontiere=
num_front;};
152 // ici il faut faire attention, car il faut que le numéro de frontière soit cohérent avec
153 // l'élément
154 // donc normalement Change_PtEI doit s'utiliser avec Change_elem_frontiere
155 void Change_PtEI(Element * ptei) { ptEl = ptei;};
156
157 // routine de récupération d'informations
158 // récup de l'element frontiere en modification éventuelle
159 // si le retour est NULL, c'est que l'élément frontière n'est pas définit
160 ElFrontiere* Eleme() const {return elem;};
161 // idem mais en constant
162 const ElFrontiere* Eleme_const() const {return elem;};
163
164 // récup du numero du maillage rattache
165 int NumMail() {return ptEl->Num_maillage();};
166 // récup du pointeur sur l'element fini qui a cree elem
167 Element * PtEI() const { return ptEl;};
168 // récup du numéro de frontière associé à l'élément fini
169 int Num_frontiere() const {return num_frontiere;};
170
171 // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
172 // si le point est sur la surface, ramène false
173 // ramene true si hors matiere, sinon false
174 // le test sur a est executer uniquement dans les cas suivants :
175 // dimension 3D et frontiere 2D
176 // dimension 3D axi et frontiere 1D
177 // dimension 2D et frontiere 1D
178 // ->> dimension 3D et frontiere 1D, pas de verif
179 // ->> autre cas ne doivent pas arriver normalement !!
180 // .... fonctions identique à celles définit dans ElFrontiere sauf pour le cas 1D
181 // où la fonction de ElFrontiere ne fonctionne pas, il faut utiliser celle_ci
182 //!!!! il n'y a pas de vérification que l'élément frontière existe !!
183 // retour de r = distance du point à la surface, ligne
184 bool BonCote_t( const Coordonnee& a,double& r) const // cas ou on utilise la frontiere a t
185 { if (ParaGlob::Dimension() != 1) return elem->BonCote_t(a,r);
186 else { r=ConstMath::grand; return (!((ElemMeca *)ptEl)->Interne_t(a));};};
187 bool BonCote_tdt( const Coordonnee& a,double& r) const // cas ou on utilise la frontiere a tdt
188 { if (ParaGlob::Dimension() != 1) return elem->BonCote_tdt(a,r);
189 else { r=ConstMath::grand; return (!((ElemMeca *)ptEl)->Interne_tdt(a));};};
190
191 // variables
192 protected:
193 ElFrontiere* elem; // un pointeur d'élément frontiere:
194 // on est obligé d'avoir un pointeur car l'élément frontière est virtuel pur
195
196 Element * ptEl; // pointeur sur l'element fini qui a cree elem
197 int num_frontiere; // le numéro de la frontière relatif à l'élément finis associé
198 // def d'une boîte d'encombrement pour les noeuds de la frontière
199 DeuxCoordonnees boite_Front; // le premier le min, le second le max

```

```

198
199     Tableau <Front*>* tabmitoyen; // tableau des elements mitoyens
200
201     // fonctions internes
202     static double prop_mini; // mini proportion à ajouter à l'encombrement
203
204 };
205 /// @} // end of group
206
207 #endif

```

## 7.118 FrontSegCub.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      10/11/01
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:      Frontiere : segment cubique avec 3 points d'integ.
39 *             On considère aussi le cas ou il s'agit de segment
40 *             axisymétrique. Dans ce cas, le support géométrique est 1D
41 *             mais l'élément réel est 2D. A priori-cela n'influe pas
42 *             les différentes méthodes, sauf la définition de la
43 *             métrique.
44 *
45 *             $
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !           but
50 *   !-----!-----!-----!-----!
51 *   !           !           !           !
52 *   !           !           !           !
53 *   !           !           !           !
54 *   !           !           !           !
55 *   !           !           !           !
56 *   !           !           !           !
57 *****/
58 #ifndef FRONTSEGCUB_H
59 #define FRONTSEGCUB_H
60
61 #include "ElFrontiere.h"
62 #include "GeomSeg.h"
63
64 /// @addtogroup Les_Elements_de_frontiere
65 /// @{
66 ///
67
68
69 class FrontSegCub : public ElFrontiere
70 {
71     public :

```

```

72 // CONSTRUCTEURS :
73 //par default
74 FrontSegCub ();
75 // fonction du tableau des noeuds sommets
76 FrontSegCub ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);
77 FrontSegCub( const FrontSegCub& a); // de copie
78 // DESTRUCTEUR :
79 ~FrontSegCub ();
80 // surcharge de l'affectation
81 ElFrontiere& operator = ( const ElFrontiere& a);
82 // retourne le type de l'element frontiere
83 string TypeFrontiere() const ;
84 // retourne l'élément géométrique attaché à l'élément frontiere
85 ElemGeomC0 const & ElementGeometrique() const {return segment;};
86 // creation d'un nouvelle element frontiere du type FrontSegCub
87 ElFrontiere * NevezElemFront() const ;
88 // creation d'un nouvelle element frontiere du type FrontSegCub
89 // avec des donnees differentes
90 ElFrontiere * NevezElemFront( const Tableau <Noeud *>& tab, const DdlElement& ddlElem) const ;
91
92 // ramene et calcul les coordonnees du point de reference de l'element
93 Coordonnee Ref();
94 // ramene une droite tangente au point de reference
95 // si indic = 1 -> une droite
96 // ces infos sont stocke et sauvegardees dans l'element
97 void TangentRef(Droite& dr, Plan& pl, int& indic);
98 // M est un point de la derniere droite tangente sauvegarde dans l'element
99 // - calcul du point M1 correspondant sur la courbe , M1 est stocke
100 // _ calcul et retour de la droite tangente au point M1
101 // si indic = 1 -> une droite
102 // ces infos sont stocke et sauvegardees dans l'element
103 void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
104 // ramene une autre droite tangente genere de maniere pseudo aleatoire
105 // si indic = 1 -> une droite,
106 // ces infos sont stocke et sauvegardees dans l'element
107 void AutreTangent(Droite& dr, Plan& pl, int& indic);
108 // ramene true si le dernier point M1 est dans l'element , sinon false
109 // le calcul est fait à eps relatif près
110 bool InSurf(const double& eps) const ;
111 // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
112 // si indic = 1 -> une droite, =2 -> un plan
113 // ramène éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
114 // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
115 Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
116 // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
117 // si le point est sur la surface, ramène false
118 // ramene true si hors matiere, sinon false
119 // le test sur a est executer uniquement dans les cas suivants :
120 // dimension 3D et frontiere 2D
121 // dimension 3D axi et frontiere 1D
122 // dimension 2D et frontiere 1D
123 // ->> dimension 3D et frontiere 1D, pas de verif
124 // ->> autre cas ne doivent pas arriver normalement !!
125 // retour de r = distance du point à la surface, ligne
126 bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
127 bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
128 // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
129 const Vecteur& Phi();
130
131 // affichage des infos de l'elements
132 void Affiche(Enum_dure temp = TEMPS_tdt) const ;
133
134 // creation et ramene des pointeurs sur les frontieres de l'element frontiere
135 // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
136 // à moins qu'il soit effacé
137 Tableau <ElFrontiere*>& Frontiere();
138
139 // ramène la métrique associée à l'élément
140 Met_abstraite * Metrique() {return met;};
141 // cas d'un élément frontiere ligne:
142 // ramène, une longueur approximative de l'élément (toujours > 0) : calculée à l'aide
143 // de la ligne représentée par une suite de segments rejoignant les noeuds
144 // ramène une valeur nulle, s'il n'y a pas de ligne
145 double LongueurApprox();
146
147 //----- lecture écriture de restart -----
148 // ceci concerne uniquement les informations spécifiques
149 // dans le cas de l'utilisation de la frontiere pour la projection d'un point sur la frontiere
150 // il s'agit donc d'un cas particulier
151 void Lecture_base_info_ElFrontiere_pour_projection(istream& ent) ;
152 void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
153
154 private :
155 // VARIABLES PROTEGEES :
156 static GeomSeg segment; // le segment de reference
157 static Met_abstraite * met; // une metrique associee

```



```

158 // de type Met_biellette ou MetAxisymetrique2D
159 static Vecteur phi; // interpolation au point courant
160 static Mat_pleine dphi; // derivees de l'interpolation au point courant
161 Coordonnee refP; // Point de reference de l'element
162 Droite droite; // droite tangente ici c'est l'element
163 Tableau <Coordonnee > d_T; // variation du vecteur tangent
164 static BaseB giB; // base naturelle de travail
165 static BaseH giH; // base duale de travail
166 Coordonnee theta; // coordonnees du point courant projete
167 Coordonnee theta_repere; // le point où l'on calcul le repère
168
169 // METHODES PROTEGEES :
170 // methodes propres a l'element
171 inline Droite DR() const { return droite; };
172 inline Coordonnee Theta() const { return theta; };
173 // definition de la metrique
174 void DefMetrique();
175 };
176 /// @} // end of group
177
178 #endif

```

## 7.119 FrontSegLine.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *
38 *      BUT:      Frontiere : segment lineaire avec un pt d'integ.
39 *              On considère aussi le cas ou il s'agit de segment
40 *              axisymétrique. Dans ce cas, le support géométrique est 1D
41 *              mais l'élément réel est 2D. A priori-cela n'influe pas
42 *              les différentes méthodes, sauf la définition de la
43 *              métrique.
44 *
45 *              *****
46 *
47 *      VERIFICATION:
48 *      ! date !   auteur !       but
49 *      -----
50 *      !       !       !
51 *
52 *      *****
53 *      MODIFICATIONS:
54 *      ! date !   auteur !       but
55 *      -----
56 *
57 *      *****/
58 #ifndef FRONTSEGLINE_H
59 #define FRONTSEGLINE_H
60

```

```

61 #include "ElFrontiere.h"
62 #include "GeomSeg.h"
63
64 /// @addtogroup Les_Elements_de_frontiere
65 /// @{
66 ///
67
68
69 class FrontSegLine : public ElFrontiere
70 {
71 public :
72 // CONSTRUCTEURS :
73 //par default
74 FrontSegLine ();
75 // fonction du tableau des noeuds sommets
76 FrontSegLine ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);
77 FrontSegLine( const FrontSegLine& a); // de copie
78 // DESTRUCTEUR :
79 ~FrontSegLine ();
80
81 // METHODES PUBLIQUES :
82
83 // surcharge de l'affectation
84 ElFrontiere& operator = ( const ElFrontiere& a);
85 // retourne le type de l'element frontiere
86 string TypeFrontiere() const ;
87 // retourne l'élément géométrique attaché à l'élément frontiere
88 ElemGeomC0 const & ElementGeometrique() const {return segment;};
89 // creation d'un nouvelle element frontiere du type FrontSegLine
90 ElFrontiere * NevezElemFront() const ;
91 // creation d'un nouvelle element frontiere du type FrontSegLine
92 // avec des donnees differentes
93 ElFrontiere * NevezElemFront( const Tableau <Noeud *>& tab, const DdlElement& ddlElem) const ;
94
95 // ramene les coordonnees du point de reference de l'element
96 Coordonnee Ref() ;
97 // ramene une droite tangente au point de reference
98 // si indic = 1 -> une droite
99 // ces infos sont stocke et sauvegardees dans l'element
100 void TangentRef(Droite& dr, Plan& pl, int& indic);
101 // M est un point de la derniere droite tangente sauvegarde dans l'element
102 // - calcul du point M1 correspondant sur la courbe , M1 est stocke
103 // _ calcul et retour de la droite tangente au point M1
104 // si indic = 1 -> une droite
105 // ces infos sont stocke et sauvegardees dans l'element
106 void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
107 // ramene une autre droite tangente genere de maniere pseudo aleatoire
108 // si indic = 1 -> une droite,
109 // ces infos sont stocke et sauvegardees dans l'element
110 void AutreTangent(Droite& dr, Plan& pl, int& indic);
111 // ramene true si le dernier point M1 est dans l'element , sinon false
112 // le calcul est fait à eps relatif près
113 bool InSurf(const double& eps) const ;
114 // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
115 // si indic = 1 -> une droite, =2 -> un plan
116 // ramène éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
117 // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
118 Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
119 // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
120 // si le point est sur la surface, ramène false
121 // ramene true si hors matiere, sinon false
122 // le test sur a est executer uniquement dans les cas suivants :
123 // dimension 3D et frontiere 2D
124 // dimension 3D axi et frontiere 1D
125 // dimension 2D et frontiere 1D
126 // ->> dimension 3D et frontiere 1D, pas de verif
127 // ->> autre cas ne doivent pas arriver normalement !!
128 // retour de r = distance du point à la surface, ligne
129 bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
130 bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
131 // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
132 const Vecteur& Phi();
133
134 // affichage des infos de l'elements
135 void Affiche(Enum_dure temp = TEMPS_tdt) const ;
136
137 // creation et ramene des pointeurs sur les frontieres de l'element frontiere
138 // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
139 // à moins qu'il soit effacé
140 Tableau <ElFrontiere*>& Frontiere();
141
142 // ramène la métrique associée à l'élément
143 Met_abstraite * Metrique() {return met;};
144 // cas d'un élément frontiere ligne:
145 // ramène, une longueur approximative de l'élément (toujours > 0) : calculée à l'aide
146 // de la ligne représentée par une suite de segments rejoignant les noeuds

```

```

147 // ramène une valeur nulle, s'il n'y a pas de ligne
148 double LongueurApprox();
149
150 //----- lecture écriture de restart -----
151 // ceci concerne uniquement les informations spécifiques
152 // dans le cas de l'utilisation de la frontière pour la projection d'un point sur la frontière
153 // il s'agit donc d'un cas particulier
154 void Lecture_base_info_ElFrontiere_pour_projection(istream& ent) ;
155 void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
156
157 private :
158 // VARIABLES PROTEGEES :
159 static GeomSeg segment; // le segment de reference
160 static Met_abstraite * met; // une metrique associee
161 // de type Met_biellette ou MetAxisymetrique2D
162 static Vecteur phi; // interpolation au point courant
163 static Mat_pleine dphi; // derivees de l'interpolation au point courant
164 Coordonnee refP; // Point de reference de l'element
165 Droite droite; // droite tangente ici c'est l'element, à la longueur près
166 Tableau <Coordonnee > d_T; // variation du vecteur tangent
167 Coordonnee Mp; // coordonnees du point courant projete
168
169 static BaseB giB; // base naturelle de travail
170 static BaseH giH; // base duale de travail
171 Coordonnee theta; // coordonnees du point courant projete
172 Coordonnee theta_repere; // le point où l'on calcul le repère
173
174 // METHODES PROTEGEES :
175 // methodes propres a l'element
176 inline Coordonnee MP() const { return Mp;};
177 inline Droite DR() const { return droite; };
178 inline Coordonnee Theta() const { return theta;};
179 // definition de la metrique
180 void DefMetrique();
181 };
182 /// @} // end of group
183
184 #endif

```

## 7.120 FrontSegQuad.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:      Frontiere : segment quadratique avec deux points d'integ.
39 *
40 *   On considère aussi le cas où il s'agit de segment
41 *   axisymétrique. Dans ce cas, le support géométrique est 1D
42 *   mais l'élément réel est 2D. A priori-cela n'influe pas
43 *   les différentes méthodes, sauf la définition de la
44 *   métrique.

```

```

45 *      *
46 *      VERIFICATION: *
47 * * *
48 *      ! date ! auteur ! but ! *
49 *      ----- *
50 *      ! ! ! ! *
51 * * *
52 *      *
53 *      MODIFICATIONS: *
54 *      ! date ! auteur ! but ! *
55 *      ----- *
56 * * *
57 *      */
58 #ifndef FRONTSEGQUAD_H
59 #define FRONTSEGQUAD_H
60
61 #include "ElFrontiere.h"
62 #include "GeomSeg.h"
63
64 /// @addtogroup Les_Elements_de_frontiere
65 /// @{
66 ///
67
68
69 class FrontSegQuad : public ElFrontiere
70 {
71 public :
72     // CONSTRUCTEURS :
73     //par default
74     FrontSegQuad ();
75     // fonction du tableau des noeuds sommets
76     FrontSegQuad ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);
77     FrontSegQuad( const FrontSegQuad& a); // de copie
78     // DESTRUCTEUR :
79     ~FrontSegQuad ();
80     // surcharge de l'affectation
81     ElFrontiere& operator = ( const ElFrontiere& a);
82     // retourne le type de l'element frontiere
83     string TypeFrontiere() const ;
84     // retourne l'élément géométrique attaché à l'élément frontiere
85     ElemGeomC0 const & ElementGeometrique() const {return segment;};
86     // creation d'un nouvelle element frontiere du type FrontSegQuad
87     ElFrontiere * NevezElemFront() const ;
88     // creation d'un nouvelle element frontiere du type FrontSegQuad
89     // avec des donnees differentes
90     ElFrontiere * NevezElemFront( const Tableau <Noeud *>& tab, const DdlElement& ddlElem) const ;
91
92     // ramene et calcul les coordonnees du point de reference de l'element
93     Coordonnee Ref();
94     // ramene une droite tangente au point de reference
95     // si indic = 1 -> une droite
96     // ces infos sont stocke et sauvegardees dans l'element
97     void TangentRef(Droite& dr, Plan& pl, int& indic);
98     // M est un point de la derniere droite tangente sauvegarde dans l'element
99     // - calcul du point M1 correspondant sur la courbe , M1 est stocke
100    // _ calcul et retour de la droite tangente au point M1
101    // si indic = 1 -> une droite
102    // ces infos sont stocke et sauvegardees dans l'element
103    void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
104    // ramene une autre droite tangente genere de maniere pseudo aleatoire
105    // si indic = 1 -> une droite,
106    // ces infos sont stocke et sauvegardees dans l'element
107    void AutreTangent(Droite& dr, Plan& pl, int& indic);
108    // ramene true si le dernier point M1 est dans l'element , sinon false
109    // le calcul est fait à eps relatif près
110    bool InSurf(const double& eps) const ;
111    // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
112    // si indic = 1 -> une droite, =2 -> un plan
113    // ramene éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
114    // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
115    Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
116    // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
117    // si le point est sur la surface, ramène false
118    // ramene true si hors matiere, sinon false
119    // le test sur a est executer uniquement dans les cas suivants :
120    // dimension 3D et frontiere 2D
121    // dimension 3D axi et frontiere 1D
122    // dimension 2D et frontiere 1D
123    // ->> dimension 3D et frontiere 1D, pas de verif
124    // ->> autre cas ne doivent pas arriver normalement !!
125    // retour de r = distance du point à la surface, ligne
126    bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
127    bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
128    // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
129    const Vecteur& Phi();

```

```

130
131 // affichage des infos de l'elements
132 void Affiche(Enum_dure temp = TEMPS_tdt) const ;
133
134 // creation et ramene des pointeurs sur les frontieres de l'element frontiere
135 // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
136 // à moins qu'il soit effacé
137 Tableau <ElFrontiere*>& Frontiere();
138
139 // ramène la métrique associée à l'élément
140 Met_abstraite * Metrique() {return met;};
141 // cas d'un élément frontière ligne:
142 // ramène, une longueur approximative de l'élément (toujours > 0) : calculée à l'aide
143 // de la ligne représentée par une suite de segments rejoignant les noeuds
144 // ramène une valeur nulle, s'il n'y a pas de ligne
145 double LongueurApprox();
146
147 //----- lecture écriture de restart -----
148 // ceci concerne uniquement les informations spécifiques
149 // dans le cas de l'utilisation de la frontière pour la projection d'un point sur la frontière
150 // il s'agit donc d'un cas particulier
151 void Lecture_base_info_ElFrontiere_pour_projection(ifstream& ent) ;
152 void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
153
154 private :
155 // VARIABLES PROTEGEES :
156 static GeomSeg segment; // le segment de reference
157 static Met_abstraite * met; // une metrique associee
158 // de type Met_biellette ou MetAxisymetrique2D
159 static Vecteur phi; // interpolation au point courant
160 static Mat_pleine dphi; // derivees de l'interpolation au point courant
161 Coordonnee refP; // Point de reference de l'element
162 Droite droite; // droite tangente ici c'est l'element
163 Tableau <Coordonnee > d_T; // variation du vecteur tangent
164 static BaseB giB; // base naturelle de travail
165 static BaseH giH; // base duale de travail
166 Coordonnee theta; // coordonnees du point courant projete
167 Coordonnee theta_repere; // le point où l'on calcul le repère
168
169 // METHODES PROTEGEES :
170 // methodes propres a l'element
171 inline Droite DR() const { return droite; };
172 inline Coordonnee Theta() const { return theta;};
173 // definition de la metrique
174 void DefMetrrique();
175 };
176 /// @} // end of group
177
178 #endif

```

## 7.121 FrontPointF.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97      *
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)      *

```

```

34 *                                     $ *
35 *   PROJET:      Herezh++                                     $ *
36 *                                     $ *
37 *****
38 *   BUT:   Frontiere : Point.                                     $ *
39 *                                     $ *
40 *   *****
41 *   VERIFICATION:                                             *
42 *   ! date !   auteur !           but           !           *
43 *   -----!-----!-----!-----!-----!-----!----- *
44 *   !           !           !           !           !           *
45 *   -----!-----!-----!-----!-----!-----!----- *
46 *   *****
47 *   MODIFICATIONS:                                           *
48 *   ! date !   auteur !           but           !           *
49 *   -----!-----!-----!-----!-----!-----!----- *
50 *   !           !           !           !           !           *
51 *   -----!-----!-----!-----!-----!-----!----- *
52 *****/
53 #ifndef FRONTPOINTF_H
54 #define FRONTPOINTF_H
55
56 #include "ElFrontiere.h"
57 #include "Met_abstraite.h"
58 #include "GeomPoint.h"
59
60 /// @addtogroup Les_Elements_de_frontiere
61 /// @{
62 ///
63
64
65 class FrontPointF : public ElFrontiere
66 {
67     public :
68         // CONSTRUCTEURS :
69         //par default
70         FrontPointF ();
71         // fonction du tableau des noeuds sommets
72         // T est un vecteur normal optionnel au plan tangent au point
73         FrontPointF ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem, Coordonnee* T = NULL);
74         FrontPointF( const FrontPointF& a); // de copie
75         // DESTRUCTEUR :
76         ~FrontPointF ();
77         // surcharge de l'affectation
78         ElFrontiere& operator = ( const ElFrontiere& a);
79         // retourne le type de l'element frontiere
80         string TypeFrontiere() const ;
81         // retourne l'élément géométrique attaché à l'élément frontiere
82         ElemGeomC0 const & ElementGeometrique() const { return point;};
83         // creation d'un nouvelle element frontiere du type FrontPointF
84         ElFrontiere * NevezElemFront() const ;
85         // creation d'un nouvelle element frontiere du type FrontPointF
86         // avec des donnees differentes
87         ElFrontiere * NevezElemFront
88             ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem) const ;
89
90         // METHODES PUBLIQUES :
91
92
93         // ramene les coordonnees du point de reference de l'element
94         // ici ramene les coordonnees du noeud a tdt
95         Coordonnee Ref() { return tabNoeud(1)->Coord2(); };
96
97         // ramene un plan tangent ou une droite tangente au point de reference
98         // si indic = 0 -> point, indic = 1 -> une droite, =2 -> un plan
99         // ces infos sont stocke et sauvegardees dans l'element de frontiere
100        // ici il s'agit uniquement d'un point, donc pas de droite ou de plan tangent
101        void TangentRef(Droite& , Plan& , int& indic) { indic = 0;};
102
103        // M est un point du dernier plan tangent sauvegarde dans l'element frontiere
104        // - calcul du point M1 correspondant sur la surface, M1 est stocke
105        // - calcul et retour du plan tangent (ou une droite tangente) au point M1
106        // si indic = 0 -> un point, indic = 1 -> une droite, =2 -> un plan
107        // ces infos sont stocke et sauvegardees dans l'element de frontiere
108        // ici il s'agit uniquement d'un point, donc pas de droite ou de plan tangent
109        void Tangent(const Coordonnee& , Coordonnee& M1, Droite& , Plan& , int& indic)
110            { M1 = tabNoeud(1)->Coord2(); indic = 0; };
111
112        // ramene un autre plan tangent ou une droite tangente genere de maniere pseudo aleatoire
113        // si indic = 0 -> un point, indic = 1 -> une droite, =2 -> un plan
114        // ces infos sont stocke et sauvegardees dans l'element de frontiere
115        // ici il s'agit uniquement d'un point, donc pas de droite ou de plan tangent
116        void AutreTangent(Droite& , Plan& , int& indic)
117            { indic = 0;};
118
119        // ramene true si le dernier point M1 est dans l'element et que sa position precedente

```

```

120 // est situee du bon cote de la frontiere , sinon false
121 // la verif sur a est executer uniquement dans les cas suivant :
122 // dimension 3D et frontiere 2D
123 // dimension 2D et frontiere 1D
124 // ->> dimension 3D et frontiere 1D, pas de verif
125 // ->> autre cas ne doivent pas arriver normalement !!
126
127 // ici ramene vrai tjours
128 bool InSurf(const double& ) const
129 { return true; };
130
131 // affichage des infos de l'elements frontiere
132 void Affiche(Enum_dure temp ) const
133 { cout << "\n element frontiere de type FrontPointF , de noeuds sommets : \n ";
134   int nbn=1;
135   switch (temp)
136     {case TEMPS_tdt: for (int i =1;i<=nbn;i++)
137       cout <<" noe: " << tabNoeud(i)->Num_noeud() << " "
138         << tabNoeud(i)->Coord2() << ", " ; break;
139     case TEMPS_t : for (int i =1;i<=nbn;i++)
140       cout <<" noe: " << tabNoeud(i)->Num_noeud() << " "
141         << tabNoeud(i)->Coord1() << ", " ; break;
142     case TEMPS_0 : for (int i =1;i<=nbn;i++)
143       cout <<" noe: " << tabNoeud(i)->Num_noeud() << " "
144         << tabNoeud(i)->Coord0() << ", " ; break;
145     default: break;
146   };
147 };
148
149 // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
150 // si indic = 0 -> un point, indic = 1 -> une droite, =2 -> un plan
151 // ici il s'agit uniquement d'un point, donc pas de droite ou de plan tangent
152 // ramene eventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
153 // dans le cas d'une ligne en 3D ramene la variation du vecteur tangent: si var_normale = true,
sinon ramene NULL
154 Tableau <Coordonnee >* DernierTangent(Droite& , Plan& , int& indic,bool )
155 { indic = 0; return NULL;};
156
157 // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
158 // si le point est sur la surface, ramene false
159 // ramene true si hors matiere, sinon false
160 // le test sur a est executer uniquement dans les cas suivants :
161 // dimension 3D et frontiere 2D
162 // dimension 3D axi et frontiere 1D
163 // dimension 2D et frontiere 1D
164 // ->> dimension 3D et frontiere 1D, pas de verif
165 // ->> autre cas ne doivent pas arriver normalement !!
166 /// ----->> sans objet ici ramene par defaut false
167 // retour de r = distance du point à la surface, ligne
168 bool BonCote_t( const Coordonnee& ,double& r) const { return false;}; // cas ou on utilise la
frontiere a t
169 bool BonCote_tdt( const Coordonnee& ,double& r) const { return false;}; // cas ou on utilise la
frontiere a tdt
170
171 // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
172 const Vecteur& Phi() { return phi_M;};
173
174 // creation et ramene des pointeurs sur les frontieres de l'element frontiere
175 // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
176 // à moins qu'il soit effacé
177 Tableau <ElFrontiere*>& Frontiere() ;
178
179 // ramene la métrique associée à l'élément
180 // normalement ne doit pas servir donc message d'erreur
181 Met_abstraite * Metrique()
182 { cout << "\n erreur on ne doit pas calculer la métrique d'un point frontiere!!";
183   cout << "\n Met_abstraite * FrontPointFMetrique() " << endl;
184   Sortie (1);
185   Met_abstraite * toto = NULL;
186   return toto;
187 };
188
189 //----- fonction publique particulière à l'élément point -----
190 // fonction permettant de renseigner un vecteur normal au plan tangent au point
191 // ce vecteur, s'il existe est utilisé par les fonctions BonCote_t et BonCote_tdt
192 void Def_Normale(const Coordonnee& V) {if (T == NULL) {T = new Coordonnee(V);} else *T=V;};
193
194 //----- lecture écriture de restart -----
195 // ceci concerne uniquement les informations spécifiques
196 // dans le cas de l'utilisation de la frontiere pour la projection d'un point sur la frontiere
197 // il s'agit donc d'un cas particulier
198 void Lecture_base_info_ElFrontiere_pour_projection(ifstream& ent) ;
199 void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
200
201 private :
202 // VARIABLES PROTEGEES :
203 static GeomPoint point; // le point de reference

```

```

204     Coordonnee+ T; // le vecteur normal au plan tangent au point s'il existe
205     Vecteur phi_M;
206     // METHODES PROTEGEES :
207
208 };
209 /// @} // end of group
210
211 #endif

```

## 7.122 FrontQuadCC.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      10/11/01
32 *
33 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:     Herezh++
36 *
37 *
38 *      BUT:        Frontiere : Facette quadrangulaire cubique complet
39 *                  avec 9 points d'integration.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      !-----!-----!-----!-----!
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      !-----!-----!-----!-----!
52 *
53 *      *****/
54 #ifndef FRONTQUADCC_H
55 #define FRONTQUADCC_H
56
57 #include "ElFrontiere.h"
58 #include "GeomQuadrangle.h"
59 #include "Met_abstraite.h"
60
61 /// @addtogroup Les_Elements_de_frontiere
62 /// @{
63 ///
64
65
66 class FrontQuadCC : public ElFrontiere
67 {
68     public :
69         // CONSTRUCTEURS :
70         //par default
71         FrontQuadCC ();
72         // fonction du tableau des noeuds sommets
73         FrontQuadCC ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);

```



```

74     FrontQuadCC( const FrontQuadCC& a); // de copie
75     // DESTRUCTEUR :
76     ~FrontQuadCC ();
77     // surcharge de l'affectation
78     ElFrontiere& operator = ( const ElFrontiere& a);
79     // retourne le type de l'element frontiere
80     string TypeFrontiere() const ;
81     // retourne l'élément géométrique attaché à l'élément frontiere
82     ElemGeomC0 const & ElementGeometrique() const { return quadrangle;};
83     // creation d'un nouvelle element frontiere du type FrontQuadCC
84     ElFrontiere * NevezElemFront() const ;
85     // creation d'un nouvelle element frontiere du type FrontQuadCC
86     // avec des donnees differentes
87     ElFrontiere * NevezElemFront( const Tableau <Noeud *>& tab,
88         const DdlElement& ddlElem) const ;
89
90     // METHODES PUBLIQUES :
91
92     // ramene et calcul les coordonnees du point de reference de l'element
93     Coordonnee Ref();
94     // ramene un plan tangent au point de reference
95     // si indic = 2 -> un plan
96     // ces infos sont stocke et sauvegardees dans l'element
97     void TangentRef(Droite& dr, Plan& pl, int& indic);
98     // M est un point du dernier plan tangent sauvegarde dans l'element
99     // - calcul du point M1 correspondant sur la surface, M1 est stocke
100    // _ calcul et retour du plan tangent au point M1
101    // si indic = 2 -> un plan
102    // ces infos sont stocke et sauvegardees dans l'element
103    void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
104    // ramene un autre plan tangent genere de maniere pseudo aleatoire
105    // si indic = 2 -> un plan
106    // ces infos sont stocke et sauvegardees dans l'element
107    void AutreTangent(Droite& dr, Plan& pl, int& indic);
108    // ramene true si le dernier point M1 est dans l'element , sinon false
109    // le calcul est fait à eps relatif près
110    bool InSurf(const double& eps) const ;
111    // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
112    // si indic = 1 -> une droite, =2 -> un plan
113    // ramène éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
114    // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
115    Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
116    // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
117    // ramene true si hors matiere, sinon false
118    // le test sur a est executer uniquement dans les cas suivants :
119    // dimension 3D et frontiere 2D
120    // dimension 2D et frontiere 1D
121    // ->> dimension 3D et frontiere 1D, pas de verif
122    // ->> autre cas ne doivent pas arriver normalement !!
123    // retour de r = distance du point à la surface, ligne
124    bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
125    bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
126    // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
127    const Vecteur& Phi();
128
129    // affichage des infos de l'elements
130    void Affiche(Enum_dure temp = TEMPS_tdt) const ;
131
132    // creation et ramene des pointeurs sur les frontieres de l'element frontiere
133    // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
134    // à moins qu'il soit effacé
135    Tableau <ElFrontiere*>& Frontiere();
136
137    // ramène la métrique associée à l'élément
138    Met_abstraite * Metrique() {return met;};
139
140    //----- lecture écriture de restart -----
141    // ceci concerne uniquement les informations spécifiques
142    // dans le cas de l'utilisation de la frontiere pour la projection d'un point sur la frontiere
143    // il s'agit donc d'un cas particulier
144    void Lecture_base_info_ElFrontiere_pour_projection(istream& ent) ;
145    void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
146
147    private :
148    // VARIABLES PROTEGEES :
149    static GeomQuadrangle quadrangle; // le quadrangle de reference
150    static Met_abstraite * met; // une metrique associee
151    static Vecteur phi; // interpolation au point courant
152    static Mat_pleine dphi; // derivees de l'interpolation au point courant
153    Coordonnee ref; // Point de reference de l'element
154    Plan plan; // plan tangent
155    Tableau <Coordonnee > d_n, D_pasnormale; // variation du vecteur normal et variable de travail
156    static BaseB giB; // base naturelle de travail
157    static BaseH giH; // base duale de travail
158    Coordonnee theta; // coordonnees du point courant projete
159    Coordonnee theta_repere; // le point où l'on calcul le repère

```

```

160
161 // METHODES PROTEGEES :
162 // methodes propres a l'element
163 inline Plan PL() const { return plan; };
164 inline Coordonnee Theta() const { return theta;};
165
166 // definition de la metrique
167 void DefMetrique();
168 };
169 /// @} // end of group
170
171 #endif

```

## 7.123 FrontQuadLine.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           23/01/97
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *   ****
38 *   BUT:   Frontiere : Facette quadrangulaire lineaire   avec 4
39 *           points d'integration.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *   *****/
54 #ifndef FRONTQUADLINE_H
55 #define FRONTQUADLINE_H
56
57 #include "ElFrontiere.h"
58 #include "GeomQuadrangle.h"
59 #include "Met_abstraite.h"
60
61 /// @addtogroup Les_Elements_de_frontiere
62 /// @{}
63 ///
64
65
66 class FrontQuadLine : public ElFrontiere
67 {
68 public :
69     // CONSTRUCTEURS :

```

```

70 //par default
71 FrontQuadLine ();
72 // fonction du tableau des noeuds sommets
73 FrontQuadLine ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);
74 FrontQuadLine( const FrontQuadLine& a); // de copie
75 // DESTRUCTEUR :
76 ~FrontQuadLine ();
77 // surcharge de l'affectation
78 ElFrontiere& operator = ( const ElFrontiere& a);
79 // retourne le type de l'element frontiere
80 string TypeFrontiere() const ;
81 // retourne l'élément géométrique attaché à l'élément frontière
82 ElemGeomC0 const & ElementGeometrique() const { return quadrangle;};
83 // creation d'un nouvelle element frontiere du type FrontQuadLine
84 ElFrontiere * NevezElemFront() const ;
85 // creation d'un nouvelle element frontiere du type FrontQuadLine
86 // avec des donnees differentes
87 ElFrontiere * NevezElemFront( const Tableau <Noeud *>& tab, const DdlElement& ddlElem)const;
88
89 // METHODES PUBLIQUES :
90
91 // ramene les coordonnees du point de reference de l'element
92 Coordonnee Ref();
93 // ramene un plan tangent au point de reference
94 // si indic = 2 -> un plan
95 // ces infos sont stocke et sauvegardees dans l'element
96 void TangentRef(Droite& dr, Plan& pl, int& indic);
97 // M est un point du dernier plan tangent sauvegarde dans l'element
98 // - calcul du point M1 correspondant sur la surface, M1 est stocke
99 // _ calcul et retour du plan tangent au point M1
100 // si indic = 2 -> un plan
101 // ces infos sont stocke et sauvegardees dans l'element
102 void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
103 // ramene un autre plan tangent genere de maniere pseudo aleatoire
104 // si indic = 2 -> un plan
105 // ces infos sont stocke et sauvegardees dans l'element
106 void AutreTangent(Droite& dr, Plan& pl, int& indic);
107 // ramene true si le dernier point M1 est dans l'element , sinon false
108 // le calcul est fait à eps relatif près
109 bool InSurf(const double& eps) const ;
110 // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
111 // si indic = 1 -> une droite, =2 -> un plan
112 // ramène éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
113 // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
114 Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
115 // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
116 // ramene true si hors matiere, sinon false
117 // le test sur a est executer uniquement dans les cas suivants :
118 // dimension 3D et frontiere 2D
119 // dimension 2D et frontiere 1D
120 // ->> dimension 3D et frontiere 1D, pas de verif
121 // ->> autre cas ne doivent pas arriver normalement !!
122 // retour de r = distance du point à la surface, ligne
123 bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
124 bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
125 // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
126 const Vecteur& Phi() ;
127
128 // affichage des infos de l'elements
129 void Affiche(Enum_dure temp = TEMPS_tdt) const ;
130
131 // creation et ramene des pointeurs sur les frontieres de l'element frontiere
132 // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
133 // à moins qu'il soit effacé
134 Tableau <ElFrontiere*>& Frontiere();
135
136 // ramène la métrique associée à l'élément
137 Met_abstraite * Metrique() {return met;};
138
139 //----- lecture écriture de restart -----
140 // ceci concerne uniquement les informations spécifiques
141 // dans le cas de l'utilisation de la frontière pour la projection d'un point sur la frontière
142 // il s'agit donc d'un cas particulier
143 void Lecture_base_info_ElFrontiere_pour_projection(ifstream& ent) ;
144 void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
145
146 private :
147 // VARIABLES PROTEGEES :
148 static GeomQuadrangle quadrangle; // le quadrangle de reference
149 static Met_abstraite * met; // une metrique associee
150 static Vecteur phi; // interpolation au point courant
151 static Mat_pleine dphi; // derivees de l'interpolation au point courant
152 Coordonnee ref; // Point de reference de l'element
153 Plan plan; // plan tangent
154 Tableau <Coordonnee > d_N,D_pasnormale; // variation du vecteur normal et variable de travail
155 static BaseB giB; // base naturelle de travail

```

```

156     static BaseH giH; // base duale de travail
157     Coordonnee theta; // coordonnees du point courant projete
158     Coordonnee theta_repere; // le point où l'on calcul le repère
159
160     // METHODES PROTEGEES :
161     // methodes propres a l'element
162     inline Plan PL() const { return plan; };
163     inline Coordonnee Theta() const { return theta;};
164     // definition de la metrique
165     void DefMetrique();
166 };
167 /// @} // end of group
168
169 #endif

```

## 7.124 FrontQuadQC.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Frontiere : Facette quadrangulaire quadratique complet
39 *               avec 4 points d'integration.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      !-----!-----!-----!
47 *      !           !           !           !
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      !-----!-----!-----!
52 *      !           !           !           !
53 *****/
54 #ifndef FRONTQUADQC_H
55 #define FRONTQUADQC_H
56
57 #include "ElFrontiere.h"
58 #include "GeomQuadrangle.h"
59 #include "Met_abstraite.h"
60
61 /// @addtogroup Les_Elements_de_frontiere
62 /// @{
63 ///
64
65
66 class FrontQuadQC : public ElFrontiere
67 {

```

```

68 public :
69     // CONSTRUCTEURS :
70     //par défaut
71     FrontQuadQC ();
72     // fonction du tableau des noeuds sommets
73     FrontQuadQC ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);
74     FrontQuadQC( const FrontQuadQC& a); // de copie
75     // DESTRUCTEUR :
76     ~FrontQuadQC ();
77     // surcharge de l'affectation
78     ElFrontiere& operator = ( const ElFrontiere& a);
79     // retourne le type de l'element frontiere
80     string TypeFrontiere() const ;
81     // retourne l'élément géométrique attaché à l'élément frontiere
82     ElemGeomC0 const & ElementGeometrique() const { return quadrangle;};
83     // creation d'un nouvelle element frontiere du type FrontQuadQC
84     ElFrontiere * NevezElemFront () const ;
85     // creation d'un nouvelle element frontiere du type FrontQuadQC
86     // avec des donnees differentes
87     ElFrontiere * NevezElemFront( const Tableau <Noeud *>& tab,
88         const DdlElement& ddlElem) const ;
89
90     // METHODES PUBLIQUES :
91
92     // ramene et calcul les coordonnees du point de reference de l'element
93     Coordonnee Ref();
94     // ramene un plan tangent au point de reference
95     // si indic = 2 -> un plan
96     // ces infos sont stocke et sauvegardees dans l'element
97     void TangentRef(Droite& dr, Plan& pl, int& indic);
98     // M est un point du dernier plan tangent sauvegarde dans l'element
99     // - calcul du point M1 correspondant sur la surface, M1 est stocke
100    // _ calcul et retour du plan tangent au point M1
101    // si indic = 2 -> un plan
102    // ces infos sont stocke et sauvegardees dans l'element
103    void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
104    // ramene un autre plan tangent genere de maniere pseudo aleatoire
105    // si indic = 2 -> un plan
106    // ces infos sont stocke et sauvegardees dans l'element
107    void AutreTangent(Droite& dr, Plan& pl, int& indic);
108    // ramene true si le dernier point M1 est dans l'element , sinon false
109    // le calcul est fait à eps relatif près
110    bool InSurf(const double& eps) const ;
111    // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
112    // si indic = 1 -> une droite, =2 -> un plan
113    // ramène éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
114    // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
115    Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
116    // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
117    // ramene true si hors matiere, sinon false
118    // le test sur a est executer uniquement dans les cas suivants :
119    // dimension 3D et frontiere 2D
120    // dimension 2D et frontiere 1D
121    // ->> dimension 3D et frontiere 1D, pas de verif
122    // ->> autre cas ne doivent pas arriver normalement !!
123    // retour de r = distance du point à la surface, ligne
124    bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
125    bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
126    // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
127    const Vecteur& Phi();
128
129    // affichage des infos de l'elements
130    void Affiche(Enum_dure temp = TEMPS_tdt) const ;
131
132    // creation et ramene des pointeurs sur les frontieres de l'element frontiere
133    // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
134    // à moins qu'il soit effacé
135    Tableau <ElFrontiere*>& Frontiere();
136
137    // ramène la métrique associée à l'élément
138    Met_abstraite * Metrique() {return met;};
139
140    //----- lecture écriture de restart -----
141    // ceci concerne uniquement les informations spécifiques
142    // dans le cas de l'utilisation de la frontiere pour la projection d'un point sur la frontiere
143    // il s'agit donc d'un cas particulier
144    void Lecture_base_info_ElFrontiere_pour_projection(ifstream& ent) ;
145    void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
146
147 private :
148     // VARIABLES PROTEGEES :
149     static GeomQuadrangle quadrangle; // le quadrangle de reference
150     static Met_abstraite * met; // une metrique associee
151     static Vecteur phi; // interpolation au point courant
152     static Mat_pleine dphi; // derivees de l'interpolation au point courant
153     Coordonnee ref; // Point de reference de l'element

```

```

154 Plan plan; // plan tangent
155 Tableau <Coordonnee > d_N,D_pasnormale; // variation du vecteur normal et variable de travail
156 static BaseB qiB; // base naturelle de travail
157 static BaseH qiH; // base duale de travail
158 Coordonnee theta; // coordonnees du point courant projete
159 Coordonnee theta_repere; // le point où l'on calcul le repère
160
161 // METHODES PROTEGEES :
162 // methodes propres a l'element
163 inline Plan PL() const { return plan; };
164 inline Coordonnee Theta() const { return theta;};
165 // definition de la metrique
166 void DefMetrique();
167 };
168 /// @} // end of group
169
170 #endif

```

## 7.125 FrontQuadQuad.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Frontiere : Facette quadrangulaire quadratique incomplete *
39 *              avec 4 points d'integration.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      !           !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      !           !           !
52 *
53 *****/
54 #ifndef FRONTQUADQUAD_H
55 #define FRONTQUADQUAD_H
56
57 #include "ElFrontiere.h"
58 #include "GeomQuadrangle.h"
59 #include "Met_abstraite.h"
60
61 /// @addtogroup Les_Elements_de_frontiere
62 /// @{
63 ///
64

```

```

65
66 class FrontQuadQuad : public ElFrontiere
67 {
68 public :
69     // CONSTRUCTEURS :
70     //par default
71     FrontQuadQuad ();
72     // fonction du tableau des noeuds sommets
73     FrontQuadQuad ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);
74     FrontQuadQuad( const FrontQuadQuad& a); // de copie
75     // DESTRUCTEUR :
76     ~FrontQuadQuad ();
77     // surcharge de l'affectation
78     ElFrontiere& operator = ( const ElFrontiere& a);
79     // retourne le type de l'element frontiere
80     string TypeFrontiere() const ;
81     // retourne l'élément géométrique attaché à l'élément frontiere
82     ElemGeomC0 const & ElementGeometrique() const { return quadrangle;};
83     // creation d'un nouvelle element frontiere du type FrontQuadQuad
84     ElFrontiere * NevezElemFront() const ;
85     // creation d'un nouvelle element frontiere du type FrontQuadQuad
86     // avec des donnees differentes
87     ElFrontiere * NevezElemFront
88         ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem) const ;
89
90     // METHODES PUBLIQUES :
91
92     // ramene et calcul les coordonnees du point de reference de l'element
93     Coordonnee Ref();
94     // ramene un plan tangent au point de reference
95     // si indic = 2 -> un plan
96     // ces infos sont stocke et sauvegardees dans l'element
97     void TangentRef(Droite& dr, Plan& pl, int& indic);
98     // M est un point du dernier plan tangent sauvegarde dans l'element
99     // - calcul du point M1 correspondant sur la surface, M1 est stocke
100    // _ calcul et retour du plan tangent au point M1
101    // si indic = 2 -> un plan
102    // ces infos sont stocke et sauvegardees dans l'element
103    void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
104    // ramene un autre plan tangent genere de maniere pseudo aleatoire
105    // si indic = 2 -> un plan
106    // ces infos sont stocke et sauvegardees dans l'element
107    void AutreTangent(Droite& dr, Plan& pl, int& indic);
108    // ramene true si le dernier point M1 est dans l'element , sinon false
109    // le calcul est fait à eps relatif près
110    bool InSurf(const double& eps) const ;
111    // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
112    // si indic = 1 -> une droite, =2 -> un plan
113    // ramène éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
114    // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
115    Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
116    // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
117    // ramene true si hors matiere, sinon false
118    // le test sur a est executer uniquement dans les cas suivants :
119    // dimension 3D et frontiere 2D
120    // dimension 2D et frontiere 1D
121    // ->> dimension 3D et frontiere 1D, pas de verif
122    // ->> autre cas ne doivent pas arriver normalement !!
123    // retour de r = distance du point à la surface, ligne
124    bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
125    bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
126    // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
127    const Vecteur& Phi();
128
129    // affichage des infos de l'elements
130    void Affiche(Enum_dure temp = TEMPS_tdt) const ;
131
132    // creation et ramene des pointeurs sur les frontieres de l'element frontiere
133    // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
134    // à moins qu'il soit effacé
135    Tableau <ElFrontiere*>& Frontiere();
136
137    // ramène la métrique associée à l'élément
138    Met_abstraite * Metrique() {return met;};
139
140    //----- lecture écriture de restart -----
141    // ceci concerne uniquement les informations spécifiques
142    // dans le cas de l'utilisation de la frontiere pour la projection d'un point sur la frontiere
143    // il s'agit donc d'un cas particulier
144    void Lecture_base_info_ElFrontiere_pour_projection(istream& ent) ;
145    void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
146
147 private :
148     // VARIABLES PROTEGEES :
149     static GeomQuadrangle quadrangle; // le quadrangle de reference
150     static Met_abstraite * met; // une metrique associee

```

```

151     static Vecteur phi; // interpolation au point courant
152     static Mat_pleine dphi; // derivees de l'interpolation au point courant
153     Coordonnee ref; // Point de reference de l'element
154     Plan plan; // plan tangent
155     Tableau <Coordonnee > d_N,D_pasnormale; // variation du vecteur normal et variable de travail
156     static BaseB giB; // base naturelle de travail
157     static BaseH giH; // base duale de travail
158     Coordonnee theta; // coordonnees du point courant projete
159     Coordonnee theta_repere; // le point où l'on calcul le repère
160
161     // METHODES PROTEGEES :
162     // methodes propres a l'element
163     inline Plan PL() const { return plan; };
164     inline Coordonnee Theta() const { return theta;};
165     // definition de la metrique
166     void DefMetrique();
167 };
168 /// @} // end of group
169
170 #endif

```

## 7.126 FrontTriaLine.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Frontiere : Facette triangulaire lineaire avec 1 points
39 *              d'integration.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      !-----!-----!-----!
47 *      !           !           !           !
48 *      !           !           !           !
49 *      *****
50 *      MODIFICATIONS:
51 *
52 *      ! date !   auteur !           but
53 *      !-----!-----!-----!
54 *      !           !           !           !
55 *      !           !           !           !
56 *      *****/
57 #ifndef FRONTTRIALINE_H
58 #define FRONTTRIALINE_H
59
60 #include "ElFrontiere.h"
61 #include "GeomTriangle.h"
62
63 /// @addtogroup Les_Elements_de_frontiere
64 /// @{}

```



```

62 ///
63
64
65 class FrontTriaLine : public ElFrontiere
66 {
67 public :
68 // CONSTRUCTEURS :
69 //par default
70 FrontTriaLine ();
71 // fonction du tableau des noeuds sommets
72 FrontTriaLine ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);
73 FrontTriaLine( const FrontTriaLine& a); // de copie
74 // DESTRUCTEUR :
75 ~FrontTriaLine ();
76 // surcharge de l'affectation
77 ElFrontiere& operator = ( const ElFrontiere& a);
78 // retourne le type de l'element frontiere
79 string TypeFrontiere() const ;
80 // retourne l'élément géométrique attaché à l'élément frontiere
81 ElemGeomC0 const & ElementGeometrique() const { return triangle;};
82 // creation d'une nouvelle element frontiere du type FrontTriaLine
83 ElFrontiere * NevezElemFront() const ;
84 // creation d'une nouvelle element frontiere du type FrontTriaLine
85 // avec des donnees differentes
86 ElFrontiere * NevezElemFront
87     ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem) const ;
88
89 // ramene et calcul les coordonnees du point de reference de l'element
90 Coordonnee Ref();
91 // ramene un plan tangent au point de reference
92 // si indic = 2 -> un plan
93 // ces infos sont stocke et sauvegardees dans l'element
94 void TangentRef(Droite& dr, Plan& pl, int& indic);
95 // M est un point du dernier plan tangent sauvegarde dans l'element
96 // - calcul du point M1 correspondant sur la surface, M1 est stocke
97 // _ calcul et retour du plan tangent au point M1
98 // si indic = 2 -> un plan
99 // ces infos sont stocke et sauvegardees dans l'element
100 void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
101 // ramene un autre plan tangent genere de maniere pseudo aleatoire
102 // si indic = 2 -> un plan
103 // ces infos sont stocke et sauvegardees dans l'element
104 void AutreTangent(Droite& dr, Plan& pl, int& indic);
105 // ramene true si le dernier point M1 est dans l'element , sinon false
106 // le calcul est fait à eps relatif près
107 bool InSurf(const double& eps) const ;
108 // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
109 // si indic = 1 -> une droite, =2 -> un plan
110 // ramène éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
111 // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
112 Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
113 // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
114 // ramene true si hors matiere, sinon false
115 // le test sur a est executer uniquement dans les cas suivants :
116 // dimension 3D et frontiere 2D
117 // dimension 2D et frontiere 1D
118 // ->> dimension 3D et frontiere 1D, pas de verif
119 // ->> autre cas ne doivent pas arriver normalement !!
120 // retour de r = distance du point à la surface, ligne
121 bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
122 bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
123 // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
124 const Vecteur& Phi();
125
126 // affichage des infos de l'elements
127 void Affiche(Enum_dure temp = TEMPS_tdt) const ;
128
129 // creation et ramene des pointeurs sur les frontieres de l'element frontiere
130 // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
131 // à moins qu'il soit effacé
132 Tableau <ElFrontiere*>& Frontiere();
133
134 // ramène la métrique associée à l'élément
135 Met_abstraite * Metrique() {return met;};
136
137 //----- lecture écriture de restart -----
138 // ceci concerne uniquement les informations spécifiques
139 // dans le cas de l'utilisation de la frontiere pour la projection d'un point sur la frontiere
140 // il s'agit donc d'un cas particulier
141 void Lecture_base_info_ElFrontiere_pour_projection(istream& ent);
142 void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort);
143
144 private :
145 // VARIABLES PROTEGEES :
146 static GeomTriangle triangle; // le triangle de reference
147 static Met_abstraite * met; // une metrique associee

```

```

148     static Vecteur phi; // interpolation au point courant
149     static Mat_pleine dphi; // derivees de l'interpolation au point courant
150     Coordonnee ref; // Point de reference de l'element
151     Plan plan; // plan tangent ici c'est l'element
152     Tableau <Coordonnee > d_N,D_pasnormale; // variation du vecteur normal et variable de travail
153     static BaseB giB; // base naturelle de travail
154     static BaseH giH; // base duale de travail
155     Coordonnee theta; // coordonnees du point courant projete
156     Coordonnee theta_repere; // le point où l'on calcul le repère
157
158     // METHODES PROTEGEES :
159     // methodes propres a l'element
160     inline Plan PL() const { return plan; };
161     inline Coordonnee Theta() const { return theta;};
162     // definition de la metrique
163     void DefMetrique();
164 };
165 /// @} // end of group
166
167 #endif

```

## 7.127 FrontTriaQuad.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          23/01/97
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *
38 *      BUT:           Frontiere : Facette triangulaire quadratique avec 3 points
39 *                   d'integration.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      !           !           !
47 *      !           !           !
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      !           !           !
52 *      !           !           !
53 *      *****/
54 #ifndef FRONTTRIAQUAD_H
55 #define FRONTTRIAQUAD_H
56
57 #include "ElFrontiere.h"
58 #include "GeomTriangle.h"
59 #include "Met_abstraite.h"
60
61 /// @addtogroup Les_Elements_de_frontiere

```

```

62 /// @{
63 ///
64
65
66 class FrontTriaQuad : public ElFrontiere
67 {
68 public :
69     // CONSTRUCTEURS :
70     //par défaut
71     FrontTriaQuad ();
72     // fonction du tableau des noeuds sommets
73     FrontTriaQuad ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem);
74     FrontTriaQuad( const FrontTriaQuad& a); // de copie
75     // DESTRUCTEUR :
76     ~FrontTriaQuad ();
77     // surcharge de l'affectation
78     ElFrontiere& operator = ( const ElFrontiere& a);
79     // retourne le type de l'element frontiere
80     string TypeFrontiere() const ;
81     // retourne l'élément géométrique attaché à l'élément frontiere
82     ElemGeomC0 const & ElementGeometrique() const { return triangle;};
83     // creation d'un nouvelle element frontiere du type FrontTriaQuad
84     ElFrontiere * NevezElemFront() const ;
85     // creation d'un nouvelle element frontiere du type FrontTriaQuad
86     // avec des donnees differentes
87     ElFrontiere * NevezElemFront
88         ( const Tableau <Noeud *>& tab, const DdlElement& ddlElem) const ;
89
90     // ramene et calcul les coordonnees du point de reference de l'element
91     Coordonnee Ref();
92     // ramene un plan tangent au point de reference
93     // si indic = 2 -> un plan
94     // ces infos sont stocke et sauvegardees dans l'element
95     void TangentRef(Droite& dr, Plan& pl, int& indic);
96     // M est un point du dernier plan tangent sauvegarde dans l'element
97     // - calcul du point M1 correspondant sur la surface, M1 est stocke
98     // _ calcul et retour du plan tangent au point M1
99     // si indic = 2 -> un plan
100    // ces infos sont stocke et sauvegardees dans l'element
101    void Tangent(const Coordonnee& M, Coordonnee& M1, Droite& dr, Plan& pl, int& indic);
102    // ramene un autre plan tangent genere de maniere pseudo aleatoire
103    // si indic = 2 -> un plan
104    // ces infos sont stocke et sauvegardees dans l'element
105    void AutreTangent(Droite& dr, Plan& pl, int& indic);
106    // ramene true si le dernier point M1 est dans l'element , sinon false
107    // le calcul est fait à eps relatif près
108    bool InSurf(const double& eps) const ;
109    // actualise et ramene le dernier plan tangent (ou droite tangente) calcule
110    // si indic = 1 -> une droite, =2 -> un plan
111    // ramène éventuellement la variation du vecteur normale pour un plan en 3D ou une ligne en 2D
112    // dans le cas d'une ligne en 3D ramène la variation du vecteur tangent: si var_normale = true,
    sinon ramène NULL
113    Tableau <Coordonnee >* DernierTangent(Droite& dr, Plan& pl, int& indic, bool avec_var=false);
114    // test si la position d'un point est du bon cote ( c-a-d hors matiere) ou non
115    // ramene true si hors matiere, sinon false
116    // le test sur a est executer uniquement dans les cas suivants :
117    // dimension 3D et frontiere 2D
118    // dimension 2D et frontiere 1D
119    // ->> dimension 3D et frontiere 1D, pas de verif
120    // ->> autre cas ne doivent pas arriver normalement !!
121    // retour de r = distance du point à la surface, ligne
122    bool BonCote_t( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a t
123    bool BonCote_tdt( const Coordonnee& a, double& r) const ; // cas ou on utilise la frontiere a tdt
124    // calcul les fonctions d'interpolation au dernier point de projection sauvegarde
125    const Vecteur& Phi();
126
127    // affichage des infos de l'elements
128    void Affiche(Enum_dure temp = TEMPS_tdt) const ;
129
130    // creation et ramene des pointeurs sur les frontieres de l'element frontiere
131    // au premier appel il y a construction, ensuite on ne fait que ramener le tableau
132    // à moins qu'il soit effacé
133    Tableau <ElFrontiere*>& Frontiere();
134
135    // ramène la métrique associée à l'élément
136    Met_abstraite * Metrique() {return met;};
137
138    //----- lecture écriture de restart -----
139    // ceci concerne uniquement les informations spécifiques
140    // dans le cas de l'utilisation de la frontiere pour la projection d'un point sur la frontiere
141    // il s'agit donc d'un cas particulier
142    void Lecture_base_info_ElFrontiere_pour_projection(ifstream& ent) ;
143    void Ecriture_base_info_ElFrontiere_pour_projection(ofstream& sort) ;
144
145 private :
146     // VARIABLES PROTEGEES :
147     static GeomTriangle triangle; // le triangle de reference

```

```

148     static Met_abstraite * met; // une metrique associee
149     static Vecteur phi; // interpolation au point courant
150     static Mat_pleine dphi; // derivees de l'interpolation au point courant
151     Coordonnee ref; // Point de reference de l'element
152     Plan plan; // plan tangent
153     Tableau <Coordonnee > d_N,D_pasnormale; // variation du vecteur normal et variable de travail
154     static BaseB giB; // base naturelle de travail
155     static BaseH giH; // base duale de travail
156     Coordonnee theta; // coordonnees du point courant projete
157     Coordonnee theta_repere; // le point où l'on calcul le repère
158
159     // METHODES PROTEGEES :
160     // methodes propres a l'element
161     inline Plan PL() const { return plan; };
162     inline Coordonnee Theta() const { return theta; };
163     // definition de la metrique
164     void DefMetrrique();
165 };
166 /// @} // end of group
167
168 #endif

```

## 7.128 Biel\_axi.h

```

1 // FICHER : Biel_axi.h
2 // CLASSE : Biel_axi
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      BUT: // La classe Biel_axi permet de declarer des elements
41 *      biellettes axisymétrique et de realiser le calcul du residu local
42 *      et de la raideur locale pour une loi de comportement donnee.
43 *      La dimension de l'espace pour un tel element est 1.
44 *
45 *      *****
46 *
47 *      VERIFICATION:
48 *
49 *      ! date !   auteur !           but
50 *      -----
51 *      !           !           !
52 *
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date !   auteur !           but
56 *      -----
57 *
58 *      *****/
59 // -----classe pour un calcul de mecanique-----
60
61

```

```

62
63 #ifndef BIEL_AXI_H
64 #define BIEL_AXI_H
65
66 #include "ParaGlob.h"
67 #include "ElemMeca.h"
68 // #include "Loi_comp_abstraite.h"
69 #include "Met_abstraite.h"
70 #include "Met_biellette.h"
71 #include "MetAxisymetrique2D.h"
72 #include "Noeud.h"
73 #include "UtilLecture.h"
74 #include "Tenseur.h"
75 #include "NevezTenseur.h"
76 #include "Deformation.h"
77 #include "ElFrontiere.h"
78 #include "GeomSeg.h"
79 #include "GeomPoint.h"
80 #include "ParaAlgoControle.h"
81 #include "FrontSegLine.h"
82 #include "Epai.h"
83
84 class ConstrucElementbiel;
85
86 /// @addtogroup groupe_des_elements_finis
87 /// @{
88 ///
89
90
91 class Biel_axi : public ElemMeca
92 {
93
94     public :
95
96         // CONSTRUCTEURS :
97         // Constructeur par défaut
98         Biel_axi ();
99
100        // Constructeur fonction d'une epaisseur et eventuellement d'un numero
101        // d'identification et de maillage
102        Biel_axi (double epai,int num_maill=0,int num_id=-3);
103
104        // Constructeur fonction d'un numero de maillage et d'identification
105        Biel_axi (int num_maill,int num_id);
106
107        // Constructeur fonction d'une epaisseur, d'un numero de maillage et d'identification,
108        // du tableau de connexite des noeuds
109        Biel_axi (double epai,int num_maill,int num_id,const Tableau<Noeud *>& tab);
110
111        // Constructeur de copie
112        Biel_axi (const Biel_axi& biel);
113
114
115        // DESTRUCTEUR :
116        ~Biel_axi ();
117
118        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
119        // méthode virtuelle
120        Element* Nevez_copie() const { Element * el= new Biel_axi(*this); return el;};
121
122        // Surcharge de l'operateur = : realise l'egalite entre deux instances de Biel_axi
123        Biel_axi& operator= (Biel_axi& biel);
124
125        // METHODES :
126        // 1) derivant des virtuelles pures
127        // Lecture des donnees de la classe sur fichier
128        void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
129
130        // Calcul du residu local et de la raideur locale,
131        // pour le schema implicite
132        Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
133
134        // Calcul du residu local a t
135        // pour le schema explicit par exemple
136        Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
137        { return Biel_axi::CalculResidu(false,pa);};
138
139        // Calcul du residu local a tdt
140        // pour le schema explicit par exemple
141        Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
142        { return Biel_axi::CalculResidu(true,pa);};
143
144        // Calcul de la matrice masse pour l'élément
145        Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
146
147        // ----- calcul dynamique -----
148        // calcul de la longueur d'arrête de l'élément minimal

```

```

149 // divisé par la célérité la plus rapide dans le matériau
150 double Long_arrete_mini_sur_c(Enum_dure temps)
151 { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
152
153 //----- calcul d'erreur, remontée des contraintes -----
154 // 1)calcul du résidu et de la matrice de raideur pour le calcul d'erreur
155 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
156 // 2) remontée aux erreurs aux noeuds
157 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
158
159 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
160 // ce tableau et specifique a l'element
161 const DdlElement & TableauDdl() const ;
162
163
164 // Libere la place occupee par le residu et eventuellement la raideur
165 // par l'appel de Libere de la classe mere et libere les differents tenseurs
166 // intermediaires cree pour le calcul et les grandeurs pointee
167 // de la raideur et du residu
168 void Libere ();
169
170 // acquisition d'une loi de comportement
171 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
172
173 // test si l'element est complet
174 // = 1 tout est ok, =0 element incomplet
175 int TestComplet();
176
177 // procedure permettant de completer l'element apres
178 // sa creation avec les donnees du bloc transmis
179 // peut etre appeler plusieurs fois
180 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
181 // Compléter pour la mise en place de la gestion de l'hourglass
182 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
183
184 // ramene l'element geometrique
185 ElemGeomC0 & ElementGeometrique() const { return doCo->segment;};
186 // ramene l'element geometrique en constant
187 const ElemGeomC0 & ElementGeometrique_const() const { return doCo->segment;};
188
189 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
190 // associé
191 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
192 // 1) cas où l'on utilise la place passée en argument
193 Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
194 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
195 void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
196
197 // -- connaissances particulières sur l'élément
198 // ramène l'épaisseur de l'élément
199 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
200 virtual double Epaisseurs(Enum_dure enu , const Coordonnee& ) {return H(enu);};
201 // ramène l'épaisseur moyenne de l'élément (indépendante du point)
202 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
203 virtual double EpaisseurMoyenne(Enum_dure enu ) {return H(enu);};
204
205 // affichage dans la sortie transmise, des variables duales "nom"
206 // dans le cas ou nom est vide, affichage de "toute" les variables
207 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
208
209 // affichage d'info en fonction de ordre
210 // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
211 void Info_com_Element(UtilLecture * entreePrinc,string& ordre,Tableau<Noeud * > * tabMaillageNoeud)
212 { return Element::Info_com_El(2,entreePrinc,ordre,tabMaillageNoeud);};
213
214 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
215 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
216 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
217 // temps: dit si c'est à 0 ou t ou tdt
218 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
219 { return PtLePlusPres(temps,enu,M);};
220
221 // recuperation des coordonnées du point de numéro d'ordre iteg pour
222 // la grandeur enu
223 // temps: dit si c'est à 0 ou t ou tdt
224 // si erreur retourne erreur à true
225 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
226 { return CoordPtInt(temps,enu,iteg,erreur);};
227
228 // récupération des valeurs au numéro d'ordre = iteg pour
229 // les grandeur enu
230 Tableau <double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
231 enu,int iteg) ;
232 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu

```

```

233 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
234 // de conteneurs quelconque associée
235 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int
iteg);
236
237 // ramene vrai si la surface numéro ns existe pour l'élément
238 // dans le cas de la biellette il n'y a pas de surface
239 bool SurfExiste(int ) const
240 { return false;};
241
242 // ramene vrai si l'arête numéro na existe pour l'élément
243 bool AreteExiste(int na) const {if (na==1) return true; else return false;};
244
245 //===== lecture écriture dans base info =====
246
247 // cas donne le niveau de la récupération
248 // = 1 : on récupère tout
249 // = 2 : on récupère uniquement les données variables (supposées comme telles)
250 void Lecture_base_info
251 (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
252 // cas donne le niveau de sauvegarde
253 // = 1 : on sauvegarde tout
254 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
255 void Ecriture_base_info(ofstream& sort,const int cas) ;
256
257 // METHODES VIRTUELLES:
258 // ----- calculs utiles dans le cadre de la recherche du flambement linéaire
259 // Calcul de la matrice géométrique et initiale
260 ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
261
262 // retourne la liste des données particulières actuellement utilisés
263 // par l'élément (actif ou non), sont exclu de cette liste les données particulières des noeuds
264 // reliés à l'élément
265 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
266 List_io <TypeQuelconque> Les_types_particuliers_internes(bool absolue) const;
267
268 // récupération de grandeurs particulières au numéro d'ordre = iteg
269 // celles-ci peuvent être quelconques
270 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
271 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
272 void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);
273
274 // inactive les ddl du problème primaire de mécanique
275 inline void Inactive_ddl_primaire()
276 {ElemMeca::Inact_ddl_primaire(doCo->tab_ddl);};
277 // active les ddl du problème primaire de mécanique
278 inline void Active_ddl_primaire()
279 {ElemMeca::Act_ddl_primaire(doCo->tab_ddl);};
280 // ajout des ddl de contraintes pour les noeuds de l'élément
281 inline void Plus_ddl_Sigma()
282 {ElemMeca::Ad_ddl_Sigma(doCo->tab_ddlErr);};
283 // inactive les ddl du problème de recherche d'erreur : les contraintes
284 inline void Inactive_ddl_Sigma()
285 {ElemMeca::Inact_ddl_Sigma(doCo->tab_ddlErr);};
286 // active les ddl du problème de recherche d'erreur : les contraintes
287 inline void Active_ddl_Sigma()
288 {ElemMeca::Act_ddl_Sigma(doCo->tab_ddlErr);};
289 // active le premier ddl du problème de recherche d'erreur : SIGMA11
290 inline void Active_premier_ddl_Sigma()
291 {ElemMeca::Act_premier_ddl_Sigma();};
292
293 // lecture de données diverses sur le flot d'entrée
294 void LectureContraintes(UtilLecture * entreePrinc)
295 {if (unefois.CalResPrem_t == 1)
296 ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
297 else
298 { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
299 unefois.CalResPrem_t = 1;
300 }
301 };
302
303 // retour des contraintes en absolu retour true si elle existe sinon false
304 bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
305 { if (unefois.CalResPrem_t == 1)
306 ElemMeca::ContraintesEnAbsolues (false,lesPtMecaInt.TabSigHH_t(),tabSig);
307 else
308 { unefois.CalResPrem_t = 1;
309 ElemMeca::ContraintesEnAbsolues (true,lesPtMecaInt.TabSigHH_t(),tabSig);
310 };
311 return true;
312 };
313
314 // 2) derivant des virtuelles
315
316

```

```

317 // retourne un tableau de ddl element, correspondant à la
318 // composante de sigma -> SIG11, pour chaque noeud qui contient
319 // des ddl de contrainte
320 // -> utilisé pour l'assemblage de la raideur d'erreur
321 inline DdlElement& Tableau_de_Sig1() const
322 {return doCo->tab_Err1Sig1;} ;
323
324 // actualisation des ddl et des grandeurs actives de t+dt vers t
325 void TdtversT();
326 // actualisation des ddl et des grandeurs actives de t vers tdt
327 void TversTdt();
328
329 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
330 // qu'une fois la remontée aux contraintes effectuées sinon aucune
331 // action. En retour la valeur de l'erreur sur l'élément
332 // type indique le type de calcul d'erreur :
333 void ErreurElement(int type,double& errElemRelative
334 ,double& numerateur, double& denominateur);
335
336 // mise à jour de la boîte d'encombrement de l'élément, suivant les axes I_a globales
337 // en retour coordonnées du point mini dans retour.Premier() et du point maxi dans .Second()
338 // la méthode est différente de la méthode générale car il faut prendre en compte l'épaisseur de
l'élément
339 virtual const DeuxCoordonnees& Boite_encombre_element(Enum_dure temps);
340
341 // calcul des seconds membres suivant les chargements
342 // cas d'un chargement volumique,
343 // force indique la force volumique appliquée
344 // retourne le second membre résultant
345 // ici on l'épaisseur de l'élément pour constituer le volume
346 // -> explicite à t
347 Vecteur SM_charge_volumique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,const
ParaAlgoControle & pa,bool sur_volume_finale_)
348 { return Biel_axi::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_)} ;
349 // -> explicite à tdt
350 Vecteur SM_charge_volumique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,const
ParaAlgoControle & pa,bool sur_volume_finale_)
351 { return Biel_axi::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_)} ;
352 // -> implicite,
353 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
354 // retourne le second membre et la matrice de raideur correspondant
355 ResRaid SMR_charge_volumique_I(const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle
& pa,bool sur_volume_finale_) ;
356
357 // cas d'un chargement surfacique, sur les frontières des éléments
358 // force indique la force surfacique appliquée
359 // numface indique le numéro de la face chargée
360 // retourne le second membre résultant
361 // -> version explicite à t
362 Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
363 { return Biel_axi::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa)} ;
364 // -> version explicite à tdt
365 Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
366 { return Biel_axi::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa)} ;
367 // -> implicite,
368 // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
369 // retourne le second membre et la matrice de raideur correspondant
370 ResRaid SMR_charge_surfacique_I
371 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const ParaAlgoControle & pa) ;
372
373 // cas d'un chargement de type pression, sur les frontières des éléments
374 // pression indique la pression appliquée
375 // numface indique le numéro de la face chargée
376 // retourne le second membre résultant
377 // -> explicite à t
378 Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
379 { return Biel_axi::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa)} ;
380 // -> explicite à tdt
381 Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
382 { return Biel_axi::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa)} ;
383 // -> implicite,
384 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
385 // retourne le second membre et la matrice de raideur correspondant
386 ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
387
388 // cas d'un chargement surfacique hydrostatique,
389 // poidvol: indique le poids volumique du liquide
390 // M_liquide : un point de la surface libre
391 // dir_normal_liquide : direction normale à la surface libre
392 // retourne le second membre résultant
393 // -> explicite à t
394 Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol

```



```

395                                     ,int numFace,const Coordonnee& M_liquide
396                                     ,const ParaAlgoControle & pa
397                                     ,bool sans_limitation)
398     { return
Biel_axi::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation));};
399     // -> explicite à tdt
400     Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
401                                           ,int numFace,const Coordonnee& M_liquide
402                                           ,const ParaAlgoControle & pa
403                                           ,bool sans_limitation)
404     { return
Biel_axi::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation));};
405     // -> implicite,
406     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
407     // retourne le second membre et la matrice de raideur correspondant
408     ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
409                                         ,int numFace,const Coordonnee& M_liquide
410                                         ,const ParaAlgoControle & pa
411                                         ,bool sans_limitation) ;
412
413     // cas d'un chargement lineique, sur les aretes frontieres des éléments
414     // force indique la force lineique appliquée
415     // numarete indique le numéro de l'arete chargée
416     // retourne le second membre résultant
417     // -> explicite à t
418     Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
419     { return Biel_axi::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa);};
420     // -> explicite à tdt
421     Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
422     { return Biel_axi::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa);};
423     // -> implicite,
424     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
425     // retourne le second membre et la matrice de raideur correspondant
426     ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
427
428     // cas d'un chargement lineique suiveuse, sur l'arrête frontiere de
429     // la bielle (2D uniquement)
430     // force indique la force lineique appliquée
431     // numarete indique le numéro de l'arete chargée
432     // retourne le second membre résultant
433     // -> explicite à t
434     Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
435     { return Biel_axi::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa);};
436     // -> explicite à tdt
437     Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
438     { return Biel_axi::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa);};
439     // -> implicite,
440     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
441     // retourne le second membre et la matrice de raideur correspondant
442     ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa) ;
443
444     // cas d'un chargement surfacique hydro-dynamique,
445     // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
446     // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc unitaire)
447     // une suivant la direction normale à la vitesse de type portance
448     // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
449     // une suivant la vitesse tangente de type frottement visqueux
450     // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
451     // coef_mul: est un coefficient multiplicateur global (de tout)
452     // retourne le second membre résultant
453     // -> explicite à t
454     Vecteur SM_charge_hydrodynamique_E_t( CourbeID* frot_fluid,const double& poidvol
455                                           , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
456                                           , CourbeID* coef_aero_t,const ParaAlgoControle &
pa)
457     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa));};
458     // -> explicite à tdt
459     Vecteur SM_charge_hydrodynamique_E_tdt( CourbeID* frot_fluid,const double& poidvol
460                                           , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
461                                           , CourbeID* coef_aero_t,const ParaAlgoControle &
pa)
462     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa));};
463     // -> implicite,
464     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
465     // retourne le second membre et la matrice de raideur correspondant

```

```

466     ResRaid SMR_charge_hydrodynamique_I( CourbelD* frot_fluid,const double& poidvol
467                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
468                                     , CourbelD* coef_aero_t,const ParaAlgoControle &
pa) ;
469
470
471 // ===== définition et/ou construction des frontières =====
472
473 // Calcul des frontieres de l'element
474 // creation des elements frontieres et retour du tableau de ces elements
475 // la création n'a lieu qu'au premier appel
476 // ou lorsque l'on force le paramètre force a true
477 // dans ce dernier cas seul les frontière effacées sont recréée
478 Tableau <ElFrontiere*> const & Frontiere(bool force = false);
479
480 // ramene l'épaisseur
481 inline double H(Enum_dure enu = TEMPS_tdt )
482 { switch (enu)
483   { case TEMPS_0: return donnee_specif.epais.epaisseur0; break;
484     case TEMPS_t: return donnee_specif.epais.epaisseur_t; break;
485     case TEMPS_tdt: return donnee_specif.epais.epaisseur_tdt; break;
486   };
487   return 0.; // cas n'arrivant normalement jamais
488 };
489
490 // ajout du tableau specific de ddl des noeuds de la biellette
491 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
492 // des noeuds constituants l'element
493 void ConstTabDdl();
494 protected:
495
496 // ==== »» methodes virtuelles dérivant d'ElemMeca =====
497 // ramene la dimension des tenseurs contraintes et déformations de l'élément
498 int Dim_sig_eps() const {return 2;};
499
500 // ----- calcul de frontières en protected -----
501
502 // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
particulière à l'élément
503 // adressage des frontières linéiques et surfacique
504 // définit dans les classes dérivées, et utilisées pour la construction des frontières
505 virtual ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
506 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
507 virtual ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
508 {return NULL;}; // il n'y a pas de surface possible
509
510 private :
511
512 // VARIABLES PRIVEES :
513
514
515
516 class DonneeCommune
517 { public :
518   DonneeCommune (GeomSeg& seg,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
519                 MetAxisymetrique2D& met_bie,
520                 Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
521                 GeomSeg& seEr,Vecteur& residu_int,Mat_pleine& raideur_int,
522                 Tableau <Vecteur* > & residus_extN,Tableau <Mat_pleine* >& raideurs_extN,
523                 Tableau <Vecteur* > & residus_extA,Tableau <Mat_pleine* >& raideurs_extA,
524                 Mat_pleine& mat_masse ,GeomSeg& seMa,int nbi,GeomSeg* segHourg
525                 ) ;
526   DonneeCommune(DonneeCommune& a);
527   ~DonneeCommune();
528   // variables
529   GeomSeg segment ; // element geometrique correspondant
530   DdlElement tab_ddl; // tableau des degres
531   //de liberte des noeuds de l'element commun a tous les
532   // elements
533   MetAxisymetrique2D met_biellette;
534   Mat_pleine matGeom ; // matrice géométrique
535   Mat_pleine matInit ; // matrice initile
536   Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
537   Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
538   Tableau < Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
539
540 // ---- concernant les frontières et particulièrement le calcul de second membre
541 GeomSeg segS; // contient les fonctions d'interpolation et les derivees
542 GeomPoint point; // " " "
543
544 //----- calcul d'erreur -----
545 DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
546 //----- d'erreur : contraintes -----
547 DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud,
548 //servant pour le calcul d'erreur : contraintes, en fait pour l'assemblage
549 Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur

```

```

550     Mat_pleine  raidErr; // raideur pour le calcul d'erreur
551     GeomSeg  segmentEr; // contient les fonctions d'interpolation et
552                // les derivees pour le calcul du hessien dans
553                // la résolution de la fonctionnelle d'erreur
554     // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
555     -----
556     // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
557     Vecteur  residu_interne;
558     Mat_pleine  raideur_interne;
559     Tableau <Vecteur* >  residus_externeN; // pour les noeuds
560     Tableau <Mat_pleine* >  raideurs_externeN; // pour les noeuds
561     Tableau <Vecteur* >  residus_externeA; // pour l' aretes
562     Tableau <Mat_pleine* >  raideurs_externeA; // pour l' aretes
563     // ----- données concernant la dynamique -----
564     Mat_pleine  matrice_masse;
565     GeomSeg  segmentMas; // contient les fonctions d'interpolation et les dérivées
566     // pour les calculs relatifs au calcul de la masse
567     // ----- blocage éventuel d'hourglass
568     // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
569     Cal_explici_hourglass
570     GeomSeg*  segmentHourg; // contient les fonctions d'interpolation
571     };
572
573     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
574     // et un pointeur sur les données statiques communes
575     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
576     // classe est défini. Son allocation est effectuée dans les classes dérivées
577     class UneFois
578     { public :
579         UneFois () ; // constructeur par défaut
580         ~UneFois () ; // destructeur
581
582         // VARIABLES :
583     public :
584         DonneeCommune * doCoMemb;
585
586         // incicateurs permettant de dimensionner seulement au premier passage
587         // utilise dans "CalculResidu" et "Calcul_implicit"
588         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
589         int CalimpPrem;
590         int dualSortbiel; // pour la sortie des valeurs au pt d'integ
591         int CalSMlin_t; // pour les seconds membres concernant les arretes
592         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
593         int CalSMRlin; // pour les seconds membres concernant les arretes
594         int CalSMsurf_t; // pour les seconds membres concernant les surfaces
595         int CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
596         int CalSMRsurf; // pour les seconds membres concernant les surfaces
597         int CalSMvol_t; // pour les seconds membres concernant les volumes
598         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
599         int CalSMvol; // pour les seconds membres concernant les volumes
600         int CalDynamique; // pour le calcul de la matrice de masse
601         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
602         // ----- sauvegarde du nombre d'élément en cours -----
603         int nbelem_in_Prog;
604     };
605
606     // -----
607
608     protected :
609         // VARIABLES PROTÉGÉES :
610         // les données spécifiques sont groupées dans une structure pour sécuriser
611         // le passage de paramètre dans init par exemple
612         class Donnee_specif
613         { public :
614             Donnee_specif() : // défaut
615                 epais(Element::epaisseur_defaut,Element::epaisseur_defaut,Element::epaisseur_defaut)
616                 ,cas_pti_nbi(0),cas_pti_nbiEr(0)
617                 ,cas_pti_nbiS(0),cas_pti_nbiMas(0)
618             {};
619             Donnee_specif(double epai) : // uniquement l'épaisseur
620                 epais(epai,epai,epai) ,cas_pti_nbi(0),cas_pti_nbiEr(0)
621                 ,cas_pti_nbiS(0),cas_pti_nbiMas(0)
622             {};
623             Donnee_specif(int casptnbi,int casptnbiEr,int casptnbiS,int casptnbiMas) : // les indicateurs
624                 epais(Element::epaisseur_defaut,Element::epaisseur_defaut,Element::epaisseur_defaut)
625                 ,cas_pti_nbi(casptnbi),cas_pti_nbiEr(casptnbiEr)
626                 ,cas_pti_nbiS(casptnbiS),cas_pti_nbiMas(casptnbiMas)
627             {};
628             Donnee_specif(double epai0,double epai_t,double epai_tdt
629                 ,int casptnbi,int casptnbiEr,int casptnbiS,int casptnbiMas) : // tous
630                 epais(epai0,epai_t,epai_tdt)
631             ,cas_pti_nbi(casptnbi),cas_pti_nbiEr(casptnbiEr),cas_pti_nbiS(casptnbiS),cas_pti_nbiMas(casptnbiMas)
632             {};
633             Donnee_specif(const Donnee_specif& a) :

```

```

634     epais(a.epais )
635     ,cas_pti_nbi(a.cas_pti_nbi),cas_pti_nbiEr(a.cas_pti_nbiEr)
636     ,cas_pti_nbiS(a.cas_pti_nbiS),cas_pti_nbiMas(a.cas_pti_nbiMas)
637     {} ; // recopie via le constructeur de copie
638 ~Donnee_specif() {} ;
639 Donnee_specif & operator = ( const Donnee_specif& a)
640 { epais = a.epais ;
641   cas_pti_nbi=a.cas_pti_nbi;cas_pti_nbiEr=a.cas_pti_nbiEr ;
642   cas_pti_nbiS=a.cas_pti_nbiS;cas_pti_nbiMas=a.cas_pti_nbiMas ;
643   return *this ;};
644 // data
645 // epaisseurs de l'element
646 Epai epais ; // épaisseur
647
648 int cas_pti_nbi ; // permet de différencier les différents cas de pt d'integ identique
649 // =0: valeur par défaut, ensuite si diff de 1 donne les différents
cas
650 // est par exemple utilisé pour différencier le cas de 3 pt sur les
arrêtes
651 // ou 3 pt en interne, dans ce cas vaut 1, il peut ainsi y avoir plus
de 2 cas
652 int cas_pti_nbiEr ; // idem pour l'erreur
653 int cas_pti_nbiS ; // idem pour le calcul de second membre surfacique
654 int cas_pti_nbiMas ; // idem pour le calcul de la matrice masse
655 };
656 Donnee_specif donnee_specif ;
657
658 // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
659 LesPtIntegMecaInterne lesPtMecaInt ;
660
661 // place memoire commune a tous les elements biellettes
662 static DonneeCommune * doCo ;
663 // idem mais pour les indicateurs qui servent pour l'initialisation
664 static UneFois unefois ;
665
666 // type structuré pour construire les éléments
667 class NombresConstruire
668 { public:
669   NombresConstruire() ;
670   int nbne ; // le nombre de noeud de l'élément
671   int nbneA ; // le nombre de noeud des aretes
672   int nbi ; // le nombre de point d'intégration pour le calcul mécanique
673   int nbiEr ; // le nombre de point d'intégration pour le calcul d'erreur
674   int nbiA ; // le nombre de point d'intégration pour le calcul de second membre linéique
675   int nbiMas ; // le nombre de point d'intégration pour le calcul de la matrice masse consistante
676   int nbiHour ; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
677 };
678 static NombresConstruire nombre_V ; // les nombres propres à l'élément
679
680 // fonction privée
681 // fonction d'initialisation servant au niveau du constructeur
682 Biel_axi::DonneeCommune * Init(Donnee_specif donnee_specif = Donnee_specif()
683                               ,bool sans_init_noeud = false) ;
684 void Def_DonneeCommune () ;
685 // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
686 void Destruction() ;
687
688 // pour l'ajout d'element dans la liste : listTypeElemen, gérée par la class Element
689 class ConstrucElementbiel : public ConstrucElement
690 { public : ConstrucElementbiel ()
691   { NouvelleTypeElement nouv(SEG_AXI,BIE1,MECA_SOLIDE_DEFORMABLE,this) ;
692     if (ParaGlob::NiveauImpression() >= 4)
693       cout << "\n initialisation Biel_axi" << endl ;
694     Element::listTypeElement.push_back(nouv) ;
695   } ;
696   Element * NouvelElement(int nb_mail,int num) // un nouvel élément sans rien
697   {Element * pt ;
698     pt = new Biel_axi (nb_mail,num) ;
699     return pt ;};
700   // ramene true si la construction de l'element est possible en fonction
701   // des variables globales actuelles: ex en fonction de la dimension
702   bool Element_possible() {return true ;};
703 };
704 static ConstrucElementbiel construcElementbiel ;
705
706 // Calcul du residu local a t ou tdt en fonction du booleen
707 Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa) ;
708 // calcul des seconds membres suivant les chargements
709 // cas d'un chargement volumique,
710 // force indique la force volumique appliquée
711 // retourne le second membre résultant
712 // ici on l'épaisseur de l'élément pour constituer le volume
713 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
714 Vecteur SM_charge_volumique_E
715 (const Coordonnee& force,Fonction_nD* pt_fonct,bool atdt,const ParaAlgoControle &
pa,bool sur_volume_finale_) ;
716 // cas d'un chargement surfacique, sur les frontières des éléments

```

```

717 // force indique la force surfacique appliquée
718 // numface indique le numéro de la face chargée
719 // retourne le second membre résultant
720 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
721 Vecteur SM_charge_surfacique_E
722 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,bool atdt,const
ParaAlgoControle & pa);
723 // cas d'un chargement de type pression, sur les frontières des éléments
724 // pression indique la pression appliquée
725 // numface indique le numéro de la face chargée
726 // retourne le second membre résultant
727 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
728 Vecteur SM_charge_pression_E
729 (double pression,Fonction_nD* pt_fonct,int numFace,bool atdt,const ParaAlgoControle &
pa);
730 // cas d'un chargement surfacique hydrostatique,
731 // poidvol: indique le poids volumique du liquide
732 // M_liquide : un point de la surface libre
733 // dir_normal_liquide : direction normale à la surface libre
734 // retourne le second membre résultant
735 // -> explicite à t
736 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
737 ,int numFace,const Coordonnee& M_liquide,bool atdt
738 ,const ParaAlgoControle & pa
739 ,bool sans_limitation);
740 // cas d'un chargement lineique, sur les aretes frontières des éléments
741 // force indique la force lineique appliquée
742 // numarete indique le numéro de l'arete chargée
743 // retourne le second membre résultant
744 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
745 Vecteur SM_charge_lineique_E
746 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
747 // cas d'un chargement lineique suiveuse, sur l'arete frontière
748 //de la biellette (2D uniquement)
749 // force indique la force lineique appliquée
750 // numarete indique le numéro de l'arete chargée
751 // retourne le second membre résultant
752 // -> explicite à t
753 Vecteur SM_charge_lineique_Suiv_E
754 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
755 // cas d'un chargement surfacique hydro-dynamique,
756 // voir méthode explicite plus haut, pour les arguments
757 // retourne le second membre résultant
758 // bool atdt : permet de spécifier à t ou a t+tdt
759 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
760 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
761 , CourbelD* coef_aero_t,bool atdt,const
ParaAlgoControle & pa) ;
762
763 // calcul de la nouvelle épaisseur moyenne finale (sans raideur)
764 // mise à jour des volumes aux pti
765 // ramène l'épaisseur moyenne calculée à atdt
766 const double& CalEpaisseurMoyenne_et_vol_pti(bool atdt);
767 };
768 /// @} // end of group
769 #endif
770
771
772
773

```

## 7.129 Biel\_axiQ.h

```

1 // FICHER : Biel_axiQ.h
2 // CLASSE : Biel_axiQ
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,

```

```

20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           20/06/2017                               *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)      *
35 *   PROJET:        Herezh++                                  *
36 *   $                                                       *
37 *   $                                                       *
38 *   $                                                       *
39 *****/
40 *   BUT: // La classe Biel_axiQ permet de declarer des elements *
41 *   biellettes axisymétrique et de realiser le calcul du residu local *
42 *   et de la raideur locale pour une loi de comportement donnee. *
43 *   La dimension de l'espace pour un tel element est 1. *
44 *   Le support géométrique de l'élément est quadratique *
45 *   *
46 *   ***** *
47 *   *
48 *   VERIFICATION: *
49 *   *
50 *   ! date ! auteur ! but ! *
51 *   ----- *
52 *   ! ! ! ! *
53 *   $ *
54 *   ***** *
55 *   MODIFICATIONS: *
56 *   ! date ! auteur ! but ! *
57 *   ----- *
58 *   $ *
59 *****/
60 // -----classe pour un calcul de mecanique-----
61
62
63
64 #ifndef BIEL_AXIQ_H
65 #define BIEL_AXIQ_H
66
67 #include "ParaGlob.h"
68 #include "ElemMeca.h"
69 // #include "Loi_comp_abstraite.h"
70 #include "Met_abstraite.h"
71 #include "Met_biellette.h"
72 #include "MetAxisymetrique2D.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "ElFrontiere.h"
79 #include "GeomSeg.h"
80 #include "GeomPoint.h"
81 #include "ParaAlgoContrôle.h"
82 #include "FrontSegQuad.h"
83 #include "Epai.h"
84
85 class ConstrucElementbiel;
86
87 /// @addtogroup groupe_des_elements_finis
88 /// @{
89 ///
90
91
92 class Biel_axiQ : public ElemMeca
93 {
94
95     public :
96
97     // CONSTRUCTEURS :
98     // Constructeur par default
99     Biel_axiQ ();
100
101     // Constructeur fonction d'une epaisseur et eventuellement d'un numero
102     // d'identification et de maillage
103     Biel_axiQ (double epai,int num_maill=0,int num_id=-3);
104
105     // Constructeur fonction d'un numero de maillage et d'identification
106     Biel_axiQ (int num_maill,int num_id);

```

```

107
108 // Constructeur fonction d'une epaisseur, d'un numero de maillage et d'identification,
109 // du tableau de connexite des noeuds
110 Biel_axiQ (double epai,int num_maill,int num_id,const Tableau<Noeud *>& tab);
111
112 // Constructeur de copie
113 Biel_axiQ (const Biel_axiQ& biel);
114
115
116 // DESTRUCTEUR :
117 ~Biel_axiQ ();
118
119 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
120 // méthode virtuelle
121 Element* Nevez_copie() const { Element * el= new Biel_axiQ(*this); return el;};
122
123 // Surcharge de l'operateur = : realise l'egalite entre deux instances de Biel_axiQ
124 Biel_axiQ& operator= (Biel_axiQ& biel);
125
126 // METHODES :
127 // 1) derivant des virtuelles pures
128 // Lecture des donnees de la classe sur fichier
129 void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
130
131 // Calcul du residu local et de la raideur locale,
132 // pour le schema implicite
133 Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
134
135 // Calcul du residu local a t
136 // pour le schema explicit par exemple
137 Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
138 { return Biel_axiQ::CalculResidu(false,pa);};
139
140 // Calcul du residu local a tdt
141 // pour le schema explicit par exemple
142 Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
143 { return Biel_axiQ::CalculResidu(true,pa);};
144
145 // Calcul de la matrice masse pour l'élément
146 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
147
148 // ----- calcul dynamique -----
149 // calcul de la longueur d'arrête de l'élément minimal
150 // divisé par la célérité la plus rapide dans le matériau
151 double Long_arrete_mini_sur_c(Enum_dure temps)
152 { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
153
154 //----- calcul d'erreur, remontée des contraintes -----
155 // 1)calcul du résidu et de la matrice de raideur pour le calcul d'erreur
156 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
157 // 2) remontée aux erreurs aux noeuds
158 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
159
160 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
161 // ce tableau est specifique a l'element
162 const DdlElement & TableauDdl() const ;
163
164
165 // Libere la place occupee par le residu et eventuellement la raideur
166 // par l'appel de Libere de la classe mere et libere les differents tenseurs
167 // intermediaires cree pour le calcul et les grandeurs pointee
168 // de la raideur et du residu
169 void Libere ();
170
171 // acquisition d'une loi de comportement
172 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
173
174 // test si l'element est complet
175 // = 1 tout est ok, =0 element incomplet
176 int TestComplet();
177
178 // procedure permettant de completer l'element apres
179 // sa creation avec les donnees du bloc transmis
180 // peut etre appeler plusieurs fois
181 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
182 // Compléter pour la mise en place de la gestion de l'hourglass
183 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
184
185 // ramene l'element geometrique
186 ElemGeomCO& ElementGeometrique() const { return doCo->segment;};
187 // ramene l'element geometrique en constant
188 const ElemGeomCO& ElementGeometrique_const() const { return doCo->segment;};
189
190 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
191 // associé
192 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
193 // 1) cas où l'on utilise la place passée en argument

```

```

193 Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
194 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
195 void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
196
197 // -- connaissances particulières sur l'élément
198 // ramène l'épaisseur de l'élément
199 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
200 virtual double Epaisseurs(Enum_dure enu , const Coordonnee& ) {return H(enu);};
201 // ramène l'épaisseur moyenne de l'élément (indépendante du point)
202 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
203 virtual double EpaisseurMoyenne(Enum_dure enu ) {return H(enu);};
204
205 // affichage dans la sortie transmise, des variables duales "nom"
206 // dans le cas ou nom est vide, affichage de "toute" les variables
207 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
208
209 // affichage d'info en fonction de ordre
210 // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
211 void Info_com_Element(UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> * tabMaillageNoeud)
212
213     { return Element::Info_com_El(2,entreePrinc,ordre,tabMaillageNoeud);};
214
215 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
216 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
217 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
218 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
219 // temps: dit si c'est à 0 ou t ou tdt
220 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
221     { return PtLePlusPres(temps,enu,M);};
222
223 // recuperation des coordonnées du point de numéro d'ordre iteg pour
224 // la grandeur enu
225 // temps: dit si c'est à 0 ou t ou tdt
226 // si erreur retourne erreur à true
227 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
228     { return CoordPtInt(temps,enu,iteg,erreur);};
229
230 // récupération des valeurs au numéro d'ordre = iteg pour
231 // les grandeur enu
232 Tableau <double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
233     enu,int iteg) ;
234
235 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
236 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
237 // de conteneurs quelconque associée
238 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int
239     iteg);
240
241 // ramene vrai si la surface numéro ns existe pour l'élément
242 // dans le cas de la biellette il n'y a pas de surface
243 bool SurfExiste(int ) const
244     { return false;};
245
246 // ramene vrai si l'arête numéro na existe pour l'élément
247 bool AreteExiste(int na) const {if (na==1) return true; else return false;};
248
249 //===== lecture écriture dans base info =====
250
251 // cas donne le niveau de la récupération
252 // = 1 : on récupère tout
253 // = 2 : on récupère uniquement les données variables (supposées comme telles)
254 void Lecture_base_info
255     (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
256 // cas donne le niveau de sauvegarde
257 // = 1 : on sauvegarde tout
258 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
259 void Ecriture_base_info(ofstream& sort,const int cas) ;
260
261 // METHODES VIRTUELLES:
262 // ----- calculs utils dans le cadre de la recherche du flambement linéaire
263 // Calcul de la matrice géométrique et initiale
264 ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
265
266 // retourne la liste des données particulières actuellement utilisés
267 // par l'élément (actif ou non), sont exclu de cette liste les données particulières des noeuds
268 // reliés à l'élément
269 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
270 // particulière
271 List_io <TypeQuelconque> Les_types_particuliers_internes(bool absolue) const;
272
273 // récupération de grandeurs particulières au numéro d'ordre = iteg
274 // celles-ci peuvent être quelconques
275 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
276 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
277 // particulière
278 void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);

```



```

275
276 // inactive les ddl du problème primaire de mécanique
277 inline void Inactive_ddl_primaire()
278 {ElemMeca::Inact_ddl_primaire(doCo->tab_ddl)};
279 // active les ddl du problème primaire de mécanique
280 inline void Active_ddl_primaire()
281 {ElemMeca::Act_ddl_primaire(doCo->tab_ddl)};
282 // ajout des ddl de contraintes pour les noeuds de l'élément
283 inline void Plus_ddl_Sigma()
284 {ElemMeca::Ad_ddl_Sigma(doCo->tab_ddlErr)};
285 // inactive les ddl du problème de recherche d'erreur : les contraintes
286 inline void Inactive_ddl_Sigma()
287 {ElemMeca::Inact_ddl_Sigma(doCo->tab_ddlErr)};
288 // active les ddl du problème de recherche d'erreur : les contraintes
289 inline void Active_ddl_Sigma()
290 {ElemMeca::Act_ddl_Sigma(doCo->tab_ddlErr)};
291 // active le premier ddl du problème de recherche d'erreur : SIGMAll
292 inline void Active_premier_ddl_Sigma()
293 {ElemMeca::Act_premier_ddl_Sigma()};
294
295 // lecture de données diverses sur le flot d'entrée
296 void LectureContraintes(UtilLecture * entreePrinc)
297 {if (unefois.CalResPrem_t == 1)
298     ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
299     else
300     { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
301       unefois.CalResPrem_t = 1;
302     }
303 };
304
305 // retour des contraintes en absolu retour true si elle existe sinon false
306 ContraintesAbsolues(Tableau <Vecteur>& tabSig)
307 { if (unefois.CalResPrem_t == 1)
308     ElemMeca::ContraintesEnAbsolues (false,lesPtMecaInt.TabSigHH_t(),tabSig);
309     else
310     { unefois.CalResPrem_t = 1;
311       ElemMeca::ContraintesEnAbsolues (true,lesPtMecaInt.TabSigHH_t(),tabSig);
312     };
313     return true;
314 };
315
316 // 2) derivant des virtuelles
317 // retourne un tableau de ddl element, correspondant à la
318 // composante de sigma -> SIG11, pour chaque noeud qui contient
319 // des ddl de contrainte
320 // -> utilisé pour l'assemblage de la raideur d'erreur
321 inline DdlElement& Tableau_de_Sig11() const
322 {return doCo->tab_Err1Sig11};
323
324 // actualisation des ddl et des grandeurs actives de t+dt vers t
325 void TdtversT();
326 // actualisation des ddl et des grandeurs actives de t vers tdt
327 void TversTdt();
328
329 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
330 // qu'une fois la remontée aux contraintes effectuées sinon aucune
331 // action. En retour la valeur de l'erreur sur l'élément
332 // type indique le type de calcul d'erreur :
333 void ErreurElement(int type,double& errElemRelative
334                   ,double& numerateur, double& denominateur);
335
336 // mise à jour de la boîte d'encombrement de l'élément, suivant les axes I_a globales
337 // en retour coordonnées du point mini dans retour.Premier() et du point maxi dans .Second()
338 // la méthode est différente de la méthode générale car il faut prendre en compte l'épaisseur de
339 // l'élément
340 virtual const DeuxCoordonnees& Boite_encombre_element(Enum_dure temps);
341
342 // calcul des seconds membres suivant les chargements
343 // cas d'un chargement volumique,
344 // force indique la force volumique appliquée
345 // retourne le second membre résultant
346 // ici on l'épaisseur de l'élément pour constituer le volume
347 // -> explicite à t
348 Vecteur SM_charge_volumique_E_t(const Coordonnee& force,Fonction_nd* pt_fonct,const
ParaAlgoContrôle & pa,bool sur_volume_finale_)
349 { return Biel_axiQ::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_)};
350 // -> explicite à tdt
351 Vecteur SM_charge_volumique_E_tdt(const Coordonnee& force,Fonction_nd* pt_fonct,const
ParaAlgoContrôle & pa,bool sur_volume_finale_)
352 { return Biel_axiQ::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_)};
353 // -> implicite,
354 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
355 // retourne le second membre et la matrice de raideur correspondant
356 ResRaid SMR_charge_volumique_I(const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoContrôle
& pa,bool sur_volume_finale_);
357

```

```

358 // cas d'un chargement surfacique, sur les frontieres des elements
359 // force indique la force surfacique appliquee
360 // numface indique le numero de la face chargee
361 // retourne le second membre resultant
362 // -> version explicite à t
363 Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
364 { return Biel_axiQ::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa); };
365 // -> version explicite à tdt
366 Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
367 { return Biel_axiQ::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa); };
368 // -> implicite,
369 // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
370 // retourne le second membre et la matrice de raideur correspondant
371 ResRaid SMR_charge_surfacique_I
372 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const ParaAlgoControle & pa) ;
373
374 // cas d'un chargement de type pression, sur les frontieres des elements
375 // pression indique la pression appliquee
376 // numface indique le numero de la face chargee
377 // retourne le second membre resultant
378 // -> explicite à t
379 Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
380 { return Biel_axiQ::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa); };
381 // -> explicite à tdt
382 Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
383 { return Biel_axiQ::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa); };
384 // -> implicite,
385 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
386 // retourne le second membre et la matrice de raideur correspondant
387 ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
388
389
390 // cas d'un chargement surfacique hydrostatique,
391 // poidvol: indique le poids volumique du liquide
392 // M_liquide : un point de la surface libre
393 // dir_normal_liquide : direction normale à la surface libre
394 // retourne le second membre resultant
395 // -> explicite à t
396 Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
397 ,const ParaAlgoControle & pa
398 ,bool sans_limitation)
399 {
400 { return
Biel_axiQ::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation); };
401 // -> explicite à tdt
402 Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
403 ,const ParaAlgoControle & pa
404 ,bool sans_limitation)
405 { return
Biel_axiQ::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation); };
406 // -> implicite,
407 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
408 // retourne le second membre et la matrice de raideur correspondant
409 ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
410 ,const ParaAlgoControle & pa
411 ,bool sans_limitation) ;
412
413
414 // cas d'un chargement lineique, sur les aretes frontieres des elements
415 // force indique la force lineique appliquee
416 // numarete indique le numero de l'arete chargee
417 // retourne le second membre resultant
418 // -> explicite à t
419 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
420 { return Biel_axiQ::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); };
421 // -> explicite à tdt
422 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
423 { return Biel_axiQ::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); };
424 // -> implicite,
425 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
426 // retourne le second membre et la matrice de raideur correspondant
427 ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
428
429
430 // cas d'un chargement lineique suiveuse, sur l'arrete frontiere de
431 // la biellette (2D uniquement)
432 // force indique la force lineique appliquee
433 // numarete indique le numero de l'arete chargee
434 // retourne le second membre resultant

```

```

435 // -> explicite à t
436 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoContrôle & pa)
437 { return Biel_axiQ::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa); };
438 // -> explicite à tdt
439 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoContrôle & pa)
440 { return Biel_axiQ::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa); };
441 // -> implicite,
442 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
443 // retourne le second membre et la matrice de raideur correspondant
444 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoContrôle & pa) ;
445
446 // cas d'un chargement surfacique hydro-dynamique,
447 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
448 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc unitaire)
449 // une suivant la direction normale à la vitesse de type portance
450 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
451 // une suivant la vitesse tangente de type frottement visqueux
452 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
453 // coef_mul: est un coefficient multiplicateur global (de tout)
454 // retourne le second membre résultant
455 // -> explicite à t
456 Vecteur SM_charge_hydrodynamique_E_t( CourbeID* frot_fluid,const double& poidvol
457 , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
458 , CourbeID* coef_aero_t,const ParaAlgoContrôle &
pa)
459 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
460
461 // -> explicite à tdt
462 Vecteur SM_charge_hydrodynamique_E_tdt( CourbeID* frot_fluid,const double& poidvol
463 , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
464 , CourbeID* coef_aero_t,const ParaAlgoContrôle &
pa)
465 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};
466
467 // -> implicite,
468 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
469 // retourne le second membre et la matrice de raideur correspondant
470 ResRaid SMR_charge_hydrodynamique_I( CourbeID* frot_fluid,const double& poidvol
471 , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
472 , CourbeID* coef_aero_t,const ParaAlgoContrôle &
pa) ;
473
474 // ===== définition et/ou construction des frontières =====
475
476 // Calcul des frontieres de l'element
477 // creation des elements frontieres et retour du tableau de ces elements
478 // la création n'a lieu qu'au premier appel
479 // ou lorsque l'on force le paramètre force a true
480 // dans ce dernier cas seul les frontière effacées sont recréée
481 Tableau <ElFrontiere*> const & Frontiere(bool force = false);
482
483 // ramene l'epaisseur
484 inline double H(Enum_dure enu = TEMPS_tdt )
485 { switch (enu)
486 { case TEMPS_0: return donnee_specif.epais.epaisseur0; break;
487 case TEMPS_t: return donnee_specif.epais.epaisseur_t; break;
488 case TEMPS_tdt: return donnee_specif.epais.epaisseur_tdt; break;
489 };
490 return 0.; // cas n'arrivant normalement jamais
491 };
492
493 // ajout du tableau specfic de ddl des noeuds de la biellette
494 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
495 // des noeuds constituant l'element
496 void ConstTabDdl();
497
498 protected:
499 // ==== » methodes virtuelles dérivant d'ElemMeca =====
500 // ramene la dimension des tenseurs contraintes et déformations de l'élément
501 int Dim_sig_eps() const {return 2;};
502
503 // ----- calcul de frontières en protected -----
504
505 // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
506 // particulière à l'élément
507 // adressage des frontières linéiques et surfacique
508 // définit dans les classes dérivées, et utilisées pour la construction des frontières
509 virtual ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)

```

```

508     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
509 virtual ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
510     {return NULL;} // il n'y a pas de surface possible
511
512
513 private :
514
515     // VARIABLES PRIVEES :
516
517
518     class DonneeCommune
519     { public :
520         DonneeCommune (GeomSeg& seg,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
521             MetAxisymetrique2D& met_bie,
522             Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
523             GeomSeg& seEr,Vecteur& residu_int,Mat_pleine& raideur_int,
524             Tableau <Vecteur* > & residus_extN,Tableau <Mat_pleine* >& raideurs_extN,
525             Tableau <Vecteur* > & residus_extA,Tableau <Mat_pleine* >& raideurs_extA,
526             Mat_pleine& mat_masse ,GeomSeg& seMa,int nbi,GeomSeg* segHourg
527         ) ;
528         DonneeCommune(DonneeCommune& a);
529         ~DonneeCommune();
530         // variables
531         GeomSeg segment ; // element geometrique correspondant
532         DdlElement tab_ddl; // tableau des degres
533         //de liberte des noeuds de l'element commun a tous les
534         // elements
535         MetAxisymetrique2D met_biellette;
536         Mat_pleine matGeom ; // matrice géométrique
537         Mat_pleine matInit ; // matrice initiale
538         Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
539         Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
540         Tableau <Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
541
542         // ---- concernant les frontières et particulièrement le calcul de second membre
543         GeomSeg segS; // contient les fonctions d'interpolation et les derivees
544         GeomPoint point; // " " "
545
546         //----- calcul d'erreur -----
547         DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
548         //----- d'erreur : contraintes -----
549         DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud,
550         //servant pour le calcul d'erreur : contraintes, en fait pour l'assemblage
551         Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
552         Mat_pleine raidErr; // raideur pour le calcul d'erreur
553         GeomSeg segmentEr; // contient les fonctions d'interpolation et
554         // les derivees pour le calcul du hessian dans
555         //la résolution de la fonctionnelle d'erreur
556         // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
557         -----
558         // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
559         Vecteur residu_interne;
560         Mat_pleine raideur_interne;
561         Tableau <Vecteur* > residus_externeN; // pour les noeuds
562         Tableau <Mat_pleine* > raideurs_externeN; // pour les noeuds
563         Tableau <Vecteur* > residus_externeA; // pour l' aretes
564         Tableau <Mat_pleine* > raideurs_externeA; // pour l' aretes
565         // ----- données concernant la dynamique -----
566         Mat_pleine matrice_masse;
567         GeomSeg segmentMas; // contient les fonctions d'interpolation et les dérivées
568         // pour les calculs relatifs au calcul de la masse
569         // ----- blocage éventuel d'hourglass
570         // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
571         Cal_explici_hourglass
572         GeomSeg* segmentHourg; // contient les fonctions d'interpolation
573     };
574
575     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
576     // et un pointeur sur les données statiques communes
577     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
578     // classe est défini. Son allocation est effectuée dans les classes dérivées
579     class UneFois
580     { public :
581         UneFois () ; // constructeur par défaut
582         ~UneFois () ; // destructeur
583
584         // VARIABLES :
585     public :
586         DonneeCommune * doCoMemb;
587
588         // indicateurs permettant de dimensionner seulement au premier passage
589         // utilise dans "CalculResidu" et "Calcul_implicit"
590         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
591         int CalimpPrem;
592         int dualSortbiel; // pour la sortie des valeurs au pt d'integ
593         int CalSMLin_t; // pour les seconds membres concernant les arretes

```

```

593         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
594         int CalSMRlin; // pour les seconds membres concernant les arretes
595         int CalSMsurf_t; // pour les seconds membres concernant les surfaces
596         int CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
597         int CalSMRsurf; // pour les seconds membres concernant les surfaces
598         int CalSMvol_t; // pour les seconds membres concernant les volumes
599         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
600         int CalSMvol; // pour les seconds membres concernant les volumes
601         int CalDynamique; // pour le calcul de la matrice de masse
602         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
603         // ----- sauvegarde du nombre d'élément en cours -----
604         int nbelem_in_Prog;
605     };
606
607
608     // -----
609
610     protected :
611     // VARIABLES PROTÉGÉES :
612     // les données spécifiques sont groupées dans une structure pour sécuriser
613     // le passage de paramètre dans init par exemple
614     class Donnee_specif
615     { public :
616         Donnee_specif() : // défaut
617             epais(Element::epaisseur_defaut,Element::epaisseur_defaut,Element::epaisseur_defaut)
618             ,cas_pti_nbi(0),cas_pti_nbiEr(0)
619             ,cas_pti_nbiS(0),cas_pti_nbiMas(0)
620         {};
621         Donnee_specif(double epai) : // uniquement l'épaisseur
622             epais(epai,epai,epai) ,cas_pti_nbi(0),cas_pti_nbiEr(0)
623             ,cas_pti_nbiS(0),cas_pti_nbiMas(0)
624         {};
625         Donnee_specif(int casptnbi,int casptnbiEr,int casptnbiS,int casptnbiMas) : // les indicateurs
626             epais(Element::epaisseur_defaut,Element::epaisseur_defaut,Element::epaisseur_defaut)
627             ,cas_pti_nbi(casptnbi),cas_pti_nbiEr(casptnbiEr)
628             ,cas_pti_nbiS(casptnbiS),cas_pti_nbiMas(casptnbiMas)
629         {};
630         Donnee_specif(double epai0,double epai_t,double epai_tdt
631             ,int casptnbi,int casptnbiEr,int casptnbiS,int casptnbiMas) : // tous
632             epais(epai0,epai_t,epai_tdt)
633         ,cas_pti_nbi(casptnbi),cas_pti_nbiEr(casptnbiEr),cas_pti_nbiS(casptnbiS),cas_pti_nbiMas(casptnbiMas)
634         {};
635         Donnee_specif(const Donnee_specif& a) :
636             epais(a.epais)
637             ,cas_pti_nbi(a.cas_pti_nbi),cas_pti_nbiEr(a.cas_pti_nbiEr)
638             ,cas_pti_nbiS(a.cas_pti_nbiS),cas_pti_nbiMas(a.cas_pti_nbiMas)
639         {}; // recopie via le constructeur de copie
640         ~Donnee_specif() {};
641         Donnee_specif & operator = ( const Donnee_specif& a)
642         { epais = a.epais;
643           cas_pti_nbi=a.cas_pti_nbi;cas_pti_nbiEr=a.cas_pti_nbiEr;
644           cas_pti_nbiS=a.cas_pti_nbiS;cas_pti_nbiMas=a.cas_pti_nbiMas;
645           return *this;};
646         // data
647         // épaisseurs de l'element
648         Epai epais; // épaisseur
649
650         int cas_pti_nbi; // permet de différencier les différents cas de pt d'integ identique
651         // =0: valeur par défaut, ensuite si diff de 1 donne les différents
652
653         cas // est par exemple utilisé pour différencier le cas de 3 pt sur les
654         arrêtes // ou 3 pt en interne, dans ce cas vaut 1, il peut ainsi y avoir plus
655         de 2 cas
656         int cas_pti_nbiEr; // idem pour l'erreur
657         int cas_pti_nbiS; // idem pour le calcul de second membre surfacique
658         int cas_pti_nbiMas; // idem pour le calcul de la matrice masse
659     };
660     Donnee_specif donnee_specif;
661
662     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
663     LesPtIntegMecaInterne lesPtMecaInt;
664
665     // place memoire commune a tous les elements biellettes
666     static DonneeCommune * doCo;
667     // idem mais pour les indicateurs qui servent pour l'initialisation
668     static UneFois unefois;
669
670     // type structuré pour construire les éléments
671     class NombresConstruire
672     { public:
673         NombresConstruire();
674         int nbne; // le nombre de noeud de l'élément
675         int nbneA ; // le nombre de noeud des arêtes
676         int nbi; // le nombre de point d'intégration pour le calcul mécanique
677         int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur

```

```

676     int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
677     int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse consistante
678     int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
679 };
680     static NombresConstruire nombre_V; // les nombres propres à l'élément
681
682 // fonction privée
683 // fonction d'initialisation servant au niveau du constructeur
684 Biel_axiQ::DonneeCommune * Init(Donnee_specif donnee_specif = Donnee_specif()
685     ,bool sans_init_noeud = false);
686
687 void Def_DonneeCommune();
688 // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
689 void Destruction();
690
691 // pour l'ajout d'élément dans la liste : listTypeElement, gérée par la class Element
692 class ConstrucElementbiel : public ConstrucElement
693 { public : ConstrucElementbiel ()
694     { NouvelleTypeElement nouv(SEG_AXI,BIE2,MECA_SOLIDE_DEFORMABLE,this);
695     if (ParaGlob::NiveauImpression() >= 4)
696     cout << "\n initialisation Biel_axiQ" << endl;
697     Element::listTypeElement.push_back(nouv);
698     };
699     Element * NouvelElement(int nb_mail,int num) // un nouvel élément sans rien
700     {Element * pt;
701     pt = new Biel_axiQ (nb_mail,num) ;
702     return pt;};
703     // ramène true si la construction de l'élément est possible en fonction
704     // des variables globales actuelles: ex en fonction de la dimension
705     bool Element_possible() {return true;};
706 };
707
708 static ConstrucElementbiel construcElementbiel;
709
710 // Calcul du résidu local a t ou tdt en fonction du booleen
711 Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
712 // calcul des seconds membres suivant les chargements
713 // cas d'un chargement volumique,
714 // force indique la force volumique appliquée
715 // retourne le second membre résultant
716 // ici on l'épaisseur de l'élément pour constituer le volume
717 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
718 Vecteur SM_charge_volumique_E
719     (const Coordonnee& force,Fonction_nD* pt_fonct,bool atdt,const ParaAlgoControle &
720     pa,bool sur_volume_finale_);
721 // cas d'un chargement surfacique, sur les frontières des éléments
722 // force indique la force surfacique appliquée
723 // numface indique le numéro de la face chargée
724 // retourne le second membre résultant
725 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
726 Vecteur SM_charge_surfacique_E
727     (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,bool atdt,const
728     ParaAlgoControle & pa);
729 // cas d'un chargement de type pression, sur les frontières des éléments
730 // pression indique la pression appliquée
731 // numface indique le numéro de la face chargée
732 // retourne le second membre résultant
733 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
734 Vecteur SM_charge_pression_E
735     (double pression,Fonction_nD* pt_fonct,int numFace,bool atdt,const ParaAlgoControle &
736     pa);
737
738 // cas d'un chargement surfacique hydrostatique,
739 // poidvol: indique le poids volumique du liquide
740 // M_liquide : un point de la surface libre
741 // dir_normal_liquide : direction normale à la surface libre
742 // retourne le second membre résultant
743 // -> explicite à t
744 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
745     ,int numFace,const Coordonnee& M_liquide,bool atdt
746     ,const ParaAlgoControle & pa
747     ,bool sans_limitation);
748 // cas d'un chargement linéique, sur les arêtes frontières des éléments
749 // force indique la force linéique appliquée
750 // numarete indique le numéro de l'arête chargée
751 // retourne le second membre résultant
752 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
753 Vecteur SM_charge_lineique_E
754     (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
755     ParaAlgoControle & pa);
756 // cas d'un chargement linéique suivieuse, sur l'arête frontière
757 //de la biellette (2D uniquement)
758 // force indique la force linéique appliquée
759 // numarete indique le numéro de l'arête chargée
760 // retourne le second membre résultant
761 // -> explicite à t
762 Vecteur SM_charge_lineique_Suiv_E
763     (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
764     ParaAlgoControle & pa);

```

```

758     // cas d'un chargement surfacique hydro-dynamique,
759     // voir méthode explicite plus haut, pour les arguments
760     // retourne le second membre résultant
761     // bool atdt : permet de spécifier à t ou a t+dt
762     Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
763     , CourbelD* coef_aero_n,int numFace,const double&
764     coef_mul
765     , CourbelD* coef_aero_t,bool atdt,const
766     ParaAlgoControle & pa) ;
767
768     // calcul de la nouvelle épaisseur moyenne finale (sans raideur)
769     // mise à jour des volumes aux pti
770     // ramène l'épaisseur moyenne calculée à atdt
771     const double& CalEpaisseurMoyenne_et_vol_pti(bool atdt);
772 };
773 /// @} // end of group
774 #endif
775
776

```

## 7.130 Biellette.h

```

1 // FICHER : Biellette.h
2 // CLASSE : Biellette
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      BUT: // La classe Biellette permet de declarer des elements
41 *      biellettes et de realiser le calcul du residu local et de la raideur*
42 *      locale pour une loi de comportement donnee. La dimension de l'espace *
43 *      pour un tel element est 1.
44 *
45 *      *****
46 *
47 *      VERIFICATION:
48 *      ! date !   auteur !           but
49 *      -----
50 *      !           !           !
51 *
52 *      *****
53 *
54 *      MODIFICATIONS:
55 *      ! date !   auteur !           but
56 *      -----
57 *
58 *      *****
59 // -----classe pour un calcul de mecanique-----
60

```

```

61
62 #ifndef BIELLETTE_H
63 #define BIELLETTE_H
64
65 #include "ParaGlob.h"
66 #include "ElemMeca.h"
67 // #include "Loi_comp_abstraite.h"
68 #include "Met_abstraite.h"
69 #include "Met_biellette.h"
70 #include "Noeud.h"
71 #include "UtilLecture.h"
72 #include "Tenseur.h"
73 #include "NevezTenseur.h"
74 #include "Deformation.h"
75 #include "ElFrontiere.h"
76 #include "GeomSeg.h"
77 #include "ParaAlgoControle.h"
78 #include "FrontSegLine.h"
79 #include "Section.h"
80
81 class ConstrucElementbiel;
82
83 /// @addtogroup groupe_des_elements_finis
84 /// @{
85 ///
86
87
88 class Biellette : public ElemMeca
89 {
90
91     public :
92
93         // CONSTRUCTEURS :
94         // Constructeur par default
95         Biellette ();
96
97         // Constructeur fonction d'une section et eventuellement d'un numero
98         // d'identification et de maillage
99         Biellette (double sect,int num_maill=0,int num_id=-3);
100
101         // Constructeur fonction d'un numero de maillage et d'identification
102         Biellette (int num_maill,int num_id);
103
104         // Constructeur fonction d'une section, d'un numero de maillage et d'identification,
105         // du tableau de connexite des noeuds
106         Biellette (double sect,int num_maill,int num_id,const Tableau<Noeud *>& tab);
107
108         // Constructeur de copie
109         Biellette (const Biellette& biel);
110
111
112         // DESTRUCTEUR :
113         ~Biellette ();
114
115         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
116         // méthode virtuelle
117         Element* Nevez_copie() const { Element * el= new Biellette(*this); return el;};
118
119         // Surcharge de l'operateur = : realise l'egalite entre deux instances de Biellette
120         Biellette& operator= (Biellette& biel);
121
122         // METHODES :
123         // 1) derivant des virtuelles pures
124         // Lecture des donnees de la classe sur fichier
125         void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
126
127         // Calcul du residu local et de la raideur locale,
128         // pour le schema implicite
129         Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
130
131         // Calcul du residu local a t
132         // pour le schema explicit par exemple
133         Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
134         { return Biellette::CalculResidu(false,pa);};
135
136         // Calcul du residu local a tdt
137         // pour le schema explicit par exemple
138         Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
139         { return Biellette::CalculResidu(true,pa);};
140
141         // Calcul de la matrice masse pour l'élément
142         Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
143
144         // ----- calcul dynamique -----
145         // calcul de la longueur d'arrête de l'élément minimal
146         // divisé par la célérité la plus rapide dans le matériau
147         double Long_arrete_mini_sur_c(Enum_dure temps)

```



```

148         { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
149
150     //----- calcul d'erreur, remontée des contraintes -----
151     // 1)calcul du résidu et de la matrice de raideur pour le calcul d'erreur
152     Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
153     // 2) remontée aux erreurs aux noeuds
154     Element::Er_ResRaid ErreurAuNoeud_ResRaid();
155
156     // retourne les tableaux de ddl associés aux noeuds, gere par l'element
157     // ce tableau et specifique a l'element
158     const DdlElement & TableauDdl() const ;
159
160
161     // Libere la place occupee par le residu et eventuellement la raideur
162     // par l'appel de Libere de la classe mere et libere les differents tenseurs
163     // intermediaires cree pour le calcul et les grandeurs pointee
164     // de la raideur et du residu
165     void Libere ();
166
167     // acquisition d'une loi de comportement
168     void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
169
170     // test si l'element est complet
171     // = 1 tout est ok, =0 element incomplet
172     int TestComplet();
173
174     // procedure permettant de completer l'element apres
175     // sa creation avec les donnees du bloc transmis
176     // peut etre appeler plusieurs fois
177     Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
178     // Compléter pour la mise en place de la gestion de l'hourglass
179     Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
180
181     // ramene l'element geometrique
182     ElemGeomCO& ElementGeometrique() const { return doCo->segment;};
183     // ramene l'element geometrique en constant
184     const ElemGeomCO& ElementGeometrique_const() const { return doCo->segment;};
185
186     // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
187     // associé
188     // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
189     // 1) cas où l'on utilise la place passée en argument
190     Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
191     // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
192     void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
193
194     // -- connaissances particulières sur l'élément
195     // ramène l'épaisseur de l'élément
196     // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
197     virtual double Section(Enum_dure enu , const Coordonnee& ) {return S(enu);};
198     // ramène l'épaisseur moyenne de l'élément (indépendante du point)
199     // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
200     virtual double SectionMoyenne(Enum_dure enu ) {return S(enu);};
201
202     // affichage dans la sortie transmise, des variables duales "nom"
203     // dans le cas ou nom est vide, affichage de "toute" les variables
204     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
205
206     // affichage d'info en fonction de ordre
207     // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
208     void Info_com_Element(UtilLecture * entreePrinc,string& ordre,Tableau<Noeud * > * tabMaillageNoeud)
209
210     { return Element::Info_com_El(2,entreePrinc,ordre,tabMaillageNoeud);};
211
212     // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
213     // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
214     // par exemple CoordPtInteg, ou Valeur_a_diff_temps
215     // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
216     // temps: dit si c'est à 0 ou t ou tdt
217     int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
218     { return PtLePlusPres(temps,enu,M);};
219
220     // recuperation des coordonnées du point de numéro d'ordre iteg pour
221     // la grandeur enu
222     // temps: dit si c'est à 0 ou t ou tdt
223     // si erreur retourne erreur à true
224     Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
225     { return CoordPtInt(temps,enu,iteg,erreur);};
226
227     // récupération des valeurs au numéro d'ordre = iteg pour
228     // les grandeur enu
229     Tableau<double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
230     enu,int iteg) ;
231
232     // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
233     // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
234     // de conteneurs quelconque associée

```

```

232 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int
iteg);
233
234 // ramene vrai si la surface numéro ns existe pour l'élément
235 // dans le cas de la biellette il n'y a pas de surface
236 bool SurfExiste(int ) const
237 { return false;};
238
239 // ramene vrai si l'arête numéro na existe pour l'élément
240 bool AreteExiste(int na) const {if (na==1) return true; else return false;};
241
242
243 //===== lecture écriture dans base info =====
244
245 // cas donne le niveau de la récupération
246 // = 1 : on récupère tout
247 // = 2 : on récupère uniquement les données variables (supposées comme telles)
248 void Lecture_base_info
249 (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
250 // cas donne le niveau de sauvegarde
251 // = 1 : on sauvegarde tout
252 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
253 void Ecriture_base_info(ofstream& sort,const int cas) ;
254
255 // METHODES VIRTUELLES:
256 // ----- calculs utils dans le cadre de la recherche du flambement linéaire
257 // Calcul de la matrice géométrique et initiale
258 ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
259
260 // retourne la liste des données particulières actuellement utilisés
261 // par l'élément (actif ou non), sont exclu de cette liste les données particulières des noeuds
262 // reliés à l'élément
263 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
264 List_io <TypeQuelconque> Les_types_particuliers_internes(bool absolue) const;
265
266 // récupération de grandeurs particulières au numéro d'ordre = iteg
267 // celles-ci peuvent être quelconques
268 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
269 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
270 void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);
271
272 // inactive les ddl du problème primaire de mécanique
273 inline void Inactive_ddl_primaire()
274 {ElemMeca::Inact_ddl_primaire(doCo->tab_ddl);};
275 // active les ddl du problème primaire de mécanique
276 inline void Active_ddl_primaire()
277 {ElemMeca::Act_ddl_primaire(doCo->tab_ddl);};
278 // ajout des ddl de contraintes pour les noeuds de l'élément
279 inline void Plus_ddl_Sigma()
280 {ElemMeca::Ad_ddl_Sigma(doCo->tab_ddlErr);};
281 // inactive les ddl du problème de recherche d'erreur : les contraintes
282 inline void Inactive_ddl_Sigma()
283 {ElemMeca::Inact_ddl_Sigma(doCo->tab_ddlErr);};
284 // active les ddl du problème de recherche d'erreur : les contraintes
285 inline void Active_ddl_Sigma()
286 {ElemMeca::Act_ddl_Sigma(doCo->tab_ddlErr);};
287 // active le premier ddl du problème de recherche d'erreur : SIGMA11
288 inline void Active_premier_ddl_Sigma()
289 {ElemMeca::Act_premier_ddl_Sigma();};
290
291 // lecture de données diverses sur le flot d'entrée
292 void LectureContraintes(UtilLecture * entreePrinc)
293 {if (unefois.CalResPrem_t == 1)
294 ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
295 else
296 { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
297 unefois.CalResPrem_t = 1;
298 }
299 };
300
301 // retour des contraintes en absolu retour true si elle existe sinon false
302 bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
303 { if (unefois.CalResPrem_t == 1)
304 ElemMeca::ContraintesEnAbsolues (false,lesPtMecaInt.TabSigHH_t(),tabSig);
305 else
306 { unefois.CalResPrem_t = 1;
307 ElemMeca::ContraintesEnAbsolues (true,lesPtMecaInt.TabSigHH_t(),tabSig);
308 };
309 return true;
310 };
311
312
313 // 2) derivant des virtuelles
314 // retourne un tableau de ddl element, correspondant à la
315 // composante de sigma -> SIG11, pour chaque noeud qui contient

```

```

316 // des ddl de contrainte
317 // -> utilisé pour l'assemblage de la raideur d'erreur
318 inline DdlElement& Tableau_de_Sigl() const
319 {return doCo->tab_Err1Sigl;} ;
320
321 // actualisation des ddl et des grandeurs actives de t+dt vers t
322 void TdtversT();
323 // actualisation des ddl et des grandeurs actives de t vers tdt
324 void TversTdt();
325
326 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
327 // qu'une fois la remontée aux contraintes effectuées sinon aucune
328 // action. En retour la valeur de l'erreur sur l'élément
329 // type indique le type de calcul d'erreur :
330 void ErreurElement(int type,double& errElemRelative
331 ,double& numerateur, double& denominateur);
332
333 // mise à jour de la boîte d'encombrement de l'élément, suivant les axes I_a globales
334 // en retour coordonnées du point mini dans retour.Premier() et du point maxi dans .Second()
335 // la méthode est différente de la méthode générale car il faut prendre en compte l'épaisseur de
l'élément
336 virtual const DeuxCoordonnees& Boite_encombre_element(Enum_dure temps);
337
338 // calcul des seconds membres suivant les chargements
339 // cas d'un chargement volumique,
340 // force indique la force volumique appliquée
341 // retourne le second membre résultant
342 // ici on l'épaisseur de l'élément pour constituer le volume
343 // -> explicite à t
344 Vecteur SM_charge_volumique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,const
ParaAlgoControle & pa,bool sur_volume_finale_)
345 { return Biellette::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_)} ;
346 // -> explicite à tdt
347 Vecteur SM_charge_volumique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,const
ParaAlgoControle & pa,bool sur_volume_finale_)
348 { return Biellette::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_)} ;
349 // -> implicite,
350 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
351 // retourne le second membre et la matrice de raideur correspondant
352 ResRaid SMR_charge_volumique_I(const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle
& pa,bool sur_volume_finale_) ;
353
354 // cas d'un chargement lineique, sur les aretes frontieres des éléments
355 // force indique la force lineique appliquée
356 // numarete indique le numéro de l'arete chargée
357 // retourne le second membre résultant
358 // -> explicite à t
359 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
360 { return Biellette::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa)} ;
361 // -> explicite à tdt
362 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
363 { return Biellette::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa)} ;
364 // -> implicite,
365 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
366 // retourne le second membre et la matrice de raideur correspondant
367 ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
368
369 // cas d'un chargement lineique suiveuse, sur l'arrêt frontiere de
370 // la biellette (2D uniquement)
371 // force indique la force lineique appliquée
372 // numarete indique le numéro de l'arete chargée
373 // retourne le second membre résultant
374 // -> explicite à t
375 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
376 { return Biellette::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa)} ;
377 // -> explicite à tdt
378 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
379 { return Biellette::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa)} ;
380 // -> implicite,
381 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
382 // retourne le second membre et la matrice de raideur correspondant
383 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa) ;
384
385 // cas d'un chargement surfacique hydro-dynamique,
386 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
387 // Fn = poids_volu * fn(V) * S * (normale*ut) * u, u étant le vecteur directeur de V (donc unitaire)
388 // une suivant la direction normale à la vitesse de type portance
389 // Ft = poids_volu * ft(V) * S * (normale*ut) * w, w unitaire, normal à V, et dans le plan n et V
390 // une suivant la vitesse tangente de type frottement visqueux
391 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
392 // coef_mul: est un coefficient multiplicateur global (de tout)

```

```

393 // retourne le second membre résultant
394 // -> explicite à t
395 Vecteur SM_charge_hydrodynamique_E_t( CourbelD* frot_fluid,const double& poidvol
396                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
397                                     , CourbelD* coef_aero_t,const ParaAlgoControle &
pa)
398 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa)};

399 // -> explicite à tdt
400 Vecteur SM_charge_hydrodynamique_E_tdt( CourbelD* frot_fluid,const double& poidvol
401                                       , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
402                                       , CourbelD* coef_aero_t,const ParaAlgoControle &
pa)
403 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa)};

404 // -> implicite,
405 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
406 // retourne le second membre et la matrice de raideur correspondant
407 ResRaid SMR_charge_hydrodynamique_I( CourbelD* frot_fluid,const double& poidvol
408                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
409                                     , CourbelD* coef_aero_t,const ParaAlgoControle &
pa) ;

410
411 // ===== définition et/ou construction des frontières =====
412
413 // Calcul des frontieres de l'element
414 // creation des elements frontieres et retour du tableau de ces elements
415 // la création n'a lieu qu'au premier appel
416 // ou lorsque l'on force le paramètre force a true
417 // dans ce dernier cas seul les frontière effacées sont recréée
418 // Tableau <ElFrontiere*> const & Frontiere(bool force = false);
419
420 // Retourne la section de l'element
421 inline double S(Enum_dure enu = TEMPS_tdt)
422 { switch (enu)
423   { case TEMPS_0: return donnee_specif.secti.section0; break;
424     case TEMPS_t: return donnee_specif.secti.section_t; break;
425     case TEMPS_tdt: return donnee_specif.secti.section_tdt; break;
426   };
427   return 0.; // cas n'arrivant normalement jamais
428 };
429
430 // ajout du tableau specific de ddl des noeuds de la biellette
431 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
432 // des noeuds constituant l'element
433 void ConstTabDdl();
434
435 protected:
436
437 // ==== »» methodes virtuelles dérivant d'ElemMeca =====
438 // ramene la dimension des tenseurs contraintes et déformations de l'élément
439 int Dim_sig_eps() const {return 1;};
440
441 // ----- calcul de frontières en protected -----
442
443 // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
particulière à l'élément
444 // adressage des frontières linéiques et surfacique
445 // définit dans les classes dérivées, et utilisées pour la construction des frontières
virtual ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
446 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
447 virtual ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
448 {return NULL;} // il n'y a pas de surface possible
449
450 private :
451
452 // VARIABLES PRIVEES :
453
454 class DonneeCommune
455 { public :
456   DonneeCommune (GeomSeg& seg,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
457                 Met_biellette& met_bie,
458                 Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
459                 GeomSeg& seEr,Vecteur& residu_int,Mat_pleine& raideur_int,
460                 Tableau <Vecteur* > & residus_extN,Tableau <Mat_pleine* >& raideurs_extN,
461                 Tableau <Vecteur* > & residus_extA,Tableau <Mat_pleine* >& raideurs_extA,
462                 Mat_pleine& mat_masse ,GeomSeg& seMa,int nbi,GeomSeg* segHourg
463                 ) ;
464   DonneeCommune(DonneeCommune& a);

```

```

469     ~DonneeCommune();
470     // variables
471     GeomSeg segment ; // element geometrique correspondant
472     DdlElement tab_ddl; // tableau des degres
473         //de liberte des noeuds de l'element commun a tous les
474         // elements
475     Met_biellette met_biellette;
476     Mat_pleine matGeom ; // matrice géométrique
477     Mat_pleine matInit ; // matrice initiale
478     Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
479     Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
480     Tableau <Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
481         // calcul d'erreur
482     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
483         // d'erreur : contraintes
484     DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud,
485         //servant pour le calcul d'erreur : contraintes, en fait pour l'assemblage
486     Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
487     Mat_pleine raidErr; // raideur pour le calcul d'erreur
488     GeomSeg segmentEr; // contient les fonctions d'interpolation et
489         // les derivees pour le calcul du hessien dans
490         //la résolution de la fonctionnelle d'erreur
491         // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
492         -----
493         // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
494     Vecteur residu_interne;
495     Mat_pleine raideur_interne;
496     Tableau <Vecteur* > residus_externeN; // pour les noeuds
497     Tableau <Mat_pleine* > raideurs_externeN; // pour les noeuds
498     Tableau <Vecteur* > residus_externeA; // pour l' aretes
499     Tableau <Mat_pleine* > raideurs_externeA; // pour l' aretes
500         // ----- données concernant la dynamique -----
501     Mat_pleine matrice_masse;
502     GeomSeg segmentMas; // contient les fonctions d'interpolation et les dérivées
503         // pour les calculs relatifs au calcul de la masse
504         // ----- blocage éventuel d'hourglass
505         // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
506         Cal_explici_hourglass
507     GeomSeg* segmentHourg; // contient les fonctions d'interpolation
508     };
509
510     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
511     // et un pointeur sur les données statiques communes
512     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
513     // classe est défini. Son allocation est effectuée dans les classes dérivées
514     class UneFois
515     { public :
516         UneFois () ; // constructeur par défaut
517         ~UneFois () ; // destructeur
518
519         // VARIABLES :
520     public :
521         DonneeCommune * doCoMemb;
522
523         // incicateurs permettant de dimensionner seulement au premier passage
524         // utilise dans "CalculResidu" et "Calcul_implicit"
525         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
526         int CalimpPrem;
527         int dualSortbiel; // pour la sortie des valeurs au pt d'integ
528         int CalSMlin_t; // pour les seconds membres concernant les arretes
529         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
530         int CalSMrlin; // pour les seconds membres concernant les arretes
531         int CalSMvol_t; // pour les seconds membres concernant les volumes
532         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
533         int CalSMvol; // pour les seconds membres concernant les volumes
534         int CalDynamique; // pour le calcul de la matrice de masse
535         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
536         // ----- sauvegarde du nombre d'élément en cours -----
537         int nbelem_in_Prog;
538     };
539
540     // -----
541
542     protected :
543         // VARIABLES PROTÉGÉES :
544         // les données spécifiques sont groupées dans une structure pour sécuriser
545         // le passage de paramètre dans init par exemple
546         class Donnee_specif
547         { public :
548             Donnee_specif() : // défaut
549                 secti(Element::section_defaut,Element::section_defaut,Element::section_defaut)
550             , fctnD_section(NULL),variation_section(true)
551             {}
552             Donnee_specif(double section) : // uniquement la section
553                 secti(section,section,section),fctnD_section(NULL),variation_section(true)

```

```

554         {} ;
555         Donnee_specif(double epai0, double epai_t, double epai_tdt, Fonction_nD* fctnD, bool variation) :
// tous
556             secti(epai0, epai_t, epai_tdt), fctnD_section(fctnD), variation_section(variation)
557         {} ;
558         Donnee_specif(const Donnee_specif& a) :
559             secti(a.secti), fctnD_section(a.fctnD_section), variation_section(a.variation_section)
560             {} ; // recopie via le constructeur de copie
561         ~Donnee_specif() {} ;
562         Donnee_specif & operator = ( const Donnee_specif& a)
563         { secti = a.secti; fctnD_section = a.fctnD_section; variation_section = a.variation_section;
564           return *this; };
565         // data
566         // sections de l'element
567         Sect secti; // épaisseur
568         Fonction_nD* fctnD_section; // permet éventuellement de prendre en compte une évolution de la
section
569         bool variation_section; // permet éventuellement de ne pas prendre en compte la variation
570     };
571     Donnee_specif donnee_specif;
572
573     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
574     LesPtIntegMecaInterne lesPtMecaInt;
575
576     // place memoire commune a tous les elements biellettes
577     static DonneeCommune * doCo;
578     // idem mais pour les indicateurs qui servent pour l'initialisation
579     static UneFois unefois;
580
581     // type structuré pour construire les éléments
582     class NombresConstruire
583     { public:
584         NombresConstruire();
585         int nbne; // le nombre de noeud de l'élément
586         int nbneA; // le nombre de noeud des aretes
587         int nbi; // le nombre de point d'intégration pour le calcul mécanique
588         int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
589         int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
590         int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse consistante
591         int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
592     };
593     static NombresConstruire nombre_V; // les nombres propres à l'élément
594
595     // fonction privée
596     // fonction d'initialisation servant au niveau du constructeur
597     Biellette::DonneeCommune * Init(Donnee_specif donnee_specif = Donnee_specif()
598                                   , bool sans_init_noeud = false);
599
600     void Def_DonneeCommune ();
601     // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
602     void Destruction();
603
604     // pour l'ajout d'element dans la liste : listTypeElement, gérée par la class Element
605     class ConstrucElementbiel : public ConstrucElement
606     { public : ConstrucElementbiel ()
607         { NouvelleTypeElement nouv(POUT, BIE1, MECA_SOLIDE_DEFORMABLE, this);
608           if (ParaGlob::NiveauImpression() >= 4)
609             cout << "\n initialisation Biellette" << endl;
610           Element::listTypeElement.push_back(nouv);
611         };
612         Element * NouvelElement(int nb_mail, int num) // un nouvel élément sans rien
613         { Element * pt;
614           pt = new Biellette (nb_mail, num) ;
615           return pt; };
616         // ramene true si la construction de l'element est possible en fonction
617         // des variables globales actuelles: ex en fonction de la dimension
618         bool Element_possible() {return true;};
619     };
620     static ConstrucElementbiel construcElementbiel;
621
622     // Calcul du residu local a t ou tdt en fonction du booleen
623     Vecteur* CalculResidu (bool atdt, const ParaAlgoControle & pa);
624     // calcul des seconds membres suivant les chargements
625     // cas d'un chargement volumique,
626     // force indique la force volumique appliquée
627     // retourne le second membre résultant
628     // ici on l'épaisseur de l'élément pour constituer le volume
629     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
630     Vecteur SM_charge_volumique_E
631     (const Coordonnee& force, Fonction_nD* pt_fonct, bool atdt, const ParaAlgoControle &
pa, bool sur_volume_finale_);
632     // cas d'un chargement lineique, sur les aretes frontieres des éléments
633     // force indique la force lineique appliquée
634     // numarete indique le numéro de l'arete chargée
635     // retourne le second membre résultant
636     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
637     Vecteur SM_charge_lineique_E
638     (const Coordonnee& force, Fonction_nD* pt_fonct, int numArete, bool atdt, const

```

```

    ParaAlgoControle & pa);
638 // cas d'un chargement lineique suivieuse, sur l'arete frontiere
639 //de la biellette (2D uniquement)
640 // force indique la force lineique appliquee
641 // numarete indique le numero de l'arete chargee
642 // retourne le second membre resultant
643 // -> explicite à t
644 Vecteur SM_charge_lineique_Suiv_E
645 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
646 // cas d'un chargement surfacique hydro-dynamique,
647 // voir methode explicite plus haut, pour les arguments
648 // retourne le second membre resultant
649 // bool atdt : permet de specifier à t ou a t+dt
650 Vecteur SM_charge_hydrodynamique_E( CourbeID* frot_fluid,const double& poidvol
651 , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
652 , CourbeID* coef_aero_t,bool atdt,const
ParaAlgoControle & pa) ;
653
654 // calcul de la nouvelle section moyenne finale (sans raideur)
655 // mise à jour des volumes aux pti
656 // ramène la section moyenne calculée à atdt
657 const double& CalSectionMoyenne_et_vol_pti(const bool atdt);
658 };
659 /// @} // end of group
660 #endif
661
662
663
664

```

## 7.131 BielletteC1.h

```

1 // FICHER : BielletteC1.h
2 // CLASSE : BielletteC1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 * DATE: 05/06/06 *
34 * * $ *
35 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
36 * * $ *
37 * PROJET: Herezh++ *
38 * * $ *
39 *****/
40 * BUT: // La classe Biellette permet de declarer des elements *
41 * biellettes de continuité C1. *
42 * * *
43 * ***** *
44 * VERIFICATION: *
45 * * *
46 * ! date ! auteur ! but ! *
47 * ----- *
48 * ! ! ! ! *
49 * * $ *
50 * ***** *

```

```

51 *      MODIFICATIONS:
52 *      ! date !      auteur !      but
53 *      -----
54 *      $
55 *      *****/
56 // -----classe pour un calcul de mecanique-----
57
58
59
60 #ifndef BIELLETTE_C1_H
61 #define BIELLETTE_C1_H
62
63 #include "ParaGlob.h"
64 #include "ElemMeca.h"
65 // #include "Loi_comp_abstraite.h"
66 #include "Met_abstraite.h"
67 #include "Met_BielletteC1.h"
68 #include "Noeud.h"
69 #include "UtilLecture.h"
70 #include "Tenseur.h"
71 #include "NevezTenseur.h"
72 #include "Deformation.h"
73 #include "ElFrontiere.h"
74 #include "GeomSeg.h"
75 #include "ParaAlgoControle.h"
76 #include "FrontSegLine.h"
77
78 class ConstrucElementbiel;
79
80 /// @addtogroup groupe_des_elements_finis
81 /// @
82 ///
83
84
85 class BielleC1 : public ElemMeca
86 {
87
88     public :
89
90         // CONSTRUCTEURS :
91         // Constructeur par defaut
92         BielleC1 ();
93
94         // Constructeur fonction d'une section et eventuellement d'un numero
95         // d'identification et de maillage
96         BielleC1 (double sect,int num_maill=0,int num_id=-3);
97
98         // Constructeur fonction d'un numero de maillage et d'un numero d'identification
99         BielleC1 (int num_maill,int num_id);
100
101         // Constructeur fonction d'une section, d'un numero de maillage et d'un numero
102         // d'identification,
103         // du tableau de connexite des noeuds
104         BielleC1 (double sect,int num_maill,int num_id,const Tableau<Noeud *>& tab);
105
106         // Constructeur de copie
107         BielleC1 (const BielleC1& biel);
108
109         // DESTRUCTEUR :
110         ~BielleC1 ();
111
112         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113         // méthode virtuelle
114         Element* Nevez_copie() const { Element * el= new BielleC1(*this); return el;};
115
116         // Surcharge de l'operateur = : realise l'egalite entre deux instances de BielleC1
117         BielleC1& operator= (BielleC1& biel);
118
119         // METHODES :
120 // 1) derivant des virtuelles pures
121         // Lecture des donnees de la classe sur fichier
122         void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
123
124         // Calcul du residu local et de la raideur locale,
125         // pour le schema implicite
126         Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
127
128         // Calcul du residu local a t
129         // pour le schema explicite par exemple
130         Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
131         { return BielleC1::CalculResidu(false,pa);};
132
133         // Calcul du residu local a tdt
134         // pour le schema explicite par exemple
135         Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
136         { return BielleC1::CalculResidu(true,pa);};

```



```

137
138 // Calcul de la matrice masse pour l'élément
139 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
140
141 // ----- calcul dynamique -----
142 // calcul de la longueur d'arrêt de l'élément minimal
143 // divisé par la célérité la plus rapide dans le matériau
144 double Long_arrete_mini_sur_c(Enum_dure temps)
145     { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
146
147 //----- calcul d'erreur, remontée des contraintes -----
148 // 1)calcul du résidu et de la matrice de raideur pour le calcul d'erreur
149 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
150 // 2) remontée aux erreurs aux noeuds
151 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
152
153 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
154 // ce tableau et specifique a l'element
155 const DdlElement & TableauDdl() const ;
156
157
158 // Libere la place occupee par le residu et eventuellement la raideur
159 // par l'appel de Libere de la classe mere et libere les differents tenseurs
160 // intermediaires cree pour le calcul et les grandeurs pointees
161 // de la raideur et du residu
162 void Libere ();
163
164 // acquisition d'une loi de comportement
165 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
166
167 // test si l'element est complet
168 // = 1 tout est ok, =0 element incomplet
169 int TestComplet();
170
171 // procedure permettant de completer l'element apres
172 // sa creation avec les donnees du bloc transmis
173 // peut etre appele plusieurs fois
174 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
175 // Compléter pour la mise en place de la gestion de l'hourglass
176 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
177
178 // ramene l'element geometrique
179 ElemGeomC0 & ElementGeometrique() const { return doCo->segment;};
180 // ramene l'element geometrique en constant
181 const ElemGeomC0 & ElementGeometrique_const() const { return doCo->segment;};
182
183 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
184 // associé
185 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
186 // 1) cas où l'on utilise la place passée en argument
187 Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
188 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
189 void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
190
191 // affichage dans la sortie transmise, des variables duales "nom"
192 // dans le cas ou nom est vide, affichage de "toute" les variables
193 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
194
195 // affichage d'info en fonction de ordre
196 // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
197 void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud * > *
198     tabMaillageNoeud)
199     { return Element::Info_com_El(2,entreePrinc,ordre,tabMaillageNoeud);};
200
201 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
202 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
203 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
204 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
205 // temps: dit si c'est à 0 ou t ou tdt
206 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
207     { return PtLePlusPres(temps,enu,M);};
208
209 // recuperation des coordonnées du point de numéro d'ordre iteg pour
210 // la grandeur enu
211 // temps: dit si c'est à 0 ou t ou tdt
212 // si erreur retourne erreur à true
213 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
214     { return CoordPtInt(temps,enu,iteg,erreur);};
215
216 // récupération des valeurs au numéro d'ordre = iteg pour
217 // les grandeurs enu
218 Tableau <double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const
219     List_io<Ddl_enum_etendu>& enu,int iteg) ;

```

```

220 // de conteneurs quelconque associée
221 void ValTensorielle_a_diff_temps(bool absolue, Enum_dure enu_t, List_io<TypeQuelconque>& enu, int
iteg);
222
223 // ramene vrai si la surface numéro ns existe pour l'élément
224 // dans le cas de la BielleC1 il n'y a pas de surface
225 bool SurfExiste(int ) const
226 { return false; };
227
228 // ramene vrai si l'arête numéro na existe pour l'élément
229 bool AreteExiste(int na) const { if (na==1) return true; else return false; };
230
231
232 //===== lecture écriture dans base info =====
233
234 // cas donne le niveau de la récupération
235 // = 1 : on récupère tout
236 // = 2 : on récupère uniquement les données variables (supposées comme telles)
237 void Lecture_base_info
238 (ifstream& ent, const Tableau<Noeud *> * tabMaillageNoeud, const int cas) ;
239 // cas donne le niveau de sauvegarde
240 // = 1 : on sauvegarde tout
241 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
242 void Ecriture_base_info(ofstream& sort, const int cas) ;
243
244 // METHODES VIRTUELLES:
245 // ----- calculs utils dans le cadre de la recherche du flambement linéaire
246 // Calcul de la matrice géométrique et initiale
247 ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
248
249 // inactive les ddl du problème primaire de mécanique
250 inline void Inactive_ddl_primaire()
251 { ElemMeca::Inact_ddl_primaire(doCo->tab_ddl); };
252 // active les ddl du problème primaire de mécanique
253 inline void Active_ddl_primaire()
254 { ElemMeca::Act_ddl_primaire(doCo->tab_ddl); };
255 // ajout des ddl de contraintes pour les noeuds de l'élément
256 inline void Plus_ddl_Sigma()
257 { ElemMeca::Ad_ddl_Sigma(doCo->tab_ddlErr); };
258 // inactive les ddl du problème de recherche d'erreur : les contraintes
259 inline void Inactive_ddl_Sigma()
260 { ElemMeca::Inact_ddl_Sigma(doCo->tab_ddlErr); };
261 // active les ddl du problème de recherche d'erreur : les contraintes
262 inline void Active_ddl_Sigma()
263 { ElemMeca::Act_ddl_Sigma(doCo->tab_ddlErr); };
264 // active le premier ddl du problème de recherche d'erreur : SIGMAll
265 inline void Active_premier_ddl_Sigma()
266 { ElemMeca::Act_premier_ddl_Sigma(); };
267
268 // lecture de données diverses sur le flot d'entrée
269 void LectureContraintes(UtilLecture * entreePrinc)
270 { if (unefois.CalResPrem_t == 1)
271 ElemMeca::LectureDesContraintes (false, entreePrinc, lesPtMecaInt.TabSigHH_t());
272 else
273 { ElemMeca::LectureDesContraintes (true, entreePrinc, lesPtMecaInt.TabSigHH_t());
274 unefois.CalResPrem_t = 1;
275 }
276 };
277
278 // retour des contraintes en absolu retour true si elle existe sinon false
279 bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
280 { if (unefois.CalResPrem_t == 1)
281 ElemMeca::ContraintesEnAbsolues (false, lesPtMecaInt.TabSigHH_t(), tabSig);
282 else
283 { unefois.CalResPrem_t = 1;
284 ElemMeca::ContraintesEnAbsolues (true, lesPtMecaInt.TabSigHH_t(), tabSig);
285 }
286 return true;
287 }
288
289
290 // 2) derivant des virtuelles
291 // retourne un tableau de ddl element, correspondant à la
292 // composante de sigma -> SIG11, pour chaque noeud qui contient
293 // des ddl de contrainte
294 // -> utilisé pour l'assemblage de la raideur d'erreur
295 inline DdlElement& Tableau_de_Sigl() const
296 { return doCo->tab_Err1Sig11; };
297
298 // actualisation des ddl et des grandeurs actives de t+dt vers t
299 void TdtversT();
300 // actualisation des ddl et des grandeurs actives de t vers tdt
301 void TversTdt();
302
303 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
304 // qu'une fois la remontée aux contraintes effectuées sinon aucune
305 // action. En retour la valeur de l'erreur sur l'élément

```

```

306 // type indique le type de calcul d'erreur :
307 void ErreurElement(int type,double& errElemRelative
308                  ,double& numerateur, double& denominateur);
309 // calcul des seconds membres suivant les chargements
310 // cas d'un chargement volumique,
311 // force indique la force volumique appliquée
312 // retourne le second membre résultant
313 // ici on l'épaisseur de l'élément pour constituer le volume
314 // -> explicite à t
315 Vecteur SM_charge_volumique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle
& pa,bool sur_volume_finale_)
316 { return BielletteC1::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_); };
317 // -> explicite à tdt
318 Vecteur SM_charge_volumique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,const
ParaAlgoControle & pa,bool sur_volume_finale_)
319 { return BielletteC1::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_); };
320 // -> implicite,
321 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
322 // retourne le second membre et la matrice de raideur correspondant
323 ResRaid SMR_charge_volumique_I(const Coordonnee& force,Fonction_nD* pt_fonct,const
ParaAlgoControle & pa,bool sur_volume_finale_);
324
325 // cas d'un chargement lineique, sur les aretes frontieres des éléments
326 // force indique la force lineique appliquée
327 // numarete indique le numéro de l'arete chargée
328 // retourne le second membre résultant
329 // -> explicite à t
330 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
331 { return BielletteC1::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); };
332 // -> explicite à tdt
333 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
334 { return BielletteC1::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); };
335 // -> implicite,
336 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
337 // retourne le second membre et la matrice de raideur correspondant
338 ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa);
339
340 // cas d'un chargement lineique suiveuse, sur l'arrête frontière de
341 // la BielletteC1 (2D uniquement)
342 // force indique la force lineique appliquée
343 // numarete indique le numéro de l'arete chargée
344 // retourne le second membre résultant
345 // -> explicite à t
346 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
347 { return BielletteC1::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa); };
348 // -> explicite à tdt
349 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
350 { return BielletteC1::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa); };
351 // -> implicite,
352 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
353 // retourne le second membre et la matrice de raideur correspondant
354 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa);
355
356 // cas d'un chargement surfacique hydro-dynamique,
357 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
358 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc
unitaire)
359 // une suivant la direction normale à la vitesse de type portance
360 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
361 // une suivant la vitesse tangente de type frottement visqueux
362 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
363 // coef_mul: est un coefficient multiplicateur global (de tout)
364 // retourne le second membre résultant
365 // -> explicite à t
366 Vecteur SM_charge_hydrodynamique_E_t( CourbelD* frot_fluid,const double& poidvol
367                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
368                                     , CourbelD* coef_aero_t,const ParaAlgoControle &
pa)
369 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
370 // -> explicite à tdt
371 Vecteur SM_charge_hydrodynamique_E_tdt( CourbelD* frot_fluid,const double& poidvol
372                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
373                                     , CourbelD* coef_aero_t,const ParaAlgoControle &
pa)
374 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};

```

```

375 // -> implicite,
376 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
377 // retourne le second membre et la matrice de raideur correspondant
378 ResRaid SMR_charge_hydrodynamique_I( CourbelID* frot_fluid,const double& poidsvol
379                                     , CourbelID* coef_aero_n,int numFace,const double&
coef_mul
380                                     , CourbelID* coef_aero_t,const ParaAlgoControle &
pa) ;
381
382
383 // ===== définition et/ou construction des frontières =====
384
385 // Calcul des frontieres de l'element
386 // creation des elements frontieres et retour du tableau de ces elements
387 // la création n'a lieu qu'au premier appel
388 // ou lorsque l'on force le paramètre force a true
389 // dans ce dernier cas seul les frontière effacées sont recrée
390 Tableau <ElFrontiere*> const & Frontiere(bool force = false);
391
392 // ramène la frontière point
393 // éventuellement création des frontieres points de l'element et stockage dans l'element
394 // si c'est la première fois sinon il y a seulement retour de l'elements
395 // a moins que le paramètre force est mis a true
396 // dans ce dernier cas la frontière effacée est recrée
397 // num indique le numéro du point à créer (numérotation EF)
398 // ElFrontiere* const Frontiere_points(int num,bool force = false);
399
400 // ramène la frontière linéique
401 // éventuellement création des frontieres linéique de l'element et stockage dans l'element
402 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
403 // a moins que le paramètre force est mis a true
404 // dans ce dernier cas la frontière effacée est recrée
405 // num indique le numéro de l'arête à créer (numérotation EF)
406 // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
407
408 // ramène la frontière surfacique
409 // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
410 // si c'est la première fois sinon il y a seulement retour de l'elements
411 // a moins que le paramètre force est mis a true
412 // dans ce dernier cas la frontière effacée est recrée
413 // num indique le numéro de la surface à créer (numérotation EF)
414 // ici normalement la fonction ne doit pas être appelée
415 // ElFrontiere* const Frontiere_surfacique(int ,bool force = false);
416
417 double Section (Enum_dure , const Coordonnee& )
418 // Retourne la section de l'element
419 { return donnee_specif.section; };
420 // ramène la section moyenne de l'élément (indépendante du point)
421 double SectionMoyenne(Enum_dure )
422 { return donnee_specif.section; };
423
424
425 // ajout du tableau specific de ddl des noeuds de la BielleC1
426 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
427 // des noeuds constituants l'element
428 void ConstTabDdl();
429 protected:
430
431 // ==== » methodes virtuelles dérivant d'ElemMeca =====
432 // ramene la dimension des tenseurs contraintes et déformations de l'élément
433 int Dim_sig_eps() const {return 1;};
434
435 // ----- calcul de frontières en protected -----
436
437 // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
particulière à l'élément
438 // adressage des frontières linéiques et surfacique
439 // définit dans les classes dérivées, et utilisées pour la construction des frontières
440 virtual ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
441 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
442 virtual ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
443 {return NULL;}; // il n'y a pas de surface possible
444
445
446 private :
447
448 // VARIABLES PRIVEES :
449
450
451 class DonneeCommune
452 { public :
453     DonneeCommune (GeomSeg& seg,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
454                   Met_BielleC1& met_bie,
455                   Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
456                   GeomSeg& seEr,Vecteur& residu_int,Mat_pleine& raideur_int,
457                   Tableau <Vecteur *> & residus_extN,Tableau <Mat_pleine *> & raideurs_extN,
458                   Tableau <Vecteur *> & residus_extA,Tableau <Mat_pleine *> & raideurs_extA,

```

```

459         Mat_pleine& mat_masse ,GeomSeg& seMa,int nbi,GeomSeg* segHourg
460     ) ;
461     DonneeCommune(DonneeCommune& a);
462     ~DonneeCommune();
463     // variables
464     GeomSeg segment ; // element geometrique correspondant
465     DdlElement tab_ddl; // tableau des degres
466         //de liberte des noeuds de l'element commun a tous les
467         // elements
468     Met_BielletteC1 met_BielletteC1;
469     Mat_pleine matGeom ; // matrice géométrique
470     Mat_pleine matInit ; // matrice initiale
471     Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
472     Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
473     Tableau < Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
474         // calcul d'erreur
475     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
476         // d'erreur : contraintes
477     DdlElement tab_ErrlSig11; // tableau du ddl SIG11 pour chaque noeud,
478         //servant pour le calcul d'erreur : contraintes, en fait pour l'assemblage
479     Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
480     Mat_pleine raidErr; // raideur pour le calcul d'erreur
481     GeomSeg segmentEr; // contient les fonctions d'interpolation et
482         // les derivees pour le calcul du hessien dans
483         //la résolution de la fonctionnelle d'erreur
484         // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
485         // -----
486         // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
487     Vecteur residu_interne;
488     Mat_pleine raideur_interne;
489     Tableau <Vecteur* > residus_externeN; // pour les noeuds
490     Tableau <Mat_pleine* > raideurs_externeN; // pour les noeuds
491     Tableau <Vecteur* > residus_externeA; // pour l' aretes
492     Tableau <Mat_pleine* > raideurs_externeA; // pour l' aretes
493         // ----- données concernant la dynamique -----
494     Mat_pleine matrice_masse;
495     GeomSeg segmentMas; // contient les fonctions d'interpolation et les dérivées
496         // pour les calculs relatifs au calcul de la masse
497         // ----- blocage éventuel d'hourglass
498         // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
499     Cal_explici_hourglass
500     GeomSeg* segmentHourg; // contient les fonctions d'interpolation
501     };
502
503     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
504     // et un pointeur sur les données statiques communes
505     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
506     // classe est défini. Son allocation est effectuée dans les classes dérivées
507     class UneFois
508     { public :
509         UneFois () ; // constructeur par défaut
510         ~UneFois () ; // destructeur
511
512         // VARIABLES :
513         public :
514         DonneeCommune * doCoMemb;
515
516         // incicateurs permettant de dimensionner seulement au premier passage
517         // utilise dans "CalculResidu" et "Calcul_implicit"
518         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
519         int CalimpPrem;
520         int dualSortbiel; // pour la sortie des valeurs au pt d'integ
521         int CalSMlin_t; // pour les seconds membres concernant les arretes
522         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
523         int CalSMrlin; // pour les seconds membres concernant les arretes
524         int CalSMvol_t; // pour les seconds membres concernant les volumes
525         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
526         int CalSMvol; // pour les seconds membres concernant les volumes
527         int CalDynamique; // pour le calcul de la matrice de masse
528         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
529         // ----- sauvegarde du nombre d'élément en cours -----
530         int nbelem_in_Prog;
531     };
532
533     // -----
534
535     protected :
536         // VARIABLES PROTÉGÉES :
537         // les données spécifiques sont groupées dans une structure pour sécuriser
538         // le passage de paramètre dans init par exemple
539         class Donnee_specif{ public :
540         Donnee_specif() :
541             section(section_defaut) {};
542         Donnee_specif(double sect) :
543             section(sect) {};

```

```

544     Donnee_specif(const Donnee_specif& a) :
545         section(a.section) {};
546     Donnee_specif & operator = ( const Donnee_specif& a)
547     { section = a.section;return *this;};
548     // data
549     double section;          // section de l'element
550     };
551     Donnee_specif donnee_specif;
552
553     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
554     LesPtIntegMecaInterne lesPtMecaInt;
555
556     // place memoire commune a tous les elements BielletteC1s
557     static DonneeCommune * doCo;
558     // idem mais pour les indicateurs qui servent pour l'initialisation
559     static UneFois unefois;
560
561     // type structuré pour construire les éléments
562     class NombresConstruire
563     { public:
564         NombresConstruire();
565         int nbne; // le nombre de noeud de l'élément
566         int nbneA ; // le nombre de noeud des aretes
567         int nbi; // le nombre de point d'intégration pour le calcul mécanique
568         int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
569         int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
570         int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse consistante
571         int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
572     };
573     static NombresConstruire nombre_V; // les nombres propres à l'élément
574
575     // fonction privée
576     // fonction d'initialisation servant au niveau du constructeur
577     void Init(Donnee_specif donnee_specif = Donnee_specif(),bool sans_init_noeud = false);
578     void Def_DonneeCommune();
579     // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
580     void Destruction();
581
582     // pour l'ajout d'element dans la liste : listTypeElement, geree par la class Element
583     class ConstrucElementbiel : public ConstrucElement
584     { public : ConstrucElementbiel ()
585         { NouvelleTypeElement nouv (POUT,HERMITE,MECA_SOLIDE_DEFORMABLE,this);
586         if (ParaGlob::NiveauImpression() >= 4)
587             cout << "\n initialisation BielletteC1" << endl;
588             Element::listTypeElement.push_back (nouv);
589         };
590         Element * NouvelElement(int num_mail,int num) // un nouvel élément sans rien
591         {Element * pt;
592         pt = new BielletteC1 (num_mail,num) ;
593         return pt;
594         };
595         // ramene true si la construction de l'element est possible en fonction
596         // des variables globales actuelles: ex en fonction de la dimension
597         bool Element_possible() {return true;};
598     };
599     static ConstrucElementbiel construcElementbiel;
600
601     // Calcul du residu local a t ou tdt en fonction du booleen
602     Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
603     // calcul des seconds membres suivant les chargements
604     // cas d'un chargement volumique,
605     // force indique la force volumique appliquée
606     // retourne le second membre résultant
607     // ici on l'épaisseur de l'élément pour constituer le volume
608     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
609     Vecteur SM_charge_volumique_E
610     (const Coordonnee& force,Fonction_nD* pt_fonct,bool atdt,const ParaAlgoControle & pa,bool
sur_volume_finale_);
611     // cas d'un chargement lineique, sur les aretes frontieres des éléments
612     // force indique la force lineique appliquée
613     // numarete indique le numéro de l'arete chargée
614     // retourne le second membre résultant
615     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
616     Vecteur SM_charge_lineique_E
617     (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
618     // cas d'un chargement lineique suiveuse, sur l'arete frontiere
619     //de la BielletteC1 (2D uniquement)
620     // force indique la force lineique appliquée
621     // numarete indique le numéro de l'arete chargée
622     // retourne le second membre résultant
623     // -> explicite à t
624     Vecteur SM_charge_lineique_Suiv_E
625     (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
626     // cas d'un chargement surfacique hydro-dynamique,
627     // voir méthode explicite plus haut, pour les arguments

```

```

628 // retourne le second membre résultant
629 // bool atdt : permet de spécifier à t ou a t+dt
630 Vecteur SM_charge_hydrodynamique_E( CourbeID* frot_fluid,const double& poidvol
631                                     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
632                                     , CourbeID* coef_aero_t,bool atdt,const
ParaAlgoControle & pa) ;
633
634 };
635 /// @} // end of group
636 #endif
637
638
639
640

```

## 7.132 BielletteQ.h

```

1 // FICHER : BielletteQ.h
2 // CLASSE : BielletteQ
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      26/01/2008
34 *
35 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT: // La classe BielletteQ permet de declarer des elements
41 *   biellettes quadratique du même type que biellette pour les éléments
42 *   linéaires. La dimension de l'espace pour un tel element est 1.
43 *
44 *   *****
45 *
46 *   VERIFICATION:
47 *
48 *   ! date ! auteur ! but
49 *   -----
50 *   ! ! !
51 *   $
52 *
53 *   MODIFICATIONS:
54 *
55 *   ! date ! auteur ! but
56 *   -----
57 *   $
58 *****/
59 // -----classe pour un calcul de mecanique-----
60
61 #ifndef BIELLETTEQ_H
62 #define BIELLETTEQ_H
63
64 #include "ParaGlob.h"
65 #include "ElemMeca.h"
66 //#include "Loi_comp_abstraite.h"

```

```

67 #include "Met_abstraite.h"
68 #include "Met_biellette.h"
69 #include "Noeud.h"
70 #include "UtilLecture.h"
71 #include "Tenseur.h"
72 #include "NevezTenseur.h"
73 #include "Deformation.h"
74 #include "ElFrontiere.h"
75 #include "GeomSeg.h"
76 #include "ParaAlgoControle.h"
77 #include "FrontSegQuad.h"
78 #include "Section.h"
79
80 class ConstrucElementbielQ;
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class BielletteQ : public ElemMeca
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93         // Constructeur par default
94         BielletteQ ();
95
96         // Constructeur fonction d'une section et eventuellement d'un numero
97         // d'identification et de maillage
98         BielletteQ (double sect,int num_maill=0,int num_id=-3);
99
100        // Constructeur fonction d'un numero de maillage et d'identification
101        BielletteQ (int num_maill,int num_id);
102
103        // Constructeur fonction d'une section, d'un numero de maillage et d'identification,
104        // du tableau de connexite des noeuds
105        BielletteQ (double sect,int num_maill,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        BielletteQ (const BielletteQ& biel);
109
110
111        // DESTRUCTEUR :
112        ~BielletteQ ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new BielletteQ(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de BielletteQ
119        BielletteQ& operator= (BielletteQ& biel);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123        // Lecture des donnees de la classe sur fichier
124        void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
125
126        // Calcul du residu local et de la raideur locale,
127        // pour le schema implicite
128        Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
129
130        // Calcul du residu local a t
131        // pour le schema explicite par exemple
132        Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
133        { return BielletteQ::CalculResidu(false,pa);};
134
135        // Calcul du residu local a tdt
136        // pour le schema explicite par exemple
137        Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
138        { return BielletteQ::CalculResidu(true,pa);};
139
140        // Calcul de la matrice masse pour l'élément
141        Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
142
143        // ----- calcul dynamique -----
144        // calcul de la longueur d'arrête de l'élément minimal
145        // divisé par la célérité la plus rapide dans le matériau
146        double Long_arrete_mini_sur_c(Enum_dure temps)
147        { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
148
149        //----- calcul d'erreur, remontée des contraintes -----
150        // 1)calcul du résidu et de la matrice de raideur pour le calcul d'erreur
151        Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
152        // 2) remontée aux erreurs aux noeuds
153        Element::Er_ResRaid ErreurAuNoeud_ResRaid();

```



```

154
155 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
156 // ce tableau et specifique a l'element
157 const DdlElement & TableauDdl() const ;
158
159
160 // Libere la place occupee par le residu et eventuellement la raideur
161 // par l'appel de Libere de la classe mere et libere les differents tenseurs
162 // intermediaires cree pour le calcul et les grandeurs pointees
163 // de la raideur et du residu
164 void Libere ();
165
166 // acquisition d'une loi de comportement
167 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
168
169 // test si l'element est complet
170 // = 1 tout est ok, =0 element incomplet
171 int TestComplet();
172
173 // procedure permettant de completer l'element apres
174 // sa creation avec les donnees du bloc transmis
175 // peut etre appelee plusieurs fois
176 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
177 // Compléter pour la mise en place de la gestion de l'hourglass
178 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
179
180 // ramene l'element geometrique
181 ElemGeomC0& ElementGeometrique() const { return doCo->segment;};
182 // ramene l'element geometrique en constant
183 const ElemGeomC0& ElementGeometrique_const() const { return doCo->segment;};
184
185 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
186 // associé
187 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
188 // 1) cas où l'on utilise la place passée en argument
189 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
190 void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
191
192 // affichage dans la sortie transmise, des variables duales "nom"
193 // dans le cas où nom est vide, affichage de "toute" les variables
194 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
195
196 // affichage d'info en fonction de ordre
197 // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
198 void Info_com_Element(UtilLecture * entreePrinc,string& ordre,Tableau<Noeud * > * tabMaillageNoeud)
199 { return Element::Info_com_El(2,entreePrinc,ordre,tabMaillageNoeud);};
200
201 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
202 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
203 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
204 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
205 // temps: dit si c'est à 0 ou t ou tdt
206 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
207 { return PtLePlusPres(temps,enu,M);};
208
209 // la grandeur enu
210 // temps: dit si c'est à 0 ou t ou tdt
211 // si erreur retourne erreur à true
212 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
213 { return CoordPtInt(temps,enu,iteg,erreur);};
214
215 // récupération des valeurs au numéro d'ordre = iteg pour
216 // les grandeur enu
217 Tableau <double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
218 enu,int iteg) ;
219
220 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
221 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
222 // de conteneurs quelconque associée
223 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int iteg);
224
225 // ramene vrai si la surface numéro ns existe pour l'élément
226 // dans le cas de la BiелletteQ il n'y a pas de surface
227 bool SurfExiste(int ) const
228 { return false;};
229
230 // ramene vrai si l'arête numéro na existe pour l'élément
231 bool AreteExiste(int na) const {if (na==1) return true; else return false;};
232
233 //===== lecture écriture dans base info =====
234
235 // cas donne le niveau de la récupération
236 // = 1 : on récupère tout
237 // = 2 : on récupère uniquement les données variables (supposées comme telles)
238 void Lecture_base_info

```

```

239     (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
240     // cas donne le niveau de sauvegarde
241     // = 1 : on sauvegarde tout
242     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
243     void Ecriture_base_info(ofstream& sort,const int cas) ;
244
245     // METHODES VIRTUELLES:
246     // ----- calculs utiles dans le cadre de la recherche du flambement linéaire
247     // Calcul de la matrice géométrique et initiale
248     ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
249
250     // retourne la liste des données particulières actuellement utilisés
251     // par l'élément (actif ou non), sont exclu de cette liste les données particulières des noeuds
252     // reliés à l'élément
253     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
254     List_io <TypeQuelconque> Les_types_particuliers_internes(bool absolue) const;
255
256     // récupération de grandeurs particulières au numéro d'ordre = iteg
257     // celles-ci peuvent être quelconques
258     // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
259     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
260     void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);
261
262     // inactive les ddl du problème primaire de mécanique
263     inline void Inactive_ddl_primaire()
264     {ElemMeca::Inact_ddl_primaire(doCo->tab_ddl);};
265     // active les ddl du problème primaire de mécanique
266     inline void Active_ddl_primaire()
267     {ElemMeca::Act_ddl_primaire(doCo->tab_ddl);};
268     // ajout des ddl de contraintes pour les noeuds de l'élément
269     inline void Plus_ddl_Sigma()
270     {ElemMeca::Ad_ddl_Sigma(doCo->tab_ddlErr);};
271     // inactive les ddl du problème de recherche d'erreur : les contraintes
272     inline void Inactive_ddl_Sigma()
273     {ElemMeca::Inact_ddl_Sigma(doCo->tab_ddlErr);};
274     // active les ddl du problème de recherche d'erreur : les contraintes
275     inline void Active_ddl_Sigma()
276     {ElemMeca::Act_ddl_Sigma(doCo->tab_ddlErr);};
277     // active le premier ddl du problème de recherche d'erreur : SIGMA11
278     inline void Active_premier_ddl_Sigma()
279     {ElemMeca::Act_premier_ddl_Sigma();};
280
281     // lecture de données diverses sur le flot d'entrée
282     void LectureContraintes(UtilLecture * entreePrinc)
283     {if (unefois.CalResPrem_t == 1)
284         ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
285     else
286         { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
287           unefois.CalResPrem_t = 1;
288         }
289     };
290
291     // retour des contraintes en absolu retour true si elle existe sinon false
292     bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
293     { if (unefois.CalResPrem_t == 1)
294         ElemMeca::ContraintesEnAbsolues(false,lesPtMecaInt.TabSigHH_t(),tabSig);
295     else
296         { unefois.CalResPrem_t = 1;
297           ElemMeca::ContraintesEnAbsolues(true,lesPtMecaInt.TabSigHH_t(),tabSig);
298         }
299     return true;
300     };
301
302
303     // 2) derivant des virtuelles
304     // retourne un tableau de ddl element, correspondant à la
305     // composante de sigma -> SIG11, pour chaque noeud qui contient
306     // des ddl de contrainte
307     // -> utilisé pour l'assemblage de la raideur d'erreur
308     inline DdlElement& Tableau_de_Sig11() const
309     {return doCo->tab_Err1Sig11;} ;
310
311     // actualisation des ddl et des grandeurs actives de t+dt vers t
312     void TdtversT();
313     // actualisation des ddl et des grandeurs actives de t vers tdt
314     void TversTdt();
315
316     // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
317     // qu'une fois la remontée aux contraintes effectuées sinon aucune
318     // action. En retour la valeur de l'erreur sur l'élément
319     // type indique le type de calcul d'erreur :
320     void ErreurElement(int type,double& errElemRelative
321         ,double& numerateur, double& denominateur);
322
323     // mise à jour de la boîte d'encombrement de l'élément, suivant les axes I_a globales
324     // en retour coordonnées du point mini dans retour.Premier() et du point maxi dans .Second()
325     // la méthode est différente de la méthode générale car il faut prendre en compte l'épaisseur de

```

```

l'élément
326 virtual const DeuxCoordonnees& Boite_encombre_element(Enum_dure temps);
327
328 // calcul des seconds membres suivant les chargements
329 // cas d'un chargement volumique,
330 // force indique la force volumique appliquée
331 // retourne le second membre résultant
332 // ici on l'épaisseur de l'élément pour constituer le volume
333 // -> explicite à t
334 Vecteur SM_charge_volumique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle &
pa,bool sur_volume_finale_)
335 { return BielleQ::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_); } ;
336 // -> explicite à tdt
337 Vecteur SM_charge_volumique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle
& pa,bool sur_volume_finale_)
338 { return BielleQ::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_); } ;
339 // -> implicite,
340 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
341 // retourne le second membre et la matrice de raideur correspondant
342 ResRaid SMR_charge_volumique_I(const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle
& pa,bool sur_volume_finale_);
343
344 // cas d'un chargement lineique, sur les aretes frontières des éléments
345 // force indique la force lineique appliquée
346 // numarete indique le numéro de l'arete chargée
347 // retourne le second membre résultant
348 // -> explicite à t
349 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
350 { return BielleQ::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); } ;
351 // -> explicite à tdt
352 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
353 { return BielleQ::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); } ;
354 // -> implicite,
355 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
356 // retourne le second membre et la matrice de raideur correspondant
357 ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa);
358
359 // cas d'un chargement lineique suivieuse, sur l'arrête frontière de
360 // la BielleQ (2D uniquement)
361 // force indique la force lineique appliquée
362 // numarete indique le numéro de l'arete chargée
363 // retourne le second membre résultant
364 // -> explicite à t
365 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
366 { return BielleQ::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa); } ;
367 // -> explicite à tdt
368 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
369 { return BielleQ::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa); } ;
370 // -> implicite,
371 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
372 // retourne le second membre et la matrice de raideur correspondant
373 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa);
374
375 // cas d'un chargement surfacique hydro-dynamique,
376 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
377 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc unitaire)
378 // une suivant la direction normale à la vitesse de type portance
379 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
380 // une suivant la vitesse tangente de type frottement visqueux
381 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
382 // coef_mul: est un coefficient multiplicateur global (de tout)
383 // retourne le second membre résultant
384 // -> explicite à t
385 Vecteur SM_charge_hydrodynamique_E_t( CourbeID* frot_fluid,const double& poidvol
, CourbeID* coef_aero_n,int numFace,const double&
coef_mul
, CourbeID* coef_aero_t,const ParaAlgoControle & pa)
386 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
387 // -> explicite à tdt
388 Vecteur SM_charge_hydrodynamique_E_tdt( CourbeID* frot_fluid,const double& poidvol
, CourbeID* coef_aero_n,int numFace,const double&
coef_mul
, CourbeID* coef_aero_t,const ParaAlgoControle & pa)
389 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};
390 // -> implicite,
391 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
392 // retourne le second membre et la matrice de raideur correspondant
393 ResRaid SMR_charge_hydrodynamique_I( CourbeID* frot_fluid,const double& poidvol

```

```

398                                     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
399                                     , CourbeID* coef_aero_t,const ParaAlgoControle & pa)
;
400
401
402 // ===== définition et/ou construction des frontières =====
403
404     // Calcul des frontieres de l'element
405     // creation des elements frontieres et retour du tableau de ces elements
406     // la création n'a lieu qu'au premier appel
407     // ou lorsque l'on force le paramètre force a true
408     // dans ce dernier cas seul les frontière effacées sont recréée
409     Tableau <ElFrontiere*> const & Frontiere(bool force = false);
410
411 // Retourne la section de l'element
412 inline double S(Enum_dure enu = TEMPS_tdt)
413 { switch (enu)
414   { case TEMPS_0: return donnee_specif.secti.section0; break;
415     case TEMPS_t: return donnee_specif.secti.section_t; break;
416     case TEMPS_tdt: return donnee_specif.secti.section_tdt; break;
417   };
418   return 0.; // cas n'arrivant normalement jamais
419 };
420
421 // ajout du tableau specific de ddl des noeuds de la BielleQ
422 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
423 // des noeuds constituant l'element
424 void ConstTabDdl();
425
426 protected:
427
428     // ==== »» methodes virtuelles dérivant d'ElemMeca =====
429     // ramene la dimension des tenseurs contraintes et déformations de l'élément
430     int Dim_sig_eps() const {return 1;};
431
432 // ----- calcul de frontières en protected -----
433
434     // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
particulière à l'élément
435     // adressage des frontières linéiques et surfacique
436     // définit dans les classes dérivées, et utilisées pour la construction des frontières
437     virtual ElFrontiere* new_frontiere_lin(int , Tableau <Noeud *> & tab, DdlElement& ddelem)
438     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
439     virtual ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
440     {return NULL; } // il n'y a pas de surface possible
441
442 private :
443
444     // VARIABLES PRIVEES :
445
446     class DonneeCommune
447     { public :
448         DonneeCommune (GeomSeg& seg,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
449             Met_bielleQ & met_bie,
450             Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
451             GeomSeg& seEr,Vecteur& residu_int,Mat_pleine& raideur_int,
452             Tableau <Vecteur*> & residus_extN,Tableau <Mat_pleine*> & raideurs_extN,
453             Tableau <Vecteur*> & residus_extA,Tableau <Mat_pleine*> & raideurs_extA,
454             Mat_pleine& mat_masse ,GeomSeg& seMa,int nbi,GeomSeg* segHourg
455         );
456         DonneeCommune(DonneeCommune& a);
457         ~DonneeCommune();
458         // variables
459         GeomSeg segment ; // element geometrique correspondant
460         DdlElement tab_ddl; // tableau des degres
461         //de liberte des noeuds de l'element commun a tous les
462         // elements
463         Met_bielleQ met_BielleQ;
464         Mat_pleine matGeom ; // matrice géométrique
465         Mat_pleine matInit ; // matrice initiale
466         Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
467         Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
468         Tableau < Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
469         // calcul d'erreur
470         DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
471         // d'erreur : contraintes
472         DdlElement tab_Err1Sig1; // tableau du ddl SIG11 pour chaque noeud,
473         //servant pour le calcul d'erreur : contraintes, en fait pour l'assemblage
474         Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
475         Mat_pleine raidErr; // raideur pour le calcul d'erreur
476         GeomSeg segmentEr; // contient les fonctions d'interpolation et
477         // les derivees pour le calcul du hessien dans
478         //la résolution de la fonctionnelle d'erreur
479         // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
480         -----

```

```

481         // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
482         Vecteur residu_interne;
483         Mat_pleine raideur_interne;
484         Tableau <Vecteur* > residus_externesN; // pour les noeuds
485         Tableau <Mat_pleine* > raideurs_externesN; // pour les noeuds
486         Tableau <Vecteur* > residus_externesA; // pour l' aretes
487         Tableau <Mat_pleine* > raideurs_externesA; // pour l' aretes
488         // ----- données concernant la dynamique -----
489         Mat_pleine matrice_masse;
490         GeomSeg segmentMas; // contient les fonctions d'interpolation et les dérivées
491         // pour les calculs relatifs au calcul de la masse
492         // ----- blocage éventuel d'hourglass
493         // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
Cal_explici_hourglass
494         GeomSeg* segmentHourg; // contient les fonctions d'interpolation
495     };
496
497
498     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
499     // et un pointeur sur les données statiques communes
500     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
501     // classe est défini. Son allocation est effectuée dans les classes dérivées
502     class UneFois
503     { public :
504         UneFois () ; // constructeur par défaut
505         ~UneFois () ; // destructeur
506
507         // VARIABLES :
508     public :
509         DonneeCommune * doCoMemb;
510
511         // incicateurs permettant de dimensionner seulement au premier passage
512         // utilise dans "CalculResidu" et "Calcul_implicit"
513         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
514         int CalimpPrem;
515         int dualSortbiel; // pour la sortie des valeurs au pt d'integ
516         int CalSMLin_t; // pour les seconds membres concernant les arretes
517         int CalSMLin_tdt; // pour les seconds membres concernant les arretes
518         int CalSMRlin; // pour les seconds membres concernant les arretes
519         int CalSMvol_t; // pour les seconds membres concernant les volumes
520         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
521         int CalSMvol; // pour les seconds membres concernant les volumes
522         int CalDynamique; // pour le calcul de la matrice de masse
523         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
524         // ----- sauvegarde du nombre d'élément en cours -----
525         int nbelem_in_Prog;
526     };
527
528
529     // -----
530
531     protected :
532         // VARIABLES PROTÉGÉES :
533         // les données spécifiques sont groupées dans une structure pour sécuriser
534         // le passage de paramètre dans init par exemple
535         class Donnee_specif
536         { public :
537             Donnee_specif() : // défaut
538                 secti(Element::section_defaut,Element::section_defaut,Element::section_defaut)
539                 , fctnD_section(NULL), variation_section(true)
540             {};
541             Donnee_specif(double section) : // uniquement la section
542                 secti(section,section,section), fctnD_section(NULL), variation_section(true)
543             {};
544             Donnee_specif(double epai0,double epai_t,double epai_tdt,Fonction_nD* fctnD,bool variation) :
545             // tous
546                 secti(epai0,epai_t,epai_tdt), fctnD_section(fctnD), variation_section(variation)
547             {};
548             Donnee_specif(const Donnee_specif& a) :
549                 secti(a.secti ), fctnD_section(a.fctnD_section), variation_section(a.variation_section)
550                 {}; // recopie via le constructeur de copie
551             ~Donnee_specif() {};
552             Donnee_specif & operator = ( const Donnee_specif& a)
553                 { secti = a.secti; fctnD_section=a.fctnD_section; variation_section=a.variation_section;
554                 return *this;};
555             // data
556             // sections de l'element
557             Sect secti; // épaisseur
558             Fonction_nD* fctnD_section; // permet éventuellement de prendre en compte une évolution de la
559             section
560             bool variation_section; // permet éventuellement de ne pas prendre en compte la variation
561         };
562
563         Donnee_specif donnee_specif;
564
565         // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
566         LesPtIntegMecaInterne lesPtMecaInt;

```

```

565
566 // place memoire commune a tous les elements BielleQs
567 static DonneCommune * doCo;
568 // idem mais pour les indicateurs qui servent pour l'initialisation
569 static UneFois unefois;
570
571 // type structuré pour construire les éléments
572 class NombresConstruire
573 { public:
574     NombresConstruire();
575     int nbne; // le nombre de noeud de l'élément
576     int nbneA ; // le nombre de noeud des aretes
577     int nbi; // le nombre de point d'intégration pour le calcul mécanique
578     int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
579     int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
580     int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse consistante
581     int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
582 };
583 static NombresConstruire nombre_V; // les nombres propres à l'élément
584
585 // fonction privée
586 // fonction d'initialisation servant au niveau du constructeur
587 BielleQ::DonneCommune * Init(Donnee_specif donnee_specif = Donnee_specif()
588                             ,bool sans_init_noeud = false);
589 void Def_DonneCommune();
590 // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
591 void Destruction();
592
593 // pour l'ajout d'element dans la liste : listTypeElement, gérée par la class Element
594 class ConstrucElementbielQ : public ConstrucElement
595 { public : ConstrucElementbielQ ()
596     { NouvelleTypeElement nouv(POUT,BIE2,MECA_SOLIDE_DEFORMABLE,this);
597     if (ParaGlob::NiveauImpression() >= 4)
598     cout << "\n initialisation BielleQ" << endl;
599     Element::listTypeElement.push_back(nouv);
600     };
601     Element * NouvelElement(int nb_mail,int num) // un nouvel élément sans rien
602     {Element * pt;
603     pt = new BielleQ (nb_mail,num) ;
604     return pt;
605     };
606     // ramene true si la construction de l'element est possible en fonction
607     // des variables globales actuelles: ex en fonction de la dimension
608     bool Element_possible() {return true;};
609 };
610 static ConstrucElementbielQ construcElementbielQ;
611
612 // Calcul du residu local a t ou tdt en fonction du booleen
613 Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
614 // calcul des seconds membres suivant les chargements
615 // cas d'un chargement volumique,
616 // force indique la force volumique appliquée
617 // retourne le second membre résultant
618 // ici on l'épaisseur de l'élément pour constituer le volume
619 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
620 Vecteur SM_charge_volumique_E
621 (const Coordonnee& force,Fonction_nD* pt_fonct,bool atdt,const ParaAlgoControle &
pa,bool sur_volume_finale_);
622 // cas d'un chargement lineique, sur les aretes frontieres des éléments
623 // force indique la force lineique appliquée
624 // numarete indique le numéro de l'arete chargée
625 // retourne le second membre résultant
626 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
627 Vecteur SM_charge_lineique_E
628 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
629 // cas d'un chargement lineique suivieuse, sur l'arete frontiere
630 //de la BielleQ (2D uniquement)
631 // force indique la force lineique appliquée
632 // numarete indique le numéro de l'arete chargée
633 // retourne le second membre résultant
634 // -> explicite à t
635 Vecteur SM_charge_lineique_Suiv_E
636 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
637 // cas d'un chargement surfacique hydro-dynamique,
638 // voir méthode explicite plus haut, pour les arguments
639 // retourne le second membre résultant
640 // bool atdt : permet de spécifier à t ou a t+dt
641 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
642 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
643 , CourbelD* coef_aero_t,bool atdt,const
ParaAlgoControle & pa) ;
644
645 // calcul de la nouvelle section moyenne finale (sans raideur)
646 // mise à jour des volumes aux pti

```

```

647 // ramène la section moyenne calculée à atdt
648 const double& CalSectionMoyenne_et_vol_pti(const bool atdt);
649
650 };
651 /// @} // end of group
652 #endif
653
654
655
656

```

## 7.133 DeformationP2D.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      25/05/98
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: Calcul des differentes grandeurs liee a la deformation
39 *   des elements poutre 2D.
40 *
41 *   La particularité de cette classe par rapport à DeformationPP
42 *   est que l'on calcul la courbure directement à partir de la
43 *   dérivée seconde des coordonnées.
44 *
45 *   La classe fonctionne comme une boite a outil. On y choisit ce
46 *   dont on a besoin. Bien faire attention a l'ordre d'appel des
47 *   differentes methodes lorsque il faut suivre une chronologie.
48 *   Cette classe calcul mais ne stock pas.
49 *
50 *   *****
51 *
52 *   VERIFICATION:
53 *   ! date !   auteur !           but
54 *   -----
55 *   !           !           !           !
56 *
57 *   *****
58 *   MODIFICATIONS:
59 *   ! date !   auteur !           but
60 *   -----
61 *
62 *****/
63 #ifndef DEFORMATIONP2D_H
64 #define DEFORMATIONP2D_H
65
66 #include "Tableau_T.h"
67 #include "Met_abstraite.h"
68 #include "DeformationPP.h"
69
70 /// @addtogroup groupe_des_deformations
71 /// @

```

```

72 ///
73
74
75 class DeformationP2D : public DeformationPP
76 {
77 public :
78
79
80 // CONSTRUCTEURS :
81 DeformationP2D () ; // par défaut ne doit pas être utilisé -> message
82 // d'erreur en phase de mise au point
83 // constructeur normal dans le cas d'un ou de plusieurs pt d'intégration
84 // tabDphi et tabPhi sont relatifs aux fonctions d'interpolation
85 // la terminaison H est relative aux grandeurs d'épaisseur, S pour la surface
86 // ----> tabD2phi est relatif aux dérivées secondes des fonctions d'interpolations sur l'axe
87
88 DeformationP2D (Met_abstraite & a, Tableau<Noeud *>& tabnoeud
89 , Tableau <Mat_pleine> const & tabDphiH, Tableau <Vecteur> const & tabPhiH
90 , Tableau <Mat_pleine> const & tabDphiS, Tableau <Vecteur> const & tabPhiS
91 , Tableau <Vecteur> const & tabD2phi);
92 // constructeur de copie
93 DeformationP2D (const DeformationP2D &);
94 // DESTRUCTEUR :
95 virtual ~DeformationP2D ();
96
97 // ===== changement de grandeurs stockées =====
98
99 // définition du déformation du même type, permet d'utiliser des types dérivées surchargées
100 virtual Deformation * Nevez_deformation (Tableau <Noeud *> & tabN) const
101 { DeformationP2D* def = new DeformationP2D (*this); def->PointeurTableauNoeud (tabN); return def; };
102
103 // Surcharge de l'opérateur = : réalise l'affectation
104 // fonction virtuelle, normalement ne devrait pas être utilisée
105 // si c'est le cas -> affichage d'un message d'erreur
106 Deformation& operator= (const Deformation& def);
107 DeformationPP& operator= (const DeformationPP& def);
108 // fonction normale
109 DeformationP2D& operator= (const DeformationP2D& def);
110 // change les numéros d'intégration de l'axe ou du plan et d'épaisseur courante
111 // ntotalaxpl : nombre total de pt d'integ de l'axe ou du plan
112 // niaxpl : nouveau point de l'axe ou du plan
113 // niepaiss : nouveau point d'épaisseur
114 // epaisseur : l'épaisseur courante
115 void ChangeNumIntegSH (int niaxpl, int niepaiss);
116
117 protected :
118 // VARIABLES PROTEGEES :
119 Tableau <Vecteur> const * tabD2phi; // dérivées secondes des fonctions d'interpolation
120 // tabD2phi(i)(j) : dérivée seconde par rapport à la seule variable l,
121 // de phi du noeud j au point d'intégration i.
122 };
123 /// @} // end of group
124
125 #endif

```

## 7.134 Met\_biellette.h

```

1 // FICHER : Met_biellette.h
2 // CLASSE : Met_biellette
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License

```



```

28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *
35 *   DATE:      15/01/97
36 *
37 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:    Herezh++
40 *
41 *   $
42 * *****
43 * La classe Met_biellette est une classe derivee de la classe Met_abstraite
44 * et permet de realiser tous les calculs lies a la metrique d'une biellette.
45 * N.B. : La dimension de l'espace est 1.
46 *
47 *   $
48 * *****
49 * VERIFICATION:
50 *
51 *   ! date !   auteur !       but
52 *   -----
53 *   !       !       !
54 *
55 *   $
56 * *****
57 * MODIFICATIONS:
58 *
59 *   ! date !   auteur !       but
60 *   -----
61 *
62 *   $
63 * *****/
64
65 // La classe Met_biellette est une classe derivee de la classe Met_abstraite
66 // et permet de realiser tous les calculs lies a la metrique d'une biellette.
67 // N.B. : La dimension de l'espace est 1.
68
69
70 #ifndef MET_BIELLETTE_H
71 #define MET_BIELLETTE_H
72
73 #include "Met_abstraite.h"
74
75 /// @addtogroup groupe_des_metrrique
76 /// @{
77 ///
78
79
80
81 class Met_biellette : public Met_abstraite
82 {
83
84     public :
85
86         // CONSTRUCTEUR :
87
88         // Constructeur par defaut
89         Met_biellette ();
90         // constructeur permettant de dimensionner unique ment certaine variables
91         // dim = dimension de l'espace, tab = liste
92         // des variables a initialiser
93         Met_biellette (int dim_base,const DdlElement& tabddl,
94             const Tableau<Enum_variable_metrrique> & tab,int nomb_noeud);
95         // constructeur de copie
96         Met_biellette (const Met_biellette&);
97         // DESTRUCTEUR :
98
99         ~Met_biellette ();
100
101         // Surcharge de l'operateur = : realise l'affectation
102         // fonction virtuelle
103         inline Met_abstraite& operator= (const Met_abstraite& met)
104             { return (Met_abstraite::operator=(met)); };
105
106     protected :
107         // METHODES protegees:
108         //==calcul des points, identique a Met_abstraite
109
110
111         //== le nombre de vecteur de base pour la biellette est 1
112         //== les fonctions de calcul de base sont donc redefini
113

```

```

114         // calcul de la base naturel a t0
115         void Calcul_giB_0
116         ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud, const
Vecteur& phi);
117         // calcul de la base naturel a t
118         void Calcul_giB_t
119         ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud, const
Vecteur& phi);
120         // calcul de la base naturel a t+dt
121         void Calcul_giB_tdt
122         ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud, const
Vecteur& phi);
123         //== calcul de la variation des jacobiens
124         void Djacobien_t(); // avant calcul de : d_giB_t et giB_t
125         void Djacobien_tdt(); // avant calcul de : d_giB_tdt et giB_tdt
126
127     };
128     /// @} // end of group
129
130
131 #endif
132
133

```

## 7.135 Met\_BielletteC1.h

```

1 // FICHER : Met_BielletteC1.h
2 // CLASSE : Met_BielletteC1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      05/06/2006
34 *
35 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 *      BUT:      La classe Met_biellette est une classe derivée de la
41 *               classe Met_abstraite et permet de réaliser tous les calculs
42 *               liés à la métrique d'une bielle de continuité C1.
43 *               N.B. : La dimension de l'espace est 1.
44 *               A chaque noeud il y a deux ddl, X et dX/dxi
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *
50 *      ! date ! auteur ! but
51 *      -----
52 *      ! ! ! !
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *      $
58 *****/

```

```

59
60
61
62 #ifndef MET_BIELLETTE_C1_H
63 #define MET_BIELLETTE_C1_H
64
65 #include "Met_abstraite.h"
66
67
68 /// @addtogroup groupe_des_metrique
69 /// @{
70 ///
71
72
73 class Met_BielletteC1 : public Met_abstraite
74 {
75
76
77     public :
78
79
80         // CONSTRUCTEUR :
81
82         // Constructeur par défaut
83         Met_BielletteC1 ();
84         // constructeur permettant de dimensionner uniquement certaines variables
85         // dim = dimension de l'espace, tab = liste
86         // des variables à initialiser
87         Met_BielletteC1 (int dim_base,const DdlElement& tabddl,
88                         const Tableau<Enum_variable_metrique> & tab,int nomb_noeud);
89         // constructeur de copie
90         Met_BielletteC1 (const Met_BielletteC1&);
91         // DESTRUCTEUR :
92
93         ~Met_BielletteC1 ();
94         // Surcharge de l'opérateur = : réalise l'affectation
95         // fonction virtuelle
96         inline Met_abstraite& operator= (const Met_abstraite& met)
97         { return (Met_abstraite::operator=(met)); };
98
99     protected :
100         // METHODES protegee:
101         //==calcul des points, identique à Met_abstraite
102
103
104         //== le nombre de vecteur de base pour la BielleteteC1 est 1
105         //== les fonctions de calcul de base sont donc redéfini
106         // les fonctions d'interpolation sont redéfinies: les premières sont relatives aux X_ar et les
107         // dernières
108         // au d X_ar/d xi, sachant qu'il y en a exactement le même nombre
109         // calcul de la base naturel a t0
110         void Calcul_giB_0
111         ( const Tableau<Noeud *>& tab_noeud,const Mat_pleine& dphi, int nombre_noeud,const
112         Vecteur& phi);
113         // calcul de la base naturel a t
114         void Calcul_giB_t
115         ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud,const
116         Vecteur& phi);
117         // calcul de la base naturel a t+dt
118         void Calcul_giB_tdt
119         ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud,const
120         Vecteur& phi);
121         //== calcul de la variation des jacobiens
122         void Djacobien_t(); // avant calcul de : d_giB_t et giB_t
123         void Djacobien_tdt(); // avant calcul de : d_giB_tdt et giB_tdt
124
125 };
126 /// @} // end of group
127
128 #endif

```

## 7.136 Met\_pout2D.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.

```

```

10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29 //
30 /*****
31 *   DATE:           4/06/98
32 *
33 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:        Herezh++
36 *
37 *
38 *   BUT: Met_Pout2D constitue une boite a outil comme met_abstraite,
39 *   mais ici c'est particulièrement dédié aux poutres 2D.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *   $
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *   $
53 *****/
54 #ifndef MET_POUT2D_H
55 #define MET_POUT2D_H
56
57 #include "Met_PiPoCo.h"
58
59
60 /// @addtogroup groupe_des_metrrique
61 /// @{
62 ///
63
64
65 class Met_Pout2D : public Met_PiPoCo
66 {
67 public :
68     // CONSTRUCTEUR :
69
70     // Constructeur par défaut
71     Met_Pout2D ();
72     // constructeur permettant de dimensionner unique ment certaine variables
73     // dim = dimension de l'espace, tab = liste
74     // des variables a initialiser
75     Met_Pout2D (int dim_base,int nbvec,const DdElement& tabddl,
76                const Tableau<Enum_variable_metrrique>& tabb,int nomb_noeud);
77     // constructeur de copie
78     Met_Pout2D (const Met_Pout2D&);
79     // DESTRUCTEUR :
80     ~Met_Pout2D ();
81
82     // METHODES PUBLIQUES :
83     // Surcharge de l'operateur = : realise l'affectation
84     // dérivant de virtuel, a ne pas employer -> message d'erreur
85     Met_abstraite& operator= (const Met_abstraite& met);
86     Met_PiPoCo& operator= (const Met_PiPoCo& met);
87     // normale
88     virtual Met_Pout2D& operator= (const Met_Pout2D& met);
89
90     // passage de la dérivée seconde
91     void DeriveeSeconde(Vecteur const & tabD2phi);
92
93
94     // ===== protege =====
95

```

```

96     protected :
97         // calcul des normales a la poutre
98         void Calcul_N_0 ();
99         void Calcul_N_t ();
100        void Calcul_N_tdt ();
101        //== les fonctions de calcul de base donc redefini
102        // calcul de la base naturel a t0
103        // pendant le traitement il y a calcul de la base aiB et aiH de l'axe median
104        // la fonction contient un argument de plus que la routine dans met_abstraite
105        void Calcul_giB_0 ( const  Tableau<Noeud *>& ,const  Mat_pleine& ,
106                          int, const  Vecteur& phi);
107        // calcul de la base naturel a t
108        void Calcul_giB_t ( const  Tableau<Noeud *>& ,const  Mat_pleine& ,
109                          int, const  Vecteur& phi);
110        // calcul de la base naturel a t+dt
111        void Calcul_giB_tdt ( const  Tableau<Noeud *>& ,const  Mat_pleine& ,
112                             int, const  Vecteur& phi);
113        //== calcul de la variation des bases
114        void D_giB_t( const  Mat_pleine& , int , const  Vecteur & phi);
115        void D_giB_tdt(const  Mat_pleine& , int , const  Vecteur & phi);
116        //-----// calcul du tenseur de courbure dans la base naturelle
117        // plusieurs cas sont etudies suivant l'instant considere
118        // a l'instant t = 0
119        Vecteur& courbure_0 (const  Tableau<Noeud *>& tab_noeud);
120        // a l'instant t
121        Vecteur& courbure_t (const  Tableau<Noeud *>& tab_noeud);
122        // a l'instant t+dt
123        Vecteur& courbure_tdt (const  Tableau<Noeud *>& tab_noeud);
124        // routine generale de calcul de la courbure
125        double courbure(const  Tableau<Coordonnee>& tab_coor,
126                      const  CoordonneeB & aiB1,const  CoordonneeH & );
127        //-----// calcul du tenseur de courbure et de sa variation
128        // plusieurs cas sont etudies suivant l'instant considere
129        // a l'instant t
130        void Dcourbure_t(const  Tableau<Noeud *>& tab_noeud,  Vecteur& curb,
131                        TabOper<Vecteur>& dcurb);
132        // a l'instant tdt
133        void Dcourbure_tdt(const  Tableau<Noeud *>& tab_noeud,
134                          Vecteur& curb,TabOper<Vecteur>& dcurb);
135        // routine generale de calcul de la courbure et de sa variation
136        // en sortie : curb , la courbure  b11,b12,b22
137        //          dcurb , la variation de courbure.
138        void Dcourbure (const  Tableau<Coordonnee>& tab_coor,const  CoordonneeB & aiB1,
139                      const  CoordonneeH & ,Tableau <BaseB>& DaiB,
140                      Vecteur& curb,TabOper <Vecteur>& dcurb);
141
142        // DONNEES PROTEGEES
143        Vecteur  const * tabD2phi; // pointeur sur les dérivées secondes courantes
144
145        // fonctions privees
146
147
148 };
149 /// @} // end of group
150
151
152 #endif
153
154

```

## 7.137 PoutSimple1.h

```

1 // FICHER : PoutSimple1.h
2 // CLASSE : PoutSimple1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty

```

```

24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      23/01/97
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT: // La classe PoutSimple1 permet de declarer des elements
41 * de flexion très simple, qui à priori, sont prévus pour fonctionner
42 * seul, c'est à dire un seul élément, ceci pour tester le flambage .
43 * La classe permet de realiser le calcul du residu local et de la
44 * raideur locale pour une loi de comportement donnee. La dimension de
45 * l'espace de référence pour un tel element est 1, celle de l'espace
46 * physique est uniquement 2. Les déplacements prévus sont U dans l'axe
47 * et W transversalement.
48 * La section de la poutre est rectangulaire, épaisseur x largeur.
49 *
50 *   *****
51 *
52 *   VERIFICATION:
53 *   ! date !   auteur !           but
54 *   -----
55 *   !           !           !
56 *   $
57 *   *****
58 *   MODIFICATIONS:
59 *   ! date !   auteur !           but
60 *   -----
61 *   $
62 *****/
63 // -----classe pour un calcul de mecanique-----
64
65
66
67 #ifndef POUTSIMPLE1_H
68 #define POUTSIMPLE1_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 // #include "Loi_comp_abstraite.h"
73 #include "Met_abstraite.h"
74 #include "Met_pout2D.h"
75 #include "Noeud.h"
76 #include "UtilLecture.h"
77 #include "ElFrontiere.h"
78 #include "GeomSeg.h"
79 #include "PiPoCo.h"
80 #include "ParaAlgoControle.h"
81 #include "FrontSegLine.h"
82
83 class ConstrucElementbiel;
84
85 /// @addtogroup groupe_des_elements_finis
86 /// @{
87 ///
88
89
90 class PoutSimple1 : public PiPoCo
91 {
92     public :
93
94         // CONSTRUCTEURS :
95         // Constructeur par default
96         PoutSimple1 ();
97
98         // Constructeur fonction de la hauteur, la largeur et eventuellement d'un numero
99         // d'identification
100        PoutSimple1 (double epais,double larg,int num_mail=0,int num_id=-3);
101
102        // Constructeur fonction d'un numero de maillage et d'identification
103        PoutSimple1 (int num_mail,int num_id);
104
105        // Constructeur fonction de la hauteur, la largeur, d'un numero d'identification,
106        // du tableau de connexite des noeuds
107        PoutSimple1 (double epais,double larg,int num_mail,int num_id,const Tableau<Noeud *>& tab);
108
109

```

```

110     // Constructeur de copie
111     PoutSimple1 (const PoutSimple1& biel);
112
113
114     // DESTRUCTEUR :
115     ~PoutSimple1 ();
116
117     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
118     // méthode virtuelle
119     Element* Nevez_copie() const { Element * el= new PoutSimple1(*this); return el;};
120
121     // Surcharge de l'operateur = : realise l'egalite entre deux instances de PoutSimple1
122     PoutSimple1& operator= (PoutSimple1& pout);
123
124     // METHODES :
125 // 1) derivant des virtuelles pures
126 // Lecture des donnees de la classe sur fichier
127     void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
128
129     // affichage d'info en fonction de ordre
130     // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
131     void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> *
132     tabMaillageNoeud)
133     { return Element::Info_com_El (2,entreePrinc,ordre,tabMaillageNoeud);};
134
135     // Calcul du residu local et de la raideur locale,
136     // pour le schema implicite
137     Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
138
139     // Calcul du residu local a t
140     // pour le schema explicit par exemple
141     Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
142     { return PoutSimple1::CalculResidu(false,pa);};
143
144     // Calcul du residu local a tdt
145     // pour le schema explicit par exemple
146     Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
147     { return PoutSimple1::CalculResidu(true,pa);};
148
149     // Calcul de la matrice masse pour l'élément
150     Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
151
152     // ----- calcul dynamique -----
153     // calcul de la longueur d'arrête de l'élément minimal
154     // divisé par la célérité la plus rapide dans le matériau
155     double Long_arrete_mini_sur_c (Enum_dure temps)
156     { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
157
158     // retourne les tableaux de ddl associés aux noeuds, gere par l'element
159     // ce tableau est specifique a l'element
160     const DdlElement & TableauDdl() const ;
161
162     // actualisation des ddl et des grandeurs actives de t+tdt vers t
163     void TdtversT();
164     // actualisation des ddl et des grandeurs actives de t vers tdt
165     void TversTdt();
166
167     // Libere la place occupee par le residu et eventuellement la raideur
168     // par l'appel de Libere de la classe mere et libere les differents tenseurs
169     // intermediaires cree pour le calcul et les grandeurs pointee
170     // de la raideur et du residu
171     void Libere ();
172
173     // acquisition d'une loi de comportement
174     void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
175
176     // test si l'element est complet
177     // = 1 tout est ok, =0 element incomplet
178     int TestComplet();
179
180     // procedure permettant de completer l'element apres
181     // sa creation avec les donnees du bloc transmis
182     // peut etre appeler plusieurs fois
183     Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
184     // Compléter pour la mise en place de la gestion de l'hourglass
185     Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc) {return
186     this;};
187
188     // ramene l'element geometrique
189     ElemGeomC0& ElementGeometrique() const { return doCoPS1->segmentL;};
190     // ramene l'element geometrique en constant
191     const ElemGeomC0& ElementGeometrique_const() const { return doCoPS1->segmentL;};
192
193     // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
194     associé

```

```

193     // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
194     // 1) cas où l'on utilise la place passée en argument
195     Coordonnee & Point_physique(const Coordonnee& c_int, Coordonnee & co, Enum_dure temps);
196     // 3) cas où l'on veut les coordonnées aux trois temps
197     void Point_physique(const Coordonnee& c_int, Tableau <Coordonnee> & t_co);
198
199     // affichage dans la sortie transmise, des variables duales "nom"
200     // dans le cas où nom est vide, affichage de "toute" les variables
201     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
202
203     //===== lecture écriture dans base info =====
204
205     // cas donne le niveau de la récupération
206     // = 1 : on récupère tout
207     // = 2 : on récupère uniquement les données variables (supposées comme telles)
208     void Lecture_base_info
209     (ifstream& ent, const Tableau<Noeud *> * tabMaillageNoeud, const int cas) ;
210     // cas donne le niveau de sauvegarde
211     // = 1 : on sauvegarde tout
212     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
213     void Ecriture_base_info(ofstream& sort, const int cas) ;
214
215     // définition du nombre maxi de point d'intégration dans l'épaisseur
216     inline int Nb_pt_int_epai()
217     { return doCoPS1->segmentL.Nbi(); };
218     // définition du nombre maxi de point d'intégration sur la surface ou
219     // dans l'axe de la poutre
220     inline int Nb_pt_int_surf()
221     { return doCoPS1->segmentH.Nbi(); };
222     // récupération de l'épaisseur
223     inline double H()
224     { return donnee_specif.epaisseur; };
225
226     // ----- calculs utils dans le cadre de la recherche du flambement linéaire
227     // Calcul de la matrice géométrique et initiale
228     ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
229
230     // inactive les ddl du problème primaire de mécanique
231     inline void Inactive_ddl_primaire()
232     { ElemMeca::Inact_ddl_primaire(doCoPS1->tab_ddl); };
233     // active les ddl du problème primaire de mécanique
234     inline void Active_ddl_primaire()
235     { ElemMeca::Act_ddl_primaire(doCoPS1->tab_ddl); };
236     // ajout des ddl de contraintes pour les noeuds de l'élément
237     inline void Plus_ddl_Sigma()
238     { ElemMeca::Ad_ddl_Sigma(doCoPS1->tab_ddlErr); };
239     // inactive les ddl du problème de recherche d'erreur : les contraintes
240     inline void Inactive_ddl_Sigma()
241     { ElemMeca::Inact_ddl_Sigma(doCoPS1->tab_ddlErr); };
242     // active les ddl du problème de recherche d'erreur : les contraintes
243     inline void Active_ddl_Sigma()
244     { ElemMeca::Act_ddl_Sigma(doCoPS1->tab_ddlErr); };
245     // active le premier ddl du problème de recherche d'erreur : SIGMA11
246     inline void Active_premier_ddl_Sigma()
247     { ElemMeca::Act_premier_ddl_Sigma(); };
248
249     // lecture de données diverses sur le flot d'entrée
250     void LectureContraintes(UtilLecture * entreePrinc)
251     { if (CalculResidu_t_PoutSimplel_met_abstraite == 1)
252         ElemMeca::LectureDesContraintes (false, entreePrinc, lesPtMecaInt.TabSigHH_t());
253         else
254         { ElemMeca::LectureDesContraintes (true, entreePrinc, lesPtMecaInt.TabSigHH_t());
255             CalculResidu_t_PoutSimplel_met_abstraite = 1;
256         }
257     };
258
259     // retour des contraintes en absolu retour true si elle existe sinon false
260     bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
261     { if (CalculResidu_t_PoutSimplel_met_abstraite == 1)
262         ElemMeca::ContraintesEnAbsolues(false, lesPtMecaInt.TabSigHH_t(), tabSig);
263         else
264         { ElemMeca::ContraintesEnAbsolues(true, lesPtMecaInt.TabSigHH_t(), tabSig);
265             CalculResidu_t_PoutSimplel_met_abstraite = 1;
266         }
267         return true; }
268
269     // ===== définition et/ou construction des frontières =====
270
271     // Calcul des frontières de l'element
272     // creation des elements frontieres et retour du tableau de ces elements
273     // la création n'a lieu qu'au premier appel
274     // ou lorsque l'on force le paramètre force a true
275     // dans ce dernier cas seul les frontière effacées sont recréée
276     Tableau <ElFrontiere*> const & Frontiere(bool force = false);
277
278     // ramène la frontière point
279     // éventuellement création des frontières points de l'element et stockage dans l'element

```



```

280 // si c'est la première fois sinon il y a seulement retour de l'elements
281 // a moins que le paramètre force est mis a true
282 // dans ce dernier cas la frontière effacée est recréée
283 // num indique le numéro du point à créer (numérotation EF)
284 // ElFrontiere* const Frontiere_points(int num,bool force = false);
285
286 // ramène la frontière linéique
287 // éventuellement création des frontières linéique de l'element et stockage dans l'element
288 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
289 // a moins que le paramètre force est mis a true
290 // dans ce dernier cas la frontière effacée est recréée
291 // num indique le numéro de l'arête à créer (numérotation EF)
292 // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
293
294 // ramène la frontière surfacique
295 // éventuellement création des frontières surfacique de l'element et stockage dans l'element
296 // si c'est la première fois sinon il y a seulement retour de l'elements
297 // a moins que le paramètre force est mis a true
298 // dans ce dernier cas la frontière effacée est recréée
299 // num indique le numéro de la surface à créer (numérotation EF)
300 // ici normalement la fonction ne doit pas être appelée
301 // ElFrontiere* const Frontiere_surfacique(int ,bool force = false);
302
303 // Retourne la section de l'element
304 double Section (Enum_dure , const Coordonnee& )
305 { return donnee_specif.epaisseur * donnee_specif.largeur; };
306 // ramène la section moyenne de l'élément (indépendante du point)
307 double SectionMoyenne (Enum_dure )
308 { return donnee_specif.epaisseur * donnee_specif.largeur; };
309
310 // 2) methodes propres a l'element
311
312 // Retourne la largeur de l'element
313 inline double Largeur ()
314 { return donnee_specif.largeur; };
315 // Retourne la hauteur de l'element
316 inline double Hauteur ()
317 { return donnee_specif.epaisseur; };
318 // les coordonnées des points d'integration dans l'epaisseur
319 inline double KSIEpais(int i) { return doCoPS1->segmentH.KSI(i);};
320
321
322 // ajout du tableau spécifique de ddl des noeuds de la poutre
323 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
324 // des noeuds constituant l'element
325 void ConstTabDdl();
326
327 protected:
328
329 // ==== »» methodes virtuelles dérivant d'ElemMeca =====
330 // ramene la dimension des tenseurs contraintes et déformations de l'élément
331 int Dim_sig_eps() const {return 1;};
332
333 // ----- calcul de frontières en protected -----
334
335 // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
particulière à l'élément
336 // adressage des frontières linéiques et surfacique
337 // définit dans les classes dérivées, et utilisées pour la construction des frontières
338 virtual ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
339 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
340 virtual ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
341 {return NULL;} // il n'y a pas de surface possible
342
343
344 private :
345
346 // VARIABLES PRIVEES :
347 // les données spécifiques sont groupées dans une structure pour sécuriser
348 // le passage de paramètre dans init par exemple
349 class Donnee_specif{ public :
350 Donnee_specif() :
351 epaisseur(epaisseur_defaut),largeur(largeur_defaut)
352 {};
353 Donnee_specif(double epaiss,double large) :
354 epaisseur(epaiss),largeur(large) {};
355 Donnee_specif(const Donnee_specif& a) :
356 epaisseur(a.epaisseur),largeur(a.largeur) {};
357 Donnee_specif & operator = ( const Donnee_specif& a)
358 { epaisseur = a.epaisseur;largeur = a.largeur;
359 return *this;};
360 // data
361 double epaisseur; // épaisseur de la poutre
362 double largeur; // largeur de la poutre
363 };
364 Donnee_specif donnee_specif;
365

```

```

366 // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
367 LesPtIntegMecaInterne lesPtMecaInt;
368
369 static Tableau <TenseurBB *> d_epsBB; // place de travail pour la variation des def
370 static Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
371
372 class DonnCommunPS1
373 { public :
374     DonnCommunPS1 (GeomSeg& segL,GeomSeg& segH,DdlElement& tab,
375                   Met_Pout2D& met_bie,
376                   Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
377                   Mat_pleine& mat_masse,int nbi);
378     DonnCommunPS1 (DonnCommunPS1& a);
379     ~DonnCommunPS1();
380     // variables
381     GeomSeg segmentL ; // element geometrique correspondant a l'axe
382     GeomSeg segmentH ; // element geometrique correspondant a l'épaisseur
383
384     DdlElement tab_ddl; // tableau des degres
385         //de liberte des noeuds de l'element commun a tous les
386         // elements
387     Met_Pout2D met_pout;
388     Mat_pleine matGeom ; // matrice géométrique
389     Mat_pleine matInit ; // matrice initiale
390     Tableau < Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
391         // calcul d'erreur
392     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
393         // d'erreur : contraintes
394
395     Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
396     Mat_pleine raidErr; // raideur pour le calcul d'erreur
397     // ----- données concernant la dynamique -----
398     Mat_pleine matrice_masse;
399     // ----- blocage éventuel d'hourglass
400     // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
401     Cal_explici_hourglass
402     GeomSeg* segmentHourg; // contient les fonctions d'interpolation
403 };
404
405 // place memoire commune a tous les elements PoutSimple1
406 static int nbNoeud ; // nombre de noeud
407 static int nbintL ; // nombre de point d'intégration selon l'axe
408 static int nbintH ; // nombre de point d'intégration selon l'épaisseur
409 static Tableau <Vecteur> tabD2phi; // par commodité
410
411 static DonnCommunPS1 * doCoPS1;
412 static int CalculResidu_t_PoutSimple1_met_abstraite; // pour dim met_pout à t
413 static int CalculResidu_tdt_PoutSimple1_met_abstraite; // pour dim met_pout à tdt
414 static int Calcul_implicit_PoutSimple1_met_abstraite; // "
415 static int Calcul_VarDualSort; // pour la sortie des valeurs au pt d'integ
416 static int CalDynamique; // pour le calcul de la matrice de masse
417 static int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
418 // pour l'ajout d'element dans la liste : listTypeElement, geree par la class Element
419 class ConstrucElementPoutSimple1 : public ConstrucElement
420 { public : ConstrucElementPoutSimple1 ()
421     { NouvelleTypeElement nouv (PS1,CUBIQUE,MECA_SOLIDE_DEFORMABLE,this);
422     if (ParaGlob::NiveauImpression() >= 4)
423     cout << "\n initialisation PoutSimple1" << endl;
424     Element::listTypeElement.push_back(nouv);
425     };
426     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
427     {Element * pt;
428     pt = new PoutSimple1 (num_maill,num) ;
429     return pt;};
430     // ramene true si la construction de l'element est possible en fonction
431     // des variables globales actuelles: ex en fonction de la dimension
432     bool Element_possible() { if (ParaGlob::Dimension() == 2) return true; else return false;};
433 };
434 static ConstrucElementPoutSimple1 construcElementPoutSimple1;
435 // fonction privée
436 void Def_DonnCommunPS1();
437 // Calcul du residu local a t ou tdt en fonction du booleen
438 Vecteur* CalculResidu (bool atdt,const ParaAlgoContrôle & pa);
439 };
440 /// @} // end of group
441 #endif
442
443
444

```

## 7.138 PoutTimo.h

```

1 // FICHER : PoutTimo.h
2 // CLASSE : PoutTimo

```

```

3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          20/05/98
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *****/
40 *      BUT: // La classe PoutTimo permet de declarer des elements
41 * poutTimo et de realiser le calcul du residu local et de la raideur
42 * locale pour une loi de comportement donnee. La dimension de l'espace
43 * pour un tel element est 1.
44 * Le modèle de poutre de Timoshenko repose sur des fonctions d'interpo-
45 * lation de l'Hermite, il y a donc 2*dim ddl par noeud, les positions
46 * et les dérivées.
47 *
48 *      *****
49 *      VERIFICATION:
50 *
51 *      ! date !   auteur !           but
52 *      -----
53 *      !           !           !
54 *      $
55 *      *****
56 *      MODIFICATIONS:
57 *      ! date !   auteur !           but
58 *      -----
59 *      $
60 *****/
61 // -----classe pour un calcul de mecanique-----
62
63
64
65 #ifndef BIELLETTE_H
66 #define BIELLETTE_H
67
68 #include "ParaGlob.h"
69 #include "ElemMeca.h"
70 // #include "Loi_comp_abstraite.h"
71 #include "Met_abstraite.h"
72 #include "Met_biellette.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "ElFrontiere.h"
79 #include "GeomSeg.h"
80 #include "FrontSegLine.h"
81
82 class ConstrucElementbiel;
83
84 /// @addtogroup groupe_des_elements_finis
85 /// @{
86 ///
87
88
89 class PoutTimo : public ElemMeca

```

```

90 {
91
92     public :
93
94         // CONSTRUCTEURS :
95         // Constructeur par defaut
96         PoutTimo ();
97
98         // Constructeur fonction d'une section et eventuellement d'un numero
99         // d'identification
100        PoutTimo (double sect,int num_id=-3);
101
102        // Constructeur fonction d'un numero d'identification
103        PoutTimo (int num_id);
104
105        // Constructeur fonction d'une section, d'un numero d'identification,
106        // du tableau de connexion des noeuds
107        PoutTimo (double sect,int num_id,const Tableau<Noeud *>& tab);
108
109        // Constructeur de copie
110        PoutTimo (PoutTimo& pout);
111
112
113        // DESTRUCTEUR :
114        ~PoutTimo ();
115
116
117        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PoutTimo
118        PoutTimo& operator= (PoutTimo& pout);
119
120        // METHODES :
121        // 1) derivant des virtuelles pures
122        // Lecture des donnees de la classe sur fichier
123        void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
124
125        // Calcul du residu local et de la raideur locale,
126        // pour le schema implicite
127        Element::ResRaid Calcul_implicit ();
128
129        // Calcul du residu local a t
130        // pour le schema explicite par exemple
131        Vecteur* CalculResidu_t ();
132
133        // retourne les tableaux de ddl gere par l'element
134        // ce tableau est specifique a l'element
135        DdlElement & TableauDdl();
136
137
138        // Libere la place occupee par le residu et eventuellement la raideur
139        // par l'appel de Libere de la classe mere et libere les differents tenseurs
140        // intermediaires cree pour le calcul et les grandeurs pointee
141        // de la raideur et du residu
142        void Libere ();
143
144        // acquisition d'une loi de comportement
145        void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
146
147        // test si l'element est complet
148        // = 1 tout est ok, =0 element incomplet
149        int TestComplet();
150
151        // procedure permettant de completer l'element apres
152        // sa creation avec les donnees du bloc transmis
153        // peut etre appele plusieurs fois
154        Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
155
156        // ramene l'element geometrique correspondant
157        ElemGeom& ElementGeometrique() { return doCo->segment;};
158
159        // affichage dans la sortie transmise, des variables duales "nom"
160        // dans le cas ou nom est vide, affichage de "toute" les variables
161        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
162
163        // Calcul des frontieres de l'element
164        // creation des elements frontieres et retour du tableau de ces elements
165        Tableau <ElFrontiere*>& Frontiere();
166        // METHODES VIRTUELLES:
167        // ----- calculs utils dans le cadre de la recherche du flambement lineaire
168        // Calcul de la matrice geometrique et initiale
169        ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale () ;
170
171        // 2) methodes propres a l'element
172
173        inline double& Section ()
174        // Retourne la section de l'element
175        { return section; };
176

```

```

177 //      inline Mat_pleine& Dphi ()
178 //      // Retourne les derivees des fonctions d'interpolation de la PoutTimo
179 //      { return doCo->dphi; };
180
181 //      // Retourne le tenseur des deformations
182 inline TenseurBB * DeformationBB ()
183     { return epsBB; };
184
185 //      // Retourne le tenseur des contraintes
186 inline TenseurHH * ContrainteHH ()
187     { return sigHH; };
188
189 // ajout du tableau specific de ddl des noeuds de la PoutTimo
190 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
191 // des noeuds constitutants l'element
192 void ConstTabDdl();
193
194 protected:
195
196 // ----- calcul de frontieres en protected -----
197
198 // --- fonction necessaire pour la construction des Frontieres linéiques ou surfaciques
199 // particuliere a l'élément
200 // adresse des frontieres linéiques et surfacique
201 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
202 virtual ElFrontiere* new_frontiere_lin(int , Tableau <Noeud *> & tab, DdlElement& ddelem)
203     { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
204 virtual ElFrontiere* new_frontiere_surf(int , Tableau <Noeud *> & tab, DdlElement& ddelem)
205     {return NULL; } // il n'y a pas de surface possible
206
207 private :
208
209 // VARIABLES PRIVEES :
210
211 double section; // section de la poutTimo
212 TenseurBB * epsBB; // deformation finale
213 TenseurHH * sigHH; // contrainte finale
214 // derivee de la deformation par rapport aux degres de liberte
215 Tableau <TenseurBB *> d_epsBB;
216
217 class DonneeCommune
218     { public :
219         DonneeCommune (GeomSeg& seg,DdlElement& tab,
220             Met_biellette& met_bie) :
221             segment(seg),tab_ddl(tab),met_biellette(met_bie)
222             ,matGeom(tab.NbDdl(),tab.NbDdl())
223             ,matInit(tab.NbDdl(),tab.NbDdl())
224             ,d2_epsBB(tab.NbDdl())
225             {};
226         // variables
227         GeomSeg segment ; // element geometrique correspondant
228         DdlElement tab_ddl; // tableau des degres
229         //de liberte des noeuds de l'element commun a tous les
230         // elements
231         Met_biellette met_biellette;
232         Mat_pleine matGeom ; // matrice géométrique
233         Mat_pleine matInit ; // matrice initiale
234         Tableau2 <TenseurBB *> d2_epsBB;
235     };
236
237 // place memoire commune a tous les elements poutTimos
238 static DonneeCommune * doCo;
239 static int CalculResidu_t_PoutTimo_met_abstraite; // pour dim met_biellette
240 static int Calcul_implicit_PoutTimo_met_abstraite; // "
241 static int Calcul_VarDualSort; // pour la sortie des valeurs au pt d'integ
242 // pour l'ajout d'element dans la liste : listTypeElement, geree par la class Element
243 class ConstrucElementpoutTimo : public ConstrucElement
244     { public : ConstrucElementpoutTimo ()
245         { NouvelleTypeElement nouv;
246           nouv.id_geom = POUT; nouv.id_interpol = BIE1;
247           cout << "\n initialisation PoutTimo" << endl;
248           nouv.el = this;
249           Element::listTypeElement.push_back(nouv);
250         };
251         Element * NouvelElement(int num)
252         {Element * pt;
253           pt = new PoutTimo (num) ;
254           return pt;};
255     };
256 static ConstrucElementpoutTimo construcElementpoutTimo;
257 // fonction privée
258 void Def_DonneeCommune ();
259 };
260 /// @} // end of group
261 #endif
262

```

263  
264

## 7.139 Def\_Umat.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          2/02/2005
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *****/
38 *      BUT: Calcul des differentes grandeurs liee a la deformation
39 *      Dans le cas des grandeurs relatives aux procédures Umat.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date !   auteur !           but
45 *      -----
46 *      !           !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      -----
52 *
53 *****/
54 #ifndef DEF_UMAT_H
55 #define DEF_UMAT_H
56
57 #include "Deformation.h"
58 #include "Met_ElemPoint.h"
59
60 /// @addtogroup groupe_des_deformations
61 /// @{
62 ///
63
64 ///      BUT: Calcul des differentes grandeurs liee a la deformation
65 ///      Dans le cas des grandeurs relatives aux procédures Umat.
66 ///
67 ///
68 /// \author      Gérard Rio
69 /// \version     1.0
70 /// \date        2/02/2005
71
72
73 class Def_Umat : public Deformation
74 {
75 public :
76
77
78     // CONSTRUCTEURS :
79     Def_Umat () ; // par default ne doit pas etre utilise -> message

```

```

80                                     // d'erreur en phase de mise au point
81 // constructeur normal dans le cas d'un ou de plusieurs pt d'integration
82 Def_Umat (Met_abstraite & ,Tableau<Noeud *>& tabnoeud,
83          Tableau <Mat_pleine> const & tabDphi,Tableau <Vecteur>const & tabPhi,
84          Enum_type_deformation type_de_deformation = DEFORMATION_STANDART);
85 // constructeur de copie
86 Def_Umat (const Def_Umat &);
87 // DESTRUCTEUR :
88 virtual ~Def_Umat () {};
89
90 // METHODES PUBLIQUES :
91
92 // Surcharge de l'operateur = : realise l'affectation
93 virtual Deformation& operator= (const Deformation& def)
94     {((Deformation*) this)->operator=(def); return *this;};
95
96
97 // définition d'une déformation du même type, permet d'utiliser des types dérivée surchargé
98 virtual Def_Umat * Nevez_deformation(Tableau <Noeud *> & tabN) const
99     { Def_Umat* def = new Def_Umat (*this);def->PointeurTableauNoeud(tabN);return def;};
100
101
102 // ----- cas d'une procédure Umat -----
103 const Met_ElemPoint::Umat_cont& Cal_defUmat(bool premier_calcul)
104     { return ((Met_ElemPoint*) metrique)->Calcul_grandeurs_Umat(premier_calcul);};
105 // mise à jour de la métrique
106 void Mise_a_jourUmat(const UmatAbaqus& umat)
107     {((Met_ElemPoint*) metrique)->Mise_a_jour(umat);};
108 // - calcul des déformations équivalentes
109 void CalDefEqui(const Tableau <double>& def_equi_t,TenseurBB & epsBB_tdt
110               ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& delta_epsBB_tdt
111               ,const Met_abstraite::Umat_cont& ex,bool premier_calcul);
112
113 protected :
114     // VARIABLES PROTEGEES :
115
116 };
117 /// @} // end of group
118
119 #endif

```

## 7.140 Deformation.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: Calcul des differentes grandeurs liee a la deformation
39 *   La classe fonctionne comme une boite a outil. On y choisit ce
40 *   dont on a besoin. Bien faire attention a l'ordre d'appel des
41 *   differentes methodes lorsque il faut suivre une chronologie.
42 *   Cette classe calcul mais ne stocke pas (ou très peu).

```

```

43 *                                     $ *
44 *      "*****" *
45 *
46 *      VERIFICATION:
47 *
48 *      ! date !   auteur !           but
49 *      -----
50 *      !           !           !
51 *      "*****" *
52 *
53 *      MODIFICATIONS:
54 *      ! date !   auteur !           but
55 *      -----
56 *      $ *
57 *      *****/
58 #ifndef DEFORMATION_H
59 #define DEFORMATION_H
60
61 #include "Tableau_T.h"
62 #include "Met_abstraite.h"
63 #include "Noeud.h"
64 #include "Enum_type_deformation.h"
65 #include "EnumTypeGradient.h"
66 #include "EnumTypeViteRotat.h"
67 #include "ThermoDonnee.h"
68 #include "TypeQuelconque.h"
69 #include "Enum_type_stocke_deformation.h"
70
71 /** @defgroup groupe_des_deformations
72 *
73 *      BUT: Calcul des differentes grandeurs liee a la deformation
74 *      La classe fonctionne comme une boite a outil. On y choisit ce
75 *      dont on a besoin. Bien faire attention a l'ordre d'appel des
76 *      differentes methodes lorsque il faut suivre une chronologie.
77 *      Cette classe calcul mais ne stocke pas (ou très peu).
78 *
79 *
80 * \author   Gérard Rio
81 * \version  1.0
82 * \date    23/01/97
83 * \brief   groupe relatif aux calculs de déformation
84 *
85 */
86
87 /// @addtogroup groupe_des_deformations
88 /// @{
89 ///
90
91 ///      BUT: Calcul des differentes grandeurs liee a la deformation
92 ///      La classe fonctionne comme une boite a outil. On y choisit ce
93 ///      dont on a besoin. Bien faire attention a l'ordre d'appel des
94 ///      differentes methodes lorsque il faut suivre une chronologie.
95 ///      Cette classe calcul mais ne stocke pas (ou très peu).
96 ///
97 ///
98 /// \author   Gérard Rio
99 /// \version  1.0
100 /// \date    23/01/97
101
102 class Deformation
103 {
104     public :
105
106         // CONSTRUCTEURS :
107         Deformation () ; // par défaut ne doit pas être utilisé -> message
108                             // d'erreur en phase de mise au point
109         // constructeur normal dans le cas d'un ou de plusieurs pt d'integration
110         Deformation (Met_abstraite &, Tableau<Noeud *>& tabnoeud,
111                     Tableau<Mat_pleine> const & tabDphi, Tableau<Vecteur>const & tabPhi,
112                     Enum_type_deformation type_de_deformation = DEFORMATION_STANDART);
113         // constructeur de copie
114         Deformation (const Deformation &);
115         // DESTRUCTEUR :
116         virtual ~Deformation ();
117
118
119
120 // -----
121 //      conteneurs basiques pour le stockage de variables spécifiques à un point de calcul
122 // -----
123
124 // stockage métrique: uniquement les grandeurs à un instant donnée t, sans les variations
125 // >>> Grandeurs réellement créées (et non des pointeurs!!) -> sert pour les déformations par exemple
126
127 class Stmet
128     { // surcharge de l'opérateur de lecture avec le type
129         friend istream & operator » (istream &, Stmet &);

```



```

129     // surcharge de l'operator d'écriture
130     friend ostream & operator << (ostream &, const Stmet &);
131     public :
132     Stmet (); // constructeur par défaut
133     // constructeur normal :
134     //     dim_base: dimension des vecteurs de base,
135     //     nbvec_base : nb de vecteur de la base naturelle
136     Stmet (int dim_base, int nbvec_base);
137     Stmet (const Stmet& ex) ; // constructeur de copie
138     ~Stmet(); // destructeur
139     Stmet& operator= (const Stmet& ex); // surcharge d'affectation
140     void Affiche(); // affichage des infos
141     // variables :
142     BaseB * giB_; BaseH * giH_; TenseurBB * gijBB_; TenseurHH * gijHH_;
143     TenseurBB * gradVmoyBB_; TenseurBB * gradVBB_; double* jacobien_;
144     };
145
146 // -----
147 //     fin: conteneurs pour le stockage de variables spécifiques à un point de calcul
148 // -----
149 // -----
150 // //     définition d'une classe conteneur spécifique à chaque point ou la déformation
151 // //     est calculée
152
153     class SaveDefResul
154     { friend class Deformation;
155     public :
156         // Constructeurs
157         SaveDefResul(); // constructeur par défaut-> à ne pas utiliser -> message d'erreur
158         SaveDefResul(const Met_abstraite& meta); // le constructeur recevable
159         SaveDefResul(const SaveDefResul& a); // constructeur de copie
160         ~SaveDefResul(); // destructeur
161
162         // surcharge de l'optérateur d'affectation
163         virtual SaveDefResul& operator= (const SaveDefResul& a); // surcharge d'affectation
164
165         // affichage des infos
166         virtual void Affiche();
167         // idem sur un ofstream
168         virtual void Affiche(ofstream& sort);
169         //===== lecture écriture dans base info =====
170         // cas donne le niveau de la récupération
171         // = 1 : on récupère tout
172         // = 2 : on récupère uniquement les données variables (supposées comme telles)
173         virtual void Lecture_base_info (ifstream& ent,const int cas);
174         // cas donne le niveau de sauvegarde
175         // = 1 : on sauvegarde tout
176         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
177         virtual void Ecriture_base_info(ofstream& sort,const int cas);
178
179         // mise à jour des informations transitoires en définitif, ou l'inverse
180         virtual void TdtversT() {meti_t = meti_tdt; (*D_BB_t)=(*D_BB_tdt);};
181         virtual void TversTdt() {meti_tdt = meti_t; (*D_BB_tdt)=(*D_BB_t);};
182         // mise à jour des grandeurs meti_00 et à t
183         // *** non car dans le cas d'un restart ce n'est pas vrai !!!! //et init par recopie des grandeurs
184         // sur t
185         virtual void MiseAJourGrandeurs_a_0(const Met_abstraite * metrique);
186         // mise à jour des grandeurs meti_tdt
187         virtual void MiseAJourGrandeurs_a_tdt(const Met_abstraite * metrique,const TenseurBB& Deps_BB);
188         // il n'y a pas de mise à jour des grandeurs à t, car celles-ci sont mise à jour à l'aide
189         // de la méthode TdtversT !!
190
191         // lecture des infos des métriques
192         const Deformation::Stmet& Meti_00() const {return meti_00;};
193         const Deformation::Stmet& Meti_t() const {return meti_t;};
194         const Deformation::Stmet& Meti_tdt() const {return meti_tdt;};
195
196         Enum_type_stocke_deformation Type_stocke() const {return enu_type;};
197
198     protected : // VARIABLES PROTEGEES :
199         Deformation::Stmet meti_00,meti_t,meti_tdt; // les infos à 0 à t et à tdt
200         TenseurBB* D_BB_t,* D_BB_tdt; // vitesse de déformation à t et tdt:
201         // on sauvegarde la vitesse, pour en particulier le cas ou on a un deltat trop petit
202         // pour éviter une indécision, on reprend la valeur précédente stockée
203
204         Enum_type_stocke_deformation enu_type; // type de stockage pour le cast lors de la verif de
205         // surcharge
206         // entre instances non différenciées
207     };
208
209     // création d'une instance de SaveDefResul et initialisation.
210     virtual SaveDefResul * New_et_Initialise() {return (new SaveDefResul(*metrique));};
211     // affichage des donnees particulieres a l'elements
212     // de matiere traite ( c-a-dire au pt calcule)
213     virtual void AfficheDataSpecif(ofstream& sort,SaveDefResul * a) const {a->Affiche(sort);};
214     // met à jour les données spécifiques du point considéré
215     void Mise_a_jour_data_specif(Deformation::SaveDefResul* don) {saveDefResul=don;};

```

```

214
215 // // fin de définition relatif à la classe conteneur spécifique à chaque point
216 // // ou la déformation est calculée
217 //-----
218
219 // définition d'une déformation du même type, permet d'utiliser des types dérivée surchargé
220 virtual Deformation * Nevez_deformation(Tableau <Noeud *> & tabN) const
221 { Deformation* def = new Deformation(*this);def->PointeurTableauNoeud(tabN);return def;} ;
222
223 // réaffectation du pointeur de tableau de noeuds interne
224 // il faut que le nouveau tableau ait la même taille que l'ancien (sert pour créer de nouveau
éléments
225 // de même type par exemple, sinon ce n'est pas la bonne solution,
226 // il vaut mieux creer une nouvelle def
227 void PointeurTableauNoeud(Tableau <Noeud *> & tabN);
228
229 // Surcharge de l'operateur = : realise l'affectation
230 virtual Deformation& operator= (const Deformation& def);
231
232 // change le type de déformation
233 virtual void Change_type_de_deformation(Enum_type_deformation type_de_def);
234
235
236 // METHODES PUBLIQUES :
237
238 // affichage des informations
239 virtual void Affiche() const;
240
241 // ----- calcul des variables primaires pour la mécanique -----
242 // calcul explicite à t : tous les parametres sont des resultats
243 virtual const Met_abstraite::Expli& Cal_explicit_t
244 // (bool gradV_instantane,TenseurBB & epsBB_t,Tableau <TenseurBB *> & d_epsBB
245 // ,TenseurBB& DepsBB,TenseurBB& DeltaEpsBB,bool premier_calcul);
246 // // def_equi_t: est la def equi au debut du pas, et def_equi est la def finale
247 // (const Tableau <double>& def_equi_t,TenseurBB & epsBB_t,Tableau <TenseurBB *> &
d_epsBB
248 // ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& DeltaEpsBB,bool
premier_calcul);
249 // calcul explicite à tdt : tous les parametres sont des resultats
250 virtual const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt
251 // ( bool gradV_instantane,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
252 // ,TenseurBB& DepsBB,TenseurBB& delta_epsBB_tdt,bool premier_calcul);
253 // (const Tableau <double>& def_equi_t,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> &
d_epsBB
254 // ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& delta_epsBB_tdt,bool
premier_calcul);
255 // cas implicite
256 virtual const Met_abstraite::Impli& Cal_implicit
257 // ( bool gradV_instantane,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
258 // ,TenseurBB& DepsBB,TenseurBB& delta_epsBB,bool premier_calcul);
259 // (const Tableau <double>& def_equi_t,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
260 // ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& delta_epsBB,bool premier_calcul);
261
262 //// ----- pour la mécanique calcul de l'élongation d'épaisseur pour les contraintes planes
-----
263 //// ce calcul doit faire suite à un appel d'une des 3 fonctions précédentes
264 // // calcul de l'élongation dans le sens de l'épaisseur. Cette déformaion n'est utile que pour des
lois en contraintes planes
265 // // - pour les lois 3D : retour d'un nombre très grand, indiquant que cette fonction est invalide
266 // // - pour les lois 2D def planes: retour de 0
267 // virtual double Elongation3_explicit() const {return ConstMath::tresgrand;};
268 // // le tableau d_lambda3 contient les dérivées de l'élongation / au ddl
269 // virtual double Elongation3_implicit(Tableau <double>& d_lambda3) const;
270 //
271
272 // ----- calcul des variables primaires autre que pour la mécanique -----
273 // ----- donc pas de retour relatif aux déformations
274 // calcul explicite à t : tous les parametres sont des resultats
275 // virtual const Met_abstraite::Expli& Cal_explicit_t(bool gradV_instantane,bool premier_calcul)
276 // {return
metrique->Cal_explicit_t(tabnoeud,gradV_instantane,(*tabDphi)(numInteg),nbNoeud,premier_calcul);};
277 virtual const Met_abstraite::Expli& Cal_explicit_t(bool premier_calcul)
278 {return
metrique->Cal_explicit_t(*tabnoeud,false,(*tabDphi)(numInteg),nbNoeud,(*tabPhi)(numInteg),premier_calcul);};
279 // calcul explicite à tdt : tous les parametres sont des resultats
280 // virtual const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt(bool gradV_instantane,bool
premier_calcul)
281 // {return
metrique->Cal_explicit_tdt(*tabnoeud,gradV_instantane,(*tabDphi)(numInteg),nbNoeud,premier_calcul);};
282 virtual const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt(bool premier_calcul)
283 {return
metrique->Cal_explicit_tdt(*tabnoeud,false,(*tabDphi)(numInteg),nbNoeud,(*tabPhi)(numInteg),premier_calcul);};
284 // cas implicite
285 // virtual const Met_abstraite::Impli& Cal_implicit(bool gradV_instantane,bool premier_calcul)
286 // {return
metrique->Cal_implicit(*tabnoeud,gradV_instantane,(*tabDphi)(numInteg),nbNoeud,pa.Var_D(),premier_calcul);};
287 // avec_var_Xi : par défaut true, si false indique que l'on ne calcule pas les variations / au ddl Xi

```

```

288     virtual const Met_abstraite::Implic& Cal_implicit(bool premier_calcul,bool avec_var_Xi=true)
289     {return
metrique->Cal_implicit(*tabnoeud,false,(*tabDphi)(numInteg),nbNoeud,(*tabPhi)(numInteg),premier_calcul,avec_var_Xi);};
290 // ----- flambage linéaire -----
291 // le calcul est identique au cas implicite sauf que l'on doit absolument calculer la dérivée
seconde de la déformation
292     virtual const Met_abstraite::flambe_lin& Cal_flambe_lin
293     ( bool gradV_instantane,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
294     ,TenseurBB& DepsBB,TenseurBB& delta_epsBB,bool premier_calcul);
295     (const Tableau <double>& def_equi_t,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
296     ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& delta_epsBB,bool premier_calcul);
297 // ----- cas de la dynamique -----
298 // calcul des grandeurs utils pour les raideurs inertielles: c'est-à-dire le calcul de la matrice
masse
299     virtual const Met_abstraite::Dynamiq& Cal_matMass()
300     { return metrique->Cal_pourMatMass(
*tabnoeud,(*tabDphi)(numInteg),nbNoeud,(*tabPhi)(numInteg));};
301 // ----- calcul de données interpolées aux même points d'integ que la mécanique -----
302 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
303     virtual double DonneeInterpoleeScalaire(Enum_ddl enu,Enum_dure temps) const
304     { return metrique->DonneeInterpoleeScalaire(*tabnoeud,(*tabPhi)(numInteg),nbNoeud,enu,temps); };
305 // ----- calcul de la variation d'une donnée interpolée par rapport aux ddl de la donnée -----
306 // aux même points d'integ que la mécanique -----
307 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
308 // en sortie : d_A
309     virtual Tableau <double >& DerDonneeInterpoleeScalaire(Tableau <double >& d_A, Enum_ddl
enu,Enum_dure temps) const
310     { return metrique->DerDonneeInterpoleeScalaire
311     (d_A,*tabnoeud,(*tabPhi)(numInteg),nbNoeud,enu); };
312 // ----- calcul du gradient d'une donnée interpolée aux même points d'integ que la mécanique
-----
313 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
314 // en sortie : gradB
315     virtual CoordonneeB& GradDonneeInterpoleeScalaire(CoordonneeB& gradB, Enum_ddl enu,Enum_dure temps)
const
316     { return
metrique->GradDonneeInterpoleeScalaire(gradB,*tabnoeud,(*tabDphi)(numInteg),nbNoeud,enu,temps); };
317 // ----- calcul de la variation du gradient d'une donnée interpolée par rapport aux ddl de la donnée
-----
318 // aux même points d'integ que la mécanique -----
319 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
320 // en sortie : d_gradB
321     virtual Tableau <CoordonneeB >& DerGradDonneeInterpoleeScalaire(Tableau <CoordonneeB >& d_gradB,
Enum_ddl enu) const
322     { return metrique->DerGradDonneeInterpoleeScalaire
323     (d_gradB,*tabnoeud,(*tabPhi)(numInteg),(*tabDphi)(numInteg),nbNoeud,enu); };
324 // ----- calcul de la variation seconde du gradient d'une donnée interpolée par rapport aux ddl de
la donnée -----
325 // aux même points d'integ que la mécanique -----
326 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
327 // en sortie : d2_gradB
328 // si d2_gradB est NULL cela signifie qu'il ne faut pas considérer cette grandeur
329     virtual Tableau2 <CoordonneeB>* Der2GradDonneeInterpoleeScalaire
330     (Tableau2 <CoordonneeB>* d2_gradTB, Enum_ddl enu) const
331     {return metrique->Der2GradDonneeInterpoleeScalaire
332     (d2_gradTB,*tabnoeud,(*tabPhi)(numInteg),(*tabDphi)(numInteg),nbNoeud,enu); };
333
334 //     virtual Vecteur& DonneeInterpoleeVecteur(Enum_ddl enu,Vecteur& v, Enum_dure temps)
335 //     { return metrique->DonneeInterpoleeVecteur(*tabnoeud,(*tabPhi)(numInteg),nbNoeud,enu,v,temps);
};
336 //     virtual TenseurBB& DonneeInterpoleeTenseurBB(Enum_ddl enu,TenseurBB& t);
337 // ----- cas de la dilatation thermique -----
338 // l'objectif est de calculer la déformation d'origine thermique et la déformation d'origine
mécanique
339 // temperature_tdt,temperature_t : températures, temperature_0: température initiale
340 // atdt : booléen qui indique si les grandeurs à tdt sont disponible ou pas
341 // avec_repercution_sur_def_meca: si oui on met à jour la déformation méca, sinon, on ne calcul que
les def thermiques
342     virtual void DeformationThermoMechanique(const double& temperature_0,const TenseurBB& gijBB,const
ThermoDonnee& dTP
343     ,const TenseurBB & epsBB_totale
344     ,TenseurBB & epsBB_therm,const double& temperature_tdt,TenseurBB &
epsBB_meca
345     ,const double& temperature_t
346     ,TenseurBB & DepsBB_totale,TenseurBB & DepsBB_therm,TenseurBB &
DepsBB_meca
347     , bool atdt, bool avec_repercution_sur_def_meca);
348 // ----- calcul et récupération de la dérivée de la vitesse de déformation virtuelle ----
349 // calcul et récupération de la dérivée de la vitesse de déformation virtuelle à tdt
350 // dans le cas d'une discrétisation classique simple, ce calcul est indépendant des coordonnées
351 // donc peut être effectué indépendamment du reste, il dépend cependant du point d'intégration
352 // dans le cas de discrétisation complexe, il faut s'assurer que les autres grandeurs utiles de la
métrique
353 // sont déjà calculé (cf. D2_gijBB_tdt() )
354 // total = true : indique que le calcul est complet, à partir des données de base
355 // c-a-d calcul de la variation de vecteur de base puis calcul de la variation
seconde

```

```

356 //           = false: le calcul est simplifié, on considère que la variation des vecteurs de base vient
357 //           juste d'être calculé, dans un calcul implicite, on ne calcul alors que la
    variation seconde
358
359 void Cal_var_def_virtuelle(bool total,Tableau2 <TenseurBB *>& d2_epsBB_tdt);
360
361 // ===== remontee aux informations =====
362 // calcul :
363 // M0 : point d'integration numInteg a t = 0
364 // Mt ou Mtdt : point d'integration final
365 // Aa0 et Aafin : matrice de passage initiale et finale dans un repere ortho tel que
366 // la nouvelle base Aa est calculee par projection de "Ipa" sur Gi
367 // gijHH et gijBB : metrique finale
368 // Aa(i,a) = Aa^i_{.a}, avec g^i = Aa^i_{.a} * Ip^a
369 // tout ce passe comme si Ip^a est la nouvelle base vers laquelle on veut évoluer
370
371 // cas sortie d'un calcul implicite
372 virtual const Met_abstraite::InfoImp RemontImp(bool absolue,Mat_pleine& Aa0,Mat_pleine& Aafin);
373 // idem sans le calcul des matrices de passage
374 virtual const Met_abstraite::InfoImp RemontImp();
375 // cas sortie d'un calcul explicite à t
376 virtual const Met_abstraite::InfoExp_t RemontExp_t(bool absolue,Mat_pleine& Aa0,Mat_pleine& Aafin);
377 // idem sans le calcul des matrices de passage
378 virtual const Met_abstraite::InfoExp_t RemontExp_t();
379 // cas sortie d'un calcul explicite à tdt
380 virtual const Met_abstraite::InfoExp_tdt RemontExp_tdt(bool absolue,Mat_pleine& Aa0,Mat_pleine&
    Aafin);
381 // idem sans le calcul des matrices de passage
382 virtual const Met_abstraite::InfoExp_tdt RemontExp_tdt();
383 // cas sortie d'un calcul à 0, t et tdt
384 virtual const Met_abstraite::Info0_t_tdt Remont0_t_tdt(bool absolue,Mat_pleine& Aa0,Mat_pleine&
    Aat,Mat_pleine& Aatdt);
385 // idem sans le calcul des matrices de passage
386 virtual const Met_abstraite::Info0_t_tdt Remont0_t_tdt();
387
388 // mêmes fct que précédemment mais dans le cas où les éléments de métrique viennent justes
389 // d'être calculés, donc les seules grandeurs qui sont réellement calculées sont les matrices
390 // Aa0 et Aafin
391 // cas sortie d'un calcul implicite
392 virtual const Met_abstraite::InfoImp RemontImpSansCalMet(bool absolue,Mat_pleine& Aa0,Mat_pleine&
    Aafin);
393 // cas sortie d'un calcul explicite à t
394 virtual const Met_abstraite::InfoExp_t RemontExp_tSansCalMet(bool absolue,Mat_pleine& Aa0,Mat_pleine&
    Aafin);
395 // cas sortie d'un calcul explicite à tdt
396 virtual const Met_abstraite::InfoExp_tdt RemontExp_tdtSansCalMet(bool absolue,Mat_pleine&
    Aa0,Mat_pleine& Aafin);
397 // cas sortie d'un calcul à 0, t et tdt
398 virtual const Met_abstraite::Info0_t_tdt Remont0_t_tdtSansCalMet(bool absolue,Mat_pleine&
    Aa0,Mat_pleine& Aat,Mat_pleine& Aatdt);
399 // cas sortie d'un calcul à 0, t et tdt, y compris les métriques pour tous les temps
400 virtual const Met_abstraite::Info_et_metrrique_0_t_tdt Remont_et_metrrique_0_t_tdtSansCalMet(bool
    absolue,Mat_pleine& Aa0,Mat_pleine& Aat,Mat_pleine& Aatdt);
401 // cas sortie d'un calcul à 0, t et tdt, y compris les métriques pour tous les temps, pas de calcul
    de matrice
402 // de passage
403 virtual const Met_abstraite::Info_et_metrrique_0_t_tdt Remont_et_metrrique_0_t_tdtSansCalMet()
    { return metrique->Recup_Info_et_metrrique_0_t_tdt(); };
404
405
406 // calcul d'une déformation au temps donné:
407 // on suppose que gijHH et gijBB ont déjà été calculé à t=0, et à temps
408 virtual void Cal_deformation (Enum_dure temps, TenseurBB & epsBB);
409
410 // ramène le type de déformation actuellement utilisé
411 Enum_type_deformation Type_de_deformation() const {return type_deformation;};
412 // ramène le type de gradient de vitesse actuellement utilisé
413 Enum_type_gradient Type_de_gradient_vitesse() const {return type_gradient_vitesse;};
414
415 // récupération des grandeurs particulière (hors ddl )
416 // correspondant à liTQ
417 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
418 virtual void Grandeur_particuliere(bool absolue,List_io<TypeQuelconque>& ) const {};
419 // récupération de la liste de tous les grandeurs particulières
420 // ces grandeurs sont ajoutées à la liste passées en paramètres
421 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
422 virtual void ListeGrandeurs_particulieres(bool absolue,List_io<TypeQuelconque>& ) const {};
423
424 // ----- fonctions utilitaires -----
425 // ramene la taille du premier element de tabPhi, ce qui represente en general
426 // le nombre de point d'integration pour les element simple
427 inline int Phil_Taille() const { return tabPhi->Taille();};
428
429 // change le numero d'integration courant
430 virtual void ChangeNumInteg(int ni) {sauve_numInteg = numInteg; numInteg = ni;};
431 // retourne le numero d'integration courant
432 int NumInteg_en_cours() const {return numInteg;};
433 // gestion du parcours de tous les points d'integration

```

```

434 virtual void PremierPtInteg();
435 virtual bool DernierPtInteg();
436 virtual void NevezPtInteg();
437 // méthode pour revenir au pti précédant
438 virtual void Retour_pti_precedant();
439
440 // calcul de la position dans le repère absolu du point d'intégration
441 virtual const Coordonnee& Position_0() // à t=0
442     {return metrique->PointM_0(*tabnoeud, (*tabPhi) (numInteg));};
443 virtual const Coordonnee& Position_t() // à t
444     {return metrique->PointM_t(*tabnoeud, (*tabPhi) (numInteg));};
445 virtual const Coordonnee& Position_tdt() // à t+dt
446     {return metrique->PointM_tdt(*tabnoeud, (*tabPhi) (numInteg));};
447 // calcul de la variation dans le repère absolu du point d'intégration
448 virtual const Tableau <Coordonnee>& Der_Pos_t() // à t
449     {return metrique->D_PointM_t(*tabnoeud, (*tabPhi) (numInteg));};
450 virtual const Tableau <Coordonnee>& Der_Pos_tdt() // à t+dt
451     {return metrique->D_PointM_tdt(*tabnoeud, (*tabPhi) (numInteg));};
452
453 // calcul de la vitesse dans le repère absolu du point d'intégration
454 // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
455 // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
456 // à priori on calcul au point d'intégration: cas de la première fonction
457 // on peut également appeler la fonction avec un numéro de noeud ou un pointeur de noeud
458 // dans ces deux derniers cas la vitesses est calculée au noeud spécifié
459 virtual const Coordonnee& VitesseM_0() {return metrique->VitesseM_0(*tabnoeud, (*tabPhi) (numInteg));};
460
461 virtual const Coordonnee& VitesseM_0(int num_n) {return metrique->VitesseM_0(*tabnoeud (num_n));};
462 virtual const Coordonnee& VitesseM_0(const Noeud* noe) {return metrique->VitesseM_0(noe);};
463
464 virtual const Coordonnee& VitesseM_t() {return metrique->VitesseM_t(*tabnoeud, (*tabPhi) (numInteg));};
465
466 virtual const Coordonnee& VitesseM_t(int num_n) {return metrique->VitesseM_t(*tabnoeud (num_n));};
467 virtual const Coordonnee& VitesseM_t(const Noeud* noe) {return metrique->VitesseM_t(noe);};
468
469 virtual const Coordonnee& VitesseM_tdt() {return
metrique->VitesseM_tdt(*tabnoeud, (*tabPhi) (numInteg));};
470 virtual const Coordonnee& VitesseM_tdt(int num_n) {return
metrique->VitesseM_tdt(*tabnoeud (num_n));};
471 virtual const Coordonnee& VitesseM_tdt(const Noeud* noe) {return metrique->VitesseM_tdt(noe);};
472
473 // calcul de la variation dans le repère absolu de la vitesse du point d'intégration
474 // ddl_vitesse : indique si oui ou non il s'agit de variation par rapport
475 // aux ddl de vitesse ou au ddl de position
476 virtual const Tableau <Coordonnee>& Der_VitesseM_t(bool& ddl_vitesse) // à t
477     {return metrique->D_VitesseM_t(*tabnoeud, (*tabPhi) (numInteg), ddl_vitesse);};
478 virtual const Tableau <Coordonnee>& Der_VitesseM_t(bool& ddl_vitesse, int num_noeud) // à t
479     {return metrique->D_VitesseM_t(*tabnoeud (num_noeud), ddl_vitesse);};
480 virtual const Tableau <Coordonnee>& Der_VitesseM_t(bool& ddl_vitesse, const Noeud* noe) // à t
481     {return metrique->D_VitesseM_t(noe, ddl_vitesse);};
482
483 virtual const Tableau <Coordonnee>& Der_VitesseM_tdt(bool& ddl_vitesse) // à t
484     {return metrique->D_VitesseM_tdt(*tabnoeud, (*tabPhi) (numInteg), ddl_vitesse);};
485 virtual const Tableau <Coordonnee>& Der_VitesseM_tdt(bool& ddl_vitesse, int num_noeud) // à t
486     {return metrique->D_VitesseM_tdt(*tabnoeud (num_noeud), ddl_vitesse);};
487 virtual const Tableau <Coordonnee>& Der_VitesseM_tdt(bool& ddl_vitesse, const Noeud* noe) // à t
488     {return metrique->D_VitesseM_tdt(noe, ddl_vitesse);};
489
490 // détermination d'une bases particulière orthonormée pour représenter les tenseurs
491 // en sortie on a une (ou plusieurs) matrices de passage, qui permettent de passer de la base
492 // curviligne
493 // à cette base particulière que l'on va appeler IPa
494 // Le choix qui est fait est de calculer le passage gH(i) -> IPa ,
495 // c-a-d Aa(i,a) = les coordonnées de gH(i) dans la base IPa,
496 // ou encore: la ligne i de Aa = les coordonnées de gH(i)
497 // g^i = Aa^i_{.a} * Ip^a
498 // NB: si on a besoin du passage de gB(i) on utilise la matrice inverse de Aa
499 //
500 // 1)cas d'une base naturelle avec un nombre de vecteur == à la dimension de l'espace:
501 // -> on retient comme base intéressante la base absolue Ia, donc ici IPa = Ia
502 // c-a-d que l'on calcul une matrice de passage pour arriver dans la base Ia à partir de gH(i)
503 // c'est le cas le plus simple qui semble naturel,
504 //
505 // 2)cas où le nombre de vecteur de la base naturelle = 2 dans un espace 3D
506 // la nouvelle base IPa est calculée par projection de "Ia" sur Galpha
507 // Premier cas: la projection du vecteur $\vec I_{1}$ sur l'élément est non nulle. Dans ce cas, cette
508 // projection
509 // est normalisée et tient lieu de premier vecteur de la base locale. Le troisième vecteur est
510 // constitué de
511 // la normale à l'élément, et le second vecteur est obtenu par produit vectoriel des deux premiers.
512 // Second cas: la projection du vecteur $\vec I_{1}$ sur l'élément est nulle, on utilise alors la
513 // projection du
514 // vecteur $\vec I_{2}$ qui tient lieu alors de premier vecteur. La suite est identique au premier
515 // cas.
516 //
517 // 3)cas où le nombre de vecteur de la base naturelle = 1 dans un espace 2D
518 // IP1 = le vecteur normé collinéaire avec gH(1) (qui est également collinéaire avec gB(1)

```

```

512 // IP2 est normal à IP1, et choisit dans le sens direct
513 //
514 // 4) cas où le nombre de vecteur de la base naturelle = 1 dans un espace 3D
515 // IP1 = le vecteur normé collinéaire avec gH(1) (qui est également collinéaire avec gB(1)
516 // IP2 et IP3 sont normaux à IP1, et choisit de manière arbitraire dans le plan normal à IP1
517 // normalement IP2 et IP3 ne servent pas pour exprimer des tenseurs qui utilisent la base naturelle
518 // c'est pour cela qu'a priori on peut les choisir de manière quelconque
519
520 // détermination des bases de passages (seules les données en entrées et sortie
521 // sont utilisées: soit Aa(i,j) la matrice de passage (= Aa0, Aa0fin)
522 virtual void BasePassage(bool absolue, const BaseB & giB0, const BaseB & giB, const BaseH & giH0,
523     const BaseH & giH, Mat_pleine & Aa0, Mat_pleine & Aa0fin);
524 // cas où l'on veut les matrices de passages à 0 t et tdt
525 virtual void BasePassage(bool absolue, const BaseB & giB0, const BaseB & giB_t, const BaseB & giB_tdt
526     , const BaseH & giH0, const BaseH & giH_t, const BaseH & giH_tdt
527     , Mat_pleine & Aa0, Mat_pleine & Aa_t, Mat_pleine & Aa_tdt);
528 // cas où l'on veut la matrice de passage à 0 uniquement
529 virtual void BasePassage(bool absolue, const BaseB & giB0, const BaseH & giH0, Mat_pleine & Aa0);
530
531 // ===== informations diverses (mais cohérentes, c-a-d, peuvent être appelées directement sans
init) =====
532 double JacobienInitial()
533 { return
metrique->JacobienInitial(*tabnoeud, (*tabDphi)(numInteg), nbNoeud, (*tabPhi)(numInteg)); };
534
535 protected :
536 // VARIABLES PROTEGEES :
537 Met_abstraite * metrique; // boite a outil pour le calcul
538 // des differents tenseurs lies a la metrique
539 // on utilise un pointeur de tableau pour tabnoeud car il faut pouvoir le réafecter,
540 // par exemple après un constructeur de copie: il faut même bien faire attention de ne pas oublier !!
541
542 Tableau<Noeud *> tabnoeud; // le tableau de noeud qui est interpolé
543 // mais on n'utilise pas forcément tous les noeuds !! le nombre de
noeud
544 // utilisé est nbNoeud
545 Tableau<Mat_pleine> const * tabDphi; // derivees de tous les fonctions d'interpolation
546 Tableau<Vecteur> const * tabPhi; // toutes les fonctions d'interpolation
547 // derivees et fonctions d'interpolations au point d'integration courant
548 // on utilise un tableau car certaine classe dérivée on
549 int nbNoeud; // nb de noeud de l'interpolation
550 int numInteg; // numero du point d'integration en cours
551 // dans le cas d'élément coque numInteg représente le pt d'integ de la surface
552 // dans le cas d'élément poutre, numInteg représente le pt d'integ de la ligne
553 int sauve_numInteg; // sauvegarde pour un retour au pti précédant avec la méthode
Retour_pti_precedant();
554
555 // identificateur interne de type de déformation, permet pour les classes dérivées
556 // de faire du cast
557 Enum_type_deformation type_deformation;
558 Enum_type_gradient type_gradient_vitesse; // idem pour le gradient de vitesse
559 // une instance de données particulière
560 SaveDefResul * saveDefResul;
561
562 // --- gestion d'index ---
563 class UtilIndexDeformation
564 { public:
565     UtilIndexDeformation();
566     // def de deux tableaux d'indices pour avoir la suite: 12 21 13 31 23 32 à partir de 6
nombres
567     Tableau<int> iii, jjj;
568 };
569 static const UtilIndexDeformation ccdex;
570 // variables intermédiaires utilisées pour les calculs relatifs aux déformations cumulées ou
logarithmiques
571
572 TenseurBH* B_BH_tr; // tenseur B_BH
573 Tableau<TenseurBH *> d_B_BH_tr; // variation du tenseur B_BH
574 Coordonnee ki; // vecteur des valeurs propres
575 Tableau<Coordonnee *> d_ki; // variation des valeurs propres
576 Tableau<TenseurBH*> Palpha_BH_tr; // projecteurs propres
577 Tableau<Tableau<TenseurBH*>> d_Palphi_BH_tr; // tableau des variation des projecteurs propres
578 int dimensionnement_tableaux; // =0 au début, =1 si dim des tab sans var, =2 si dim des tab avec
var
579
580 //----- partie statique ou sont stocké tous les cas différents des variables précédentes
581 static list<Tableau<TenseurBH *>> list_var_tens_BH;
582 static list<Tableau<Tableau<TenseurBH*>>> list_var_var_tens_BH;
583 static list<Tableau<Coordonnee *>> list_tab_coor;
584 static int nombre_d_instance_deformation;
585
586 // METHODES PROTEGEES :
587 // calcul implicite dans le cas d'une déformation d'Almansi
588 const Met_abstraite::Impli & Cal_implicit_Almansi (bool gradV_instantane
589     , TenseurBB & epsBB_tdt, Tableau<TenseurBB *> & d_epsBB
590     , TenseurBB & DepsBB, TenseurBB & delta_epsBB, bool premier_calcul
591     , const Met_abstraite::Impli & ex);

```



```

591 // calcul implicite dans le cas d'une déformation Logarithmique
592 const Met_abstraite::Impli& Cal_implicit_Logarithmique (bool gradV_instantane
593 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
594 ,TenseurBB& DepsBB,TenseurBB& delta_epsBB,bool premier_calcul
595 ,const Met_abstraite::Impli& ex);
596 // calcul implicite dans le cas d'une déformation tensorielle cumulée
597 const Met_abstraite::Impli& Cal_implicit_def_cumule (bool gradV_instantane
598 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
599 ,TenseurBB& DepsBB,TenseurBB& delta_epsBB,bool premier_calcul
600 ,const Met_abstraite::Impli& ex);
601 // calcul explicite dans le cas d'une déformation d'Almansi
602 const Met_abstraite::Expli& Cal_explicit_Almansi (bool gradV_instantane
603 ,TenseurBB & epsBB_t,Tableau <TenseurBB *> & d_epsBB,TenseurBB& DepsBB
604 ,TenseurBB& delta_epsBB,bool premier_calcul,const Met_abstraite::Expli& ex);
605 // calcul explicite dans le cas d'une déformation logarithmique
606 const Met_abstraite::Expli& Cal_explicit_Logarithmique (bool gradV_instantane
607 ,TenseurBB & epsBB_t,Tableau <TenseurBB *> & d_epsBB,TenseurBB& DepsBB
608 ,TenseurBB& delta_epsBB,bool premier_calcul,const Met_abstraite::Expli& ex);
609 // calcul explicite dans le cas d'une déformation tensorielle cumulée
610 const Met_abstraite::Expli& Cal_explicit_def_cumule (bool gradV_instantane
611 ,TenseurBB & epsBB_t,Tableau <TenseurBB *> & d_epsBB,TenseurBB& DepsBB
612 ,TenseurBB& delta_epsBB,bool premier_calcul,const Met_abstraite::Expli& ex);
613 // calcul explicite à t et tdt dans le cas d'une déformation d'Almansi
614 const Met_abstraite::Expli_t_tdt& Cal_explicit_Almansi_tdt (bool gradV_instantane
615 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB,TenseurBB& DepsBB
616 ,TenseurBB& delta_epsBB,bool premier_calcul,const Met_abstraite::Expli_t_tdt& ex);
617 // calcul explicite à t et tdt dans le cas d'une déformation logarithmique
618 const Met_abstraite::Expli_t_tdt& Cal_explicit_logarithmique_tdt (bool gradV_instantane
619 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB,TenseurBB& DepsBB
620 ,TenseurBB& delta_epsBB,bool premier_calcul,const Met_abstraite::Expli_t_tdt& ex);
621 // calcul explicite à t et tdt dans le cas d'une déformation tensorielle cumulée
622 const Met_abstraite::Expli_t_tdt& Cal_explicit_def_cumule_tdt (bool gradV_instantane
623 ,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB,TenseurBB& DepsBB
624 ,TenseurBB& delta_epsBB,bool premier_calcul,const Met_abstraite::Expli_t_tdt& ex);
625 // ----- calcul uniquement de la déformation -----
626 // on suppose que gijHH et gijBB ont déjà été calculé à t=0, et à temps
627 // calcul de la deformation au temps donné dans le cas d'une déformation d'Almansi
628 void Cal_Almansi_auTemps(Enum_dure temps, TenseurBB & epsBB);
629 // calcul de la deformation au temps donné dans le cas d'une déformation logarithmique
630 void Cal_logarithmique_auTemps(Enum_dure temps, TenseurBB & epsBB);
631 // calcul de la deformation au temps donné dans le cas d'une déformation tensorielle cumulée
632 void Cal_def_cumule_auTemps(Enum_dure temps, TenseurBB & epsBB);
633 // --- méthodes de vérifications ---
634 void VerifCal_implicit (bool gradV_instantane,const Met_abstraite::Impli & ex,TenseurBB& DepsBB);
635 // indicateur utilisé par VerifCal_deflog
636 static int indic_VerifCal_implicit;
637
638 // calcul d'une vitesse de rotation cohérente avec un référentiel objectif particulier
639 void Cal_vite_rota_objectif(EnumTypeViteRotat type_rotation,bool variation,const TenseurBB& gradVBB
640 ,const Tableau <TenseurBB *> & d_gradVBB
641 ,const TenseurBH& gijHH_0,const TenseurBB& gij_BB,Tableau <TenseurBB *> & d_gijBB
642 ,const TenseurHH& gij_HH,Tableau <TenseurHH *> & d_gijHH
643 ,const TenseurBB& D_BB, Tableau <TenseurBB *> & d_D_BB
644 ,TenseurBB& OmegaBB,Tableau <TenseurBB*> & d_OmegaBB);
645 // fonction f et f' utilisé dans la méthode Cal_vite_rota_objectif:
646 double F_pour_rotat_objectif(EnumTypeViteRotat type_rotation,const double& x);
647 double Fprime_pour_rotat_objectif(EnumTypeViteRotat type_rotation,const double& x);
648 // calcul des valeurs propres, les projections propres, et les variations d'un tenseurs BH
649 // utilisé dans Cal_vite_rota_objectif
650 // cas_ki indique le cas des valeurs et vecteurs propres (cf TenseurBH)
651 // quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
652 // dans ce cas les valeurs propres de retour sont nulles par défaut
653 // dim = 1, cas=1 pour une valeur propre;
654 // dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques
655 // dim = 3 , cas = 1 si 3 val propres différentes, cas = 0 si les 3 sont identiques,
656 // cas = 2 si ki(1)=ki(2), cas = 3 si ki(2)=ki(3)
657 void Val_et_projection_prop_tenseur(const TenseurBH & B_BH,const Tableau <TenseurBH *> & d_B_BH
658 ,Coordonnee& ki,Tableau <TenseurBH*> & Palpha_BH,bool variation
659 ,Tableau <Coordonnee > & d_ki, Tableau <Tableau <TenseurBH*> > & d_Palphi_BH
660 ,int& cas_ki);
661 // intégration de la vitesse de rotation
662 void Integ_vitesse_rotation();
663 // dimensionnement ou vérif des dimensions des variables intermédiaires
664 // utilisées pour les mesures cumulées et/ou log
665 void DimensionnementVarLog(int dima,bool avec_var,int nbvar);
666 private: // ---- fonctions internes de gestion de dimension et autres----
667 // verif si tableau de tenseur est bien dimensionné, ou alloue si nécessaire
668 Tableau <TenseurBH *>* ExisteTabTenseur_BH_tr(int nbvar,Tableau <TenseurBH *>* tab_Tens,int dima)
669 const ;
670 // verif si le tableau de tableau de tenseur est bien dimensionné, ou alloue si nécessaire
671 // dima: dimension des tenseur et du premier tableau, nbvar: dimension du tableau extérieur
672 Tableau <Tableau <TenseurBH*> *>* ExisteTabTabTenseur_BH_tr(int dima,Tableau <Tableau <TenseurBH*> >
673 >* tab_Tens,int nbvar) const ;
674 // verif si tableau de Coordonnee est bien dimensionné, ou alloue si nécessaire
675 Tableau <Coordonnee*>* ExisteTabCoor_tr(int nbvar,Tableau <Coordonnee*>* tab_coor,int dima) const ;
676 // fonction qui calcule le tenseur logarithmique et sa variation éventuelle
677 void Cal_Logarithmique (const TenseurBB & gijBB_0,const TenseurHH& gijHH_0,Tableau <TenseurBB *> &

```

```

d_epsBB
676         ,const TenseurBB& gijBB,const TenseurHH& gijHH,TenseurBB & epsBB
677         ,const Tableau <TenseurBB *>& d_gijBB,bool variation);
678 // méthode de vérification par différences finies du calcul des différentes dérivées

679 void VerifCal_deflog(bool gradV_instantane,const Met_abstraite::Impli & ex,TenseurBB& epsBB_tdt
680                   ,Tableau <TenseurBB *> & d_epsBB_tdt);
681
682 };
683 /// @} // end of group
684
685 #endif

```

## 7.141 DeformationPP.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           25/05/98
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *
38 *   BUT: Calcul des differentes grandeurs liee a la deformation
39 *   d'elements poutres et plaques classiques.
40 *   Par rapport à la classe Deformation de base, ici on considère
41 *   deux directions : l'épaisseur, et l'axe ou le plan
42 *   dans ces deux directions, il y a des fcts d'interpolation parti-
43 *   culières.
44 *   La classe fonctionne comme une boite a outil. On y choisit ce
45 *   dont on a besoin. Bien faire attention a l'ordre d'appel des
46 *   differentes methodes lorsque il faut suivre une chronologie.
47 *   Cette classe calcul mais ne stock pas.
48 *
49 *   *****
50 *
51 *   VERIFICATION:
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !           !
55 *   $
56 *
57 *   MODIFICATIONS:
58 *   ! date !   auteur !           but
59 *   -----
60 *   $
61 *****/
62 #ifndef DEFORMATIONPP_H
63 #define DEFORMATIONPP_H
64
65 #include "Tableau_T.h"
66 #include "Met_abstraite.h"
67 #include "Deformation.h"
68
69 /// @addtogroup groupe_des_deformations

```



```

70 /// @{
71 ///
72
73 /// BUT: Calcul des differentes grandeurs liee a la deformation
74 /// d'elements poutres et plaques classiques.
75 /// Par rapport à la classe Deformation de base, ici on considère
76 /// deux directions : l'épaisseur, et l'axe ou le plan
77 /// dans ces deux directions, il y a des fcns d'interpolation parti-
78 /// culières.
79 /// La classe fonctionne comme une boite a outil. On y choisit ce
80 /// dont on a besoin. Bien faire attention a l'ordre d'appel des
81 /// differentes methodes lorsque il faut suivre une chronologie.
82 /// Cette classe calcul mais ne stock pas.
83 ///
84 ///
85 /// \author Gérard Rio
86 /// \version 1.0
87 /// \date 25/05/98
88
89 class DeformationPP : public Deformation
90 {
91 public :
92
93
94 // CONSTRUCTEURS :
95 DeformationPP () ; // par default ne doit pas etre utilise -> message
96 // d'erreur en phase de mise au point
97 // constructeur normal dans le cas d'un ou de plusieurs pt d'integration
98 // tabDphi et tabPhi sont relatifs aux fonctions d'interpolation
99 // la terminaison H est relative aux grandeurs d'épaisseur, S pour la surface
100 DeformationPP (Met_abstraite & ,Tableau<Noeud *>& tabnoeud
101 ,Tableau <Mat_pleine> const & tabDphiH,Tableau <Vecteur> const & tabPhiH
102 ,Tableau <Mat_pleine> const & tabDphiS,Tableau <Vecteur> const & tabPhiS);
103 // constructeur de copie
104 DeformationPP (const DeformationPP &);
105 // DESTRUCTEUR :
106 virtual ~DeformationPP ();
107
108 // définition du déformation du même type, permet d'utiliser des types dérivée surchargé
109 virtual Deformation * Nevez_deformation(Tableau <Noeud *> & tabN) const
110 { DeformationPP* def = new DeformationPP(*this);def->PointeurTableauNoeud(tabN);return def;} ;
111
112 // METHODES PUBLIQUES :
113 // Surcharge de l'operateur = : realise l'affectation
114 // fonction virtuelle, normalement ne devrait pas être utilisé
115 // si c'est le cas -> affichage d'un message d'erreur
116 Deformation& operator= (const Deformation& def);
117 // fonction normale
118 virtual DeformationPP& operator= (const DeformationPP& def);
119
120 // ===== changement de grandeurs stockees =====
121 // change les numeros d'integration de l'axe ou du plan et d'epaisseur courant
122 // ntotalaxpl : nombre total de pt d'integ de l'axe ou du plan
123 // niaxpl : nouveau point de l'axe ou du plan
124 // niepaiss : nouveau point d'epaisseur
125 // epaisseur : l'epaisseur courante
126 virtual void ChangeNumIntegSH(int niaxpl, int niepaiss);
127
128 // ===== calcul des raideurs =====
129 // calcul explicite à t : tous les parametres sont de resultats
130 const Met_abstraite::Expli& Cal_explicit_t
131 ( const Tableau <double>& def_equi_t,TenseurBB & epsBB_t,Tableau <TenseurBB *> &
d_epsBB
132 ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& DeltaEpsBB,bool
premier_calcul);
133 // calcul explicite à tdt : tous les parametres sont de resultats
134 const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt
135 ( const Tableau <double>& def_equi_t,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> &
d_epsBB
136 ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& delta_epsBB_tdt,bool
premier_calcul);
137 // cas implicite
138 const Met_abstraite::Impli& Cal_implicit
139 ( const Tableau <double>& def_equi_t,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
140 ,Tableau <double>& def_equi,Tableau2 <TenseurBB *>& d2_epsBB_tdt,TenseurBB& DepsBB,TenseurBB&
delta_epsBB,bool premier_calcul);
141
142 // ----- calcul des variables primaires autre que pour la mécanique -----
143 // ----- donc par de retour relatif aux déformations -----
144 // calcul explicite à t : tous les parametres sont des resultats
145 const Met_abstraite::Expli& Cal_explicit_t(bool premier_calcul);
146 // calcul explicite à tdt : tous les parametres sont des resultats
147 const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt(bool premier_calcul);
148 // cas implicite
149 const Met_abstraite::Impli& Cal_implicit(bool premier_calcul);
150
151 // ===== remontee aux informations =====

```

```

152 // calcul :
153 // M0 : point d'integration numInteg a t = 0
154 // Mt ou Mtdt : point d'integration final
155 // Aa0 et AaFin : matrice de passage initiale et finale dans un repere ortho tel que
156 // la nouvelle base Aa est calculee par projection de "Ipa" sur Gi
157 // gijHH et gijBB : metrique finale
158 // Aa(i,a) = Aa^i_{.a}, avec g^i = Aa^i_{.a} * Ip^a
159 // tout ce passe comme si Ip^a est la nouvelle base vers laquelle on veut évoluer
160
161 // cas sortie d'un calcul implicite
162 const Met_abstraite::InfoImp RemontImp(bool absolue,Mat_pleine& Aa0,Mat_pleine& AaFin);
163 // idem sans le calcul des matrices de passage
164 const Met_abstraite::InfoImp RemontImp();
165 // cas sortie d'un calcul explicite à t
166 const Met_abstraite::InfoExp_t RemontExp_t(bool absolue,Mat_pleine& Aa0,Mat_pleine& AaFin);
167 // idem sans le calcul des matrices de passage
168 const Met_abstraite::InfoExp_t RemontExp_t();
169 // cas sortie d'un calcul explicite à tdt
170 const Met_abstraite::InfoExp_tdt RemontExp_tdt(bool absolue,Mat_pleine& Aa0,Mat_pleine& AaFin);
171 // idem sans le calcul des matrices de passage
172 const Met_abstraite::InfoExp_tdt RemontExp_tdt();
173
174 // gestion du parcours de tous les points d'integration
175 virtual void PremierPtInteg();
176 virtual bool DernierPtInteg();
177 virtual void NevezPtInteg();
178
179 // methode virtuelle : determination des bases de passages
180 // (seules les donnees en entrees et sortie sont utilisees )
181 void BasePassage(bool absolue,const BaseB & giB0,const BaseB & giB,const BaseH & giH0,
182                 const BaseH & giH,
183                 Mat_pleine& Aa0,Mat_pleine& AaFin);
184
185 protected :
186 // VARIABLES PROTEGEES :
187
188 Tableau <Mat_pleine> const * tabDphiH; // derivees des fonctions d'interpolation suivant H
189 Tableau <Vecteur> const * tabPhiH; // les fonctions d'interpolation suivant H
190 // les tableaux tabDphi et tabPhi definis dans la classe mere Deformation, sont utilisees
191 // pour le stockage de l'interpolation suivant l'axe ou le plan.
192
193 // +++++ des variables qui vont varier au cours du calcul: variables transitoires
194 // on les definit en static pour qui n'encombre pas la memoire, et on les definit
195 // dans la classe pour qui soient accessibles à tous les methodes
196 static int numInteg_ep; // numero du point d'integration en cours dans l'epaisseur
197 // numInteg defini dans la classe mere represente le numero du point d'integration
198 // en cours sur l'axe ou sur le plan
199 static int nbtotalep; // nombre total de pt integ de epaisseur
200 static int nbtotalaxpl; // nombre total de pt integ de sur l'axe ou sur le plan
201 // nbNoeud defini dans la classe mere, represente le nombre de noeud de l'axe ou du plan
202 // mais pas de l'epaisseur
203 static Vecteur theta_z; // vecteur intermediaire dont l'element 1 = la cote du point courant
204 // +++++ fin des variables qui vont varier au cours du calcul: variables transitoires
205
206 static Tableau < Mat_pleine const * > taDphi; // derivees des fonctions d'interpolation courantes
207 static Tableau <Vecteur const * > taPhi; // les fonctions d'interpolation courantes
208
209
210 // METHODES PROTEGEES :
211 void BasePassage(BaseB & giB0,BaseB & giB,BaseH & giH0,BaseH & giH,
212                 Mat_pleine& Aa0,Mat_pleine& AaFin);
213 // applique les consequences d'un changement de numero de point d'integration
214 // par exemple effectue via NevezPtInteg() ou ChangeNumIntegSH
215 void AppliquePtInteg();
216 };
217 /// @} // end of group
218
219 #endif

```

## 7.142 Met\_abstraite.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //

```

```

15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *   *****/
38 *   BUT: Met_abstraite constitue une boite a outil qui permet
39 *   d'obtenir les diverses grandeurs liées a la metrique.
40 *   Se rappeler qu'avant la recuperation d'une grandeur il faut la
41 *   calculer* En phase de mise au point c'a-d avec l'existence de la
42 *   variable MISE_AU_POINT il y a verification des tailles etc, en phase
43 *   normale il n'y a plus de verif !
44 *
45 *   *****/
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !           but
50 *   -----
51 *   !           !           !
52 *   *****/
53 *
54 *   MODIFICATIONS:
55 *
56 *   ! date !   auteur !           but
57 *   -----
58 *
59 *   *****/
60
61 #ifndef MET_ABSTRAITE_H
62 #define MET_ABSTRAITE_H
63
64 #include <math.h>
65 #include "ParaGlob.h"
66 #include "DdlElement.h"
67 #include "Mat_pleine.h"
68 #include "Tableau_3D.h"
69 #include "Tenseur.h"
70 #include "NevezTenseur.h"
71 #include "Base.h"
72 #include "Noeud.h"
73 #include "Tableau2_T.h"
74 #include "Enum_variable_metrrique.h"
75 #include "Enum_dure.h"
76
77 /** @defgroup groupe_des_metrrique
78 *
79 *   BUT: Met_abstraite constitue une boite a outil qui permet
80 *   d'obtenir les diverses grandeurs liées a la metrique.
81 *   Se rappeler qu'avant la recuperation d'une grandeur il faut la
82 *   calculer* En phase de mise au point c'a-d avec l'existence de la
83 *   variable MISE_AU_POINT il y a verification des tailles etc, en phase
84 *   normale il n'y a plus de verif !
85 *   La metrique est concue pour etre commune a toute une classe d'element
86 *   d'ou une gestion de la memoire particuliere, vu la taille des different tableaux
87 *   en fonctionnement normal les methodes ne testent pas les differentes tailles et si
88 *   les tableaux sont correctement alloue ce qui impose de bien gerer l'allocation
89 *   a l'aide du constructeur et des routines public de gestion, ceci a chaque changement
90 *   de pb par exemple passage d'explicit a implicit
91 *
92 *   \author   Gérard Rio
93 *   \version  1.0
94 *   \date    23/01/97
95 *   \brief   groupe relatif aux calculs de métriques
96 */
97
98 /// @addtogroup groupe_des_metrrique
99 /// @{
100 ///

```

```

101
102
103 // la metrique est concue pour etre commune a toute une classe d'element
104 // d'ou une gestion de la memoire particuliere, vu la taille des different tableaux
105 // en fonctionnement normal les methodes ne testent pas les differentes tailles et si
106 // les tableaux sont correctement alloues ce qui impose de bien gerer l'allocation
107 // a l'aide du constructeur et des routines public de gestion, ceci a chaque changement
108 // de pb par exemple passage d'explicit a implicit
109
110
111 class Met_abstraite
112 {
113     public :
114
115         // Constructeur par défaut
116         Met_abstraite ();
117         // constructeur permettant de dimensionner uniquement certaines variables
118         // dim = dimension de l'espace, nbvec = nb de vecteur des bases, tab = liste
119         // des variables a initialiser
120         Met_abstraite (int dim_base,int nbvec_base,const DdlElement& tabddl,
121                       const Tableau<Enum_variable_metrrique>& tab,int nomb_noeud
122                       ,int nbddl_sup_xi = 0);
123         // Constructeur de copie :
124         Met_abstraite (const Met_abstraite& );
125
126         // DESTRUCTEUR VIRTUEL :
127
128         virtual ~Met_abstraite ();
129         // METHODES PUBLIQUES :
130         // Surcharge de l'operateur = : realise l'affectation
131         virtual Met_abstraite& operator= (const Met_abstraite& met);
132
133         // affichage minimale des éléments de métrique de base
134         virtual void Affiche() const;
135
136         // ----- calculs -----
137
138         //=====
139         // définition des différentes classes conteneurs, utilisés pour ramener les résultats après calcul |
140         #include "Met_abstraite_struc_donnees.h"
141         //=====
142
143         // calcul de Expli: cas entre 0 et t, toutes les grandeurs sont a 0 ou t
144         // gradV_instantane : true : calcul du gradient de vitesse instantannée
145         // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
146         // false: uniquement les grandeurs à t+dt sont calculées
147         virtual const Expli & Cal_explicit_t(const Tableau<Noeud *>& tab_noeud,bool gradV_instantane,
148         const Mat_pleine& tabDphi,
149         int nombre_noeud,const Vecteur& phi,bool premier_calcul);
150         // calcul de Expli_t_tdt cas explicite entre t=0 et tdt, toutes les grandeurs sont a 0 ou tdt
151         // gradV_instantane : true : calcul du gradient de vitesse instantannée
152         // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
153         // false: uniquement les grandeurs à t+dt sont calculées
154         virtual const Expli_t_tdt & Cal_explicit_tdt(const Tableau<Noeud *>& tab_noeud,bool
155         gradV_instantane, const Mat_pleine& tabDphi,
156         int nombre_noeud,const Vecteur& phi,bool premier_calcul);
157         // récup du conteneur Expli_t_dt
158         virtual const Expli_t_tdt & Conteneur_Expli_t_tdt() const {return ex_expli_t_tdt;};
159         // calcul de impli. Par défaut tous les termes de variation sont calculés (le D...)
160         // mais dans certain cas ils ne sont pas nécessaire. le paramètre avec_D est alors
161         // a mettre à false et le calcul de toutes les variations est omis
162         // gradV_instantane : true : calcul du gradient de vitesse instantannée
163         // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
164         // false: uniquement les grandeurs à t+dt sont calculées
165         // avec_var_Xi : par défaut true, si false indique que l'on ne calcule pas les variations / au ddl
166         // Xi
167         virtual const Impli & Cal_implicit(const Tableau<Noeud *>& tab_noeud,bool gradV_instantane,
168         const Mat_pleine& tabDphi,
169         int nombre_noeud,const Vecteur& phi,bool premier_calcul,bool avec_var_Xi = true);
170         // calcul de gijHH_0 et de giH_0 juste après le calcul implicite ou explicite (sinon il y a erreur)
171         virtual const gijHH_0_et_giH_0 & Cal_gijHH_0_et_giH_0_apres_im_expli();
172         // récupération de grandeurs sauvegardées par ailleurs à 0 et t (et non les pointeurs)
173         void Recup_grandeur_0_t(BaseB & ggiB_0,BaseH & ggiH_0,BaseB & ggiB_t,BaseH & ggiH_t
174         ,TenseurBB & ggiBB_0,TenseurHH & ggiJHH_0,TenseurBB & ggiJB_t,TenseurHH &
175         ggiJHH_t
176         ,TenseurBB & ggradVmoyBB_t,double & gjacobien_t,double & gjacobien_0)
177         {ex_impli.Recup_grandeur_0_t(ggiB_0,ggiH_0,ggiB_t,ggiH_t,ggiJB_t,ggiJHH_0,ggiJB_t
178         ,ggiJHH_t,ggradVmoyBB_t,gjacobien_t,gjacobien_0);
179         }; // met à jour en même temps tous les autres conteneurs pour les grandeurs équivalentes
180         // récupération de grandeurs sauvegardées par ailleurs à 0 (et non les pointeurs)
181         void Recup_grandeur_0(BaseB & ggiB_0,BaseH & ggiH_0,TenseurBB & ggiJB_0,TenseurHH & ggiJHH_0
182         ,double & gjacobien_0)
183         {ex_expli.Recup_grandeur_0(ggiB_0,ggiH_0,ggiJB_0,ggiJHH_0,gjacobien_0);
184         }; // met à jour en même temps tous les autres conteneurs pour les grandeurs équivalentes
185
186         // construction d'une métrique umat
187         // - construction d'une métrique qui dépend du paramétrage matériel = X_(0)^a

```

```

183 // - en fonction d'une métrique normale fonction de l'interpolation sur
184 // les éléments de référence c'est-à-dire theta^i
185 virtual const Umat_cont& Construction_Umat(const Met_abstraite::Impli& umat_cont);
186 // idem à partir d'une umat_cont
187 virtual const Umat_cont& Construction_Umat(const Met_abstraite::Umat_cont& umat_cont);
188 // récup du conteneur umat
189 virtual const Umat_cont& Conteneur_Umat() const {return umat_cont;};
190
191 // ----- cas de la dynamique -----
192 // calcul pour la matrice masse
193 virtual const Dynamiq& Cal_pourMatMass(const Tableau<Noeud *>& tab_noeud, const Mat_pleine&
tabDphi
194                                     ,int nombre_noeud,const Vecteur& phi);
195 // ----- flambage linéaire -----
196 // calcul des termes de la classe flambe_lin
197 virtual const flambe_lin& Cal_flambe_lin(const Tableau<Noeud *>& tab_noeud,bool gradV_instantane,
const Mat_pleine& tabDphi,
198                                     int nombre_noeud,const Vecteur& phi,bool premier_calcul)
199 { return (flambe_lin&) Cal_implicit(tab_noeud,gradV_instantane,tabDphi,nombre_noeud
200                                     ,phi,premier_calcul);};
201
202 // ----- calcul de données interpolées aux même points d'integ que la mécanique
-----
203 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
204 virtual double DonneeInterpoleeScalaire
205     (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi,int nombre_noeud
206     ,Enum_ddl enu,Enum_dure temps) const;
207 // ----- calcul de la variation d'une donnée interpolée par rapport aux ddl de la donnée -----
208 // aux même points d'integ que la mécanique -----
209 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
210 // en sortie : d_A
211 virtual Tableau<double >& DerDonneeInterpoleeScalaire
212     (Tableau<double >& d_A, const Tableau<Noeud *>& tab_noeud,const Vecteur& phi
213     ,int nombre_noeud,Enum_ddl enu) const;
214 // ----- calcul du gradient d'une donnée interpolée aux même points d'integ que la
mécanique -----
215 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
216 // en sortie : gradB
217 virtual CoordonneeB& GradDonneeInterpoleeScalaire
218     (CoordonneeB& gradB,const Tableau<Noeud *>& tab_noeud,const Mat_pleine& dphi
219     ,int nombre_noeud,Enum_ddl enu,Enum_dure temps) const;
220
221 // ----- calcul de la variation du gradient d'une donnée interpolée par rapport aux ddl de la
donnée -----
222 // aux même points d'integ que la mécanique -----
223 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
224 // en sortie : d_gradB
225 virtual Tableau<CoordonneeB >& DerGradDonneeInterpoleeScalaire
226     (Tableau<CoordonneeB >& d_gradB,const Tableau<Noeud *>& tab_noeud
227     ,const Vecteur& phi,const Mat_pleine& dphi,int nombre_noeud,Enum_ddl enu) const;
228 // ----- calcul de la variation seconde du gradient d'une donnée interpolée par rapport aux ddl
de la donnée -----
229 // aux même points d'integ que la mécanique -----
230 // et utilisées par la mécanique (ex: température, taux de cristallinité, vitesse, etc..)
231 // en sortie : d2_gradB
232 // si d2_gradB est NULL cela signifie qu'il ne faut pas considérer cette grandeur
233 virtual Tableau2<CoordonneeB*> Der2GradDonneeInterpoleeScalaire
234     (Tableau2<CoordonneeB*> d2_gradB,const Tableau<Noeud *>& tab_noeud
235     ,const Vecteur& phi,const Mat_pleine& dphi,int nombre_noeud,Enum_ddl enu) const;
236 // Vecteur& DonneeInterpoleeVecteur
237 // (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi,int nombre_noeud
238 // ,Enum_ddl enu,Vecteur& v, Enum_dure temps);
239
240 // ----- variation seconde de la métrique -----
241 // calcul et récupération de la variation seconde de la métrique à t+dt
242 // total = true : indique que le calcul est complet, à partir des données de base
243 // c-a-d calcul de la variation de vecteur de base puis calcul de la variation
seconde
244 // = false: le calcul est simplifié, on considère que la variation des vecteurs de base
vient
245 // juste d'être calculé, dans un calcul implicite, on ne calcul alors que la
variation seconde
246 const Tableau2<TenseurBB * > & CalVar2GijBBtdt(bool total,const Mat_pleine& dphi,int
nombre_noeud,const Vecteur& phi);
247
248
249 // ===== remontee aux informations =====
250
251 // cas sortie d'infoImp
252 // calcul des termes de la classe InfoImp
253 virtual const InfoImp& Cal_InfoImp(const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,
const Vecteur& phi, int nombre_noeud);
254 // récup du conteneur qui a déjà été calculé
255 virtual const InfoImp& Recup_InfoImp() {return ex_InfoImp;};
256
257 // cas sortie d'infoExp à 0 et t
258 // calcul des termes de la classe InfoExp

```

```

260     virtual const InfoExp_t& Cal_InfoExp_t(const Tableau<Noeud *>& tab_noeud, const Mat_pleine&
tabDphi,
261         const Vecteur& phi, int nombre_noeud);
262     // récup du conteneur qui a déjà été calculé
263     virtual const InfoExp_t& Recup_InfoExp_t() {return ex_InfoExp_t;};
264
265     // cas sortie d'infoExp à 0 et tdt
266     // calcul des termes de la classe InfoExp
267     virtual const InfoExp_tdt& Cal_InfoExp_tdt(const Tableau<Noeud *>& tab_noeud, const Mat_pleine&
tabDphi,
268         const Vecteur& phi, int nombre_noeud);
269     // récup du conteneur qui a déjà été calculé
270     virtual const InfoExp_tdt& Recup_InfoExp_tdt() {return ex_InfoExp_tdt;};
271
272     // cas sortie d'info à 0 t et tdt: point, giB, giH, calcul des termes de la classe Info0_t_tdt
273     virtual const Info0_t_tdt& Cal_Info0_t_tdt(const Tableau<Noeud *>& tab_noeud, const Mat_pleine&
tabDphi
274         ,const Vecteur& phi, int nombre_noeud);
275     // récup du conteneur qui a déjà été calculé
276     virtual const Info0_t_tdt& Recup_Info0_t_tdt() {return ex_Info0_t_tdt;};
277
278     // récup du conteneur avec métrique qui a déjà été calculé
279     virtual const Info_et_metrrique_0_t_tdt& Recup_Info_et_metrrique_0_t_tdt() {return umat_cont;};
280
281     // cas du calcul de la déformation d'Almansi uniquement
282     class Pour_def_Almansi{public: TenseurBB * gijBB;TenseurBB * gijBB_0;
283         Pour_def_Almansi(): gijBB(NULL), gijBB_0(NULL) {};
284         Pour_def_Almansi(const Pour_def_Almansi& a ) : gijBB(a.gijBB), gijBB_0(a.gijBB_0) {};
285     };
286     virtual const Pour_def_Almansi Cal_pour_def_Almansi_au_temps
287         (Enum_dure temps,const Tableau<Noeud *>& tab_noeud,const Mat_pleine& dphi
288         ,int nombre_noeud,const Vecteur& phi);
289     // cas du calcul de la déformation logarithmique uniquement
290     class Pour_def_log{public: TenseurBB * gijBB;TenseurHH * gijHH;TenseurBB * gijBB_0;TenseurHH *
gijHH_0;
291         Pour_def_log():gijBB(NULL),gijHH(NULL),gijBB_0(NULL),gijHH_0(NULL) {};
292         Pour_def_log(const Pour_def_log& a) :
293             gijBB(a.gijBB),gijHH(a.gijHH),gijBB_0(a.gijBB_0),gijHH_0(a.gijHH_0) {};
294     };
295     virtual const Pour_def_log Cal_pour_def_log_au_temps
296         (Enum_dure temps,const Tableau<Noeud *>& tab_noeud,const Mat_pleine& dphi
297         ,int nombre_noeud,const Vecteur& phi);
298
299     // ===== utilise par le contact et autres =====
300     // calcul d'un point M en fonction de phi et des coordonnees a 0
301     virtual const Coordonnee & PointM_0 // ( stockage a t=0)
302         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
303
304     // calcul d'un point M en fonction de phi et des coordonnees a t
305     virtual const Coordonnee & PointM_t // ( stockage a t=t)
306         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
307
308     // calcul de la variation d'un point M par rapport aux ddl ,
309     // en fonction de phi et des coordonnees a t
310     virtual const Tableau<Coordonnee> & D_PointM_t // ( stockage a t)
311         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
312
313     // calcul d'un point M en fonction de phi et des coordonnees a tdt
314     virtual const Coordonnee & PointM_tdt // ( stockage a tdt)
315         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
316
317     // calcul de la variation d'un point M par rapport aux ddl ,
318     // en fonction de phi et des coordonnees a tdt
319     virtual const Tableau<Coordonnee> & D_PointM_tdt // ( stockage a tdt)
320         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
321
322     // calcul de la vitesse du point M en fonction de phi et des coordonnees a 0
323     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
324     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
325     virtual const Coordonnee & VitesseM_0 // ( stockage a t=0)
326         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
327     // idem mais au noeud passé en paramètre
328     virtual const Coordonnee & VitesseM_0 (const Noeud* noeud);
329
330     // calcul de la vitesse du point M en fonction de phi et des coordonnees a t
331     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
332     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
333     virtual const Coordonnee & VitesseM_t // ( stockage a t=t)
334         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
335     // idem mais au noeud passé en paramètre
336     virtual const Coordonnee & VitesseM_t (const Noeud* noeud);
337
338     // calcul de la variation de la vitesse du point M par rapport aux ddl ,
339     // en fonction de phi et des coordonnees a t
340     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
341     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
342     // ddl_vitesse : indique si oui ou non il s'agit de variation par rapport

```

```

343         //          aux ddl de vitesse ou au ddl de position
344     virtual const Tableau<Coordonnee> & D_VitesseM_t // ( stockage a t)
345     (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi,bool ddl_vitesse);
346     // idem mais au noeud passé en paramètre
347     virtual const Tableau<Coordonnee> & D_VitesseM_t (const Noeud* noeud,bool ddl_vitesse);
348
349     // calcul de la vitesse du point M en fonction de phi et des coordonnees a tdt
350     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
351     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
352     // ddl_vitesse : indique si oui ou non il s'agit de variation par rapport
353     //          aux ddl de vitesse ou au ddl de position
354     virtual const Coordonnee & VitesseM_tdt // ( stockage a tdt)
355     (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
356     // idem mais au noeud passé en paramètre
357     virtual const Coordonnee & VitesseM_tdt (const Noeud* noeud);
358
359     // calcul de la variation de la vitesse du point M par rapport aux ddl ,
360     // en fonction de phi et des coordonnees a tdt
361     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
362     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
363     // ddl_vitesse : indique si oui ou non il s'agit de variation par rapport
364     //          aux ddl de vitesse ou au ddl de position
365     virtual const Tableau<Coordonnee> & D_VitesseM_tdt // ( stockage a tdt)
366     (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi,bool ddl_vitesse);
367     // idem mais au noeud passé en paramètre
368     virtual const Tableau<Coordonnee> & D_VitesseM_tdt (const Noeud* noeud,bool ddl_vitesse);
369
370     // calcul de la base naturel , au point correspondant au dphi en fonction des coord a t=0
371     virtual const BaseB& BaseNat_0 // ( stockage a t=0)
372     (const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,const Vecteur& phi);
373
374     // calcul de la base naturel , au point correspondant au dphi en fonction des coord a t
375     virtual const BaseB& BaseNat_t // ( stockage a t)
376     (const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,const Vecteur& phi);
377
378     // calcul de la base naturel , au point correspondant au dphi en fonction des coord a tdt
379     virtual const BaseB& BaseNat_tdt // ( stockage a tdt)
380     (const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,const Vecteur& phi);
381
382     // calcul de la variation de la base naturel, au point correspondant au dphi en fonction des coord
383     // a tdt
384     // nécessite que la base naturelle ait été calculée auparavant
385     virtual const Tableau<BaseB>& d_BaseNat_tdt // ( stockage a tdt)
386     (const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,const Vecteur& phi);
387
388     // calcul de la base naturelle et duale ( stockage a t=0) en fonction des coord a 0
389     virtual void BaseND_0(const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,
390     const Vecteur& phi,BaseB& bB,BaseH& bH);
391
392     // calcul de la base naturelle et duale ( stockage a t) en fonction des coord a t
393     virtual void BaseND_t(const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,
394     const Vecteur& phi,BaseB& bB,BaseH& bH);
395
396     // calcul de la base naturelle et duale ( stockage a tdt) en fonction des coord a tdt
397     virtual void BaseND_tdt(const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,
398     const Vecteur& phi,BaseB& bB,BaseH& bH);
399
400     // ===== informations diverses (mais cohérentes, c-a-d, peuvent être appelées directement sans
401     // init) =====
402     double JacobienInitial(const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi
403     ,int nombre_noeud,const Vecteur& phi);
404
405     // ----- gestion de la memoire -----
406
407     // Ajout d'initialisation de differentes variables : liste dans tab
408     virtual void PlusInitVariables(Tableau<Enum_variable_metrrique>& tab) ;
409     // initialisation de la dimension , du nb de vecteur et du tableau de ddl
410     // cette fonction n'agi qu'une fois , lors des autres appels elle ne fait rien
411     virtual void Dim_NbVec(int dim_base,int nbvec_base,DdlElement & tabddl,int nb_noeud_interpol);
412     // ramène les informations de taille utilisées par la classe
413     const int & Dim_vec_base() const {return dim_base;}; // dimension des vecteurs de base
414     const int & Nbvec_des_bases() const {return nbvec_base;}; // nb de vecteur des bases
415     const int & Nombre_de_noeud() const {return nomb_noeud;}; // nombre de noeud de l'interpolation
416     const DdlElement& Tab_ddl_element() const {return tab_ddl;}; // tableau de ddl de l'element
417
418     //===== Methodes protegees =====
419     protected:
420     //==calcul des points, par default la dimension des points est celle du probleme
421     //== transmise par ParaGlob
422     // calcul du point a t0
423     virtual void Calcul_M0
424     ( const Tableau<Noeud *>& tab_noeud,const Vecteur& phi, int nombre_noeud);
425     // calcul du point a t
426     virtual void Calcul_Mt
427     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
428     // calcul des variations du point a t
429     virtual void Calcul_d_Mt

```



```

428     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
429 // calcul du point a tdt
430 virtual void Calcul_Mtdt
431     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
432 // calcul des variations du point a tdt
433 virtual void Calcul_d_Mtdt
434     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
435
436 // calcul de la vitesse du point a t0 en fonction des ddl existants de vitesse
437 virtual void Calcul_V0
438     ( const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
439 // idem mais au noeud passé en paramètre
440 virtual void Calcul_V0 (const Noeud* noeud);
441 // calcul de la vitesse du point a t en fonction des ddl existants de vitesse
442 virtual void Calcul_Vt
443     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
444 // idem mais au noeud passé en paramètre
445 virtual void Calcul_Vt (const Noeud* noeud);
446 // calcul des variations de la vitesse du point a t en fonction des ddl existants de vitesse
447 virtual void Calcul_d_Vt
448     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
449 // idem mais au noeud passé en paramètre
450 virtual void Calcul_d_Vt (const Noeud* noeud);
451 // calcul de la vitesse du point a tdt en fonction des ddl existants de vitesse
452 virtual void Calcul_Vtdt
453     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
454 // idem mais au noeud passé en paramètre
455 virtual void Calcul_Vtdt (const Noeud* noeud);
456 // calcul des variations de la vitesse du point a tdt en fonction des ddl existants de vitesse
457 virtual void Calcul_d_Vtdt
458     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
459 // idem mais au noeud passé en paramètre
460 virtual void Calcul_d_Vtdt (const Noeud* noeud);
461
462 // calcul de la vitesse moyenne du point a t0 en fonction des ddl de position
463 virtual void Calcul_V_moy0
464     ( const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
465 // idem mais au noeud passé en paramètre
466 virtual void Calcul_V_moy0 (const Noeud* noeud);
467 // calcul de la vitesse moyenne du point a t en fonction des ddl de position
468 virtual void Calcul_V_moyt
469     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
470 // idem mais au noeud passé en paramètre
471 virtual void Calcul_V_moyt (const Noeud* noeud);
472 // calcul des variations de la vitesse moyenne du point a t en fonction des ddl de position
473 virtual void Calcul_d_V_moyt
474     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
475 // idem mais au noeud passé en paramètre
476 virtual void Calcul_d_V_moyt (const Noeud* noeud);
477 // calcul de la vitesse moyenne du point a tdt en fonction des ddl de position
478 virtual void Calcul_V_moytdt
479     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
480 // idem mais au noeud passé en paramètre
481 virtual void Calcul_V_moytdt (const Noeud* noeud);
482 // calcul des variations de la vitesse moyenne du point a tdt en fonction des ddl de position
483 virtual void Calcul_d_V_moytdt
484     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
485 // idem mais au noeud passé en paramètre
486 virtual void Calcul_d_V_moytdt (const Noeud* noeud);
487
488 //== par default le nombre de vecteur de base est celui de la dimension du
489 //== probleme transmis par paraglob
490 // calcul de la base naturel a t0
491 virtual void Calcul_giB_0
492     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi, int nombre_noeud, const Vecteur&
493     phi);
494 // calcul de la base naturel a t
495 virtual void Calcul_giB_t
496     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi, int nombre_noeud, const Vecteur&
497     phi);
498 // calcul de la base naturel a t+dt
499 virtual void Calcul_giB_tdt
500     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi, int nombre_noeud, const Vecteur&
501     phi);
502 //== par default la metrique est de type (nbvecteur de la base) au carre
503 //== ce qui permet d'avoir une metrique 2*2 en 3D
504 //== !! necessite d'avoir calcule la base naturelle corespondante auparavant
505 // Calcul des composantes covariantes de la metrique de la base naturelle a 0 :
506 virtual void Calcul_gijBB_0 ();
507 // Calcul des composantes covariantes de la metrique de la base naturelle a t :
508 virtual void Calcul_gijBB_t ();
509 // Calcul des composantes covariantes de la metrique de la base naturelle a tdt :
510 virtual void Calcul_gijBB_tdt ();
511 // Calcul des composantes contravariantes de la metrique de la base naturelle a 0 :
512 virtual void Calcul_gijHH_0 ();
513 // Calcul des composantes contravariantes de la metrique de la base naturelle a t :

```



```

512 virtual void Calcul_gijHH_t () ;
513 // Calcul des composantes contravariantes de la metrique de la base naturelle a tdt :
514 virtual void Calcul_gijHH_tdt () ;
515
516 //== par défaut le gradient de vitesse est de type (nbvecteurde la base)au carre
517 //== nécessite d'avoir calculé les vecteurs giB avant
518 // calcul du gradient de vitesse à t
519 virtual void Calcul_gradVBB_t
520     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
521 // calcul gradient de vitesse à t+dt
522 virtual void Calcul_gradVBB_tdt
523     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
524 //== dans le cas où il n'y a pas de ddl de vitesse on peut utiliser les vitesses moyennes
525 //== correspondant à delta x^ar/delta t
526 // calcul du gradient de vitesse moyen à t
527 // dans le cas où les ddl à tdt n'existent pas -> utilisation de la vitesse sécante entre 0
    et t !!
528 virtual void Calcul_gradVBB_moyen_t
529     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
530 // calcul du gradient de vitesse moyen entre t et t+dt
531 virtual void Calcul_gradVBB_moyen_tdt
532     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
533
534 //== par default le nombre de vecteur de la base duale est identique
535 //== a celui de la base naturelle ainsi on peu etre es 3d est avoir uniquement
536 //== un seul vecteur de base (cas d'une biellette par exemple
537 //== mais il faut surcharger le cacul de la base
538 // !!! important : il est necessaire de calculer les composantes covariantes
539 // puis contravariantes de la metrique avant de calculer les bases duales
540 // calcul de la base duale a t0
541 virtual void Calcul_giH_0();
542 // calcul de la base duale a t
543 virtual void Calcul_giH_t();
544 // calcul de la base duale a t+dt
545 virtual void Calcul_giH_tdt();
546
547 //== calcul du jacobien aux differents temps , il est necessaire d'avoir
548 //== calcule le tenseur metrique correspondant
549 virtual void Jacobien_0();
550 virtual void Jacobien_t();
551 virtual void Jacobien_tdt();
552
553 //== calcul de la variation des bases
554 //== par default le nombre de ddl est egal a la dimension du pb
555 //== * le nombre de vecteur de la base
556 virtual void D_giB_t( const Mat_pleine& tabDphi,
557     int nbnoeu,const Vecteur & phi); // avant calcul de : giB_t
558 virtual void D_giB_tdt( const Mat_pleine& tabDphi,
559     int nbnoeu,const Vecteur & phi); // avant calcul de : giB_tdt
560 virtual void D_giH_t(); // avant calcul de : d_giB_t et giH_t
561 virtual void D_giH_tdt(); // avant calcul de : d_giB_tdt et giH_tdt
562
563 //== calcul de la variation du gradient
564 // par rapport aux composantes V^ar (et non les X^ar )
565 virtual void DgradVBB_t( const Mat_pleine& dphi); // avant calcul de : giB_t
566 // par rapport aux composantes V^ar (et non les X^ar )
567 virtual void DgradVBB_tdt(const Mat_pleine& dphi); // avant calcul de : giB_tdt
568 // par rapport aux composantes X^ar (et non les V^ar )
569 virtual void DgradVmoyBB_t(const Mat_pleine& dphi,const Tableau<Noeud *>& tab_noeud); //
calcul variation du gradient moyen à t
570 // par rapport aux composantes X^ar (et non les V^ar )
571 virtual void DgradVmoyBB_tdt(const Mat_pleine& dphi); // calcul variation du gradient moyen à
tdt
572
573 //== calcul de la variation des metriques
574 virtual void D_gijBB_t(); // avant calcul de : d_giB_t et giB_t, Prepa_calcul_DgradVmoy_t()
575 virtual void D_gijBB_tdt(); // avant calcul de : d_giB_tdt et giB_tdt, Prepa_calcul_DgradVmoy_tdt()
576 virtual void D_gijHH_t(); // avant calcul de : d_giH_t et giH_t
577 virtual void D_gijHH_tdt(); // avant calcul de : d_giH_tdt et giH_tdt
578
579 //== calcul de la variation seconde des metriques
580 // ici on ne considère que la variation première des vecteurs de base, c-a-d que
581 // l'on considère que leurs variations secondes sont nulles
582 // ce qui est vrai dans le cas d'une cinématique simple
583 virtual void D2_gijBB_tdt(); // avant calcul de : d_giB_tdt et giB_tdt
584
585 //== calcul de la variation des jacobiens
586 virtual void Djacobien_t(); // avant calcul de : d_giB_t et giB_t
587 virtual void Djacobien_tdt(); // avant calcul de : d_giB_tdt et giB_tdt
588
589 // donnee protegees
590 DdlElement tab_ddl; // tableau de ddl de l'element
591 int nbddl_sup_xi; // nombre de ddl supplémentaire à ajouter aux ddl normaux en x,v et gamma
592 // par défaut = 0, mais les classes dérivées ex: les coques
593 // peuvent changer ce nombre, ce qui permet de réserver de la place pour des ddls
594 // spécifiques, exemple l'épaisseur, dans le cas des coques
595 int dim_base ; // dimension des vecteurs de base

```

```

596 int nbvec_base; // nb de vecteur des bases
597 int nomb_noeud; // nombre de noeud de l'interpolation
598 Tableau<Enum_variable_metrrique> tab; // tableau
599
600 Coordonnee * M0; // point a t=0
601 Coordonnee * Mt ; // point a l'instant t
602 Coordonnee * Mtdt; // point a l'instant t+dt
603 Tableau <Coordonnee> * d_Mt ; // derivees au point a l'instant t, par rapport aux ddl
604 Tableau <Coordonnee> * d_Mtdt; // derivees au point a l'instant t+dt, par rapport aux ddl
605
606 Coordonnee * V0; // vitesse du point a t=0
607 Coordonnee * Vt ; // vitesse du point a l'instant t
608 Coordonnee * Vtdt; // vitesse du point a l'instant t+dt
609 Tableau <Coordonnee> * d_Vt ; // derivees de la vitesse du point a l'instant t, par rapport aux
ddl
610 Tableau <Coordonnee> * d_Vtdt; // derivees de la vitesse du point a l'instant t+dt, par rapport
aux ddl
611
612 Tableau<Coordonnee> V_moy_t,V_moy_tdt; // variable interne: vitesse moyenne à t et tdt en chaque
noeud r,
613 // calculé dans le calcul du gradient utilisé, dans le calcul de la variation du
gradient moyen
614 BaseB * dmatV_moy_t,* dmatV_moy_tdt; // variable interne: dérivée par rapport au repère
615 // matériel (d/dteta_i) au point interpolé
616
617 BaseB * giB_0 ; // base naturelle a t=0
618 BaseB * giB_t ; // base naturelle a t
619 BaseB * giB_tdt ; // base naturelle a t+dt
620
621 BaseH * giH_0; // vecteur de la base duale a 0
622 BaseH * giH_t; // vecteur de la base duale a t
623 BaseH * giH_tdt; // vecteur de la base duale a t+dt
624
625 TenseurBB * gijBB_0; // composantes de la metrique de la base naturelle a t
626 TenseurBB * gijBB_t; // composantes de la metrique de la base naturelle a t
627 TenseurBB * gijBB_tdt; // composantes de la metrique de la base naturelle a tdt
628 TenseurHH * gijHH_0; // composantes de la metrique de la base duale a t=0
629 TenseurHH * gijHH_t; // composantes de la metrique de la base duale a t
630 TenseurHH * gijHH_tdt; // composantes de la metrique de la base duale a t+dt
631
632 double jacobien_0; // jacobien a l'instant t=0
633 double jacobien_t; // jacobien a l'instant t
634 double jacobien_tdt; // jacobien a l'instant t+dt
635
636 Tableau <BaseB> * d_giB_t; // derivees de la base naturelle a t par rapport
// aux degres de liberte de position
637 Tableau <BaseB> * d_giB_tdt; // derivees de la base naturelle a t+dt par rapport
// aux degres de liberte de position
638
639 Tableau <BaseH> * d_giH_t; // derivees de la base duale a t par rapport
// aux degres de liberte de position
640
641 Tableau <BaseH> * d_giH_tdt; // derivees de la base duale a t+dt par rapport
// aux degres de liberte de position
642
643 Tableau <TenseurBB * > d_gijBB_t; // derivees de la metrique de la base naturelle
// a t par rapport aux degres de liberte de position
644 Tableau <TenseurBB * > d_gijBB_tdt; // derivees de la metrique de la base naturelle
// a t+dt par rapport aux degres de liberte de position
645 Tableau2 <TenseurBB * > d2_gijBB_tdt; // derivees seconde de la metrique de la base naturelle
// a t+dt par rapport aux degres de liberte de position
646 Tableau <TenseurHH * > d_gijHH_t; // derivees de la metrique de la base duale a t
647 Tableau <TenseurHH * > d_gijHH_tdt; // derivees de la metrique de la base duale a t+dt
648
649 Vecteur d_jacobien_t ; // derivees du jacobien à t par rapport aux degres de liberte de position
650
651 Vecteur d_jacobien_tdt ; // derivees du jacobien à tdt par rapport aux degres de liberte de position
652
653 // *** convention : grad(i,j) = vilj ***
654 TenseurBB * gradVmoyBB_t; // composantes du gradient de vitesse moyen à t
655
656 TenseurBB * gradVmoyBB_tdt; // composantes du gradient de vitesse moyen à tdt
657 TenseurBB * gradVBB_t; // composantes du gradient de vitesse instantatnée à t
658
659 TenseurBB * gradVBB_tdt; // composantes du gradient de vitesse instantatnée à tdt
660
661 Tableau <TenseurBB * > d_gradVmoyBB_t; // dérivée du gradient de vitesse moyen à t par rapport aux
ddl de vitesse
662 Tableau <TenseurBB * > d_gradVmoyBB_tdt; // dérivée du gradient de vitesse moyen à tdt par rapport
aux ddl de vitesse
663 Tableau <TenseurBB * > d_gradVBB_t; // dérivée du gradient de vitesse instantatnée à t par rapport
aux ddl de vitesse
664 Tableau <TenseurBB * > d_gradVBB_tdt; // dérivée du gradient de vitesse instantatnée à tdt par
rapport aux ddl de vitesse
665
666 // maintenant définition des différents conteneurs de retour des méthodes globales
667 Impli ex_impli; // cas d'un calcul implicite
668
669
670

```

```

671 Expli ex_expli; // calcul explicite 0 t
672 Expli_t_tdt ex_expli_t_tdt; // calcul explicite 0 t tdt
673 gijHH_0_et_giH_0 ex_gijHH_0_et_giH_0; // comme le nom l'indique !
674 Dynamiq ex_Dynamiq; // calcul pour la dynamique (partie spécifique à quelques grandeurs)
675 flambe_lin ex_flambe_lin; // grandeurs pour le flambement linéaire
676 Umat_cont umat_cont; // calcul relatif aux Umat
677 InfoImp ex_InfoImp; // sortie d'info en implicite
678 InfoExp_t ex_InfoExp_t; // sortie d'info en explicite à t
679 InfoExp_tdt ex_InfoExp_tdt; // sortie d'info en explicite à tdt
680 Info0_t_tdt ex_Info0_t_tdt; // sortie d'info à 0 t et tdt
681
682 // fonctions privees
683 // Calcul des composantes de la metrique d'une base naturelle:
684 void Calcul_gijBB (BaseB & giB, TenseurBB & gijBB) ;
685 void Calcul_giH (BaseB* giB, TenseurHH* gijHH, BaseH * giH);
686 //== Existe, recherche si un element de l'enumeration existe
687 // retourne vrai si x appartient au tableau false sinon
688 bool Existe (Tableau<Enum_variable_metrrique>& tab, const Enum_variable_metrrique& x);
689 int posi_tab; // pour le programme existe
690 //== Existe_num, recherche si un element de l'enumeration existe
691 // retourne le numéro si x appartient au tableau, 0 sinon
692 int Existe_num (Tableau<Enum_variable_metrrique>& tab, const Enum_variable_metrrique& x);
693 // allocation de la memoire
694 void Allocation ();
695 // deallocation de la memoire
696 void Deallocation(); // de toutes les grandeurs
697 // uniquement des grandeurs de tib
698 void Deallocation (Tableau<Enum_variable_metrrique>& tib);
699 // copie en fonction de l'instance passée
700 // concerne que les grandeurs pointées
701 void Copie_met (const Met_abstraite & a);
702 // initialisation des conteneurs de retour des méthodes
703 void Init_conteneur_de_retour_methode();
704 // changement de nbddl_sup_xi par les classes dérivées
705 void Change_nbddl_sup_xi (int nbddl_sup_xi_new) {nbddl_sup_xi=nbddl_sup_xi_new;};
706
707
708 };
709 /// @} // end of group
710
711
712 #endif

```

## 7.143 Met\_abstraite\_struc\_donnees.h

```

1 /*! \file Met_abstraite_stuc_donnees.h
2 \brief def des structures de données de passage d'informations pour les métriques
3 */
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *
35 * DATE: 15/01/97 *
36 * * $ *
37 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
38 * * $ *
39 * PROJET: Herezh++ *
40 * * $ *

```

```

41 *****
42 * deuxième partie du fichier Met_abstraite.h
43 * contient en fait toutes les structures de données de passage d'informations
44 *
45 *      *****
46 *
47 * VERIFICATION:
48 * ! date ! auteur ! but
49 * -----
50 * ! ! !
51 *
52 *
53 *      *****
54 * MODIFICATIONS:
55 *
56 * ! date ! auteur ! but
57 * -----
58 *
59 *
60 *
61 *****/
62
63
64 // deuxième partie du fichier Met_abstraite.h
65 // contient en fait toutes les structures de données de passage d'informations
66
67 // CLASSE CONTENEUR : cas explicite entre 0 et t, toutes les grandeurs sont a 0 ou t
68 // les données sont publiques
69 class Expli_t_tdt;
70 /// @} // end of group
71
72 /// @addtogroup groupe_des_metrique
73 /// @{
74 ///
75
76 class Expli
77 { public :
78     Expli () : // constructeur par défaut
79         giB_0 (NULL), giH_0 (NULL), giB_t (NULL), giH_t (NULL),
80
81         giJBB_0 (NULL), giJHH_0 (NULL), giJBB_t (NULL), giJHH_t (NULL), gradVBB_t (NULL), gradVmoyBB_t (NULL), d_giJBB_t (NULL),
82         jacobien_t (NULL), jacobien_0 (NULL)
83     {};
84     Expli // constructeur normal
85     (BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t, TenseurBB *
86     ggiJBB_0, TenseurHH * ggiJHH_0, TenseurBB * ggiJBB_t,
87     TenseurHH * ggiJHH_t, TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVBB_t,
88     Tableau <TenseurBB *>* gd_giJBB_t, double* gjacobien_t, double* gjacobien_0) :
89         giB_0 (ggiB_0), giH_0 (ggiH_0), giB_t (ggiB_t), giH_t (ggiH_t)
90
91         , giJBB_0 (ggiJBB_0), giJHH_0 (ggiJHH_0), giJBB_t (ggiJBB_t), giJHH_t (ggiJHH_t), gradVBB_t (ggradVBB_t)
92         , gradVmoyBB_t (ggradVmoyBB_t), d_giJBB_t (gd_giJBB_t)
93         , jacobien_t (gjacobien_t), jacobien_0 (gjacobien_0)
94     {};
95     Expli (const Expli& ex) : // constructeur de copie
96         giB_0 (ex.giB_0), giH_0 (ex.giH_0), giB_t (ex.giB_t), giH_t (ex.giH_t),
97         giJBB_0 (ex.giJBB_0), giJHH_0 (ex.giJHH_0), giJBB_t (ex.giJBB_t), giJHH_t (ex.giJHH_t)
98         , gradVBB_t (ex.gradVBB_t), gradVmoyBB_t (ex.gradVmoyBB_t), d_giJBB_t (ex.d_giJBB_t),
99         jacobien_t (ex.jacobien_t), jacobien_0 (ex.jacobien_0)
100     {};
101     Expli& operator= (const Expli& ex) // surcharge d'affectation
102     {giB_0=ex.giB_0; giH_0=ex.giH_0; giB_t=ex.giB_t; giH_t=ex.giH_t;
103     giJBB_0=ex.giJBB_0; giJHH_0=ex.giJHH_0; giJBB_t=ex.giJBB_t; giJHH_t=ex.giJHH_t;
104     gradVmoyBB_t=ex.gradVmoyBB_t; gradVBB_t=ex.gradVBB_t, d_giJBB_t=ex.d_giJBB_t;
105     jacobien_t=ex.jacobien_t; jacobien_0=ex.jacobien_0;
106     return *this;
107 };
108 // une fonction qui permet de produire un objet Expli_t_tdt mais en mettant seulement
109 les
110 // grandeur calculées à t commune à Expli_t_tdt
111 Expli_t_tdt T_dans_tdt () const
112 {return Expli_t_tdt (giB_0, giH_0, giB_t, giH_t, NULL, NULL, giJBB_0, giJHH_0, giJBB_t, giJHH_t
113 , NULL, NULL, gradVmoyBB_t, NULL, NULL, NULL, NULL, jacobien_t, jacobien_0);
114 };
115 // mise à jour des grandeurs
116 void Mise_a_jour_grandeur (BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t,
117 TenseurBB * ggiJBB_0, TenseurHH * ggiJHH_0, TenseurBB * ggiJBB_t, TenseurHH *
118 ggiJHH_t, TenseurBB * ggradVmoyBB_t,
119 TenseurBB * ggradVBB_t, Tableau <TenseurBB *>* gd_giJBB_t, double*
120 gjacobien_t, double* gjacobien_0)
121 {giB_0=ggiB_0; giH_0=ggiH_0; giB_t=ggiB_t; giH_t=ggiH_t;
122 giJBB_0=ggiJBB_0; giJHH_0=ggiJHH_0; giJBB_t=ggiJBB_t; giJHH_t=ggiJHH_t;
123 gradVmoyBB_t=ggradVmoyBB_t; gradVBB_t=ggradVBB_t, d_giJBB_t=gd_giJBB_t;
124 jacobien_t=gjacobien_t; jacobien_0=gjacobien_0;
125 };
126 // mise à jour des grandeurs à 0 sauf les variations (pas de tableau et pas à t)

```

```

121 // à partir de grandeurs stockées par ailleurs (il s'agit ici donc d'affectation de
contenu et
122 // non de pointeur !)
123 void Recup_grandeur_0(BaseB & ggiB_0, BaseH & ggiH_0, TenseurBB& ggiBB_0, TenseurHH&
ggiHH_0
124 , double& gjacobien_0)
125 {*ggiB_0=ggiB_0; *ggiH_0=ggiH_0; *ggiBB_0=ggiBB_0; *ggiHH_0=ggiHH_0;
126 *jacobien_0=gjacobien_0;
127 };
128
129 // variables
130 BaseB * giB_0; BaseH * giH_0; BaseB * giB_t; BaseH * giH_t;
131 TenseurBB * giBB_0; TenseurHH* giHH_0; TenseurBB * giBB_t; TenseurHH * giHH_t;
132 TenseurBB * gradVmoyBB_t; TenseurBB * gradVBB_t; Tableau <TenseurBB *> d_giBB_t;
133 double *jacobien_t; double *jacobien_0;
134 };
135 /// @} // end of group
136
137
138 /// @addtogroup groupe_des_metrrique
139 /// @{
140 ///
141
142 // --- CLASSE CONTENEUR : cas d'une procédure Umat
143 // les variables sont publiques
144 class Umat_cont
145 { public :
146     Umat_cont () : // constructeur par défaut
147         giB_0(NULL), giH_0(NULL), giB_t(NULL), giH_t(NULL), giB_tdt(NULL), giH_tdt(NULL)
148         , giBB_0(NULL), giHH_0(NULL), giBB_t(NULL), giHH_t(NULL), giBB_tdt(NULL), giHH_tdt(NULL)
149         , gradVmoyBB_t(NULL), gradVmoyBB_tdt(NULL), gradVBB_tdt(NULL)
150         , jacobien_tdt(NULL), jacobien_t(NULL), jacobien_0(NULL)
151         {};
152     Umat_cont // constructeur normal
153     (BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t, BaseB * ggiB_tdt, BaseH
*ggiH_tdt
154     , TenseurBB * ggiBB_0, TenseurHH* ggiHH_0, TenseurBB * ggiBB_t, TenseurHH* ggiHH_t
155     , TenseurBB * ggiBB_tdt, TenseurHH * ggiHH_tdt
156     , TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVmoyBB_tdt, TenseurBB * ggradVBB_tdt
157     , double* gjacobien_tdt, double* gjacobien_t, double* gjacobien_0
158     ) :
159
160     giB_0(ggiB_0), giH_0(ggiH_0), giB_t(ggiB_t), giH_t(ggiH_t), giB_tdt(ggiB_tdt), giH_tdt(ggiH_tdt)
161     , giBB_0(ggiBB_0), giHH_0(ggiHH_0), giBB_t(ggiBB_t), giHH_t(ggiHH_t)
162     , giBB_tdt(ggiBB_tdt), giHH_tdt(ggiHH_tdt)
163     , gradVmoyBB_t(ggradVmoyBB_t), gradVmoyBB_tdt(ggradVmoyBB_tdt), gradVBB_tdt(ggradVBB_tdt)
164     , jacobien_tdt(gjacobien_tdt), jacobien_t(gjacobien_t), jacobien_0(gjacobien_0)
165     {};
166     Umat_cont (const Umat_cont& ex) : // constructeur de copie
167
168     giB_0(ex.giB_0), giH_0(ex.giH_0), giB_t(ex.giB_t), giH_t(ex.giH_t), giB_tdt(ex.giB_tdt), giH_tdt(ex.giH_tdt)
169     , giBB_0(ex.giBB_0), giHH_0(ex.giHH_0), giBB_t(ex.giBB_t), giHH_t(ex.giHH_t)
170     , giBB_tdt(ex.giBB_tdt), giHH_tdt(ex.giHH_tdt)
171     , gradVmoyBB_t(ex.gradVmoyBB_t), gradVmoyBB_tdt(ex.gradVmoyBB_tdt), gradVBB_tdt(ex.gradVBB_tdt)
172     , jacobien_tdt(ex.jacobien_tdt), jacobien_t(ex.jacobien_t), jacobien_0(ex.jacobien_0)
173     {};
174     Umat_cont& operator= (const Umat_cont& ex) // surcharge d'affectation
175     {
176     giB_0=ex.giB_0; giH_0=ex.giH_0; giB_t=ex.giB_t; giH_t=ex.giH_t; giB_tdt=ex.giB_tdt; giH_tdt=ex.giH_tdt;
177     giBB_0=ex.giBB_0; giHH_0=ex.giHH_0; giBB_t=ex.giBB_t; giHH_t=ex.giHH_t;
178     giBB_tdt=ex.giBB_tdt; giHH_tdt=ex.giHH_tdt;
179
180     gradVmoyBB_t=ex.gradVmoyBB_t; gradVmoyBB_tdt=ex.gradVmoyBB_tdt; gradVBB_tdt=ex.gradVBB_tdt;
181     jacobien_tdt=ex.jacobien_tdt; jacobien_t=ex.jacobien_t; jacobien_0=ex.jacobien_0;
182     return *this;
183     };
184
185     // mise à jour des grandeurs
186     void Mise_a_jour_grandeur(BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t
187     , BaseB * ggiB_tdt, BaseH * ggiH_tdt
188     , TenseurBB * ggiBB_0, TenseurHH* ggiHH_0, TenseurBB * ggiBB_t, TenseurHH* ggiHH_t
189     , TenseurBB * ggiBB_tdt, TenseurHH * ggiHH_tdt
190     , TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVmoyBB_tdt, TenseurBB * ggradVBB_tdt
191     , double* gjacobien_tdt, double* gjacobien_t, double* gjacobien_0)
192     { giB_0=ggiB_0; giH_0=ggiH_0; giB_t=ggiB_t; giH_t=ggiH_t; giB_tdt=ggiB_tdt; giH_tdt=ggiH_tdt;
193     giBB_0=ggiBB_0; giHH_0=ggiHH_0; giBB_t=ggiBB_t; giHH_t=ggiHH_t; giBB_tdt=ggiBB_tdt; giHH_tdt=ggiHH_tdt;
194     gradVmoyBB_t=ggradVmoyBB_t; gradVmoyBB_tdt=ggradVmoyBB_tdt; gradVBB_tdt=ggradVBB_tdt;
195     jacobien_tdt=gjacobien_tdt; jacobien_t=gjacobien_t; jacobien_0=gjacobien_0;
196     };
197
198     // méthode permettant un passage des grandeurs de l'ordre 2 à l'ordre 3, this doit donc avoir
des tenseur d'ordre 3
199     // il y a recopie des valeurs et bool plusZero, indique s'il faut ou non compléter avec des 0
200     // ---->> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur générée dans
le cas de mise_au_point
201     // ---->> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 2, ils doivent également être

```

```

non nul au niveau de l'ordre 3, sinon erreur
197 // on se réfère donc aux grandeurs de l'argument : ex
198 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
199 // = 1 -> on transfère les grandeurs à t et tdt
200 // = 2 -> on transfère les grandeurs à tdt
201 void Passage_de_Ordre2_vers_Ordre3(const Umat_cont& ex,bool plusZero,int
type_recopie);
202 // méthode permettant un passage des grandeurs de l'ordre 3 à l'ordre 2, this doit
donc avoir des tenseur d'ordre 2
203 // il y a recopie des valeurs: contrairement à la méthode
Passage_de_Ordre2_vers_Ordre3, comme la cible est plus petite que l'origine
204 // on n'a pas besoin de compléter avec des 0
205 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur
générée dans le cas de mise_au_point
206 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 2, ils doivent
également être non nul au niveau de l'ordre 3, sinon erreur
207 // c'est donc les éléments de this, qui servent de modèle,
contrairement à la méthode duale:Passage_de_Ordre2_vers_Ordre3
208 // pour laquelle c'était les éléments de ex qui servaient de
modèle
209 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
210 // = 1 -> on transfère les grandeurs à t et tdt
211 // = 2 -> on transfère les grandeurs à tdt
212 void Passage_de_Ordre3_vers_Ordre2(const Umat_cont& ex,int type_recopie);
213
214 // méthode permettant un passage des grandeurs de l'ordre 1 à l'ordre 3, this doit donc avoir
des tenseur d'ordre 3
215 // il y a recopie des valeurs et bool plusZero, indique s'il faut ou non compléter avec des 0
216 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur générée dans
le cas de mise_au_point
217 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent également être
non nul au niveau de l'ordre 3, sinon erreur
218 // on se réfère donc aux grandeurs de l'argument : ex
219 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
220 // = 1 -> on transfère les grandeurs à t et tdt
221 // = 2 -> on transfère les grandeurs à tdt
222 void Passage_de_Ordre1_vers_Ordre3(const Umat_cont& ex,bool plusZero,int
type_recopie);
223 // méthode permettant un passage des grandeurs de l'ordre 3 à l'ordre 1, this doit
donc avoir des tenseur d'ordre 1
224 // il y a recopie des valeurs: contrairement à la méthode
Passage_de_Ordre1_vers_Ordre3, comme la cible est plus petite que l'origine
225 // on n'a pas besoin de compléter avec des 0
226 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur
générée dans le cas de mise_au_point
227 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent
également être non nul au niveau de l'ordre 3, sinon erreur
228 // c'est donc les éléments de this, qui servent de modèle,
contrairement à la méthode duale:Passage_de_Ordre1_vers_Ordre3
229 // pour laquelle c'était les éléments de ex qui servaient de
modèle
230 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
231 // = 1 -> on transfère les grandeurs à t et tdt
232 // = 2 -> on transfère les grandeurs à tdt
233 void Passage_de_Ordre3_vers_Ordre1(const Umat_cont& ex,int type_recopie);
234
235 // méthode permettant un passage des grandeurs de l'ordre 1 à l'ordre 2, this doit donc avoir
des tenseur d'ordre 2
236 // il y a recopie des valeurs et bool plusZero, indique s'il faut ou non compléter avec des 0
237 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur générée dans
le cas de mise_au_point
238 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent également être
non nul au niveau de l'ordre 2, sinon erreur
239 // on se réfère donc aux grandeurs de l'argument : ex
240 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
241 // = 1 -> on transfère les grandeurs à t et tdt
242 // = 2 -> on transfère les grandeurs à tdt
243 void Passage_de_Ordre1_vers_Ordre2(const Umat_cont& ex,bool plusZero,int
type_recopie);
244 // méthode permettant un passage des grandeurs de l'ordre 2 à l'ordre 1, this doit
donc avoir des tenseur d'ordre 1
245 // il y a recopie des valeurs: contrairement à la méthode
Passage_de_Ordre1_vers_Ordre2, comme la cible est plus petite que l'origine
246 // on n'a pas besoin de compléter avec des 0
247 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur
générée dans le cas de mise_au_point
248 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent
également être non nul au niveau de l'ordre 2, sinon erreur
249 // c'est donc les éléments de this, qui servent de modèle,
contrairement à la méthode duale:Passage_de_Ordre1_vers_Ordre2
250 // pour laquelle c'était les éléments de ex qui servaient de
modèle
251 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
252 // = 1 -> on transfère les grandeurs à t et tdt
253 // = 2 -> on transfère les grandeurs à tdt
254 void Passage_de_Ordre2_vers_Ordre1(const Umat_cont& ex,int type_recopie);
255

```

```

256 // ----- variables -----
257 BaseB * giB_0; BaseH * giH_0; BaseB * giB_t; BaseH * giH_t; BaseB * giB_tdt; BaseH * giH_tdt;
258 TenseurBB * gijBB_0; TenseurHH * gijHH_0; TenseurBB * gijBB_t; TenseurHH * gijHH_t;
259 TenseurBB * gijBB_tdt; TenseurHH * gijHH_tdt;
260 TenseurBB * gradVmoyBB_t; TenseurBB * gradVmoyBB_tdt; TenseurBB * gradVBB_tdt;
261 double* jacobien_tdt; double* jacobien_t; double* jacobien_0;
262 };
263 /// @} // end of group
264
265 // --- fin CLASSE CONTENEUR : cas d'une procédure Umat
266
267
268 /// @addtogroup groupe_des_metrique
269 /// @{
270 ///
271
272 // CLASSE CONTENEUR : cas explicite entre 0 et tdt, les grandeurs sont a 0 ou t et tdt
273 // les variables sont publiques
274 class Expli_t_tdt
275 { public :
276     Expli_t_tdt () : // constructeur par défaut
277         giB_0(NULL), giH_0(NULL), giB_t(NULL), giH_t(NULL), giB_tdt(NULL), giH_tdt(NULL)
278         , gijBB_0(NULL), gijHH_0(NULL), gijBB_t(NULL), gijHH_t(NULL), gijBB_tdt(NULL), gijHH_tdt(NULL)
279         , gradVmoyBB_t(NULL), gradVmoyBB_tdt(NULL), gradVBB_tdt(NULL), d_gijBB_tdt(NULL)
280         , jacobien_tdt(NULL), jacobien_t(NULL), jacobien_0(NULL)
281     {};
282     Expli_t_tdt // constructeur normal
283     (BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t, BaseB * ggiB_tdt, BaseH * ggiH_tdt
284     , TenseurBB * ggiBB_0, TenseurHH * ggiHH_0, TenseurBB * ggiBB_t, TenseurHH * ggiHH_t
285     , TenseurBB * ggiBB_tdt, TenseurHH * ggiHH_tdt
286     , TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVmoyBB_tdt, TenseurBB * ggradVBB_tdt
287     , Tableau <TenseurBB *>* gd_gijBB_tdt, double* gjacobien_tdt, double* gjacobien_t, double*
288     gjacobien_0) :
289     giB_0(ggiB_0), giH_0(ggiH_0), giB_t(ggiB_t), giH_t(ggiH_t), giB_tdt(ggiB_tdt), giH_tdt(ggiH_tdt)
290     , gijBB_0(ggiBB_0), gijHH_0(ggiHH_0), gijBB_t(ggiBB_t), gijHH_t(ggiHH_t)
291     , gijBB_tdt(ggiBB_tdt), gijHH_tdt(ggiHH_tdt)
292     , gradVmoyBB_t(ggradVmoyBB_t), gradVmoyBB_tdt(ggradVmoyBB_tdt), gradVBB_tdt(ggradVBB_tdt)
293     , d_gijBB_tdt(gd_gijBB_tdt)
294     , jacobien_tdt(gjacobien_tdt), jacobien_t(gjacobien_t), jacobien_0(gjacobien_0)
295     {};
296     Expli_t_tdt (const Expli_t_tdt& ex) : // constructeur de copie
297
298     giB_0(ex.giB_0), giH_0(ex.giH_0), giB_t(ex.giB_t), giH_t(ex.giH_t), giB_tdt(ex.giB_tdt), giH_tdt(ex.giH_tdt)
299     , gijBB_0(ex.gijBB_0), gijHH_0(ex.gijHH_0), gijBB_t(ex.gijBB_t), gijHH_t(ex.gijHH_t)
300     , gijBB_tdt(ex.gijBB_tdt), gijHH_tdt(ex.gijHH_tdt)
301     , gradVmoyBB_t(ex.gradVmoyBB_t), gradVmoyBB_tdt(ex.gradVmoyBB_tdt), gradVBB_tdt(ex.gradVBB_tdt)
302     , d_gijBB_tdt(ex.d_gijBB_tdt)
303     , jacobien_tdt(ex.jacobien_tdt), jacobien_t(ex.jacobien_t), jacobien_0(ex.jacobien_0)
304     {};
305     Expli_t_tdt& operator= (const Expli_t_tdt& ex) // surcharge d'affectation
306
307     {giB_0=ex.giB_0; giH_0=ex.giH_0; giB_t=ex.giB_t; giH_t=ex.giH_t; giB_tdt=ex.giB_tdt; giH_tdt=ex.giH_tdt;
308     gijBB_0=ex.gijBB_0; gijHH_0=ex.gijHH_0; gijBB_t=ex.gijBB_t; gijHH_t=ex.gijHH_t;
309     gijBB_tdt=ex.gijBB_tdt; gijHH_tdt=ex.gijHH_tdt;
310     gradVmoyBB_t=ex.gradVmoyBB_t; gradVmoyBB_tdt=ex.gradVmoyBB_tdt; gradVBB_tdt=ex.gradVBB_tdt;
311     d_gijBB_tdt=ex.d_gijBB_tdt;
312     jacobien_tdt=ex.jacobien_tdt; jacobien_t=ex.jacobien_t; jacobien_0=ex.jacobien_0;
313     return *this;};
314
315     // une fonction qui permet de produire un objet expli mais en mettant les grandeurs
316     // à tdt de l'objet expli_t_tdt à la place des grandeurs à t de l'objet expli
317     Expli Tdt_dans_t() const
318     { return
319     Expli(giB_0, giH_0, giB_tdt, giH_tdt, gijBB_0, gijHH_0, gijBB_tdt, gijHH_tdt, gradVmoyBB_tdt
320     , gradVBB_tdt, d_gijBB_tdt, jacobien_tdt, jacobien_0);
321     };
322     // mise à jour des grandeurs
323     void Mise_a_jour_grandeur(BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t
324     , BaseB * ggiB_tdt, BaseH * ggiH_tdt
325     , TenseurBB * ggiBB_0, TenseurHH * ggiHH_0, TenseurBB * ggiBB_t, TenseurHH * ggiHH_t
326     , TenseurBB * ggiBB_tdt, TenseurHH * ggiHH_tdt
327     , TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVmoyBB_tdt, TenseurBB * ggradVBB_tdt
328     , Tableau <TenseurBB *>* gd_gijBB_tdt, double* gjacobien_tdt, double* gjacobien_t, double*
329     gjacobien_0)
330     {giB_0=ggiB_0; giH_0=ggiH_0; giB_t=ggiB_t; giH_t=ggiH_t; giB_tdt=ggiB_tdt; giH_tdt=ggiH_tdt;
331     gijBB_0=ggiBB_0; gijHH_0=ggiHH_0; gijBB_t=ggiBB_t; gijHH_t=ggiHH_t; gijBB_tdt=ggiBB_tdt; gijHH_tdt=ggiHH_tdt;
332     gradVmoyBB_t=ggradVmoyBB_t; gradVmoyBB_tdt=ggradVmoyBB_tdt; gradVBB_tdt=ggradVBB_tdt;
333     d_gijBB_tdt=gd_gijBB_tdt;
334     jacobien_tdt=gjacobien_tdt; jacobien_t=gjacobien_t; jacobien_0=gjacobien_0;
335     };
336     // mise à jour des grandeurs à 0 et à t sauf les variations (pas de tableau et pas à tdt)
337     // à partir de grandeurs stockées par ailleurs (il s'agit ici donc d'affectation de contenu
338     et
339     // non de pointeur !)
340     void Recup_grandeur_0_t(BaseB & ggiB_0, BaseH & ggiH_0, BaseB & ggiB_t, BaseH & ggiH_t
341     , TenseurBB& ggiBB_0, TenseurHH& ggiHH_0, TenseurBB& ggiBB_t, TenseurHH& ggiHH_t

```



```

336         ,TenseurBB & ,double& gjacobien_t,double& gjacobien_0)
337     {*giB_0=ggiB_0;*giH_0=ggiH_0;*giB_t=ggiB_t;*giH_t=ggiH_t;
338     *gijBB_0=gijBB_0;*gijHH_0=gijHH_0;*gijBB_t=gijBB_t;*gijHH_t=gijHH_t;
339     *jacobien_t=gjacobien_t;*jacobien_0=gjacobien_0;/**gradVmoyBB_t=ggradVmoyBB_t;
340     };
341     // méthode permettant un passage des grandeurs de l'ordre 2 à l'ordre 3, this doit donc avoir
des tenseur d'ordre 3
342     // il y a recopie des valeurs et bool plusZero, indique s'il faut ou non compléter avec des 0
343     // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur générée dans
le cas de mise_au_point
344     // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 2, ils doivent également être
non nul au niveau de l'ordre 3, sinon erreur
345     // on se réfère donc aux grandeurs de l'argument : ex
346     // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
347     // = 1 -> on transfère les grandeurs à t et tdt
348     // = 2 -> on transfère les grandeurs à tdt
349     void Passage_de_Ordre2_vers_Ordre3(const Expli_t_tdt& ex,bool plusZero, int
type_recopie);
350     // méthode permettant un passage des grandeurs de l'ordre 3 à l'ordre 2, this doit
donc avoir des tenseur d'ordre 2
351     // il y a recopie des valeurs: contrairement à la méthode
Passage_de_Ordre2_vers_Ordre3, comme la cible est plus petite que l'origine
352     // on n'a pas besoin de compléter avec des 0
353     // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur
générée dans le cas de mise_au_point
354     // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 2, ils doivent
également être non nul au niveau de l'ordre 3, sinon erreur
355     // c'est donc les éléments de this, qui servent de modèle,
contrairement à la méthode duale:Passage_de_Ordre2_vers_Ordre3
356     // pour laquelle c'était les éléments de ex qui servaient de
modèle
357     // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
358     // = 1 -> on transfère les grandeurs à t et tdt
359     // = 2 -> on transfère les grandeurs à tdt
360     void Passage_de_Ordre3_vers_Ordre2(const Expli_t_tdt& ex, int type_recopie);
361
362     // méthode permettant un passage des grandeurs de l'ordre 1 à l'ordre 3, this doit donc avoir
des tenseur d'ordre 3
363     // il y a recopie des valeurs et bool plusZero, indique s'il faut ou non compléter avec des 0
364     // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur générée dans le
cas de mise_au_point
365     // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent également être
non nul au niveau de l'ordre 3, sinon erreur
366     // on se réfère donc aux grandeurs de l'argument : ex
367     // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
368     // = 1 -> on transfère les grandeurs à t et tdt
369     // = 2 -> on transfère les grandeurs à tdt
370     void Passage_de_Ordre1_vers_Ordre3(const Expli_t_tdt& ex,bool plusZero, int
type_recopie);
371     // méthode permettant un passage des grandeurs de l'ordre 3 à l'ordre 1, this doit
donc avoir des tenseur d'ordre 1
372     // il y a recopie des valeurs: contrairement à la méthode
Passage_de_Ordre1_vers_Ordre3, comme la cible est plus petite que l'origine
373     // on n'a pas besoin de compléter avec des 0
374     // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur
générée dans le cas de mise_au_point
375     // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent
également être non nul au niveau de l'ordre 3, sinon erreur
376     // c'est donc les éléments de this, qui servent de modèle,
contrairement à la méthode duale:Passage_de_Ordre1_vers_Ordre3
377     // pour laquelle c'était les éléments de ex qui servaient de
modèle
378     // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
379     // = 1 -> on transfère les grandeurs à t et tdt
380     // = 2 -> on transfère les grandeurs à tdt
381     void Passage_de_Ordre3_vers_Ordre1(const Expli_t_tdt& ex, int type_recopie);
382
383     // récupération d'une instance Umat_cont, contenant les grandeurs associés à this
384     Umat_cont & Recup_Umat_cont(Umat_cont& ex)const
385     { ex.giB_0 = giB_0;ex.giH_0=
giH_0;ex.giB_t=giB_t;ex.giH_t=giH_t;ex.giB_tdt=giB_tdt;ex.giH_tdt=giH_tdt;
386     ex.gijBB_0=gijBB_0; ex.gijHH_0=gijHH_0; ex.gijBB_t=gijBB_t;ex.gijHH_t=gijHH_t;
387     ex.gijBB_tdt=gijBB_tdt;ex.gijHH_tdt=gijHH_tdt;
388     ex.gradVmoyBB_t=gradVmoyBB_t;ex.gradVmoyBB_tdt=gradVmoyBB_tdt;ex.gradVBB_tdt=gradVBB_tdt;
389     ex.jacobien_tdt=jacobien_tdt;ex.jacobien_t=jacobien_t;ex.jacobien_0=jacobien_0;
390     return ex;
391     };
392
393     // ----- variables
394     BaseB * giB_0; BaseH * giH_0; BaseB * giB_t;BaseH * giH_t;BaseB * giB_tdt;BaseH * giH_tdt;
395     TenseurBB * gjjBB_0; TenseurHH * gjjHH_0; TenseurBB * gjjBB_t;TenseurHH * gjjHH_t;
396     TenseurBB * gjjBB_tdt;TenseurHH * gjjHH_tdt;
397     TenseurBB * gradVmoyBB_t;TenseurBB * gradVmoyBB_tdt;TenseurBB * gradVBB_tdt;
398     Tableau <TenseurBB *>* d_gijBB_tdt;
399     double* jacobien_tdt;double* jacobien_t;double* jacobien_0;
400     };
401 /// @} // end of group

```



```

402
403
404
405 /// @addtogroup groupe_des_metrique
406 /// @{
407 ///
408
409 // CLASSE CONTENEUR : cas implicite -----
410 // calcul des termes de la classe impli
411 class Impli
412 { public :
413     Impli () : // constructeur par défaut
414         giB_0(NULL), giH_0(NULL), giB_t(NULL), giH_t(NULL), giB_tdt(NULL), d_giB_tdt(NULL), giH_tdt(NULL)
415
416         , d_giH_tdt(NULL), giJBB_0(NULL), giJHH_0(NULL), giJBB_t(NULL), giJHH_t(NULL), giJBB_tdt(NULL), giJHH_tdt(NULL)
417
418         , gradVmoyBB_t(NULL), gradVmoyBB_tdt(NULL), gradVBB_tdt(NULL), d_giJBB_tdt(NULL), d2_giJBB_tdt(NULL)
419         , d_giJHH_tdt(NULL),
420         jacobien_tdt(NULL), jacobien_t(NULL), jacobien_0(NULL), d_jacobien_tdt(NULL)
421         , d_gradVmoyBB_t(NULL), d_gradVmoyBB_tdt(NULL), d_gradVBB_t(NULL), d_gradVBB_tdt(NULL)
422         {};
423     Impli // constructeur normal
424         (BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t, BaseB * ggiB_tdt, Tableau <BaseB>
425         * d_ggiB_tdt
426         , BaseH * ggiH_tdt, Tableau <BaseH> * d_ggiH_tdt
427         , TenseurBB* ggiJBB_0, TenseurHH* ggiJHH_0, TenseurBB* ggiJBB_t
428         , TenseurHH* ggiJHH_t, TenseurBB* ggiJBB_tdt, TenseurHH* ggiJHH_tdt
429         , TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVmoyBB_tdt, TenseurBB * ggradVBB_tdt
430         , Tableau <TenseurBB *>* gd_giJBB_tdt, Tableau2 <TenseurBB *> * gd2_giJBB_tdt
431         , Tableau <TenseurHH *>* gd_giJHH_tdt
432         , double* gjacobien, double* gjacobien_t, double* gjacobien_0, Vecteur* gd_jacobien_tdt
433         , Tableau <TenseurBB *>* gd_gradVmoyBB_t, Tableau <TenseurBB *>* gd_gradVmoyBB_tdt
434         , Tableau <TenseurBB *>* gd_gradVBB_t, Tableau <TenseurBB *>* gd_gradVBB_tdt
435         ) :
436
437         giB_0(ggiB_0), giH_0(ggiH_0), giB_t(ggiB_t), giH_t(ggiH_t), giB_tdt(ggiB_tdt), d_giB_tdt(d_ggiB_tdt),
438         giH_tdt(ggiH_tdt), d_giH_tdt(d_ggiH_tdt),
439         giJBB_0(ggiJBB_0), giJHH_0(ggiJHH_0)
440         , giJBB_t(ggiJBB_t), giJHH_t(ggiJHH_t), giJBB_tdt(ggiJBB_tdt), giJHH_tdt(ggiJHH_tdt)
441         , gradVmoyBB_t(ggradVmoyBB_t), gradVmoyBB_tdt(ggradVmoyBB_tdt), gradVBB_tdt(ggradVBB_tdt),
442         d_giJBB_tdt(gd_giJBB_tdt), d2_giJBB_tdt(gd2_giJBB_tdt), d_giJHH_tdt(gd_giJHH_tdt),
443
444         jacobien_tdt(gjacobien), jacobien_t(gjacobien_t), jacobien_0(gjacobien_0), d_jacobien_tdt(gd_jacobien_tdt)
445         , d_gradVmoyBB_t(d_gradVmoyBB_t), d_gradVmoyBB_tdt(d_gradVmoyBB_tdt)
446         , d_gradVBB_t(d_gradVBB_t), d_gradVBB_tdt(d_gradVBB_tdt)
447         {};
448     Impli (const Impli& ex) : // constructeur de copie
449
450         giB_0(ex.giB_0), giH_0(ex.giH_0), giB_t(ex.giB_t), giH_t(ex.giH_t), giB_tdt(ex.giB_tdt), d_giB_tdt(ex.d_giB_tdt)
451         , giH_tdt(ex.giH_tdt), d_giH_tdt(ex.d_giH_tdt),
452         giJBB_0(ex.giJBB_0), giJHH_0(ex.giJHH_0), giJBB_t(ex.giJBB_t), giJHH_t(ex.giJHH_t)
453         , giJBB_tdt(ex.giJBB_tdt), giJHH_tdt(ex.giJHH_tdt)
454
455         , gradVmoyBB_t(ex.gradVmoyBB_t), gradVmoyBB_tdt(ex.gradVmoyBB_tdt), gradVBB_tdt(ex.gradVBB_tdt),
456         d_giJBB_tdt(ex.d_giJBB_tdt), d2_giJBB_tdt(ex.d2_giJBB_tdt), d_giJHH_tdt(ex.d_giJHH_tdt),
457
458         jacobien_tdt(ex.jacobien_tdt), jacobien_t(ex.jacobien_t), jacobien_0(ex.jacobien_0), d_jacobien_tdt(ex.d_jacobien_tdt)
459         , d_gradVmoyBB_t(ex.d_gradVmoyBB_t), d_gradVmoyBB_tdt(ex.d_gradVmoyBB_tdt)
460         , d_gradVBB_t(ex.d_gradVBB_t), d_gradVBB_tdt(ex.d_gradVBB_tdt)
461         {};
462     Impli& operator= (const Impli& ex) // surcharge d'affectation
463
464     {giB_0=ex.giB_0; giH_0=ex.giH_0; giB_t=ex.giB_t; giH_t=ex.giH_t; giB_tdt=ex.giB_tdt; d_giB_tdt=ex.d_giB_tdt;
465     giH_tdt=ex.giH_tdt; d_giH_tdt=ex.d_giH_tdt;
466     giJBB_0=ex.giJBB_0; giJHH_0=ex.giJHH_0; giJBB_t=ex.giJBB_t; giJHH_t=ex.giJHH_t;
467     giJBB_tdt=ex.giJBB_tdt; giJHH_tdt=ex.giJHH_tdt;
468
469     gradVmoyBB_t=ex.gradVmoyBB_t; gradVmoyBB_tdt=ex.gradVmoyBB_tdt; gradVBB_tdt=ex.gradVBB_tdt;
470     d_giJBB_tdt=ex.d_giJBB_tdt; d2_giJBB_tdt=ex.d2_giJBB_tdt; d_giJHH_tdt=ex.d_giJHH_tdt;
471     jacobien_tdt=ex.jacobien_tdt; jacobien_t=ex.jacobien_t; jacobien_0=ex.jacobien_0;
472     d_jacobien_tdt=ex.d_jacobien_tdt;
473     d_gradVmoyBB_t=ex.d_gradVmoyBB_t; d_gradVmoyBB_tdt=ex.d_gradVmoyBB_tdt;
474     d_gradVBB_t=ex.d_gradVBB_t; d_gradVBB_tdt=ex.d_gradVBB_tdt;
475     return *this;
476     };
477     // mise à jour des grandeurs
478     void Mise_a_jour_grandeur
479         (BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t
480         , BaseB * ggiB_tdt, Tableau <BaseB> * d_ggiB_tdt
481         , BaseH * ggiH_tdt, Tableau <BaseH> * d_ggiH_tdt
482         , TenseurBB* ggiJBB_0, TenseurHH* ggiJHH_0, TenseurBB* ggiJBB_t, TenseurHH* ggiJHH_t
483         , TenseurBB* ggiJBB_tdt, TenseurHH* ggiJHH_tdt
484         , TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVmoyBB_tdt, TenseurBB * ggradVBB_tdt
485         , Tableau <TenseurBB *>* gd_giJBB_tdt, Tableau2 <TenseurBB *> * gd2_giJBB_tdt
486         , Tableau <TenseurHH *>* gd_giJHH_tdt
487         , double* gjacobien, double* gjacobien_t, double* gjacobien_0, Vecteur*

```

```

gd_jacobien_tdt
478     ,Tableau <TenseurBB * >* gd_gradVmoyBB_t,Tableau <TenseurBB * >*
gd_gradVmoyBB_tdt
479     ,Tableau <TenseurBB * >* gd_gradVBB_t,Tableau <TenseurBB * >*
gd_gradVBB_tdt
480     )
481
482 {giB_0=ggiB_0;giH_0=ggiH_0;giB_t=ggiB_t;giH_t=ggiH_t;giB_tdt=ggiB_tdt;d_giB_tdt=d_ggiB_tdt;
483     giH_tdt=ggiH_tdt;d_giH_tdt=d_ggiH_tdt;
484     gijBB_0=ggijBB_0;gijHH_0=ggijHH_0;gijBB_t=ggijBB_t;gijHH_t=ggijHH_t;
485     gijBB_tdt=ggijBB_tdt;gijHH_tdt=ggijHH_tdt;
486     gradVmoyBB_t=ggradVmoyBB_t;gradVmoyBB_tdt=ggradVmoyBB_tdt;gradVBB_tdt=ggradVBB_tdt;
487     d_gijBB_tdt=gd_gijBB_tdt;d2_gijBB_tdt=gd2_gijBB_tdt;d_gijHH_tdt=gd_gijHH_tdt;
488     jacobien_tdt=gjacobien;jacobien_t=gjacobien_t;jacobien_0=gjacobien_0;
489     d_gradVmoyBB_t=gd_gradVmoyBB_t;d_gradVmoyBB_tdt=gd_gradVmoyBB_tdt;
490     d_gradVBB_t=gd_gradVBB_t;d_gradVBB_tdt=gd_gradVBB_tdt;
491     };
492 // mise à jour des grandeurs à 0 et à t sauf les variations (pas de tableau et pas à tdt)
493 // à partir de grandeurs stockées par ailleurs (il s'agit ici donc d'affectation de contenu
et
494 // non de pointeur !)
495 void Recup_grandeur_0_t(BaseB & ggiB_0,BaseH & ggiH_0,BaseB & ggiB_t,BaseH & ggiH_t
496     ,TenseurBB& ggijBB_0,TenseurHH& ggijHH_0,TenseurBB& ggijBB_t,TenseurHH&
ggijHH_t
497     ,TenseurBB & ,double& gjacobien_t,double& gjacobien_0
498     )
499     { *giB_0=ggiB_0;*giH_0=ggiH_0;*giB_t=ggiB_t;*giH_t=ggiH_t;
500     *gijBB_0=ggijBB_0;*gijHH_0=ggijHH_0;*gijBB_t=ggijBB_t;*gijHH_t=ggijHH_t;
501     *jacobien_t=gjacobien_t;*jacobien_0=gjacobien_0;/**gradVmoyBB_t=ggradVmoyBB_t;
502     };
503 // méthode permettant un passage des grandeurs de l'ordre 2 à l'ordre 3, this doit donc avoir
des tenseur d'ordre 3
504 // il y a recopie des valeurs et bool plusZero, indique s'il faut ou non compléter avec des 0
505 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur générée dans
le cas de mise_au_point
506 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 2, ils doivent également être
non nul au niveau de l'ordre 3, sinon erreur
507 // on se réfère donc aux grandeurs de l'argument : ex
508 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
509 // = 1 -> on transfère les grandeurs à t et tdt
510 // = 2 -> on transfère les grandeurs à tdt
511 void Passage_de_Ordre2_vers_Ordre3(const Impli& ex,bool plusZero, int
type_recopie);
512 // méthode permettant un passage des grandeurs de l'ordre 3 à l'ordre 2, this doit
donc avoir des tenseur d'ordre 2
513 // il y a recopie des valeurs: contrairement à la méthode
Passage_de_Ordre2_vers_Ordre3, comme la cible est plus petite que l'origine
514 // on n'a pas besoin de compléter avec des 0
515 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur
générée dans le cas de mise_au_point
516 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 2, ils doivent
également être non nul au niveau de l'ordre 3, sinon erreur
517 // c'est donc les éléments de this, qui servent de modèle,
contrairement à la méthode duale:Passage_de_Ordre2_vers_Ordre3
518 // pour laquelle c'était les éléments de ex qui servaient de
modèle
519 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
520 // = 1 -> on transfère les grandeurs à t et tdt
521 // = 2 -> on transfère les grandeurs à tdt
522 void Passage_de_Ordre3_vers_Ordre2(const Impli& ex, int type_recopie);
523
524 // méthode permettant un passage des grandeurs de l'ordre 1 à l'ordre 3, this doit donc avoir
des tenseur d'ordre 3
525 // il y a recopie des valeurs et bool plusZero, indique s'il faut ou non compléter avec des 0
526 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur générée dans
le cas de mise_au_point
527 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent également être
non nul au niveau de l'ordre 3, sinon erreur
528 // on se réfère donc aux grandeurs de l'argument : ex
529 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
530 // = 1 -> on transfère les grandeurs à t et tdt
531 // = 2 -> on transfère les grandeurs à tdt
532 void Passage_de_Ordre1_vers_Ordre3(const Impli& ex,bool plusZero, int
type_recopie);
533 // méthode permettant un passage des grandeurs de l'ordre 3 à l'ordre 1, this doit
donc avoir des tenseur d'ordre 1
534 // il y a recopie des valeurs: contrairement à la méthode
Passage_de_Ordre1_vers_Ordre3, comme la cible est plus petite que l'origine
535 // on n'a pas besoin de compléter avec des 0
536 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur
générée dans le cas de mise_au_point
537 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent
également être non nul au niveau de l'ordre 3, sinon erreur
538 // c'est donc les éléments de this, qui servent de modèle,
contrairement à la méthode duale:Passage_de_Ordre1_vers_Ordre3
539 // pour laquelle c'était les éléments de ex qui servaient de

```

```

modèle
540 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
541 //               = 1 -> on transfère les grandeurs à t et tdt
542 //               = 2 -> on transfère les grandeurs à tdt
543 //               void Passage_de_Ordre3_vers_Ordre1(const Impli& ex, int type_recopie);
544
545 // méthode permettant un passage des grandeurs de l'ordre 1 à l'ordre 2, this doit donc avoir
des tenseur d'ordre 2
546 // il y a recopie des valeurs et bool plusZero, indique s'il faut ou non compléter avec des 0
547 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur générée dans
le cas de mise_au_point
548 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent également être
non nul au niveau de l'ordre 2, sinon erreur
549 // on se réfère donc aux grandeurs de l'argument : ex
550 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
551 //               = 1 -> on transfère les grandeurs à t et tdt
552 //               = 2 -> on transfère les grandeurs à tdt
553 //               void Passage_de_Ordre1_vers_Ordre2(const Impli& ex, bool plusZero, int
type_recopie);
554 // méthode permettant un passage des grandeurs de l'ordre 2 à l'ordre 1, this doit
donc avoir des tenseur d'ordre 1
555 // il y a recopie des valeurs: contrairement à la méthode
Passage_de_Ordre1_vers_Ordre2, comme la cible est plus petite que l'origine
556 // on n'a pas besoin de compléter avec des 0
557 // ---> pour les tableaux il faut qu'ils aient la même dimension, sinon erreur
générée dans le cas de mise_au_point
558 // ---> pour pointeur, s'ils sont non-nuls au niveau de l'ordre 1, ils doivent
également être non nul au niveau de l'ordre 2, sinon erreur
559 // c'est donc les éléments de this, qui servent de modèle,
contrairement à la méthode duale:Passage_de_Ordre1_vers_Ordre2
560 // pour laquelle c'était les éléments de ex qui servaient de
modèle
561 // type recopie : = 0 -> on transfère les grandeurs à 0, t et tdt
562 //               = 1 -> on transfère les grandeurs à t et tdt
563 //               = 2 -> on transfère les grandeurs à tdt
564 //               void Passage_de_Ordre2_vers_Ordre1(const Impli& ex, int type_recopie);
565
566 // récupération d'une instance Umat_cont, contenant les grandeurs associés à this
567 Umat_cont & Recup_Umat_cont(Umat_cont& ex) const
568 { ex.giB_0 = giB_0; ex.giH_0 =
giH_0; ex.giB_t=giB_t; ex.giH_t=giH_t; ex.giB_tdt=giB_tdt; ex.giH_tdt=giH_tdt;
569 ex.gijBB_0=gijBB_0; ex.gijHH_0=gijHH_0; ex.gijBB_t=gijBB_t; ex.gijHH_t=gijHH_t;
570 ex.gijBB_tdt=gijBB_tdt; ex.gijHH_tdt=gijHH_tdt;
571 ex.gradVmoyBB_t=gradVmoyBB_t; ex.gradVmoyBB_tdt=gradVmoyBB_tdt; ex.gradVBB_tdt=gradVBB_tdt;
572 ex.jacobien_tdt=jacobien_tdt; ex.jacobien_t=jacobien_t; ex.jacobien_0=jacobien_0;
573 return ex;
574 };
575
576 // ----- variables ----- :
577
578 BaseB * giB_0; BaseH * giH_0; BaseB * giB_t; BaseH * giH_t; BaseB * giB_tdt; Tableau <BaseB> *
d_giB_tdt;
579 BaseH * giH_tdt; Tableau <BaseH> * d_giH_tdt;
580 TenseurBB * gjjBB_0; TenseurHH * gjjHH_0;
581 TenseurBB * gjjBB_t; TenseurHH * gjjHH_t; TenseurBB * gjjBB_tdt; TenseurHH * gjjHH_tdt;
582 TenseurBB * gradVmoyBB_t; TenseurBB * gradVmoyBB_tdt; TenseurBB * gradVBB_tdt; Tableau <TenseurBB * >
* d_gijBB_tdt;
583 Tableau2 <TenseurBB * > * d2_gijBB_tdt;
584 Tableau <TenseurHH * > * d_gijHH_tdt;
585 double* jacobien_tdt; double* jacobien_t; double* jacobien_0;
586 Vecteur* d_jacobien_tdt;
587 Tableau <TenseurBB * > * d_gradVmoyBB_t; Tableau <TenseurBB * > * d_gradVmoyBB_tdt;
588 Tableau <TenseurBB * > * d_gradVBB_t; Tableau <TenseurBB * > * d_gradVBB_tdt;
589 };
590 /// @} // end of group
591
592
593
594 /// @addtogroup groupe_des_métrique
595 /// @{
596 ///
597
598 // classe pour le calcul de gjjHH_0 et de giH_0 juste après le calcul implicite
599 // (sinon il y a erreur)
600 class gjjHH_0_et_giH_0
601 { public:
602     gjjHH_0_et_giH_0 () : giH_0(NULL), gjjHH_0(NULL) {}; // constructeur par défaut
603     gjjHH_0_et_giH_0 // constructeur normal
604     (BaseH * ggiH_0, TenseurHH * ggijjHH_0) : giH_0(ggiH_0), gjjHH_0(ggijjHH_0) {};
605     gjjHH_0_et_giH_0 (const gjjHH_0_et_giH_0& ex) : // constructeur de copie
606     giH_0(ex.giH_0), gjjHH_0(ex.gijjHH_0) {};
607     gjjHH_0_et_giH_0& operator= (const gjjHH_0_et_giH_0& ex) // surcharge d'affectation
608     {giH_0=ex.giH_0; gjjHH_0=ex.gijjHH_0; return *this;};
609     // mise à jour des grandeurs
610     void Mise_a_jour_grandeur(BaseH * ggiH_0, TenseurHH * ggijjHH_0)
611     {giH_0=ggijjHH_0; gjjHH_0=ggijjHH_0;};
612 // ----- variables -----

```

```

613     TenseurHH * gijHH_0; BaseH * giH_0;
614 };
615 /// @} // end of group
616
617
618 /// @addtogroup groupe_des_metrique
619 /// @{
620 ///
621
622 // CLASSE CONTENEUR : cas de la dynamique
623 // calcul des termes de la classe Dynamiq
624 class Dynamiq
625 { public :
626     Dynamiq () : // constructeur par défaut
627         giB_0(NULL), giB_tdt(NULL), gijBB_0(NULL), gijBB_tdt(NULL)
628         , jacobien_tdt(NULL), jacobien_0(NULL)
629     {};
630     Dynamiq // constructeur normal
631         (BaseB * ggiB_0, BaseB * ggiB_tdt, TenseurBB * ggiBB_0, TenseurBB * ggiBB_tdt
632         , double* gjacobien_tdt, double* gjacobien_0
633         ) :
634         giB_0(ggiB_0), giB_tdt(ggiB_tdt), gijBB_0(ggiBB_0), gijBB_tdt(ggiBB_tdt)
635         , jacobien_tdt(gjacobien_tdt), jacobien_0(gjacobien_0)
636     {};
637     Dynamiq (const Dynamiq& ex) : // constructeur de copie
638         giB_0(ex.giB_0), giB_tdt(ex.giB_tdt), gijBB_0(ex.gijBB_0), gijBB_tdt(ex.gijBB_tdt)
639         , jacobien_tdt(ex.jacobien_tdt), jacobien_0(ex.jacobien_0)
640     {};
641
642     Dynamiq& operator= (const Dynamiq& ex) // surcharge d'affectation
643     { giB_0=ex.giB_0; giB_tdt=ex.giB_tdt; gijBB_0=ex.gijBB_0; gijBB_tdt=ex.gijBB_tdt;
644       jacobien_tdt=ex.jacobien_tdt; jacobien_0=ex.jacobien_0;
645       return *this;
646     };
647     // mise à jour des grandeurs
648     void Mise_a_jour_grandeur(BaseB * ggiB_0, BaseB * ggiB_tdt, TenseurBB * ggiBB_0, TenseurBB *
649     ggiBB_tdt
650     , double* gjacobien_tdt, double* gjacobien_0)
651     { giB_0=ggiB_0; giB_tdt=ggiB_tdt; gijBB_0=ggiBB_0; gijBB_tdt=ggiBB_tdt;
652       jacobien_tdt=gjacobien_tdt; jacobien_0=gjacobien_0;
653     };
654     // ----- variables -----
655     BaseB * giB_0; BaseB * giB_tdt;
656     TenseurBB * gijBB_0; TenseurBB * gijBB_tdt;
657     double* jacobien_tdt; double* jacobien_0;
658 };
659 /// @} // end of group
660
661
662 /// @addtogroup groupe_des_metrique
663 /// @{
664 ///
665
666 class flambe_lin : public Impli
667 { public :
668     flambe_lin() : // constructeur par défaut
669         Impli()
670     {};
671     flambe_lin // constructeur avec tous les paramètres
672     (BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t, BaseB *
673     ggiB_tdt, Tableau <BaseB> * d_ggiB_tdt
674     , BaseH * ggiH_tdt, Tableau <BaseH> * d_ggiH_tdt
675     , TenseurBB* ggiBB_0, TenseurHH* ggiHH_0, TenseurBB* ggiBB_t, TenseurHH* ggiHH_t
676     , TenseurBB* ggiBB_tdt, TenseurHH* ggiHH_tdt
677     , TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVmoyBB_tdt, TenseurBB * ggradVBB_tdt
678     , Tableau <TenseurBB * >* gd_gijBB_tdt, Tableau2 <TenseurBB * >* gd2_gijBB_tdt
679     , Tableau <TenseurHH * >* gd_gijHH_tdt
680     , double* gjacobien, double* gjacobien_t, double* gjacobien_0, Vecteur* gd_jacobien_tdt
681     , Tableau <TenseurBB * >* gd_gradVmoyBB_t, Tableau <TenseurBB * >* gd_gradVmoyBB_tdt
682     , Tableau <TenseurBB * >* gd_gradVBB_t, Tableau <TenseurBB * >* gd_gradVBB_tdt
683     ) :
684         Impli(ggiB_0, ggiH_0, ggiB_t, ggiH_t, ggiB_tdt, d_ggiB_tdt, ggiH_tdt, d_ggiH_tdt
685     , ggiBB_0, ggiHH_0, ggiBB_t, ggiHH_t, ggiBB_tdt, ggiHH_tdt, ggradVmoyBB_t, ggradVmoyBB_tdt, ggradVBB_tdt
686     , gd_gijBB_tdt, gd2_gijBB_tdt, gd_gijHH_tdt, gjacobien, gjacobien_t, gjacobien_0, gd_jacobien_tdt
687     , gd_gradVmoyBB_t, gd_gradVmoyBB_tdt
688     , gd_gradVBB_t, gd_gradVBB_tdt
689     )
690     {};
691     flambe_lin(const flambe_lin & ex) : // constructeur de copie
692         Impli(ex)
693     {};
694     flambe_lin& operator= (const flambe_lin& ex) // surcharge d'affectation
695     { this->Impli::operator= ((Impli&) ex);

```

```

696         return *this;
697     };
698     // mise à jour des grandeurs
699     void Mise_a_jour_grandeur
700         (BaseB * ggiB_0, BaseH * ggiH_0, BaseB * ggiB_t, BaseH * ggiH_t
701          , BaseB * ggiB_tdt, Tableau <BaseB> * d_ggiB_tdt
702          , BaseH * ggiH_tdt, Tableau <BaseH> * d_ggiH_tdt
703          , TenseurBB* ggiBB_0, TenseurHH* ggiHH_0, TenseurBB* ggiBB_t, TenseurHH*
ggiHH_t, TenseurBB* ggiBB_tdt, TenseurHH* ggiHH_tdt
704          , TenseurBB * ggradVmoyBB_t, TenseurBB * ggradVmoyBB_tdt, TenseurBB * ggradVBB_tdt
705          , Tableau <TenseurBB *>* gd_gijBB_tdt, Tableau2 <TenseurBB *> * gd2_gijBB_tdt
706          , Tableau <TenseurHH *>* gd_gijHH_tdt
707          , double* gjacobien, double* gjacobien_t, double* gjacobien_0, Vecteur*
gd_jacobien_tdt
708          , Tableau <TenseurBB * >* gd_gradVmoyBB_t, Tableau <TenseurBB * >*
gd_gradVmoyBB_tdt
709          , Tableau <TenseurBB * >* gd_gradVBB_t, Tableau <TenseurBB * >*
gd_gradVBB_tdt
710     )
711     {
712     Impli::Mise_a_jour_grandeur(ggiB_0, ggiH_0, ggiB_t, ggiH_t, ggiB_tdt, d_ggiB_tdt, ggiH_tdt, d_ggiH_tdt
713     , ggiBB_0, ggiHH_0, ggiBB_t, ggiHH_t, ggiBB_tdt, ggiHH_tdt, ggradVmoyBB_t, ggradVmoyBB_tdt, ggradVBB_tdt
714     , gd_gijBB_tdt, gd2_gijBB_tdt, gd_gijHH_tdt, gjacobien, gjacobien_t, gjacobien_0, gd_jacobien_tdt
715     , gd_gradVmoyBB_t, gd_gradVmoyBB_tdt
716     , gd_gradVBB_t, gd_gradVBB_tdt);
717     };
718     // ----- variables (idem Impli)
719     };
720     /// @} // end of group
721
722
723     /// @addtogroup groupe_des_metrique
724     /// @{
725     ///
726
727     class InfoImp
728     { public :
729         InfoImp(): // constructeur par défaut
730
731         M0(NULL), Mtdt(NULL), giB_0(NULL), giB_tdt(NULL), giH_0(NULL), giH_tdt(NULL), gijHH_tdt(NULL), gijBB_tdt(NULL)
732         , gijBB_0(NULL), gijHH_0(NULL)
733         {};
734         InfoImp // constructeur avec toutes les variables
735         (Coordonnee *gM0, Coordonnee *gMtdt, BaseB * ggiB_0, BaseB * ggiB_tdt,
736          BaseH * ggiH_0, BaseH * ggiH_tdt, TenseurHH * ggiHH_tdt, TenseurBB * ggiBB_tdt
737          , TenseurBB * ggiBB_0, TenseurHH * ggiHH_0):
738         M0(gM0), Mtdt(gMtdt), giB_0(ggiB_0), giB_tdt(ggiB_tdt), giH_0(ggiH_0), giH_tdt(ggiH_tdt)
739         , gijHH_tdt(ggiHH_tdt), gijBB_tdt(ggiBB_tdt), gijBB_0(ggiBB_0), gijHH_0(ggiHH_0)
740         {};
741         InfoImp (const InfoImp & ex): // constructeur de copie
742
743         M0(ex.M0), Mtdt(ex.Mtdt), giB_0(ex.giB_0), giB_tdt(ex.giB_tdt), giH_0(ex.giH_0), giH_tdt(ex.giH_tdt)
744         , gijHH_tdt(ex.gijHH_tdt), gijBB_tdt(ex.gijBB_tdt), gijBB_0(ex.gijBB_0), gijHH_0(ex.gijHH_0)
745         {};
746         InfoImp& operator= (const InfoImp& ex) // surcharge d'affectation
747         {
748         M0=ex.M0; Mtdt=ex.Mtdt; giB_0=ex.giB_0; giB_tdt=ex.giB_tdt; giH_0=ex.giH_0; giH_tdt=ex.giH_tdt;
749
750         gijHH_tdt=ex.gijHH_tdt; gijBB_tdt=ex.gijBB_tdt; gijBB_0=ex.gijBB_0; gijHH_0=ex.gijHH_0;
751         return *this;
752         };
753         // mise à jour des grandeurs
754         void Mise_a_jour_grandeur(Coordonnee *gM0, Coordonnee *gMtdt, BaseB * ggiB_0, BaseB * ggiB_tdt,
755          BaseH * ggiH_0, BaseH * ggiH_tdt, TenseurHH * ggiHH_tdt, TenseurBB * ggiBB_tdt
756          , TenseurBB * ggiBB_0, TenseurHH * ggiHH_0)
757         { M0=gM0; Mtdt=gMtdt; giB_0=ggiB_0; giB_tdt=ggiB_tdt; giH_0=ggiH_0; giH_tdt=ggiH_tdt;
758          gijHH_tdt=ggiHH_tdt; gijBB_tdt=ggiBB_tdt; gijBB_0=ggiBB_0; gijHH_0=ggiHH_0;
759         };
760
761         // ----- variables -----
762         Coordonnee *M0; Coordonnee *Mtdt; BaseB * giB_0; BaseB * giB_tdt;
763         BaseH * giH_0; BaseH * giH_tdt; TenseurHH * gijHH_tdt; TenseurBB * gijBB_tdt;
764         TenseurBB * gijBB_0; TenseurHH * gijHH_0;
765     };
766     /// @} // end of group
767
768
769     /// @addtogroup groupe_des_metrique
770     /// @{
771     ///
772
773     class InfoExp_t
774     { public :
775         InfoExp_t(): // constructeur par défaut

```

```

772 M0 (NULL), Mt (NULL), giB_0 (NULL), giB_t (NULL), giH_0 (NULL), giH_t (NULL), giJHH_t (NULL), giJBB_t (NULL)
773     , giJBB_0 (NULL), giJHH_0 (NULL)
774     };
775     InfoExp_t // constructeur avec toutes les variables
776     (Coordonnee *gM0, Coordonnee *gMt, BaseB * ggiB_0, BaseB * ggiB_t,
777      BaseH * ggiH_0, BaseH * ggiH_t, TenseurHH * ggiJHH_t, TenseurBB * ggiJBB_t
778      , TenseurBB * ggiJBB_0, TenseurHH * ggiJHH_0) :
779     M0 (gM0), Mt (gMt), giB_0 (ggiB_0), giB_t (ggiB_t), giH_0 (ggiH_0), giH_t (ggiH_t)
780     , giJHH_t (ggiJHH_t), giJBB_t (ggiJBB_t), giJBB_0 (ggiJBB_0), giJHH_0 (ggiJHH_0)
781     {};
782     InfoExp_t (const InfoExp_t & ex): // constructeur de copie
783     M0 (ex.M0), Mt (ex.Mt), giB_0 (ex.giB_0), giB_t (ex.giB_t), giH_0 (ex.giH_0), giH_t (ex.giH_t)
784     , giJHH_t (ex.giJHH_t), giJBB_t (ex.giJBB_t), giJBB_0 (ex.giJBB_0), giJHH_0 (ex.giJHH_0)
785     {};
786     InfoExp_t& operator= (const InfoExp_t& ex) // surcharge d'affectation
787     { M0=ex.M0; Mt=ex.Mt; giB_0=ex.giB_0; giB_t=ex.giB_t; giH_0=ex.giH_0; giH_t=ex.giH_t;
788       giJHH_t=ex.giJHH_t; giJBB_t=ex.giJBB_t; giJBB_0=ex.giJBB_0; giJHH_0=ex.giJHH_0;
789       return *this;
790     };
791     // mise à jour des grandeurs
792     void Mise_a_jour_grandeur(Coordonnee *gM0, Coordonnee *gMt, BaseB * ggiB_0, BaseB * ggiB_t,
793      BaseH * ggiH_0, BaseH * ggiH_t, TenseurHH * ggiJHH_t, TenseurBB * ggiJBB_t
794      , TenseurBB * ggiJBB_0, TenseurHH * ggiJHH_0)
795     { M0=gM0; Mt=gMt; giB_0=ggiB_0; giB_t=ggiB_t; giH_0=ggiH_0; giH_t=ggiH_t;
796       giJHH_t=ggiJHH_t; giJBB_t=ggiJBB_t; giJBB_0=ggiJBB_0; giJHH_0=ggiJHH_0;
797     };
798
799     // ----- variables -----
800     Coordonnee *M0; Coordonnee *Mt; BaseB * giB_0; BaseB * giB_t;
801     BaseH * giH_0; BaseH * giH_t; TenseurHH * giJHH_t; TenseurBB * giJBB_t;
802     TenseurBB * giJBB_0; TenseurHH * giJHH_0;
803     };
804     /// @} // end of group
805
806
807     /// @addtogroup groupe_des_metrique
808     /// @{
809     ///
810
811     class InfoExp_tdt
812     { public :
813         InfoExp_tdt(): // constructeur par défaut
814
815         M0 (NULL), Mt (NULL), giB_0 (NULL), giB_tdt (NULL), giH_0 (NULL), giH_tdt (NULL), giJHH_tdt (NULL), giJBB_tdt (NULL)
816         , giJBB_0 (NULL), giJHH_0 (NULL)
817         {};
818         InfoExp_tdt // constructeur avec toutes les variables
819         (Coordonnee *gM0, Coordonnee *gMtdt, BaseB * ggiB_0, BaseB * ggiB_tdt,
820          BaseH * ggiH_0, BaseH * ggiH_tdt, TenseurHH * ggiJHH_tdt, TenseurBB * ggiJBB_tdt
821          , TenseurBB * ggiJBB_0, TenseurHH * ggiJHH_0
822          ):
823         M0 (gM0), Mt (gMtdt), giB_0 (ggiB_0), giB_tdt (ggiB_tdt), giH_0 (ggiH_0), giH_tdt (ggiH_tdt)
824         , giJHH_tdt (ggiJHH_tdt), giJBB_tdt (ggiJBB_tdt), giJBB_0 (ggiJBB_0), giJHH_0 (ggiJHH_0)
825         {};
826         InfoExp_tdt (const InfoExp_tdt & ex): // constructeur de copie
827
828         M0 (ex.M0), Mt (ex.Mtdt), giB_0 (ex.giB_0), giB_tdt (ex.giB_tdt), giH_0 (ex.giH_0), giH_tdt (ex.giH_tdt)
829         , giJHH_tdt (ex.giJHH_tdt), giJBB_tdt (ex.giJBB_tdt), giJBB_0 (ex.giJBB_0), giJHH_0 (ex.giJHH_0)
830         {};
831         InfoExp_tdt& operator= (const InfoExp_tdt& ex) // surcharge d'affectation
832         {
833         M0=ex.M0; Mt=ex.Mtdt; giB_0=ex.giB_0; giB_tdt=ex.giB_tdt; giH_0=ex.giH_0; giH_tdt=ex.giH_tdt;
834
835         giJHH_tdt=ex.giJHH_tdt; giJBB_tdt=ex.giJBB_tdt; giJBB_0=ex.giJBB_0; giJHH_0=ex.giJHH_0;
836         return *this;
837         };
838         // mise à jour des grandeurs
839         void Mise_a_jour_grandeur(Coordonnee *gM0, Coordonnee *gMtdt, BaseB * ggiB_0, BaseB * ggiB_tdt,
840          BaseH * ggiH_0, BaseH * ggiH_tdt, TenseurHH * ggiJHH_tdt, TenseurBB * ggiJBB_tdt
841          , TenseurBB * ggiJBB_0, TenseurHH * ggiJHH_0)
842         { M0=gM0; Mt=gMtdt; giB_0=ggiB_0; giB_tdt=ggiB_tdt; giH_0=ggiH_0; giH_tdt=ggiH_tdt;
843           giJHH_tdt=ggiJHH_tdt; giJBB_tdt=ggiJBB_tdt; giJBB_0=ggiJBB_0; giJHH_0=ggiJHH_0;
844         };
845
846         // ----- variables -----
847         Coordonnee *M0; Coordonnee *Mtdt; BaseB * giB_0; BaseB * giB_tdt;
848         BaseH * giH_0; BaseH * giH_tdt; TenseurHH * giJHH_tdt; TenseurBB * giJBB_tdt;
849         TenseurBB * giJBB_0; TenseurHH * giJHH_0;
850     };
851     /// @} // end of group
852
853     /// @addtogroup groupe_des_metrique
854     /// @{
855     ///
856

```

```

854 class Info0_t_tdt
855 { public :
856     Info0_t_tdt(): // constructeur par défaut
857         M0(NULL),Mt(NULL),Mtdt(NULL),giB_0(NULL),giB_t(NULL),giB_tdt(NULL)
858         ,giH_0(NULL),giH_t(NULL),giH_tdt(NULL)
859         {};
860     Info0_t_tdt // constructeur avec toutes les variables
861         (Coordonnee *gM0,Coordonnee *gMt,Coordonnee *gMtdt,BaseB * ggiB_0,BaseB * ggiB_t,BaseB
* ggiB_tdt,
862         BaseH * ggiH_0, BaseH * ggiH_t,BaseH * ggiH_tdt):
863         M0(gM0),Mt(gMt),Mtdt(gMtdt),giB_0(ggiB_0),giB_t(ggiB_t),giB_tdt(ggiB_tdt)
864         ,giH_0(ggiH_0),giH_t(ggiH_t),giH_tdt(ggiH_tdt)
865         {};
866     Info0_t_tdt (const Info0_t_tdt & ex): // constructeur de copie
867         M0(ex.M0),Mt(ex.Mt),Mtdt(ex.Mtdt),giB_0(ex.giB_0),giB_t(ex.giB_t),giB_tdt(ex.giB_tdt)
868         ,giH_0(ex.giH_0),giH_t(ex.giH_t),giH_tdt(ex.giH_tdt)
869         {};
870     Info0_t_tdt& operator= (const Info0_t_tdt& ex) // surcharge d'affectation
871     { M0=ex.M0;Mt=ex.Mt;Mtdt=ex.Mtdt;giB_0=ex.giB_0;giB_t=ex.giB_t;giB_tdt=ex.giB_tdt;
872       giH_0=ex.giH_0;giH_t=ex.giH_t;giH_tdt=ex.giH_tdt;
873       return *this;
874     };
875     // mise à jour des grandeurs
876     void Mise_a_jour_grandeur(Coordonnee *gM0,Coordonnee *gMt,Coordonnee *gMtdt
877         ,BaseB * ggiB_0,BaseB * ggiB_t,BaseB * ggiB_tdt,
878         BaseH * ggiH_0, BaseH * ggiH_t,BaseH * ggiH_tdt)
879     { M0=gM0;Mt=gMt;Mtdt=gMtdt;giB_0=ggiB_0;giB_t=ggiB_t;giB_tdt=ggiB_tdt;
880       giH_0=ggiH_0;giH_t=ggiH_t;giH_tdt=ggiH_tdt;
881     };
882
883     // ----- variables -----
884     Coordonnee *M0; Coordonnee *Mt;Coordonnee *Mtdt;BaseB * giB_0; BaseB * giB_t;BaseB * giB_tdt;
885     BaseH * giH_0; BaseH * giH_t;BaseH * giH_tdt;
886 };
887 /// @} // end of group
888
889
890 /// @addtogroup groupe_des_metrique
891 /// @{
892 ///
893
894 // CLASSE CONTENEUR : pour le transfert d'information grandeurs à t = 0, t et tdt et
895 // toutes les métriques associées
896 // la classe Umat_cont, contient les mêmes informations qui sont utiles
897 // dans l'utilisation d'information déjà calculées
898 typedef Umat_cont Info_et_metrique_0_t_tdt;
899
900 /// @} // end of group
901
902
903
904
905

```

## 7.144 Met\_PiPoCo.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29

```

```

30 /*****
31 *   DATE:           4/06/98
32 *
33 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:        Herezh++
36 *
37 * *****/
38 *   BUT: Met_PiPoCo constitue une boîte a outil comme met_abstraite,
39 *   mais ici c'est particulièrement dédié aux poutres, plaques et
40 *   coques, qui nécessitent deux interpolations, une dans le plan ou
41 *   l'axe , et une suivant l'épaisseur.
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   -----
49 *   !           !           !
50 *   *****
51 *
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *   !           !           !
57 *   *****/
58 #ifndef MET_PIPOCO_H
59 #define MET_PIPOCO_H
60 #include "Met_abstraite.h"
61 #include "TabOper_T.h"
62
63 /// @addtogroup groupe_des_metrique
64 /// @{
65 ///
66
67
68 class Met_PiPoCo : public Met_abstraite
69 {
70 public :
71     // CONSTRUCTEUR :
72
73     // Constructeur par défaut
74     Met_PiPoCo ();
75     // constructeur permettant de dimensionner les variables
76     // dim = dimension de l'espace, nbvec = nombre de vecteur de la base naturelle
77     // tab = liste des variables a initialiser, nomb_noeud: le nombre de noeud
78     Met_PiPoCo (int dim_base,int nbvec,const DdlElement& tabddl,
79               const Tableau<Enum_variable_metrique> & tab,int nomb_noeud);
80     // constructeur de copie
81     Met_PiPoCo (const Met_PiPoCo&);
82     // DESTRUCTEUR :
83     ~Met_PiPoCo ();
84
85     // METHODES PUBLIQUES :
86
87     // Surcharge de l'operateur = : realise l'affectation
88     // dérivant de virtuel, a ne pas employer -> message d'erreur
89     Met_abstraite& operator= (const Met_abstraite& met);
90     // normale
91     virtual Met_PiPoCo& operator= (const Met_PiPoCo& met);
92
93     // ----- calculs -----
94     // cas explicite à t, toutes les grandeurs sont a 0 ou t
95     // calcul des termes : gijBB_0, gijBB_t, gijHH_t, d_gijBB_t,jacobien_t
96     // gradV_instantane : true : calcul du gradient de vitesse instantannée
97     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
98     // false: uniquement les grandeurs à t+dt sont calculées
99     const Met_abstraite::Expli& Cal_explicit_t
100     ( const Tableau<Noeud *>& tab_noeud,bool gradV_instantane,
101       Tableau<Mat_pleine const *>& tabdphi,int nombre_noeud,
102       Tableau<Vecteur const *>& tabphi,bool premier_calcul);
103     // calcul simplifie, a utiliser lorsque l'on se sert des grandeurs
104     // liers a la facette deja calculée, ainsi que la courbure
105     // gradV_instantane : true : calcul du gradient de vitesse instantannée
106     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
107     // false: uniquement les grandeurs à t+dt sont calculées
108     const Met_abstraite::Expli& Cal_explicit_simple_t
109     ( const Tableau<Noeud *>& tab_noeud,bool gradV_instantane,
110       Tableau<Mat_pleine const *>& tabdphi,int nombre_noeud,
111       Tableau<Vecteur const *>& tabphi,bool premier_calcul);
112     // cas explicite à tdt, toutes les grandeurs sont a 0 ou tdt
113     // calcul des termes : gijBB_0, gijBB_tdt, gijHH_tdt, d_gijBB_tdt,jacobien_tdt
114     // gradV_instantane : true : calcul du gradient de vitesse instantannée
115     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées

```



```

116         // false: uniquement les grandeurs à t+dt sont calculées
117     const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt
118         ( const Tableau<Noeud *>& tab_noeud, bool gradV_instantane,
119           Tableau<Mat_pleine const *>& tabdphi, int nombre_noeud,
120           Tableau<Vecteur const *>& tabphi, bool premier_calcul);
121     // calcul simplifie, a utiliser lorsque l'on se sert des grandeurs
122     // liers a la facette deja calculee, ainsi que la courbure
123     // gradV_instantane : true : calcul du gradient de vitesse instantannée
124     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
125     // false: uniquement les grandeurs à t+dt sont calculées
126     const Met_abstraite::Expli_t_tdt& Cal_explicit_simple_tdt
127         ( const Tableau<Noeud *>& tab_noeud, bool gradV_instantane,
128           Tableau<Mat_pleine const *>& tabdphi, int nombre_noeud,
129           Tableau<Vecteur const *>& tabphi, bool premier_calcul);
130     // cas implicite
131     // calcul des termes de la classe impli
132     // gradV_instantane : true : calcul du gradient de vitesse instantannée
133     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
134     // false: uniquement les grandeurs à t+dt sont calculées
135     const Met_abstraite::Impli& Cal_implicit
136         ( const Tableau<Noeud *>& tab_noeud, bool gradV_instantane,
137           Tableau<Mat_pleine const *>& tabdphi, int nombre_noeud,
138           Tableau<Vecteur const *>& tabphi, bool premier_calcul);
139     // calcul simplifie, a utiliser lorsque l'on se sert des grandeurs
140     // liers a la facette deja calculee, ainsi que la courbure et sa variation
141     // gradV_instantane : true : calcul du gradient de vitesse instantannée
142     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
143     // false: uniquement les grandeurs à t+dt sont calculées
144     const Met_abstraite::Impli& Cal_implicit_simple
145         ( const Tableau<Noeud *>& tab_noeud, bool gradV_instantane,
146           Tableau<Mat_pleine const *>& tabdphi, int nombre_noeud,
147           Tableau<Vecteur const *>& tabphi, bool premier_calcul);
148
149     // ----- remontee aux informations -----
150
151     // cas sortie d'infoImp
152     // calcul des termes de la classe InfoImp
153     const Met_abstraite::InfoImp& Cal_InfoImp( const Tableau<Noeud *>& tab_noeud,
154         Tableau<Mat_pleine const *>& tabdphi,
155         Tableau<Vecteur const *>& tabphi, int nombre_noeud);
156     // cas sortie d'infoExp
157     // calcul des termes de la classe InfoExp_t
158     const Met_abstraite::InfoExp_t& Cal_InfoExp_t( const Tableau<Noeud *>& tab_noeud,
159         Tableau<Mat_pleine const *>& tabdphi,
160         Tableau<Vecteur const *>& tabphi, int nombre_noeud);
161     // calcul des termes de la classe InfoExp_tdt
162     const Met_abstraite::InfoExp_tdt& Cal_InfoExp_tdt( const Tableau<Noeud *>& tab_noeud,
163         Tableau<Mat_pleine const *>& tabdphi,
164         Tableau<Vecteur const *>& tabphi, int nombre_noeud);
165
166     // ----- gestion de la memoire -----
167
168     // Ajout d'initialisation de differentes variables : liste dans tab
169     void PlusInitVariables(Tableau<Enum_variable_metrique>& tab) ;
170
171     // ===== protege =====
172
173     protected :
174     // METHODES protegees:
175     // calcul des normales a la facette
176     virtual void Calcul_N_0 ();
177     virtual void Calcul_N_t ();
178     virtual void Calcul_N_tdt ();
179     //==calcul des points, arguments identiques a ceux de Met_abstraite
180     // calcul du point a different temps
181     void Calcul_M0 ( const Tableau<Noeud *>& , const Vecteur& phi, int );
182     void Calcul_Mt ( const Tableau<Noeud *>& , const Vecteur& phi, int );
183     void Calcul_Mtdt ( const Tableau<Noeud *>& , const Vecteur& phi, int );
184
185     //== les fonctions de calcul de base sont donc redefini
186     // calcul de la base naturel a t0
187     // pendant le traitement il y a calcul de la base aiB et aiH de la facette plane
188     // la fonction contient un argument de plus que la routine dans met_abstraite
189     virtual void Calcul_giB_0
190     ( const Tableau<Noeud *>& , const Mat_pleine& , int, const Vecteur& phi);
191     // calcul de la base naturel a t
192     virtual void Calcul_giB_t
193     ( const Tableau<Noeud *>& , const Mat_pleine& , int, const Vecteur& phi);
194     // calcul de la base naturel a t+dt
195     virtual void Calcul_giB_tdt
196     ( const Tableau<Noeud *>& , const Mat_pleine& , int, const Vecteur& phi);
197     //== calcul de la variation des bases
198     virtual void D_giB_t( const Mat_pleine& , int , const Vecteur & phi);
199     virtual void D_giB_tdt( const Mat_pleine& , int , const Vecteur & phi);
200     //-----// calcul du tenseur de courbure dans la base naturelle
201     // plusieurs cas sont etudies suivant l'instant considere
202     // a l'instant t = 0

```

```

203 virtual Vecteur& courbure_0 ( const Tableau<Noeud *>& tab_noeud) = 0;
204 // a l'instant t
205 virtual Vecteur& courbure_t ( const Tableau<Noeud *>& tab_noeud) = 0;
206 // a l'instant t+dt
207 virtual Vecteur& courbure_tdt ( const Tableau<Noeud *>& tab_noeud) = 0;
208 //-----// calcul du tenseur de courbure et de sa variation
209 // plusieurs cas sont etudies suivant l'instant considere
210 // a l'instant t
211 virtual void Dcourbure_t
212 ( const Tableau<Noeud *>& tab_noeud,Vecteur& curb,TabOper<Vecteur>& dcurb) = 0;
213 // a l'instant tdt
214 virtual void Dcourbure_tdt
215 ( const Tableau<Noeud *>& tab_noeud,Vecteur& curb,TabOper<Vecteur>& dcurb) = 0;
216
217 // DONNEES PROTEGEES
218
219 Tableau <Coordonnee> Ia; // base absolue
220 // calcul de la courbure et de sa variation eventuelle
221 Vecteur curb_0,curb_t,curb_tdt;
222 TabOper<Vecteur> dcurb_t,dcurb_tdt;
223 // calcul de la normale
224 Coordonnee N_0,N_t,N_tdt;
225
226 //il s'agit ici des infos concernant la partie mediane ( axe ou facette)
227
228 Coordonnee * P0; // point a t=0
229 Coordonnee * Pt ; // point a l'instant t
230 Coordonnee * Ptdt; // point a l'instant t+dt BaseB Ia; // la base absolue
231 Tableau <Coordonnee> * d_Pt ; // derivees au point a l'instant t, par rapport aux ddl
232 Tableau <Coordonnee> * d_Ptdt; // derivees au point a l'instant t+dt, par rapport aux ddl
233
234 BaseB * aiB_0 ; // base naturelle a t=0
235 BaseB * aiB_t ; // base naturelle a t
236 BaseB * aiB_tdt ; // base naturelle a t+dt
237
238 BaseH * aiH_0; // vecteur de la base duale a 0
239 BaseH * aiH_t; // vecteur de la base duale a t
240 BaseH * aiH_tdt; // vecteur de la base duale a t+dt
241
242 TenseurBB * aijBB_0; // composantes de la metrique de la base naturelle a t
243 TenseurBB * aijBB_t; // composantes de la metrique de la base naturelle a t
244 TenseurBB * aijBB_tdt; // composantes de la metrique de la base naturelle a tdt
245 TenseurHH * aijHH_0; // composantes de la metrique de la base duale a t=0
246 TenseurHH * aijHH_t; // composantes de la metrique de la base duale a t
247 TenseurHH * aijHH_tdt; // composantes de la metrique de la base duale a t+dt
248
249 double ajacobien_0; // jacobien a l'instant t=0
250 double ajacobien_t; // jacobien a l'instant t
251 double ajacobien_tdt; // jacobien a l'instant t+dt
252
253 Tableau <BaseB> * d_aiB_t; // derivees de la base naturelle par rapport
254 // aux degres de liberte a t
255 Tableau <BaseB> * d_aiB_tdt; // derivees de la base naturelle par rapport
256 // aux degres de liberte a t+dt
257
258 Tableau <BaseH> * d_aiH_t; // derivees de la base duale par rapport
259 // aux degres de liberte a t
260 Tableau <BaseH> * d_aiH_tdt; // derivees de la base duale par rapport
261 // aux degres de liberte a t+dt
262
263 Tableau <TenseurBB * > d_aijBB_t; // derivees de la metrique de la base naturelle
264 // a t par rapport aux degres de liberte
265 Tableau <TenseurBB * > d_aijBB_tdt; // derivees de la metrique de la base naturelle
266 // a t+dt par rapport aux degres de liberte
267 Tableau <TenseurHH * > d_aijHH_t; // derivees de la metrique de la base duale a t
268 Tableau <TenseurHH * > d_aijHH_tdt; // derivees de la metrique de la base duale a t+dt
269
270 Vecteur d_ajacobien_t ; // derivees du jacobien par rapport aux degres de liberte
271 // par rapport aux degres de liberte a t
272 Vecteur d_ajacobien_tdt ; // derivees du jacobien par rapport aux degres de liberte
273 // par rapport aux degres de liberte a tdt
274
275 // fonctions privees
276 // allocation de la memoire
277 void AllocPiPoCo ();
278 // deallocation de la memoire complete, uniquement des grandeurs specifiques
279 // à Met_PiPoCo
280 void DeAllocPiPoCo();
281 // uniquement des grandeurs de tib
282 void DeAllocPiPoCo(Tableau<Enum_variable_metrrique>& tib);
283 // copie en fonction de l'instance passée
284 // concerne que les grandeurs pointees
285 void Copie_metPiPoCo(const Met_PiPoCo& a);
286
287
288 };
289 /// @} // end of group

```

```

290
291
292 #endif
293
294

```

## 7.145 MetAxisymetrique2D.h

```

1 // FICHER : MetAxisymetrique2D.h
2 // CLASSE : MetAxisymetrique2D
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      29/12/2004
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *
40 *   BUT:      La classe MetAxisymetrique2D derive de Met_abstraite, et
41 *            est pécifique aux éléments axisymétriques 2D.
42 *            La dimension de l'espace 2 ou 3
43 *            dim = 2: l'axe de rotation est z, et l'élément est
44 *            défini dans x y: en résumé: l'élément est 2D avec un élé-
45 *            ment géométrique ici uniquement 1D (suivant x).
46 *            dim = 3: l'axe de rotation est y, et l'élément est
47 *            défini dans x y: en résumé: l'élément est 2D avec un élé-
48 *            ment géométrique ici uniquement 1D (suivant x)
49 *
50 *            *****
51 *
52 *   VERIFICATION:
53 *
54 *   ! date !   auteur !           but
55 *   !-----!-----!-----!-----!
56 *
57 *            *****
58 *   MODIFICATIONS:
59 *
60 *   ! date !   auteur !           but
61 *   !-----!-----!-----!-----!
62 *
63 *            *****/
64
65 #ifndef MET_AXI_SYMETRIQUE2D_H
66 #define MET_AXI_SYMETRIQUE2D_H
67
68 #include "Met_abstraite.h"
69
70
71 /// @addtogroup groupe_des_metrique
72 /// @{
73 ///
74
75
76 class MetAxisymetrique2D : public Met_abstraite

```

```

77 {
78
79
80 public :
81
82
83     // CONSTRUCTEUR :
84
85     // Constructeur par défaut
86     MetAxisymetrique2D ();
87     // constructeur permettant de dimensionner uniquement certaines variables
88     // ici la dimension est 2 ou 3, le nombre de vecteur par contre est fixe: 2
89     // des variables a initialiser
90     MetAxisymetrique2D (int dim_base,const DdlElement& tabddl,
91                         const Tableau<Enum_variable_metrrique> & tab,int nomb_noeud);
92     // constructeur de copie
93     MetAxisymetrique2D (const MetAxisymetrique2D&);
94     // DESTRUCTEUR :
95
96     ~MetAxisymetrique2D ();
97 // Surcharge de l'operateur = : realise l'affectation
98 // fonction virtuelle
99 inline Met_abstraite& operator= (const Met_abstraite& met)
100     { return (Met_abstraite::operator=(met));};
101
102 protected :
103 // METHODES protegees:
104 //==calcul des points, identique a Met_abstraite
105
106 //===== Methodes protegees =====
107 protected:
108 //==calcul des points, la dimension des points est 3 ici
109 // calcul des variations du point a tdt
110 virtual void Calcul_d_Mtdt
111     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
112 // calcul des variations de la vitesse du point a t en fonction des ddl existants de vitesse
113 virtual void Calcul_d_Vt
114     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
115 // idem mais au noeud passé en paramètre
116 virtual void Calcul_d_Vt (const Noeud* noeud);
117 // calcul des variations de la vitesse du point a tdt en fonction des ddl existants de vitesse
118 virtual void Calcul_d_Vtdt
119     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
120 // idem mais au noeud passé en paramètre
121 virtual void Calcul_d_Vtdt (const Noeud* noeud);
122 // calcul des variations de la vitesse moyenne du point a t en fonction des ddl de position
123 virtual void Calcul_d_V_moyt
124     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
125 // idem mais au noeud passé en paramètre
126 virtual void Calcul_d_V_moyt (const Noeud* noeud);
127 // calcul des variations de la vitesse moyenne du point a tdt en fonction des ddl de position
128 virtual void Calcul_d_V_moytdt
129     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi, int nombre_noeud);
130 // idem mais au noeud passé en paramètre
131 virtual void Calcul_d_V_moytdt (const Noeud* noeud);
132
133 //== le nombre de vecteur de base 2 ou 3, avec le dernier vecteur suivant y ou z
134 // selon la dimension: 2 ou 3
135 //== les fonctions de calcul de base sont donc redefini
136 // calcul de la base naturel a t0
137 void Calcul_giB_0
138     ( const Tableau<Noeud *>& tab_noeud,const Mat_pleine& dphi, int nombre_noeud,const
139     Vecteur& phi);
140 // calcul de la base naturel a t
141 void Calcul_giB_t
142     ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud,const
143     Vecteur& phi);
144 // calcul de la base naturel a t+dt
145 void Calcul_giB_tdt
146     ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud,const
147     Vecteur& phi);
148 //== calcul du jacobien aux differents temps , il est necessaire d'avoir
149 //== calcule le tenseur metrique correspondant
150 // virtual void Jacobien_0();
151 // virtual void Jacobien_t();
152 // virtual void Jacobien_tdt();
153
154 //== calcul de la variation des bases
155 void D_giB_t( const Mat_pleine& tabDphi,
156             int nbnoeu,const Vecteur & phi); // avant calcul de : giB_t
157 void D_giB_tdt( const Mat_pleine& tabDphi,
158              int nbnoeu,const Vecteur & phi); // avant calcul de : giB_tdt
159
160 //== par défaut le gradient de vitesse est de type (nbvecteurdela base)au carre
161 //== nécessite d'avoir calculé les vecteurs giB avant
162 // calcul du gradient de vitesse à t

```

```

161     void Calcul_gradVBB_t
162     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
163     // calcul gradient de vitesse à t+dt
164     void Calcul_gradVBB_tdt
165     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
166     //== dans le cas où il n'y a pas de ddl de vitesse on peut utiliser les vitesses moyennes
167     //== correspondant à  $\Delta x^{\text{ar}}/\Delta t$ 
168     // calcul du gradient de vitesse moyen à t
169     // dans le cas où les ddl à tdt n'existent pas -> utilisation de la vitesse sécante entre 0 et
t !!
170     void Calcul_gradVBB_moyen_t
171     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
172     // calcul du gradient de vitesse moyen entre t et t+dt
173     void Calcul_gradVBB_moyen_tdt
174     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
175
176     //== calcul de la variation du gradient
177     // par rapport aux composantes  $V^{\text{ar}}$  (et non les  $X^{\text{ar}}$ )
178     void DgradVBB_t(const Mat_pleine& dphi); // avant calcul de : giB_t
179     // par rapport aux composantes  $V^{\text{ar}}$  (et non les  $X^{\text{ar}}$ )
180     void DgradVBB_tdt(const Mat_pleine& dphi); // avant calcul de : giB_tdt
181     // par rapport aux composantes  $X^{\text{ar}}$  (et non les  $V^{\text{ar}}$ )
182     void DgradVmoyBB_t(const Mat_pleine& dphi,const Tableau<Noeud *>& tab_noeud); // calcul variation
du gradient moyen à t
183     // par rapport aux composantes  $X^{\text{ar}}$  (et non les  $V^{\text{ar}}$ )
184     void DgradVmoyBB_tdt(const Mat_pleine& dphi); // calcul variation du gradient moyen à tdt
185
186
187     //----- données protégées -----
188     protected:
189     // les rayons
190     double rho_0,rho_t,rho_tdt;
191
192
193 };
194 /// @} // end of group
195
196
197 #endif
198
199

```

## 7.146 MetAxisymetrique3D.h

```

1 // FICHER : MetAxisymetrique3D.h
2 // CLASSE : MetAxisymetrique3D
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *    DATE:      29/12/2004
34 *
35 *    AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
36 *
37 *    PROJET:    Herezh++
38 *
39 *
40 *    BUT:      La classe MetAxisymetrique3D derive de Met_abstraite, et
41 *             est spécifique aux éléments axisymétriques.

```

```

42 *          La dimension de l'espace 3 *
43 *          dim = 3: l'axe de rotation est y, et l'élément est *
44 *          défini dans x y: en résumé: l'élément est 3D avec un élé- *
45 *          ment géométrique 2D. *
46 *          $ *
47 *          ***** *
48 *          *
49 *          VERIFICATION: *
50 *          ! date ! auteur ! but ! *
51 *          ----- *
52 *          ! ! ! ! *
53 *          $ *
54 *          ***** *
55 *          MODIFICATIONS: *
56 *          ! date ! auteur ! but ! *
57 *          ----- *
58 *          $ *
59 *****/
60
61
62 #ifndef MET_AXI_SYMETRIQUE3D_H
63 #define MET_AXI_SYMETRIQUE3D_H
64
65 #include "Met_abstraite.h"
66
67
68 /// @addtogroup groupe_des_metrique
69 /// @{
70 ///
71
72
73 class MetAxisymetrique3D : public Met_abstraite
74 {
75
76
77     public :
78
79
80         // CONSTRUCTEUR :
81
82         // Constructeur par défaut
83         MetAxisymetrique3D ();
84         // constructeur permettant de dimensionner unique ment certaine variables
85         // ici la dimension 3,
86         // des variables a initialiser
87         MetAxisymetrique3D (const DdlElement& tabddl,
88                             const Tableau<Enum_variable_metrique> & tab,int nomb_noeud);
89         // constructeur de copie
90         MetAxisymetrique3D (const MetAxisymetrique3D&);
91         // DESTRUCTEUR :
92
93         ~MetAxisymetrique3D ();
94         // Surcharge de l'operateur = : realise l'affectation
95         // fonction virtuelle
96         inline Met_abstraite& operator= (const Met_abstraite& met)
97         { return (Met_abstraite::operator=(met)); };
98
99     protected :
100         // METHODES protegees:
101         //==calcul des points, identique a Met_abstraite
102
103         //===== Methodes protegees =====
104     protected:
105         //==calcul des points, la dimension des points est 3 ici
106
107         //== le nombre de vecteur de base 3, le troisième est suivant k,
108         // les deux autres vecteurs de base sont calculées avec la surface de l'élément axisymétrique
109         //== les fonctions de calcul de base sont donc redefini
110         // calcul de la base naturel a t0
111         void Calcul_giB_0
112         ( const Tableau<Noeud *>& tab_noeud,const Mat_pleine& dphi, int nombre_noeud,const
113           Vecteur& phi);
114         // calcul de la base naturel a t
115         void Calcul_giB_t
116         ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud,const
117           Vecteur& phi);
118         // calcul de la base naturel a t+dt
119         void Calcul_giB_tdt
120         ( const Tableau<Noeud *>& tab_noeud, const Mat_pleine& dphi, int nombre_noeud,const
121           Vecteur& phi);
122         //== calcul du jacobien aux differents temps , il est necessaire d'avoir
123         //== calcule le tenseur metrique correspondant
124         // virtual void Jacobien_0();
125         // virtual void Jacobien_t();
126         // virtual void Jacobien_tdt();

```

```

125
126 //== calcul de la variation des bases
127 void D_giB_t( const Mat_pleine& tabDphi,
128             int nbnoeu,const Vecteur & phi); // avant calcul de : giB_t
129 void D_giB_tdt( const Mat_pleine& tabDphi,
130              int nbnoeu,const Vecteur & phi); // avant calcul de : giB_tdt
131
132 //== par défaut le gradient de vitesse est de type (nbvecteurdela base)au carre
133 //== nécessite d'avoir calculé les vecteurs giB avant
134 // calcul du gradient de vitesse à t
135 void Calcul_gradVBB_t
136     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
137 // calcul gradient de vitesse à t+dt
138 void Calcul_gradVBB_tdt
139     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
140 //== dans le cas où il n'y a pas de ddl de vitesse on peut utiliser les vitesses moyennes
141 //== correspondant à delta x^ar/delta t
142 // calcul du gradient de vitesse moyen à t
143 // dans le cas où les ddl à tdt n'existent pas -> utilisation de la vitesse sécante entre 0 et
144 t !! void Calcul_gradVBB_moyen_t
145     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
146 // calcul du gradient de vitesse moyen entre t et t+dt
147 void Calcul_gradVBB_moyen_tdt
148     (const Tableau<Noeud *>& tab_noeud, const Mat_pleine& tabDphi,int nombre_noeud);
149
150 //== calcul de la variation du gradient
151 // par rapport aux composantes V^ar (et non les X^ar )
152 void DgradVBB_t( const Mat_pleine& dphi); // avant calcul de : giB_t
153 // par rapport aux composantes V^ar (et non les X^ar )
154 void DgradVBB_tdt(const Mat_pleine& dphi); // avant calcul de : giB_tdt
155 // par rapport aux composantes X^ar (et non les V^ar )
156 void DgradVmoyBB_t(const Mat_pleine& dphi,const Tableau<Noeud *>& tab_noeud); // calcul variation
157 // par rapport aux composantes X^ar (et non les V^ar )
158 void DgradVmoyBB_tdt(const Mat_pleine& dphi); // calcul variation du gradient moyen à tdt
159
160 //----- données protégées -----
161 protected:
162
163
164 };
165 /// @} // end of group
166
167
168 #endif
169
170

```

## 7.147 PiPoCo.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      4/06/98
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *

```

```

35 *   PROJET:      Herezh++                               *
36 *                                                     $ *
37 *****
38 *   BUT: Définir une classe de base pour les plaques et poutres. *
39 *   Entre autres, cela permet d'avoir des fonctions et du stockage *
40 *   générales.                                         *
41 *                                                     $ *
42 *   ***** *
43 *   VERIFICATION:                                     *
44 *   ! date !   auteur !         but                   ! *
45 *   ----- *
46 *   !         !         !         !                   ! *
47 *   !         !         !         !                   ! *
48 *   !         !         !         !                   $ *
49 *   ***** *
50 *   MODIFICATIONS:                                   *
51 *   ! date !   auteur !         but                   ! *
52 *   ----- *
53 *   !         !         !         !                   $ *
54 *****/
55 #ifndef PIPOCO_H
56 #define PIPOCO_H
57
58 #include "ElemMeca.h"
59 #include "DeformationPP.h"
60
61 class ConstrucElementbiel;
62
63 class PiPoCo : public ElemMeca
64 {
65 public :
66 // CONSTRUCTEURS :
67 // Constructeur par défaut fonction eventuellement de numeros d'identification
68 PiPoCo (int num_mail=0,int num_id=-3) : ElemMeca (num_mail,num_id) {};
69 // Constructeur : un numero de maillage et d'identification et le tableau de connexite des noeuds
70
71 PiPoCo (int num_mail,int num_id,const Tableau<Noeud *>& tab) : ElemMeca (num_mail, num_id, tab)
72 {};
73 // Constructeur : un numero de maillage, d'identification ,la geometrie ,le type d'interpolation
74 PiPoCo (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt) :
75 ElemMeca (num_mail,num_id,id_interp_elt,id_geom_elt) {};
76 // Constructeur idem si-dessus mais des chaines de caractères
77 PiPoCo (int num_mail,int num_id,char* nom_interpol,char* nom_geom) :
78 ElemMeca (num_mail,num_id,nom_interpol,nom_geom) {};
79 // Constructeur utile quand toutes les donnees de la classe Element sont connues
80 // 1) avec des identificateurs d'énumération
81 PiPoCo (int num_mail,int num_id,const Tableau<Noeud *>& tab,Enum_interpol id_interp_elt,
82 Enum_geom id_geom_elt) :
83 ElemMeca (num_mail,num_id,tab,id_interp_elt,id_geom_elt) {};
84 // 2) avec des identificateurs = chaines de caractères
85 PiPoCo (int num_mail,int num_id,const Tableau<Noeud *>& tab,char* nom_interpol,char* nom_geom) :
86 ElemMeca (num_mail,num_id,tab,nom_interpol,nom_geom) {};
87 // Constructeur de copie
88 PiPoCo (const PiPoCo& ps) : ElemMeca (ps) {};
89
90 // DESTRUCTEUR :
91
92 // ===== methodes publiques =====
93
94 // ramene vrai si la surface numéro ns existe pour l'élément
95 // dans le cas des poutres il n'y a pas de surface
96 bool SurfExiste(int ) const
97 { return false;};
98
99 // ramene vrai si l'arête numéro na existe pour l'élément
100 bool AreteExiste(int na) const {if (na==1) return true; else return false;};
101
102 // calcul si un point est a l'interieur de l'element ou non
103 // il faut que M est la dimension globale
104 // les trois fonctions sont pour l'etude a t=0, t et tdt
105 // retour : =0 le point est externe, =1 le point est interne ,
106 // = 2 le point est sur la frontière à la précision près
107 // coor_locales : s'il est différent de NULL, est affecté des coordonnées locales calculées,
108 // uniquement précises si le point est interne
109 int Interne_0(const Coordonnee& M,Coordonnee* coor_locales=NULL);
110 int Interne_t(const Coordonnee& M,Coordonnee* coor_locales=NULL);
111 int Interne_tdt(const Coordonnee& M,Coordonnee* coor_locales=NULL);
112 //1) methodes virtuelles
113 // définition du nombre maxi de point d'intégration dans l'épaisseur
114 virtual int Nb_pt_int_epai() = 0;
115 // définition du nombre maxi de point d'intégration sur la surface ou
116 // dans l'axe de la poutre
117 virtual int Nb_pt_int_surf() = 0;
118 // récupération de l'épaisseur
119 virtual double H() = 0;

```



```

119
120 // ----- affichage ou récupération d'informations -----
121 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
122 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
123 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
124 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
125 // temps: dit si c'est à 0 ou t ou tdt
126 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
127 { // méthode ici à faire, car ici spécifique au fait d'avoir des pt d'integ dans l'épaisseur
128     cout << "\n non implanté , PiPoCo::PointLePlusPres(Enum_dure...";
129     Sortie(1);
130     return PtLePlusPres(temps,enu,M);};
131
132 // recuperation des coordonnées du point de numéro d'ordre iteg pour
133 // la grandeur enu
134 // temps: dit si c'est à 0 ou t ou tdt
135 // si erreur retourne erreur à true
136 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
137 { // méthode ici à faire, car ici spécifique au fait d'avoir des pt d'integ dans l'épaisseur
138     cout << "\n non implanté , PiPoCo::CoordPtInteg(Enum_dure...";
139     Sortie(1);
140     return CoordPtInt(temps,enu,iteg,erreur);};
141
142 // récupération des valeurs au numéro d'ordre = iteg pour
143 // les grandeur enu
144 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
145 Tableau<double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
146     enu,int iteg) { // méthode ici à faire, car ici spécifique au fait d'avoir des pt
147     d'integ dans l'épaisseur
148     cout << "\n non implanté , PiPoCo::Valeur_a_diff_temps(Enum_dure...";
149     Sortie(1);
150     return ElemMeca::Valeur_multi(absolue,enu_t,enu,iteg,1); // "
151 };
152 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
153 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
154 // de conteneurs quelconque associée
155 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure ,List_io<TypeQuelconque>& ,int )
156 { // méthode ici à faire, car ici spécifique au fait d'avoir des pt d'integ dans l'épaisseur
157     cout << "\n non implanté , PiPoCo::ValTensorielle_a_diff_temps(Enum_dure...";
158     Sortie(1);
159 };
160
161 // 2) methodes découlant de virtuelles
162 // ----- calculs utils dans le cadre de la recherche du flambement linéaire
163 // Calcul de la matrice géométrique et initiale
164 ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa);
165
166 // 3) methodes propres a l'element
167 // les coordonnees des points d'integration dans l'epaisseur
168 virtual double KSIEpais(int i) =0;
169
170 // ===== methodes protégées utilisables par les classes derivees =====
171 protected :
172
173 // 4) non virtuelles
174 // Calcul du residu local et de la raideur locale, pour le schema implicite
175 // cald_Dvirtuelle = indique si l'on doit calculer la dérivée de la vitesse de déformation virtuelle
176 void Cal_implicitPiPoCo (DdlElement & tab_ddl,Tableau<TenseurBB *> & d_epsBB
177     ,Tableau<Tableau2<TenseurBB *>> & d2_epsBB,Tableau<TenseurHH *> & d_sigHH
178     ,int nbintS,Vecteur& poidsS,int nbintH,Vecteur& poidsH
179     ,const ParaAlgoControle & pa,bool cald_Dvirtuelle);
180
181 // Calcul du residu local a l'instant t ou tdt
182 // atdt = true : calcul à tdt, valeur par défaut
183 // = false: calcul à t
184 void Cal_explicitPiPoCo (DdlElement & tab_ddl,Tableau<TenseurBB *> & d_epsBB,int nbintS
185     ,Vecteur& poidsS,int nbintH,Vecteur& poidsH,const ParaAlgoControle & pa,bool atdt);
186
187 // Calcul de la matrice géométrique et de la matrice initiale
188 // cette fonction est éventuellement appelée par les classes dérivées
189 // ddl represente les degres de liberte specifiques a l'element
190 // epsBB = deformation, sigHH = contrainte, d_epsbb = variation des def
191 // nbint = nb maxi de pt d'integration , poids = poids d'integration
192 // S pour surface et H pour épaisseur -> nbint et poids
193 // cald_Dvirtuelle = indique si l'on doit calculer la dérivée de la vitesse de déformation virtuelle
194
195 void Cal_matGeom_InitPiPoCo (Mat_pleine & matGeom, Mat_pleine & matInit,DdlElement & tab_ddl
196     ,Tableau<TenseurBB *> & d_epsBB, Tableau<Tableau2<TenseurBB *>> & d2_epsBB
197     ,Tableau<TenseurHH *> & d_sigHH,int nbintS,Vecteur& poidsS,int nbintH,Vecteur& poidsH
198     ,const ParaAlgoControle & pa,bool cald_Dvirtuelle);
199
200 private: // pour éviter les modifications par les classes dérivées
201     static TenseurHH * sig_bulk_pourPiPoCo_HH; // variable de travail pour le bulk
202 };
203

```

```
202 #endif
```

## 7.148 ElemMeca.h

```
1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:   Défini l'element generique de mecanique.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date !   auteur !           but
44 *   -----
45 *   !           !           !
46 *   $
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date !   auteur !           but
50 *   -----
51 *   $
52 *****/
53 #ifndef ELEMMECA_H
54 #define ELEMMECA_H
55
56 #include "Element.h"
57 #include "Tenseur.h"
58 #include "NevezTenseur.h"
59 #include "Deformation.h"
60 #include "Loi_comp_abstraite.h"
61 #include "Enum_calcul_masse.h"
62 #include "Basiques.h"
63 #include "Enum_dure.h"
64 #include "CompThermoPhysiqueAbstraite.h"
65 #include "CompFrotAbstraite.h"
66 #include "LesPtIntegMecaInterne.h"
67 #include "Enum_StabHourglass.h"
68 #include "LesChargeExtSurElement.h"
69 #include "Temps_CPU_HZpp.h"
70 #include "Enum_StabMembrane.h"
71
72
73 /// @addtogroup groupe_des_elements_finis
74 /// @{
75 ///
76
77
78 class ElemMeca : public Element
79 {
80 public :
```

```

81 // VARIABLES PUBLIQUES :
82
83 // CONSTRUCTEURS :
84 ElemMeca ();
85 // Constructeur utile quand le numero de maillage et d'identification de l'element est connu
86 ElemMeca (int num_maill,int num_id) ;
87 // Constructeur utile quand le numero de maillage et d'identification et le tableau des noeuds
88 // de l'element sont connus
89 ElemMeca (int num_maill,int num_id,const Tableau<Noeud *>& tab);
90 // Constructeur utile quand le numero de maillage et d'identification est connu,
91 // ainsi que la geometrie et le type d'interpolation de l'element
92 ElemMeca (int num_maill,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string
info="");
93 // Constructeur utile quand le numero de maillage et d'identification est connu,
94 // ainsi que la geometrie et le type d'interpolation de l'element
95 ElemMeca (int num_maill,int num_id,char* nom_interpol,char* nom_geom,string info="");
96 // Constructeur utile quand toutes les donnees de la classe Element sont connues
97 ElemMeca (int num_maill,int num_id,const Tableau<Noeud *>& tab,Enum_interpol id_interp_elt,
Enum_geom id_geom_elt,string info="");
98 // Constructeur utile quand toutes les donnees de la classe Element sont connues
100 ElemMeca (int num_maill,int num_id,const Tableau<Noeud *>& tab,char* nom_interpol,
char* nom_geom,string info="");
101
102 // Constructeur de copie
103 ElemMeca (const ElemMeca& elt);
104
105 // DESTRUCTEUR :
106 ~ElemMeca ();
107
108 // METHODES PUBLIQUES :
109 // test si l'element est complet
110 // = 1 tout est ok, =0 element incomplet
111 int TestComplet();
112
113 // calcul si un point est a l'interieur de l'element ou non
114 // il faut que M est la dimension globale
115 // les trois fonctions sont pour l'etude a t=0, t et tdt
116 // retour : =0 le point est externe, =1 le point est interne ,
117 // = 2 le point est sur la frontiere a la precision pres
118 // coor_locales : s'il est different de NULL, est affecte des coordonnees locales calculees,
119 // uniquement precises si le point est interne
120 int Interne_0(const Coordonnee& M,Coordonnee* coor_locales=NULL);
121 int Interne_t(const Coordonnee& M,Coordonnee* coor_locales=NULL);
122 int Interne_tdt(const Coordonnee& M,Coordonnee* coor_locales=NULL);
123
124 // recuperation des energies integrees sur l'elements, resultants d'un precedent calcul
125 // explicite, ou implicite
126 const EnergieMeca& EnergieTotaleElement() const {return energie_totale;};
127
128 // test pour savoir si le calcul de contrainte en absolu est possible
129 bool ContrainteAbsoluePossible();
130
131 // METHODES VIRTUELLES:
132
133 // retourne la liste de tous les types de ddl interne actuellement utilises
134 // par l'element (actif ou non), sont exclu de cette liste les ddl des noeuds
135 // relies a l'element (ddl implique grandeur uniquement scalaire !)
136 // par contre sont inclus les ddl venant de l'element, qui sont represente directement aux noeuds
137 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particuliere
138 virtual List_io <Ddl_enum_etendu> Les_type_de_ddl_interne(bool absolue) const ;
139 // idem pour les grandeurs evoluees c'est-a-dire directement sous forme de vecteur, tenseurs
....
140 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particuliere
141 virtual List_io <TypeQuelconque> Les_type_evolees_interne(bool absolue) const ;
142 // idem pour les donnees particulieres
143 virtual List_io <TypeQuelconque> Les_types_particuliers_interne(bool absolue) const;
144
145 // retourne la liste de toutes les grandeurs quelconques relatives aux faces de
146 // l'element (actif ou non),
147 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particuliere
148 virtual List_io <TypeQuelconque> Les_type_quelconque_de_face(bool absolue) const;
149 // retourne la liste de toutes les grandeurs quelconques relatives aux arêtes de
150 // l'element (actif ou non),
151 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particuliere
152 virtual List_io <TypeQuelconque> Les_type_quelconque_de_arete(bool absolue) const;
153
154
155 // ----- calculs utils dans le cadre de la recherche du flambement linéaire
156 // dans un premier temps uniquement virtuelles, ensuite se sera virtuelle pure pour éviter
157 // les oublis de définition ----> AMODIFIER !!!!!!!
158 // Calcul de la matrice géométrique et initiale
159 class MatGeomInit // pour le retour des pointeurs sur des matrices stockées
160 // une paire par classe d'éléments
161 { public : MatGeomInit(Mat_pleine * matG,Mat_pleine * matI) :

```

```

162         matGeom(matG),matInit(matI) {});
163         Mat_pleine * matGeom;Mat_pleine * matInit ;
164     };
165     virtual MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
166
167 // ----- calcul d'erreur, calculs du champs de contrainte continu -----
168 // ajout des ddl de contraintes pour les noeuds de l'élément
169 virtual void Plus_ddl_Sigma() = 0;
170 // inactive les ddl du problème de recherche d'erreur : les contraintes
171 virtual void Inactive_ddl_Sigma() = 0;
172 // active les ddl du problème de recherche d'erreur : les contraintes
173 virtual void Active_ddl_Sigma() = 0 ;
174 // active le premier ddl du problème de recherche d'erreur : SIGMA11
175 virtual void Active_premier_ddl_Sigma() = 0 ;
176 // retourne un tableau de ddl element, correspondant à la
177 // composante de sigma -> SIG11, pour chaque noeud qui contient
178 // des ddl de contrainte
179 // -> utilisé pour l'assemblage de la raideur d'erreur
180 //!!!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en
181 // virtuelle pure !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
182 virtual DdlElement& Tableau_de_Sig1() const ;
183 // retourne un tableau de ddl element, correspondant à la
184 // composante d'erreur -> ERREUR, pour chaque noeud
185 // -> utilisé pour l'assemblage de la raideur d'erreur
186 //dans cette version tous les noeuds sont supposés avoir un ddl erreur
187 // dans le cas contraire il faut redéfinir la fonction dans l'élément terminal
188 virtual DdlElement Tableau_de_ERREUR() const ;
189 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
190 // qu'une fois la remontée aux contraintes effectuées sinon aucune
191 // action. En retour la valeur de l'erreur sur l'élément
192 // type indique le type de calcul d'erreur :
193 // = 1 : erreur = (int (delta sigma):(delta sigma) dv)/(int sigma:sigma dv)
194 // le numerateur et le denominateur sont tel que :
195 // errElemRelative = numerateur / denominateur , si denominateur different de 0
196 // sinon denominateur = numerateur si numerateur est different de 0, sinon
197 // tous sont nuls mais on n'effectue pas la division
198 //!!!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en
199 // virtuelle pure !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
200 virtual void ErreurElement(int type,double& errElemRelative
201 ,double& numerateur, double& denominateur);
202 // les 3 routines qui suivent sont virtuelles, car la définition
203 //qui est faite dans ElemMeca.cp considère qu'il y a un ddl erreur
204 // par noeud de l'élément de manière systématique,
205 // avec le status virtuel on peut définir dans la classe dérivée
206 // un cas particulier
207 // ajout des ddl d'erreur pour les noeuds de l'élément
208 virtual void Plus_ddl_Erreur() ;
209 // inactive les ddl d'erreur
210 virtual void Inactive_ddl_Erreur() ;
211 // active les ddl d'erreur
212 virtual void Active_ddl_Erreur() ;
213 // test pour savoir si l'erreur a été calculée
214 bool ErreurDejaCalculee()
215 { if (sigErreur == NULL)
216     return false;
217     else return true;};
218 // sortie de l'erreur à l'élément
219 double Erreur( )
220 { return (*sigErreur);};
221
222 // lecture de données diverses sur le flot d'entrée
223 // l'implantation est faite dans les classe dérivées
224 virtual void LectureContraintes(UtilLecture * entreePrinc) =0 ;
225
226 // retour des contraintes en absolu retour true si elle existe sinon false
227 virtual bool ContraintesAbsolues(Tableau <Vecteur>& tabSig) = 0;
228
229 // ----- calcul dynamique -----
230
231 // calcul de la longueur d'arrête de l'élément minimal
232 // divisé par la célérité dans le matériau
233 virtual double Long_arrete_mini_sur_c(Enum_dure temps) = 0;
234 // cas du bulk viscosity
235 double E_elem_bulk_t,E_elem_bulk_tdt,P_elem_bulk;
236 static void ActiveBulkViscosity(int choix) {bulk_viscosity=choix;};
237 static void InactiveBulkViscosity() {bulk_viscosity=false;};
238 static void ChangeCoefsBulkViscosity(const DeuxDoubles & coef)
239 { c_traceBulk=coef.un;c_carreBulk=coef.deux;};
240
241 // initialisation pour le calcul de la matrice masse dans le cas de l'algorithme
242 // de relaxation dynamique avec optimisation en continu de la matrice masse
243 // casMass_relax: permet de choisir entre différentes méthodes de calcul de la masse
244 void InitCalculMatriceMassePourRelaxationDynamique(int casMass_relax);
245 // phase de calcul de la matrice masse dans le cas de l'algo de relaxation dynamique
246 // mi=fonction de (alpha*K+beta*mu+gamma*Isig/3+theta/2*Sig_mises
247 // ep: epaisseur, K module de compressibilité, mu: module de cisaillement, Isig trace de sigma,
248 // Sig_mises la contrainte de mises

```

```

249 // casMass_relax: permet de choisir entre différentes méthodes de calcul de la masse
250 void CalculMatriceMassePourRelaxationDynamique
251     (const double& alph, const double& beta, const double & lambda
252     ,const double & gamma,const double & theta, int casMass_relax);
253
254 // ----- informations annexes -----
255
256 // récupération de la base locales au noeud noe, pour le temps: temps
257 const BaseB & Gib_elemecca(Enum_dure temps, const Noeud * noe);
258
259 // récupération de grandeurs particulières au numéro d'ordre = iteg
260 // celles-ci peuvent être quelconques
261 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
262 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
263 void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);
264
265 // récupération de grandeurs particulières pour une face au numéro d'ordre = iteg
266 // celles-ci peuvent être quelconques
267 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
268 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
269 void Grandeur_particuliere_face (bool absolue,List_io<TypeQuelconque>& liTQ,int face, int iteg);
270
271 // récupération de grandeurs particulières pour une arête au numéro d'ordre = iteg
272 // celles-ci peuvent être quelconques
273 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
274 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
275 void Grandeur_particuliere_arete (bool absolue,List_io<TypeQuelconque>& liTQ,int arete, int
iteg);
276
277 // récupération de la loi de frottement, dans le cas où elle n'existe pas
278 // retour d'un pointeur nul
279 CompProtAbstraite* LoiDeFrottement() const {return loiProt;};
280
281 // récupération de la loi de comportement pour information, dans le cas où elle n'existe pas
282 // retour d'un pointeur nul
283 const Loi_comp_abstraite* LoiDeComportement() const {return loiComp;};
284
285 // recup du module de compressibilité moyen de l'élément
286 double CompressibiliteMoyenne() const {return lesPtIntegMecaInterne->CompressibiliteMoyenne();};
287
288 // --- transfert des grandeurs des points d'intégration aux noeuds
289 // transfert de ddl des points d'intégrations (de tous) d'un éléments (on ajoute aux noeuds,
on ne remplace pas)
290 // les ddl doivent déjà exister aux noeuds sinon erreur
291 // il doit s'agir du même type de répartition de pt d'integ pour toutes les grandeurs
292 // tab_val(i)(j) : valeur associée au i ième pt d'integ et au j ième ddl_enum_etendu
293 void TransfertAjoutAuNoeuds(const List_io < Ddl_enum_etendu >& lietendu
, const Tableau <Tableau <double> > & tab_val,int cas);
294
295 // transfert de type quelconque des points d'intégrations (de tous) aux noeuds d'un éléments (on
ajoute aux noeuds,
296 // on ne remplace pas). Les types quelconques doivent déjà exister
297 // un tableau dans tab_liQ correspondent aux grandeurs quelconque pour tous les pt integ,
298 // tab_liQ(i) pour le pt d'integ i
299 // liQ_travail: est une liste de travail qui sera utilisée dans le transfert
300
301 void TransfertAjoutAuNoeuds(const Tableau <List_io < TypeQuelconque > > & tab_liQ
, List_io < TypeQuelconque > & liQ_travail,int cas);
302
303 // accumulation aux noeuds de grandeurs venant de l'éléments vers ses noeuds (exemple la pression
appliquée)
304 // autres que celles aux pti classiques, mais directement disponibles
305 // le contenu du conteneur stockées dans liQ est utilisé en variable intermédiaire
306 void Accumul_aux_noeuds(const List_io < Ddl_enum_etendu >& lietendu
, List_io < TypeQuelconque > & liQ,int cas);
307
308
309 // activation du calcul des invariants de contraintes, qui seront calculé à chaque
310 // fois que l'on calcul les contraintes au travers de la loi de comportement
311 void ActivCalculInvariantsContraintes();
312 // idem pour la déformation
313 void ActivCalculInvariantsDeformation();
314 // idem pour la vitesse de déformation
315 void ActivCalculInvariantsVitesseDeformation();
316
317 // modification de l'orientation de l'élément en fonction de cas_orientation
318 // =0: inversion simple (sans condition) de l'orientation
319 // si cas_orientation est diff de 0: on calcul le jacobien aux différents points d'intégration
320 // 1. si tous les jacobiens sont négatifs on change d'orientation
321 // 2. si tous les jacobiens sont positifs on ne fait rien
322 // 3. si certains jacobiens sont positifs et d'autres négatifs message
323 // d'erreur et on ne fait rien
324 // ramène true: s'il y a eu changement effectif, sinon false
325 bool Modif_orient_elem(int cas_orientation);
326
327 // calcul éventuel de la normale à un noeud

```

```

328 // ce calcul existe pour les éléments 2D, 1D axi, et aussi pour les éléments 1D
329 // qui possède un repère d'orientation
330 // en retour coor = la normale si coor.Dimension() est = à la dimension de l'espace
331 // si le calcul n'existe pas --> coor.Dimension() = 0
332 // ramène un entier :
333 // == 1 : calcul normal
334 // == 0 : problème de calcul -> coor.Dimension() = 0
335 // == 2 : indique que le calcul n'est pas licite pour le noeud passé en paramètre
336 // c'est le cas par exemple des noeuds extérieurs pour les éléments SFE
337 // mais il n'y a pas d'erreur, c'est seulement que l'élément n'est pas ad hoc pour
338 // calculer la normale à ce noeud là
339 // temps: indique à quel moment on veut le calcul
340 // pour des éléments particulier (ex: SFE) la méthode est surchargée
341 virtual int CalculNormale_noeud(Enum_dure temps, const Noeud& noe, Coordonnee& coor);
342
343 // calcul si un point est à l'intérieur de l'élément ou non
344 // il faut que M est la dimension globale
345 // retour : =0 le point est externe, =1 le point est interne,
346 // = 2 le point est sur la frontière à la précision près
347 // coor_locales : s'il est différent de NULL, est affecté des coordonnées locales calculées,
348 // uniquement précises si le point est interne
349 int Interne(Enum_dure temps, const Coordonnee& M, Coordonnee* coor_locales=NULL);
350 // -- connaissances particulières sur l'élément
351 // ramène l'épaisseur de l'élément
352 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
353 virtual double Epaisseurs(Enum_dure, const Coordonnee&) {return 0.};
354 // ramène l'épaisseur moyenne de l'élément (indépendante du point)
355 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
356 virtual double EpaisseurMoyenne(Enum_dure) {return 0.};
357 // ramène la section de l'élément
358 // =0. si la notion de section ne veut rien dire pour l'élément
359 virtual double Section(Enum_dure, const Coordonnee&) {return 0.};
360 // ramène la section moyenne de l'élément (indépendante du point)
361 // =0. si la notion de section ne veut rien dire pour l'élément
362 virtual double SectionMoyenne(Enum_dure) {return 0.};
363 // // modifie l'épaisseur Moyenne à tdt
364 // virtual void Modifie_epaisseur_moyenne_tdt(const double& h_t)
365 // {cout << "\n erreur** la methode Modifie_epaisseur_moyenne(.. n'existe pas "; Sortie(1);};
366
367 // fonction a renseigner par les classes dérivées, concernant les répercussions
368 // éventuelles due à la suppression de tous les frontières
369 // nums_i : donnent les listes de frontières supprimées
370 virtual void Prise_en_compte_des_consequences_suppression_tous_frontieres();
371 // idem pour une frontière (avant qu'elle soit supprimée)
372 virtual void Prise_en_compte_des_consequences_suppression_une_frontiere(ElFrontiere* elemFront);
373
374 // ----- calcul de frontières -----
375
376 // ramène la frontière point
377 // éventuellement création des frontieres points de l'élément et stockage dans l'élément
378 // si c'est la première fois sinon il y a seulement retour de l'éléments
379 // a moins que le paramètre force est mis a true
380 // dans ce dernier cas la frontière effacée est recréée
381 // num indique le numéro du point à créer (numérotation EF)
382 virtual ElFrontiere* const Frontiere_points(int num, bool force);
383
384 // ramène la frontière linéique
385 // éventuellement création des frontieres linéique de l'élément et stockage dans l'élément
386 // si c'est la première fois et en 3D sinon il y a seulement retour de l'éléments
387 // a moins que le paramètre force est mis a true
388 // dans ce dernier cas la frontière effacée est recréée
389 // num indique le numéro de l'arête à créer (numérotation EF)
390 virtual ElFrontiere* const Frontiere_lineique(int num, bool force);
391
392 // ramène la frontière surfacique
393 // éventuellement création des frontieres surfacique de l'élément et stockage dans l'élément
394 // si c'est la première fois sinon il y a seulement retour de l'éléments
395 // a moins que le paramètre force est mis a true
396 // dans ce dernier cas la frontière effacée est recréée
397 // num indique le numéro de la surface à créer (numérotation EF)
398 virtual ElFrontiere* const Frontiere_surfacique(int num, bool force);
399
400 // ----- init éventuelle avant le chargement -----
401 // initialisation éventuelle, nécessaire avant d'appliquer l'ensemble des charges
402 // par exemple des stockages intermédiaires
403 virtual void Initialisation_avant_chargement();
404
405 // mise à jour éventuel de repère d'anisotropie
406 virtual void Mise_a_jour_repere_anisotropie
407 (BlocGen & bloc, LesFonctions_nD* lesFonctionsnD);
408
409
410 //=====
411 protected :
412 //=====
413 // METHODES PROTEGEES utilisables par les classes derivees :
414

```

```

415         // Calcul des frontieres de l'element
416         // creation des elements frontieres et retour du tableau de ces elements
417         // la création n'a lieu qu'au premier appel
418         // ou lorsque l'on force le paramètre force a true
419         // dans ce dernier cas seul les frontière effacées sont recréée
420         // cas :
421         // = 0 -> on veut toutes les frontières
422         // = 1 -> on veut uniquement les surfaces
423         // = 2 -> on veut uniquement les lignes
424         // = 3 -> on veut uniquement les points
425         // = 4 -> on veut les surfaces + les lignes
426         // = 5 -> on veut les surfaces + les points
427         // = 6 -> on veut les lignes + les points
428         Tableau <ElFrontiere*> const & Frontiere_elemecca(int cas, bool force = false);
429
430     // -----
431     // cas où l'on intègre que selon une liste (un axe, un plan, un volume)
432     // -----
433         // Calcul du residu local et de la raideur locale,
434         // pour le schema implicite d'ou a l'instant t + dt
435         // ddl represente les degres de liberte specifiques a l'element
436         // tabDepsBB = vitesse de deformation, tabDeltaEpsBB = incrément de def entre t et t+dt
437         // cald_Dvirtuelle = indique si l'on doit calculer la dérivée de la vitesse de déformation
virtuelle
438         void Cal_implicit (DdlElement & tab_ddl,Tableau <TenseurBB *> & d_epsBB
439             ,Tableau < Tableau2 <TenseurBB *> > d2_epsBB,Tableau <TenseurHH *>& d_sigHH,int nbint
440             ,const Vecteur& poids,const ParaAlgoControle & pa,bool cald_Dvirtuelle);
441
442         // Calcul du residu local a l'instant t ou tdt
443         // atdt = true : calcul à tdt, valeur par défaut
444         // = false: calcul à t
445         // ddl represente les degres de liberte specifiques a l'element
446         // nbint = nb de pt d'integration , poids = poids d'integration
447         void Cal_explicit (DdlElement & ddl,Tableau <TenseurBB *>& d_epsBB,int nbint
448             ,const Vecteur& poids,const ParaAlgoControle & pa,bool atdt=true);
449
450         // Calcul de la matrice géométrique et de la matrice initiale
451         // cette fonction est éventuellement appelée par les classes dérivées
452         // ddl represente les degres de liberte specifiques a l'element
453         // nbint = nb de pt d'integration , poids = poids d'integration
454         // cald_Dvirtuelle = indique si l'on doit calculer la dérivée de la vitesse de déformation
virtuelle
455         void Cal_matGeom_Init (Mat_pleine & matGeom, Mat_pleine & matInit
456             ,DdlElement & ddl,Tableau <TenseurBB *>& d_epsBB,Tableau < Tableau2 <TenseurBB *> >
d2_epsBB
457             ,Tableau <TenseurHH *>& d_sigHH,int nbint,const Vecteur& poids
458             ,const ParaAlgoControle & pa,bool cald_Dvirtuelle);
459
460         // Calcul de la matrice masse selon différent choix donné par type_matrice_masse,
461         // a l'instant initial.
462         void Cal_Mat_masse (DdlElement & tab_ddl,Enum_calcul_masse type_matrice_masse,
463             int nbint,const Tableau <Vecteur*> & taphi,int nbne
464             ,const Vecteur& poids);
465
466     // -----
467     // cas où l'on intègre selon deux listes (ex un axe et un plan, etc.)
468     // -----
469         // Calcul du residu local et de la raideur locale,
470         // pour le schema implicite d'ou a l'instant t + dt
471         // ddl represente les degres de liberte specifiques a l'element
472         void Cal_implicitap (DdlElement & tab_ddl,Tableau <TenseurBB *> & d_epsBB
473             ,Tableau < Tableau2 <TenseurBB *> > d2_epsBB,Tableau <TenseurHH *>&
d_sigHH
474             ,int nbint1,Vecteur& poids1,int nbint2,const Vecteur& poids2
475             ,const ParaAlgoControle & pa);
476
477         // Calcul du residu local a l'instant t ou tdt
478         // atdt = true : calcul à tdt, valeur par défaut
479         // = false: calcul à t
480         // ddl represente les degres de liberte specifiques a l'element
481         // d_epsbb = variation des def
482         // nbint = nb de pt d'integration , poids = poids d'integration
483         void Cal_explicitap (DdlElement & ddl,Tableau <TenseurBB *>& d_epsBB
484             ,int nbint,const Vecteur& poids,bool atdt=true);
485
486         // Calcul de la matrice géométrique et de la matrice initiale
487         // cette fonction est éventuellement appelée par les classes dérivées
488         // ddl represente les degres de liberte specifiques a l'element
489         // d_epsbb = variation des def
490         // nbint = nb de pt d'integration , poids = poids d'integration
491         void Cal_matGeom_Initap (Mat_pleine & matGeom, Mat_pleine & matInit
492             ,DdlElement & ddl,Tableau <TenseurBB *>& d_epsBB,Tableau < Tableau2 <TenseurBB *> >
d2_epsBB
493             ,Tableau <TenseurHH *>& d_sigHH,int nbint,const Vecteur& poids);
494
495     // ----- calcul de second membre -----
496         // calcul des seconds membres suivant les chargements

```



```

497 // cas d'un chargement surfacique, sur les frontières des éléments
498 // force indique la force surfacique appliquée
499 // retourne le second membre résultant
500 // nSurf : le numéro de la surface externe
501 // calcul à l'instant tdt ou t en fonction de la variable atdt
502 Vecteur& SM_charge_surf_E (DdlElement & ddls,int nSurf
503 ,const Tableau <Vecteur>& taphi,int nbne
504 ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
505 ,const ParaAlgoControle & pa,bool atdt=true);
506 // idem SM_charge_surf_E mais -> implicite,
507 // pa : permet de déterminer si l'on fait ou non le calcul de la contribution à la raideur
508 // retourne le second membre et la matrice de raideur correspondant
509 Element::ResRaid SMR_charge_surf_I (DdlElement & ddls,int nSurf
510 ,const Tableau <Vecteur>& taphi,int nbne
511 ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
512 ,const ParaAlgoControle & pa);
513 // calcul des seconds membres suivant les chargements
514 // cas d'un chargement pression, sur les frontières des éléments
515 // pression indique la pression appliquée
516 // la fonction nD est utilisée que si elle ne dépend pas strictement de grandeurs globales
517 // retourne le second membre résultant
518 // nSurf : le numéro de la surface externe
519 // calcul à l'instant tdt ou t en fonction de la variable atdt
520 Vecteur& SM_charge_pres_E (DdlElement & ddls,int nSurf
521 ,const Tableau <Vecteur>& taphi,int nbne
522 ,const Vecteur& poids,double pression,Fonction_nD* pt_fonct
523 ,const ParaAlgoControle & pa,bool atdt=true);
524 // idem SM_charge_pres_E mais -> implicite,
525 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
526 // retourne le second membre et la matrice de raideur correspondant
527 // la fonction nD est utilisée que si elle ne dépend pas strictement de grandeurs globales
528 Element::ResRaid SMR_charge_pres_I (DdlElement & ddls,int nSurf
529 ,const Tableau <Vecteur>& taphi,int nbne
530 ,const Vecteur& poids,double pression,Fonction_nD* pt_fonct
531 ,const ParaAlgoControle & pa);
532 // cas d'un chargement lineique, sur les arêtes frontières des éléments
533 // force indique la force lineique appliquée
534 // retourne le second membre résultant
535 // nArete : le numéro de l'arête externe
536 // calcul à l'instant tdt ou t en fonction de la variable atdt
537 Vecteur& SM_charge_line_E (DdlElement & ddls,int nArete
538 ,const Tableau <Vecteur>& taphi,int nbne
539 ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
540 ,const ParaAlgoControle & pa,bool atdt=true);
541 // idem SM_charge_line_E mais -> implicite,
542 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
543 // retourne le second membre et la matrice de raideur correspondant
544 Element::ResRaid SMR_charge_line_I (DdlElement & ddls,int nArete
545 ,const Tableau <Vecteur>& taphi,int nbne
546 ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
547 ,const ParaAlgoControle & pa);
548
549 // cas d'un chargement lineique suiveur, sur les arêtes frontières des éléments
550 // pas valable pour des éléments 3D !
551 // force indique la force lineique appliquée
552 // retourne le second membre résultant
553 // nArete : le numéro de l'arête externe
554 // calcul à l'instant tdt ou t en fonction de la variable atdt
555 Vecteur& SM_charge_line_Suiv_E (DdlElement & ddls,int nArete
556 ,const Tableau <Vecteur>& taphi,int nbne
557 ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
558 ,const ParaAlgoControle & pa,bool atdt=true);
559 // idem SM_charge_line_E mais -> implicite,
560 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
561 // retourne le second membre et la matrice de raideur correspondant
562 Element::ResRaid SMR_charge_line_Suiv_I (DdlElement & ddls,int nArete
563 ,const Tableau <Vecteur>& taphi,int nbne
564 ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
565 ,const ParaAlgoControle & pa);
566
567
568 // cas d'un chargement surfacique suiveur, sur les surfaces de l'élément
569 // la direction varie selon le système suivant: on définit les coordonnées matérielles
570 // de la direction, ce qui sert ensuite à calculer les nouvelles directions. L'intensité
571 // elle est constante.
572 // force indique la force surfacique appliquée
573 // retourne le second membre résultant
574 // nSurf : le numéro de la surface externe
575 // calcul à l'instant tdt ou t en fonction de la variable atdt
576 Vecteur& SM_charge_surf_Suiv_E (DdlElement & ddls,int nSurf
577 ,const Tableau <Vecteur>& taphi,int nbne
578 ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
579 ,const ParaAlgoControle & pa,bool atdt=true);
580 // idem SM_charge_surf_Suiv_E mais -> implicite,
581 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
582 // retourne le second membre et la matrice de raideur correspondant
583 Element::ResRaid SMR_charge_surf_Suiv_I (DdlElement & ddls,int nSurf

```



```

584                                     ,const Tableau <Vecteur>& taphi,int nbne
585                                     ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
586                                     ,const ParaAlgoControle & pa);
587
588 // cas d'un chargement volumique, sur l'élément
589 // force indique la force volumique appliquée
590 // retourne le second membre résultant
591 // calcul à l'instant tdt ou t en fonction de la variable atdt
592 Vecteur& SM_charge_vol_E (DdlElement & ddls
593                          ,const Tableau <Vecteur>& taphi,int nbne
594                          ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
595                          ,const ParaAlgoControle & pa,bool sur_volume_finale_,bool atdt=true);
596
597 // idem SM_charge_vol_E mais -> implicite,
598 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
599 // retourne le second membre et la matrice de raideur correspondant
600 void SMR_charge_vol_I (DdlElement & ddls
601                      ,const Tableau <Vecteur>& taphi,int nbne
602                      ,const Vecteur& poids,const Coordonnee& force,Fonction_nD* pt_fonct
603                      ,const ParaAlgoControle & pa,bool sur_volume_finale_);
604
605 // cas d'un chargement hydrostatique, sur les surfaces de l'élément
606 // la charge dépend de la hauteur à la surface libre du liquide déterminée par un point
607 // et une direction normale à la surface libre:
608 // nSurf : le numéro de la surface externe
609 // poidvol: indique le poids volumique du liquide
610 // M_liquide : un point de la surface libre
611 // dir_normal_liquide : direction normale à la surface libre
612 // sans_limitation: indique si l'on doit limiter aux positions négatives
613 // retourne le second membre résultant
614 // calcul à l'instant tdt ou t en fonction de la variable atdt
615 Vecteur& SM_charge_hydro_E (DdlElement & ddls,int nSurf
616                            ,const Tableau <Vecteur>& taphi,int nbne
617                            ,const Vecteur& poids
618                            ,const Coordonnee& dir_normal_liquide,const double& poidvol
619                            ,const Coordonnee& M_liquide,bool sans_limitation
620                            ,const ParaAlgoControle & pa,bool atdt=true);
621 // idem SM_charge_hydro_E mais -> implicite,
622 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
623 // retourne le second membre et la matrice de raideur correspondant
624 Element::ResRaid SMR_charge_hydro_I (DdlElement & ddls,int nSurf
625                                     ,const Tableau <Vecteur>& taphi,int nbne
626                                     ,const Vecteur& poids
627                                     ,const Coordonnee& dir_normal_liquide,const double& poidvol
628                                     ,const Coordonnee& M_liquide,bool sans_limitation
629                                     ,const ParaAlgoControle & pa);
630
631 // cas d'un chargement aero-hydrodynamique, sur les frontières de l'élément
632 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
633 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc
unitaire)
634 // une suivant la direction normale à la vitesse de type portance
635 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
636 // une suivant la vitesse tangente de type frottement visqueux
637 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
638 // retourne le second membre résultant
639 // calcul à l'instant tdt ou t en fonction de la variable atdt
640 // coef_mul: est un coefficient multiplicateur global (de tout)
641 Vecteur& SM_charge_hydrodyn_E (const double& poidvol,const Tableau <Vecteur>& taphi,int nbne
642                               ,CourbeID* frot_fluid,const Vecteur& poids
643                               ,CourbeID* coef_aero_n,int numfront,const double& coef_mul
644                               ,CourbeID* coef_aero_t,const ParaAlgoControle & ,bool
atdt=true);
645 // idem SM_charge_hydrodyn_E mais -> implicite,
646 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
647 // retourne le second membre et la matrice de raideur correspondant
648 Element::ResRaid SM_charge_hydrodyn_I (const double& poidvol,const Tableau <Vecteur>& taphi,int
nbne
649                                       ,CourbeID* frot_fluid,const Vecteur& poids,DdlElement & ddls
650                                       ,CourbeID* coef_aero_n,int numfront,const double& coef_mul
651                                       ,CourbeID* coef_aero_t,const ParaAlgoControle & pa);
652
653 // ----- calcul de frontières en protected -----
654
655 // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
particulière à l'élément
656 // adressage des frontières linéiques et surfacique
657 // définit dans les classes dérivées, et utilisées pour la construction des frontières
658 virtual ElFrontiere* new_frontiere_lin(int num,Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
659 virtual ElFrontiere* new_frontiere_surf(int num,Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
660
661 // ----- stabilisation d'hourglass -----
662 // calcul d'élément de contrôle d'hourglass associée à un comportement
663 void Init_hourglass_comp(const ElemGeomC0& elgeHour, const string & str_precision
664                        ,LoiAbstraiteGeneral * loiHourglass,const BlocGen & bloc);
665
666 // stabilisation pour un calcul implicit

```

```

667 void Cal_implicit_hourglass();
668
669 // stabilisation pour un calcul explicite
670 void Cal_explicit_hourglass(bool atdt);
671
672 // ----- stabilisation transversale éventuelle de membrane ou de biel
673 // -----
674 // utilisée et alimentées par les classes dérivées: ElemMeca sert de conteneur ce qui permet
675 // d'éviter de redéfinir des conteneurs locaux aux éléments membranes et biel
676 class StabMembBiel // conteneur local pour un stockage globale de toutes les grandeurs
concernées
677 { public :
678 // par défaut
679 StabMembBiel();
680 // fonction du nombre de noeud ou pti
681 StabMembBiel(int nbnoe);
682 // fonction d'une valeur numérique et du nom éventuel d'une fonction
683 StabMembBiel(double v, string * s);
684 // de copie
685 StabMembBiel(const StabMembBiel& a);
686 ~StabMembBiel();
687 StabMembBiel& operator=(const StabMembBiel& a);
688 // --- acces aux données en inline
689 Enum_StabMembraneBiel& Type_stabMembrane() {return type_stabMembrane;}
690 bool& Aa_calculer() {return a_calculer;};
691 double& Valgamma() {return gamma;};
692 Fonction_nD* Pt_fct_gamma() {return pt_fct_gamma;};
693 void Change_pt_fct_gamma(Fonction_nD* fct) {pt_fct_gamma=fct;};
694
695 //beta
696 const double& Beta() const {return beta;};
697 void Change_beta(double bet) {beta = bet;};
698 //f_mini
699 const double& F_mini() const {return f_mini;};
700 void Change_f_mini(double f) {f_mini = f;};
701 // d_maxi
702 const double& D_maxi() const {return d_maxi;};
703 void Change_d_maxi(double d) {d_maxi = d;};
704
705 // l'intensité de la force de stabilisation au noeud i ou au pti i
706 double& FF_StabMembBiel(int i) {return F_StabMembBiel(i);};
707 double& FF_StabMembBiel_t(int i) {return F_StabMembBiel_t(i);};
708 // l'énergie développée par la stabilisation au noeud i ou au pti i
709 double& EE_StabMembBiel(int i) {return E_StabMembBiel(i);};
710 // l'énergie développée par la stabilisation au noeud i ou au pti i
711 double& EE_StabMembBiel_t(int i) {return E_StabMembBiel_t(i);};
712
713 // énergie totale développée sur l'élément pour la stabilisation
714 double EE_total_StabMembBiel() const ;
715 double EE_total_StabMembBiel_t() const ;
716
717 // valeur de référence pour calculer l'intensité de stabilisation
718 // (identique pour tous les pti), typiquement == le max de la raideur par exemple
719 double& Stab_ref() {return stab_ref;};
720
721 string * Nom_fctnD() {return nom_fctnD;};
722 // actualisation de la force de stabilisation de t+dt vers t
723 void TdtversT();
724 // actualisation de la force de stabilisation de t vers tdt
725 void TversTdt();
726
727 // changement du nombre de pti ou de noeuds, suivant le type de stabilisation
728 void Change_nb_pti(int nbnoe);
729 // le nb de pti ou de noeud du conteneur
730 int Taille() const {return F_StabMembBiel.Taille();};
731
732 // --- données
733 protected:
734 Enum_StabMembraneBiel type_stabMembrane; // indique le type de stabilisation
735 bool a_calculer; // indique si oui ou non il faut calculer la force de stabilisation
736 double gamma; // si pt_fct_gamma == NULL -> valeur numérique lue
737 // si pt_fct_gamma != NULL -> contient la dernière valeur d'alpha calculée via
738 pt_fct_gamma
739 Fonction_nD* pt_fct_gamma; // si non null, c'est cette fonction que l'on utilise sinon
740 // c'est la valeur numérique gamma
741 double stab_ref; // valeur de référence pour calculer l'intensité de stabilisation
742 // (identique pour tous les pti ou noeuds), typiquement == le max de la raideur
743 par exemple // peut changer ensuite pendant le calcul, suivant le type de stabilisation
744
745 //--- gestion éventuelle des maxi mini ----
746 // si l'intensité de la stabilisation est supérieure à beta * intensite_Fext
747 // on limite (cas où intensite_Fext >= F_mini)
748 double beta;
749 double f_mini; // force mini pour détection de force externe
750 double d_maxi; // limitation éventuelle sur un déplacement maxi

```

```

750
751 // --- stockage des infos -----
752 Tableau <double> F_StabMembBiel; // la force généralisé de stabilisation actuelle à chaque
noeud
753 Tableau <double> F_StabMembBiel_t; // la force généralisé de stabilisation à t
754 Tableau <double> E_StabMembBiel; // l'énergie développée par la stabilisation à chaque
noeud
755 Tableau <double> E_StabMembBiel_t; // l'énergie développée par la stabilisation à t à
chaque noeud
756
757 string * nom_fctnD; // sert uniquement lors de la lecture base info, en attendant de
définir
758 //pt_fct_gamma
759 };
760
761 public : // retour de l'énergie totale de stabilisation éventuelle de membrane et ou de biel
762 double Energie_stab_membBiel()
763 {if (pt_StabMembBiel == NULL)
764 {return 0.;}
765 else
766 {return pt_StabMembBiel->EE_total_StabMembBiel();};
767 };
768
769 protected :
770 StabMembBiel* pt_StabMembBiel; // pointeur éventuellement non nul
771 double maxi_F_t; // grandeur de travail utiliser par Cal_implicit_StabMembBiel
772 Mat_pleine* matD; // raideur éventuelle, définie et supprimée dans la classe dérivée relative à
un EF particulier
773 Vecteur* resD; // résidu éventuelle, défini et supprimé dans la classe dérivée relative à un
EF particulier
774 // mise à jour de "a_calculer" en fonction du contexte
775 void Mise_a_jour_A_calculer_force_stab();
776 // stabilisation pour un calcul implicit,
777 // iteg -> donne le numéro du dernier pti sur lequel on a travaillé, auquel met est associé
778 // iteg == 0 : un seul calcul global, et on est à la suite d'une boucle
779 // correspond au calcul d'alpha: == stab_ref
780 // iteg == -1 : fin d'un calcul avec boucle sur tous les pti, et on est à la suite de la
boucle
781 // iteg entre 1 et nbint: on est dans une boucle de pti
782 // nbint : le nombre maxi de pti
783 // poid_volume : si iteg > 0 : le poids d'intégration
784 // si iteg <= 0 : l'intégrale de l'élément (ici la surface totale)
785 // noeud_a_prendre_en_compte: si diff de null indique les noeuds à prendre en compte
786 // : noeud_a_prendre_en_compte(i) = 0 --> le noeud i n'est pas à prendre en compte
787 // doit avoir la taille du nombre total de noeud de l'élément
788 void Cal_implicit_StabMembBiel(int iteg, const Met_abstraite::Implic & met, const int & nbint
, const double & poid_volume, Tableau <int> * noeud_a_prendre_en_compte);
789 // stabilisation pour un calcul explicite
790 // iteg -> donne le numéro de pti sur lequel on travaille
791 void Cal_explicit_StabMembBiel(int iteg, const Met_abstraite::Expli_t_tdt met, const int & nbint
, const double & poid_volume, Tableau <int> * noeud_a_prendre_en_compte);
792
793 private :
794 // static Tableau< Vecteur> vec_StabMembBiel; // tableau de vecteurs de travail, éventuellement
défini
795 // ----- fin stabilisation transversale éventuelle de membrane ou de biel
796 // -----
797
798 public :
799 // récupération de l'énergie d'hourglass éventuelle
800 // uniquement pour un pas de temps
801 double Energie_Hourglass() const {return E_Hourglass;};
802 // idem pour l'énergie et la puissance de bulk viscosity
803 double Energie_Bulk() const {return E_elem_bulk_tdt;};
804 double Puissance_Bulk() const {return P_elem_bulk;};
805 // idem éventuellement pour l'intensité de la force de stabilisation transversale au pti i
806 double Force_StabMembBiel(int i) const
807 {if (pt_StabMembBiel != NULL) return pt_StabMembBiel->FF_StabMembBiel(i); else return 0.};;
808 // idem éventuellement pour l'énergie de stabilisation transversale
809 double Energie_StabMembBiel(int i) const
810 {if (pt_StabMembBiel != NULL) return pt_StabMembBiel->EE_StabMembBiel(i); else return 0.};;
811 // récupération des temps cpu
812 Temps_CPU_HZpp Temps_lois_comportement() const
813 { Temps_CPU_HZpp inter; int taill = lesPtIntegMecaInterne->NbPti()+1;
814 for (int i=1;i<taill;i++) inter += (*lesPtIntegMecaInterne)(i).Tps_cpu_loi_comp_const();
815 return inter;};
816 Temps_CPU_HZpp Temps_metrrique_K_SM() const
817 { Temps_CPU_HZpp inter; int taill = lesPtIntegMecaInterne->NbPti()+1;
818 for (int i=1;i<taill;i++) inter += (*lesPtIntegMecaInterne)(i).TpsMetrrique_const();
819 return inter;};
820
821 // modification de l'erreur relative
822 void Change_erreur_relative(double & nevez_erreur_rel)
823 { if (sigErreur_relative == NULL) sigErreur_relative=new double;
824 *sigErreur_relative = nevez_erreur_rel;
825 };
826 // récup de l'erreur et l'erreur relative
827 const double Erreur_sigma() const

```

```

828     { if (sigErreur == NULL) return 0.;
829       else return *sigErreur;
830     };
831     const double Erreur_sigma_relative() const
832     { if (sigErreur_relative == NULL) return 0.;
833       else return *sigErreur_relative;
834     };
835
836     protected :
837     // ----- calcul d'erreur, remontée des contraintes -----
838
839     // calcul du résidu et de la matrice de raideur pour le calcul d'erreur
840     // cas d'une intégration suivant une seule liste
841     void SigmaAuNoeud_ResRaid(const int nbne,const Tableau <Vecteur>& taphi
842         ,const Vecteur& poids,Tableau <Vecteur **& resErr,Mat_pleine& raidErr
843         ,const Tableau <Vecteur>& taphiEr,const Vecteur& poidsEr );
844
845     // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
846     // qu'une fois la remontée aux contraintes effectuées sinon aucune
847     // action. pour les autres arguments de retour, idem ErreurElement
848     // qui est la fonction generique, les autres variables sont spécifiques
849     // a l'element.
850     void Cal_ErrElem(int type,double& errElemRelative
851         ,double& numerateur, double& denominateur,const int nbne,const Tableau <Vecteur>& taphi
852         ,const Vecteur& poids,const Tableau <Vecteur>& taphiEr,const Vecteur& poidsEr);
853
854     // calcul de l'erreur aux noeuds. Contrairement au cas des contraintes
855     // seul le résidu est calculé. Cas d'une intégration suivant une liste
856     void Cal_ErrAuxNoeuds(const int nbne, const Tableau <Vecteur>& taphi,
857         const Vecteur& poids,Tableau <Vecteur **& resErr );
858
859
860     // -----
861     // affichage dans la sortie transmise, des variables duales "nom"
862     // aux differents points d'integration
863     // dans le cas ou nom est vide, affichage de "toute" les variables
864     // cas = 1 -> premier passage pour de l'implicit
865     // cas = 2 -> premier passage pour de l'explicit
866     // cas = 11 -> passage autre que le premier pour de l'implicit
867     // cas = 12 -> passage autre que le premier pour de l'explicit
868     void VarDualSort(ofstream& sort, Tableau<string>& nom,int nbint,int cas);
869     // utilitaires de VarDualSort
870     // affiche en fonction d'indic les differentes variables et appel
871     // AffDefContiD en fonction de la dimension i
872     //
873     void AffDefCont( ofstream& sort,Loi_comp_abstraite::SaveResul * saveDon,
874         TenseurBB& eps0BB,TenseurBB& epsBB,TenseurBB& DepsBB,TenseurBB& DeltaEpsBB,
875         TenseurHH& sigHH,
876         TenseurHB& epsHB,TenseurHB& sigHB,
877         Coordonnee& valPropreEps,Coordonnee& valPropreDeps,Coordonnee&
878         valPropreSig,
879         double Mises,TenseurBB& epsAlmBB,TenseurBB& epslogBB, int indic);
880     void AffDefCont1D( ofstream& sort,Loi_comp_abstraite::SaveResul * saveDon,
881         TenseurBB& eps0BB,TenseurBB& epsBB,TenseurBB& DepsBB,TenseurBB& DeltaEpsBB,
882         TenseurHH& sigHH,
883         TenseurHB& epsHB,TenseurHB& sigHB,
884         Coordonnee& valPropreEps,Coordonnee& valPropreDeps,Coordonnee&
885         valPropreSig,
886         double Mises,TenseurBB& epsAlmBB,TenseurBB& epslogBB,int indic);
887     void AffDefCont2D( ofstream& sort,Loi_comp_abstraite::SaveResul * saveDon,
888         TenseurBB& eps0BB,TenseurBB& epsBB,TenseurBB& DepsBB,TenseurBB& DeltaEpsBB,
889         TenseurHH& sigHH,
890         TenseurHB& epsHB,TenseurHB& sigHB,
891         Coordonnee& valPropreEps,Coordonnee& valPropreDeps,Coordonnee&
892         valPropreSig,
893         double Mises,TenseurBB& epsAlmBB,TenseurBB& epslogBB,int indic);
894     void AffDefCont3D( ofstream& sort,Loi_comp_abstraite::SaveResul * saveDon,
895         TenseurBB& eps0BB,TenseurBB& epsBB,TenseurBB& DepsBB,TenseurBB& DeltaEpsBB,
896         TenseurHH& sigHH,
897         TenseurHB& epsHB,TenseurHB& sigHB,
898         Coordonnee& valPropreEps,Coordonnee& valPropreDeps,Coordonnee&
899         valPropreSig,
900         double Mises,TenseurBB& epsAlmBB,TenseurBB& epslogBB,int indic);
901
902     // -----
903
904     // --- méthodes relatives aux calculs d'erreurs -----
905     // ---- utilisées par les classes dérivées -----
906     // ajout des ddl relatif aux contraintes pour les noeuds de l'élément
907     void Ad_ddl_Sigma(const DdlElement& tab_ddlErr);
908     // inactive les ddl du problème primaire de mécanique
909     void Inact_ddl_primaire(DdlElement& tab_ddl);
910     // active les ddl du problème primaire de mécanique

```

```

911 void Act_ddl_primaire(DdlElement& tab_ddl);
912 // inactive les ddl du problème de recherche d'erreur : les contraintes
913 void Inact_ddl_Sigma(DdlElement& tab_ddlErr);
914 // active les ddl du problème de recherche d'erreur : les contraintes
915 void Act_ddl_Sigma(DdlElement& tab_ddlErr);
916 // active le premier ddl du problème de recherche d'erreur : SIGMA11
917 void Act_premier_ddl_Sigma();
918
919 // -----
920 // lecture des contraintes sur le flot d'entrée
921 void LectureDesContraintes
922     (bool cas,UtilLecture * entreePrinc,Tableau <TenseurHH *>& tabSigHH);
923
924 // retour des contraintes en absolu retour true si elle existe sinon false
925 void ContraintesEnAbsolues
926     (bool cas,Tableau <TenseurHH *>& tabSigHH,Tableau <Vecteur>& tabSig);
927
928
929 // ----- lecture écriture dans base info -----
930 // programmes utilisés par les classes dérivées
931
932 // cas donne le niveau de la récupération
933 // = 1 : on récupère tout
934 // = 2 : on récupère uniquement les données variables (supposées comme telles)
935 void Lecture_bas_inf
936     (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
937 // cas donne le niveau de sauvegarde
938 // = 1 : on sauvegarde tout
939 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
940 void Ecriture_bas_inf(ofstream& sort,const int cas) ;
941
942 // ----- utilitaires pour la dynamique
943 // calcul de la longueur d'arrête de l'élément minimal
944 // divisé par la célérité la plus rapide dans le matériau
945 // appelé par les classes dérivées
946 // nb_noeud : =0 indique que l'on utilise tous les noeuds du tableau de noeuds
947 //           = un nombre > 0, indique le nombre de noeuds à utiliser au début du tableau
948 double Interne_Long_arrete_mini_sur_c(Enum_dure temps,int nb_noeud=0);
949
950 // recuperation des coordonnées du point d'intégration numéro = iteg pour
951 // la grandeur enu
952 // temps: dit si c'est à 0 ou t ou tdt
953 // si erreur retourne erreur à true
954 Coordonnee CoordPtInt(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur);
955
956 // recuperation des coordonnées du point d'intégration numéro = iteg pour
957 // la face : face
958 // temps: dit si c'est à 0 ou t ou tdt
959 // si erreur retourne erreur à true
960 Coordonnee CoordPtIntFace(int face, Enum_dure temps,int iteg,bool& erreur);
961 // recuperation des coordonnées du point d'intégration numéro = iteg pour
962 // la face : face
963 // temps: dit si c'est à 0 ou t ou tdt
964 // si erreur retourne erreur à true
965 Coordonnee CoordPtIntArete(int arete, Enum_dure temps,int iteg,bool& erreur);
966
967 // procedure permettant de completer l'element, en ce qui concerne les variables gérés
968 // par ElemMeca, apres sa creation avec les donnees du bloc transmis
969 // peut etre appeler plusieurs fois
970 Element* Complete_ElemMeca(BlocGen & bloc,LesFonctions_nD* lesFonctionsND);
971
972 // retourne le numero du pt d'ing le plus près ou est exprimé la grandeur enum
973 // temps: dit si c'est à 0 ou t ou tdt
974 int PtLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M);
975
976 // récupération des valeurs au numéro d'ordre = iteg pour
977 // les grandeur enu
978 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
979 // NB Importante: il faut faire attention à ce que ces métriques soient identiques à celles qui
ont servit
980 // pour le calcul des tenseurs: en particulier si c'est utilisé pour calculer les grandeurs pour
le chargement
981 // il faut s'assurer que ce sont les "mêmes pti" qui servent pour la charge et pour la raideur !!
982 Tableau <double> Valeur_multi(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>& enu
983     ,int iteg,int cas
984     );
985
986 // récupération des valeurs Tensorielles (et non scalaire comme avec Valeur_multi)
987 // au numéro d'ordre = iteg pour les grandeur enu
988 // enu contient les grandeurs de retour
989 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
990 // NB Importante: il faut faire attention à ce que ces métriques soient identiques à celles qui
ont servit
991 // pour le calcul des tenseurs: en particulier si c'est utilisé pour calculer les grandeurs pour
le chargement

```

```

992 // il faut s'assurer que ce sont les "mêmes pti" qui servent pour la charge et pour la raideur !!
993 void Valeurs_Tensorielles(bool absolue, Enum_dure enu_t,List_io<TypeQuelconque>& enu
994 ,int iteg,int cas
995 ) ;
996 // récupération de valeurs interpolées pour les grandeur enu ou directement calculées
997 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
998 // une seule des 3 métriques doit-être renseigné, les autres doivent être un pointeur nul
999 Tableau <double> Valeur_multi_interpoler_ou_calculer
1000 (bool absolue, Enum_dure temps,const List_io<Ddl_enum_etendu>& enu
1001 ,Deformation & defor
1002 ,const Met_abstraite::Impli* ex_impli
1003 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
1004 ,const Met_abstraite::Expli* ex_expli
1005 );
1006 // récupération de valeurs interpolées pour les grandeur enu
1007 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
1008 // une seule des 3 métriques doit-être renseigné, les autres doivent être un pointeur nul
1009 void Valeurs_Tensorielles_interpoler_ou_calculer
1010 (bool absolue, Enum_dure temps,List_io<TypeQuelconque>& enu
1011 ,Deformation & defor
1012 ,const Met_abstraite::Impli* ex_impli
1013 ,const Met_abstraite::Expli_t_tdt* ex_expli_tdt
1014 ,const Met_abstraite::Expli* ex_expli
1015 );
1016
1017 // ==== »» methodes virtuelles definis dans les classes mères =====
1018 // ramene l'element geometrique correspondant au ddl passé en paramètre
1019 virtual ElemGeomCO& ElementGeometrie(Enum_ddl ddl) const ;
1020 // ramène le nombre de grandeurs génératrices pour un pt d'integ, correspondant à un type
enuméré
1021 // peut-être surchargé pour des éléments particuliers
1022 virtual int NbGrandeurGene(Enum_ddl enu) const;
1023
1024 // ==== »» methodes virtuelles definis dans les classes dérivées =====
1025 // ramene la dimension des tenseurs contraintes et déformations de l'élément
1026 virtual int Dim_sig_eps() const = 0;
1027
1028 // VARIABLES PROTEGEES :
1029
1030 //---- variables: contrainte, déformations etc.. aux points d'intégrations -----
1031 // lesPtIntegMecaInterne pointe vers une entité entièrement gérés par les classes dérivées
1032 // contenant les grandeurs mécaniques (contraintes, déformations etc.) stockées aux points
d'intégration
1033 LesPtIntegMecaInterne * lesPtIntegMecaInterne;
1034
1035 //----- les charges externes éventuelles -----
1036 LesChargeExtSurElement * lesChargeExtSurEle;
1037
1038
1039 Loi_comp_abstraite * loiComp; // loi de comportement mécanique defini dans les classes
derivees
1040 CompThermoPhysiqueAbstraite * loiTP; // éventuellement une loi de comportement thermo
physique
1041 // defini dans les classes derivees
1042 CompFrotAbstraite * loiFrot; // éventuellement une loi de comportement au frottement pour
ses frontières
1043
1044 int dilatation; // indique si oui ou non on tiend compte de la dilatation thermique
1045 Met_abstraite * met; // definition spécifique dans les classes derivees
1046 Deformation * def; // definition spécifique dans les classes derivees
1047 // mais relative à la déformation mécanique
1048 Deformation * defEr; // idem que def mais pour la remonte aux contraintes
1049 Tableau <Deformation *> defSurf; // idem mais pour les déformations des surfaces
1050 // externes (frontières) si cela est nécessaire
1051 Tableau <Deformation *> defArete; // idem mais pour les déformations des arretes
1052 // externes (frontières) si cela est nécessaire
1053 Deformation * defMas; // idem que def mais pour le calcul de la matrice masse
1054 double* sigErreur; // erreur sur l'élément dans le calcul des contraintes;
1055 double* sigErreur_relative; // idem en valeur relative
1056 Tableau <Loi_comp_abstraite::SaveResul *> tabSaveDon; // donnée particulière à la loi mécanique
1057 Tableau <CompThermoPhysiqueAbstraite::SaveResul *> tabSaveTP; // donnée parti à la loi thermo
physique
1058 Tableau <Deformation::SaveDefResul *> tabSaveDefDon; // donnée particulière pour la déf
1059 Tableau <EnergieMeca > tab_energ,tab_energ_t; //les différentes énergies mécaniques mises en
jeux
1060 // dimensionné dans les classes dérivées, a t+dt, et a t
1061 EnergieMeca energie_totale_t,energie_totale; // le bilan sur l'élément de l'énergie
1062 bool premier_calcul_meca_impli_expli; // au premier calcul soit en implicite, explicite, mat
geom
1063 // toutes les grandeurs à t=0 de la métrique sont calculées et stockées dans la déformation
1064 // ensuite elles ne sont plus calculées -> premier_calcul_meca_impli_expli sert pour
l'indiquer
1065 double masse_volumique; // masse volumique de l'élément
1066 //--- stabilisation d'hourglass éventuelle
1067 Enum_StabHourglass type_stabHourglass; // méthode de stabilisation

```

```

1068     double E_Hourglass;
1069
1070 private:
1071     //--- stabilisation d'hourglass éventuelle
1072     Tableau <ElemMeca*> tab_elHourglass; // tableau de pointeurs éventuels sur des éléments
servant à la stabilisation
1073     // les éléments sont définies au travers de l'appel à la méthode
Cal_hourglass_comp, par une classe dérivée,
1074     double coefStabHourglass; // coef de stabilisation
1075     Mat_pleine* raid_hourglass_transitoire; // raideur transitoire d'hourglass, éventuellement
définie
1076     static Tableau< Vecteur>     vec_hourglass; // tableau de vecteurs de travail, éventuellement
défini
1077
1078
1079     // ----- pour faciliter les routines Interne_t _0 et _tdt + paramètres de contrôle -----
1080     static const Coordonnee & (Met_abstraite::*PointM)
1081     (const Tableau<Noeud *>& tab_noeud, const Vecteur& phi);
1082     static void (Met_abstraite::*BaseND)
1083     (const Tableau<Noeud *>& tab_noeud,
1084      const Mat_pleine& dphi, const Vecteur& phi, BaseB& bB, BaseH& bH);
1085
1086     // ----- cas du bulk viscosity -----
1087 private: // pour éviter les modifications par les classes dérivées
1088     static int bulk_viscosity; // indique si oui ou non on utilise le bulk viscosity
1089     // en plus = 1: fonctionnement normal, =2 fonctionnement quelque soit
1090
1091     // le signe de IDeps
1092     static double c_traceBulk; // coeff de la trace de D pour le bulk
1093     static double c_carreBulk; // coeff du carré de la trace de D pour le bulk
1094     static TenseurHH * sig_bulkHH; // variable de travail pour le bulk
1095     // ----- pour le calcul de la masse pour la méthode de relaxation dynamique
1096     static Ddl_enum_etendu masse_relax_dyn; // simplement pour le définir une fois pour toute
1097
1098 protected:
1099     // pour une utilisation par les classes dérivées
1100     inline bool Bulk_visco_actif() {return bulk_viscosity;};
1101     inline double& C_traceBulk() {return c_traceBulk;};
1102     inline double& C_carreBulk() {return c_carreBulk;};
1103     // calcul de la contrainte modifiée en fonction du bulk viscosity
1104     // ramène l'énergie et la puissance par unité de volume, dépensée par le bulk
1105     DeuxDoubles ModifContrainteBulk(Enum_dure temps, const TenseurHH& gijHH_tdt, TenseurHH &
sigHH, const TenseurBB & DepsBB_tdt
1106     , TenseurHH & sigbulkHH);
1107     // actualisation des ddl et des grandeurs actives de t+dt vers t
1108     void TdtversT();
1109     // actualisation des ddl et des grandeurs actives de t vers tdt
1110     void TversTdt();
1111     // calcul des intégrales de volume et de volume + temps
1112     // cas = 1 : pour un appel après un calcul implicite
1113     // cas = 2 : pour un appel après un calcul explicite
1114     void Calcul_Integ_vol_et_temps(int cas, const double& poid_jacobien, int iteg);
1115     // au premier pti init éventuel des intégrales de volume et temps
1116     void Init_Integ_vol_et_temps();
1117
1118     // ---- utilitaires interne
1119 private:
1120     // calcul du mini du module d'young équivalent pour tous les points d'intégrations
1121     double Calcul_mini_E_qui(Enum_dure temps);
1122     // calcul du maxi du module d'young équivalent pour tous les points d'intégrations
1123     double Calcul_maxi_E_qui(Enum_dure temps);
1124     // définition d'un repère d'anisotropie à un point d'intégration
1125     BaseH DefRepereAnisotropie(int nb_pti, LesFonctions_nD* lesFonctionsnD, const BlocGen & bloc);
1126 };
1127 /// @} // end of group
1128
1129
1130
1131 #endif

```

## 7.149 ElemPoint.h

```

1 // FICHER : ElemPoint.h
2 // CLASSE : ElemPoint
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //

```



```

13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 * UNIVERSITE DE BRETAGNE SUD (UBS) --- I.U.P/I.U.T. DE LORIENT *
34 *****/
35 * LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M) *
36 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
37 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
38 *****/
39 * DATE: 24/02/2005 *
40 * * $ *
41 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
42 * Tel 0297874571 fax : 02.97.87.45.72 *
43 * * $ *
44 * PROJET: Herezh++ *
45 * * $ *
46 *****/
47 * BUT: La classe ElemPoint permet de declarer des elements *
48 * ElemPoint. *
49 * L'élément comporte un noeud qui est identique au point d'inté- *
50 * gration. *
51 * *
52 * ***** *
53 * VERIFICATION: *
54 * *
55 * ! date ! auteur ! but ! *
56 * ----- ! *
57 * ! ! ! ! *
58 * * $ *
59 * ***** *
60 * MODIFICATIONS: *
61 * ! date ! auteur ! but ! *
62 * ----- ! *
63 * * $ *
64 *****/
65
66
67 #ifndef ELEM_POINT_H
68 #define ELEM_POINT_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 // #include "Loi_comp_abstraite.h"
73 #include "Met_abstraite.h"
74 #include "Met_ElemPoint.h"
75 #include "Noeud.h"
76 #include "UtilLecture.h"
77 #include "Tenseur.h"
78 #include "NevezTenseur.h"
79 #include "Deformation.h"
80 #include "ElFrontiere.h"
81 #include "GeomPoint.h"
82 #include "ParaAlgoContrôle.h"
83 #include "UmatAbaqus.h"
84
85 //class ConstrucElemPoint;
86
87 class ElemPoint : public ElemMeca
88 {
89
90 public :
91
92 // CONSTRUCTEURS :
93 // les tenseurs on par défaut (dimension==1) la dimension de l'espace
94 // sinon il ont la dimension donnée
95 // Constructeur par défaut
96 ElemPoint (int dimension=-1);
97 // Constructeur fonction d'un numero de maillage et d'identification
98 ElemPoint (int num_mail,int num_id,int dimension=-1);

```



```

99
100
101     // Constructeur de copie
102     ElemPoint (const ElemPoint& elem);
103
104
105     // DESTRUCTEUR :
106     virtual ~ElemPoint ();
107
108     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
109     // méthode virtuelle
110     Element* Nevez_copie() const { Element * el= new ElemPoint(*this); return el;};
111
112     // Surcharge de l'operateur = : realise l'egalite entre deux instances de ElemPoint
113     ElemPoint& operator= (ElemPoint& biel);
114
115     // METHODES :
116 // 1) derivant des virtuelles pures
117     // Lecture des donnees de la classe sur fichier
118     void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
119
120     // Calcul du residu local et de la raideur locale,
121     // pour le schema implicite
122     Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
123
124     // Calcul du residu local a t
125     // pour le schema explicite par exemple
126     Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
127     { return ElemPoint::CalculResidu(false,pa);};
128
129     // Calcul du residu local a tdt
130     // pour le schema explicite par exemple
131     Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
132     { return ElemPoint::CalculResidu(true,pa);};
133
134 // Calcul de la matrice masse pour l'élément
135 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
136
137 // ----- calcul dynamique -----
138 // calcul de la longueur d'arrête de l'élément minimal
139 // divisé par la célérité la plus rapide dans le matériau
140 double Long_arrete_mini_sur_c(Enum_dure )
141 { return 0.;};
142
143 //----- calcul d'erreur, remontée des contraintes -----
144 // 1)calcul du résidu et de la matrice de raideur pour le calcul d'erreur
145 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
146 // 2) remontée aux erreurs aux noeuds
147 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
148
149 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
150 // ce tableau est spécifique a l'element
151 const DdlElement & TableauDdl() const ;
152
153 // Libere la place occupee par le residu et eventuellement la raideur
154 // par l'appel de Libere de la classe mere et libere les differents tenseurs
155 // intermediaires cree pour le calcul et les grandeurs pointee
156 // de la raideur et du residu
157 void Libere ();
158
159 // acquisition d'une loi de comportement
160 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
161
162 // test si l'element est complet
163 // = 1 tout est ok, =0 element incomplet
164 int TestComplet();
165
166 // procedure permettant de completer l'element apres
167 // sa creation avec les donnees du bloc transmis
168 // peut etre appele plusieurs fois
169 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
170 // Compléter pour la mise en place de la gestion de l'hourglass
171 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc) {return
this;};
172
173 // ramene l'element geometrique
174 ElemGeomC0& ElementGeometrique() const { return unefois->doCoMemb->ptpoint;};
175 // ramene l'element geometrique en constant
176 const ElemGeomC0& ElementGeometrique_const() const { return unefois->doCoMemb->ptpoint;};
177
178 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
179 // associé
180 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
181 // 1) cas où l'on utilise la place passée en argument
182 Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
183 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
184 void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);

```

```

184
185 // affichage dans la sortie transmise, des variables duales "nom"
186 // dans le cas ou nom est vide, affichage de "toute" les variables
187 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
188
189 // affichage d'info en fonction de ordre
190 // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
191 void Info_com_Element(UtilLecture * entreePrinc,string& ordre,Tableau<Noeud * > *
tabMaillageNoeud)
192 { return Element::Info_com_El(2,entreePrinc,ordre,tabMaillageNoeud);};
193
194 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
195 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
196 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
197 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
198 // temps: dit si c'est à 0 ou t ou tdt
199 int PointLePlusPres(Enum_dure ,Enum_ddl , const Coordonnee& )
200 { return 1;}; // ici c'est uniquement le point de l'élément
201
202 // recuperation des coordonnées du point de numéro d'ordre iteg pour
203 // la grandeur enu
204 // temps: dit si c'est à 0 ou t ou tdt
205 // si erreur retourne erreur à true
206 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur);
207
208 // récupération des valeurs au numéro d'ordre = iteg pour
209 // les grandeur enu
210 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
211 Tableau <double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
enu,int iteg) ;
212
213 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
214 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
215 // de conteneurs quelconque associée
216 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
217 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int iteg);
218
219 // ramene vrai si la surface numéro ns existe pour l'élément
220 // dans le cas de la ElemPoint il n'y a pas de surface
221 bool SurfExiste(int ) const
222 { return false;};
223
224 // ramene vrai si l'arête numéro na existe pour l'élément
225 // dans le cas de la ElemPoint il n'y a pas d'arête
226 bool AreteExiste(int ) const
227 { return false;};
228 //===== spécifique à l'interface abaqus =====
229 // on associe un noeud à l'élément, remplace le noeud existant, s'il existe déjà
230 virtual void Associer_noeud (Noeud * noeu);
231
232 // une classe de travail spécifiques
233 class inNeNpti
234 { public: int* incre; // numéro d'incrément
235 int* step; // numéro du step
236 int* nbe; // numéro d'élément
237 int* nbpti; // numéro du point d'intégration
238 double* temps_tdt; // temps courant
239 double* delta_t; // incrément courant de temps
240 string* nom_loi; // nom de la loi
241 };
242
243 // lecture des grandeurs umat transmises par abaqus sur le pipe
244 static const ElemPoint::inNeNpti& Lecture_Abaqus(bool utilisation_umat_interne)
245 {unefois_Point.doCoMemb->umatAbaqus.LectureDonneesUmat
246 (utilisation_umat_interne,ParaGlob::NiveauImpression());return unefois_Point.doCoMemb->inne;};
247 // récup du inNeNpti en cours
248 static const ElemPoint::inNeNpti& IncreeElemPtint_encours() {return unefois_Point.doCoMemb->inne;};
249 // écriture du résultat sur le pipe
250 void Ecriture_Abaqus(bool utilisation_umat_interne)
251 {unefois->doCoMemb->umatAbaqus.EcritureDonneesUmat
252 (utilisation_umat_interne,ParaGlob::NiveauImpression());};
253 // calcul de l'UMat pour abaqus
254 void CalculUmatAbaqus(ParaAlgoControle & pa);
255 // initialisation éventuelle: ajout de point d'intégration si nécessaire
256 // les tenseurs ont la dimension 3
257 virtual void InitialisationUmatAbaqus()
258 {InitialisationUmatAbaqus_interne(3);};
259 // récup du nombre de fois où l'élément est appelé depuis le début
260 // de l'incrément. Correspond environ au nombre d'itération
261 int NbIteration() const {return nb_appelsCalculUmat;};
262
263 //===== lecture écriture dans base info =====
264
265 // cas donne le niveau de la récupération
266 // = 1 : on récupère tout
267 // = 2 : on récupère uniquement les données variables (supposées comme telles)
268 void Lecture_base_info

```

```

269     (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
270     // cas donne le niveau de sauvegarde
271     // = 1 : on sauvegarde tout
272     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
273     void Ecriture_base_info(ofstream& sort,const int cas) ;
274
275     // METHODES VIRTUELLES:
276     // ----- calculs utiles dans le cadre de la recherche du flambement linéaire
277     // Calcul de la matrice géométrique et initiale
278     ElemMeca::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
279
280     // inactive les ddl du problème primaire de mécanique
281     inline void Inactive_ddl_primaire()
282     {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};
283     // active les ddl du problème primaire de mécanique
284     inline void Active_ddl_primaire()
285     {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl);};
286     // ajout des ddl de contraintes pour les noeuds de l'élément
287     inline void Plus_ddl_Sigma()
288     {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
289     // inactive les ddl du problème de recherche d'erreur : les contraintes
290     inline void Inactive_ddl_Sigma()
291     {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
292     // active les ddl du problème de recherche d'erreur : les contraintes
293     inline void Active_ddl_Sigma()
294     {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
295     // active le premier ddl du problème de recherche d'erreur : SIGMA11
296     inline void Active_premier_ddl_Sigma()
297     {ElemMeca::Act_premier_ddl_Sigma();};
298
299     // lecture de données diverses sur le flot d'entrée
300     void LectureContraintes(UtilLecture * entreePrinc);
301
302     // retour des contraintes en absolu retour true si elle existe sinon false
303     bool ContraintesAbsolues(Tableau <Vecteur>& tabSig);
304
305
306     // 2) derivant des virtuelles
307     // retourne un tableau de ddl element, correspondant à la
308     // composante de sigma -> SIG11, pour chaque noeud qui contient
309     // des ddl de contrainte
310     // -> utilisé pour l'assemblage de la raideur d'erreur
311     inline DdlElement& Tableau_de_Sig1() const
312     {return unefois->doCoMemb->tab_Err1Sig11;};
313
314     // actualisation des ddl et des grandeurs actives de t+dt vers t
315     void TdtversT();
316     // actualisation des ddl et des grandeurs actives de t vers tdt
317     void TversTdt();
318
319     // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
320     // qu'une fois la remontée aux contraintes effectuées sinon aucune
321     // action. En retour la valeur de l'erreur sur l'élément
322     // type indique le type de calcul d'erreur :
323     void ErreurElement(int type,double& errElemRelative
324     ,double& numerateur, double& denominateur);
325
326     // ===== définition et/ou construction des frontières =====
327
328     // Calcul des frontieres de l'element
329     // creation des elements frontieres et retour du tableau de ces elements
330     // la création n'a lieu qu'au premier appel
331     // ou lorsque l'on force le paramètre force a true
332     // dans ce dernier cas seul les frontière effacées sont recréée
333     Tableau <ElFrontiere*> const & Frontiere(bool force = false);
334
335     // ramène la frontière point
336     // éventuellement création des frontieres points de l'element et stockage dans l'element
337     // si c'est la première fois sinon il y a seulement retour de l'elements
338     // a moins que le paramètre force est mis a true
339     // dans ce dernier cas la frontière effacée est recréée
340     // num indique le numéro du point à créer (numérotation EF)
341     // ElFrontiere* const Frontiere_points(int num,bool force = false);
342
343     // ramène la frontière linéique
344     // éventuellement création des frontieres linéique de l'element et stockage dans l'element
345     // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
346     // a moins que le paramètre force est mis a true
347     // dans ce dernier cas la frontière effacée est recréée
348     // num indique le numéro de l'arête à créer (numérotation EF)
349     // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
350
351     // ramène la frontière surfacique
352     // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
353     // si c'est la première fois sinon il y a seulement retour de l'elements
354     // a moins que le paramètre force est mis a true
355     // dans ce dernier cas la frontière effacée est recréée

```

```

356         // num indique le numéro de la surface à créer (numérotation EF)
357         // ici normalement la fonction ne doit pas être appelée
358 //      ElFrontiere* const Frontiere_surfacique(int ,bool force = false);
359
360 // 3) methodes propres a l'element
361
362 // ajout du tableau specific de ddl des noeuds de la ElemPoint
363 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
364 // des noeuds constitutants l'element
365 void ConstTabDdl();
366
367 protected:
368
369         // ==== »» methodes virtuelles dérivant d'ElemMeca =====
370         // ramene la dimension des tenseurs contraintes et déformations de l'élément
371         int Dim_sig_eps() const {return 1;};
372
373 // ----- calcul de frontieres en protected -----
374
375 // --- fonction nécessaire pour la construction des Frontieres linéiques ou surfaciques particulière
376 // à l'élément
377 // adressage des frontieres linéiques et surfacique
378 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
379 virtual ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
380 { return NULL;}; // il n'y a pas de ligne possible
381 virtual ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
382 {return NULL;}; // il n'y a pas de surface possible
383
384 public:
385
386 class DonneeCommune
387 { public :
388     DonneeCommune (GeomPoint& pteg,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
389     Met_ElemPoint& met_point,
390     Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
391     GeomPoint& pteEr,Vecteur& residu_int,Mat_pleine& raideur_int,
392     Tableau <Vecteur* > & residu_extN,Tableau <Mat_pleine* >& raideurs_extN,
393     Mat_pleine& mat_masse ,GeomPoint& pteMa,UmatAbaqus& umatAbaqus
394     ,int dimension ,int nbi );
395     DonneeCommune(DonneeCommune& a);
396     ~DonneeCommune();
397     // variables
398     GeomPoint ptpoint ; // element geometrique correspondant
399     DdlElement tab_ddl; // tableau des degres
400         //de liberte des noeuds de l'element commun a tous les
401         // elements
402     Met_ElemPoint met_ElemPoint;
403     Mat_pleine matGeom ; // matrice géométrique
404     Mat_pleine matInit ; // matrice initiale
405     Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
406     Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
407     Tableau < Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
408         // calcul d'erreur
409     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
410         // d'erreur : contraintes
411     DdlElement tab_Err1Sig1; // tableau du ddl SIG11 pour chaque noeud,
412         //servant pour le calcul d'erreur : contraintes, en fait pour l'assemblage
413     Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
414     Mat_pleine raidErr; // raideur pour le calcul d'erreur
415     GeomPoint pteEr; // contient les fonctions d'interpolation et
416         // les derivees pour le calcul du hessien dans
417         //la résolution de la fonctionnelle d'erreur
418         // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
419 -----
420         // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
421     Vecteur residu_interne;
422     Mat_pleine raideur_interne;
423     Tableau <Vecteur* > residu_externeN; // pour les noeuds
424     Tableau <Mat_pleine* > raideurs_externeN; // pour les noeuds
425     // ----- données concernant la dynamique -----
426     Mat_pleine matrice_masse;
427     GeomPoint pteMas; // contient les fonctions d'interpolation et les dérivées
428         // pour les calculs relatifs au calcul de la masse
429     // -- particularités pour la routine Umat pour Abaqus
430     UmatAbaqus umatAbaqus;
431     inNeNpti inne; // indices pointant directement sur des données de umatAbaqus
432 };
433
434 // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
435 // et un pointeur sur les données statiques communes
436 // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
437 // classe est défini. Son allocation est effectuée dans les classes dérivées
438 class UneFois
439 { public :
440     UneFois () ; // constructeur par défaut

```

```

441         ~UneFois () ; // destructeur
442
443         // VARIABLES :
444         public :
445         DonneeCommune * doCoMemb;
446
447         // incicateurs permettant de dimensionner seulement au premier passage
448         // utilise dans "CalculResidu" et "Calcul_implicit"
449         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
450         int CalimpPrem;
451         int dualSortbiel; // pour la sortie des valeurs au pt d'integ
452         int CalSmlin_t; // pour les seconds membres concernant les arretes
453         int CalSmlin_tdt; // pour les seconds membres concernant les arretes
454         int CalSMRlin; // pour les seconds membres concernant les arretes
455         int CalDynamique; // pour le calcul de la matrice de masse
456         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
457         // ----- sauvegarde du nombre d'élément en cours -----
458         int nbelem_in_Prog;
459     };
460
461     // -----
462
463     protected :
464         // VARIABLES PROTÉGÉES :
465         // pour minimiser la place on définit une classe des contraintes et déformations locale
466         // dans le cas de l'umat, l'instance de la classe n'est pas affectée
467
468         // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
469         LesPtIntegMecaInterne lesPtMecaInt;
470
471
472         // le nombre de point d'intégration pour le calcul mécanique
473         int nbi; // ce nombre peut évoluer pendant le calcul, il n'est donc pas
474         // commun à tous les éléments
475         //-- cas de l'utilisation Umat
476         int nb_appelsCalculUmat; // stocke le nombre de fois où l'élément est appelé depuis le début
477         // de l'incrément. Correspond environ au nombre d'itération
478
479         // place memoire commune a tous les elements ElemPoints
480         static DonneeCommune * doCo_Point;
481
482         // idem mais pour les indicateurs qui servent pour l'initialisation
483         static UneFois unefois_Point;
484
485         // on utilise une variable intermédiaire car on a des classes dérivées
486         // du coup pour différencier les données communes, on utilise une variable spécifique
487         // qui donc ne peut pas être static
488         UneFois * unefois;
489
490         // type structuré pour construire les éléments
491         class NombresConstruire
492         { public:
493             NombresConstruire();
494             int nbne; // le nombre de noeud de l'élément
495             int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
496             int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse consistante
497         };
498         static NombresConstruire nombre_V; // les nombres propres à l'élément
499
500         // --- fonctions protégées
501
502         // fonction d'initialisation servant au niveau du constructeur
503         // dim_tenseur: = la dimension des tenseurs
504         void Init (int dim_tenseur);
505         // construction de données communes: correspond à un new
506         DonneeCommune* Def_DonneeCommune(int dim_tenseur);
507         // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
508         void Destruction();
509         // changement du nombre de point d'intégration
510         void ChangeNombrePtinteg(int nevez_nbi, int dim_tens);
511         // pour l'ajout d'element dans la liste : listTypeElement, geree par la class Element
512         class ConstrucElemPoint : public ConstrucElement
513         { public : ConstrucElemPoint ()
514             { NouvelleTypeElement nouv (POINT,CONSTANT,MECA_SOLIDE_DEFORMABLE,this);
515               if (ParaGlob::NiveauImpression() >= 4)
516                 cout << "\n initialisation ElemPoint" << endl;
517               Element::listTypeElement.push_back(nouv);
518             };
519             Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
520             {Element * pt;
521               pt = new ElemPoint (num_maill,num) ;
522               return pt;};
523             // ramene true si la construction de l'element est possible en fonction
524             // des variables globales actuelles: ex en fonction de la dimension
525             bool Element_possible() {return true;};
526         };
527         static ConstrucElemPoint construcElemPoint;

```

```

528 // Calcul du residu local a t ou tdt en fonction du boolean
529 Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
530
531 //----- pour des classes dérivées -----
532 // CONSTRUCTEURS :
533 // les tenseurs on par défaut (dimension==1) la dimension de l'espace
534 // sinon il ont la dimension donnée
535 ElemPoint (ElemPoint::UneFois& unefois, Enum_geom nouveau_id, int dimension=-1);
536 // Constructeur fonction d'un numero de maillage et d'identification
537 ElemPoint (ElemPoint::UneFois& unefois, Enum_geom nouveau_id, int num_mail,int num_id,int
    dimension=-1);
538
539 // initialisation éventuelle: ajout de point d'intégration si nécessaire
540 // les tenseurs on par défaut (dimension==1) la dimension de l'espace
541 // utilisation par Point et les classes dérivées
542 void InitialisationUmatAbaqus_interne(int dimension = -1);
543
544
545 };
546 #endif
547
548
549
550

```

## 7.150 ElemPoint\_CP.h

```

1 // FICHER : ElemPoint.h
2 // CLASSE : ElemPoint
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          24/02/2005
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *      *****/
40 *      BUT: La classe ElemPoint_CP permet de declarer des elements
41 *      ElemPoint associé à des contraintes plane.
42 *      L'élément comporte un noeud qui est identique au point d'inté-
43 *      gration.
44 *
45 *      *****/
46 *
47 *      VERIFICATION:
48 *      ! date ! auteur ! but
49 *      -----
50 *      ! ! !
51 *
52 *      *****/
53 *      MODIFICATIONS:
54 *      ! date ! auteur ! but
55 *      -----
56 *
57 *****/

```

```

58
59
60 #ifndef ELEM_POINT_CP_H
61 #define ELEM_POINT_CP_H
62
63 #include "ElemPoint.h"
64
65
66 /// @addtogroup groupe_des_elements_finis
67 /// @{
68 ///
69
70
71 class ElemPoint_CP : public ElemPoint
72 {
73
74     public :
75
76         // CONSTRUCTEURS :
77         // les tenseurs on par défaut la dimension 2
78         // Constructeur par défaut
79         ElemPoint_CP () : ElemPoint(unefois_CP,POINT_CP,2)
80     { };
81         // Constructeur fonction d'un numero de maillage et d'identification
82         ElemPoint_CP (int num_mail,int num_id):
83         ElemPoint(unefois_CP,POINT_CP,num_mail,num_id,2)
84     { };
85
86         // Constructeur de copie
87         ElemPoint_CP (const ElemPoint_CP& elem):
88         ElemPoint(elem)
89     { };
90
91
92         // DESTRUCTEUR :
93
94         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
95         // méthode virtuelle
96         Element* Nevez_copie() const { Element * el= new ElemPoint_CP(*this); return el;};
97
98         // Surcharge de l'opérateur = : realise l'egalite entre deux instances de ElemPoint_CP
99         ElemPoint_CP& operator= (ElemPoint_CP& biel);
100
101         // initialisation éventuelle: ajout de point d'intégration si nécessaire
102         // les tenseurs ont la dimension 2
103         virtual void InitialisationUmatAbaqus()
104         {InitialisationUmatAbaqus_interne(2);};
105
106         // lecture des grandeurs umat transmises par abaqus sur le pipe
107         static const ElemPoint_CP::inNeNpti& Lecture_Abaqus(bool utilisation_umat_interne)
108         {unefois_CP.doCoMemb->umatAbaqus.LectureDonneesUmat
109         (utilisation_umat_interne,ParaGlob::NiveauImpression());return unefois_CP.doCoMemb->inne;};
110         // récup du inNeNpti en cours
111         static const ElemPoint_CP::inNeNpti& IncreElemPtint_encours() {return unefois_CP.doCoMemb->inne;};
112
113         // 3) methodes propres a l'element
114
115     protected:
116
117         // ==== »» methodes virtuelles dérivant d'ElemMeca =====
118         // ramene la dimension des tenseurs contraintes et déformations de l'élément
119         int Dim_sig_eps() const {return 2;};
120
121         // ----- calcul de frontières en protected -----
122
123     private :
124
125         // VARIABLES PRIVEES :
126
127
128         // -----
129
130     protected :
131         // VARIABLES PROTÉGÉES :
132
133         // place memoire commune a tous les elements ElemPoint_CP
134         static ElemPoint::DonneeCommune * doCo_CP;
135         // idem mais pour les indicateurs qui servent pour l'initialisation
136         static ElemPoint::UneFois unefois_CP;
137
138         // les nombres propres à l'élément: idem ceux de Point
139         // static NombresConstruire nombre_V;
140
141         // pour l'ajout d'element dans la liste : listTypeElemen, geree par la class Element
142         class ConstrucElemPoint_CP : public ConstrucElement
143         { public : ConstrucElemPoint_CP ()
144         { NouvelleTypeElement nouv(POINT_CP,CONSTANT,MECA_SOLIDE_DEFORMABLE,this);

```

```

145         if (ParaGlob::NiveauImpression() >= 4)
146             cout << "\n initialisation ElemPoint_CP " << endl;
147         Element::listTypeElement.push_back(nouv);
148     };
149     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
150     {Element * pt;
151       pt = new ElemPoint_CP (num_maill,num) ;
152       return pt;};
153     // ramene true si la construction de l'element est possible en fonction
154     // des variables globales actuelles: ex en fonction de la dimension
155     bool Element_possible() {return true;};
156 };
157 static ConstrucElemPoint_CP construcElemPoint_CP;
158
159
160 };
161 #endif
162
163
164
165

```

## 7.151 Met\_ElemPoint.h

```

1 // FICHER : Met_ElemPoint.h
2 // CLASSE : Met_ElemPoint
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          24/02/2005
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *****/
40 *      BUT: Métrique pour le cas particulier des éléments comportants
41 *          un seul noeud, la dimension est 1 ou 2 ou 3.
42 *      Ici, par défaut la métrique est l'identité de l'espace, sauf si
43 *      on impose une métrique différente, qui peut alors être de
44 *      dimension quelconque.
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *      ! date ! auteur ! but
50 *      -----
51 *      ! ! !
52 *      $
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *      $
58 *****/
59
60

```



```

61
62 #ifndef MET_ELEM_POINT_H
63 #define MET_ELEM_POINT_H
64
65 #include "Met_abstraite.h"
66 #include "UmatAbaqus.h"
67
68 /// @addtogroup groupe_des_metrique
69 /// @{
70 ///
71
72
73
74 class Met_ElemPoint : public Met_abstraite
75 {
76
77
78     public :
79
80
81         // CONSTRUCTEUR :
82
83         // Constructeur par default
84         Met_ElemPoint ();
85         // constructeur permettant de dimensionner uniquement certaine variables
86         // dim = dimension de l'espace, tab = liste
87         // des variables a initialiser
88         Met_ElemPoint (int dim_base,int nbvec,const DdlElement& tabddl,
89             const Tableau<Enum_variable_metrique> & tab,int nomb_noeud);
90         // constructeur spécifique pour un point avec une métrique imposée
91         // ici on ne donne pas le nombre de noeud ce qui permet de faire la différence
92         // avec le constructeur précédent
93         Met_ElemPoint (int dim_base,int nbvec,const DdlElement& tabddl,
94             const Tableau<Enum_variable_metrique>& tab);
95         // constructeur de copie
96         Met_ElemPoint (const Met_ElemPoint&);
97         // DESTRUCTEUR :
98
99         ~Met_ElemPoint ();
100        // Surcharge de l'operateur = : realise l'affectation
101        // fonction virtuelle
102        inline Met_abstraite& operator= (const Met_abstraite& met)
103            { return (Met_abstraite::operator=(met)); };
104
105        // ----- spécifiques aux opérations Umat -----
106        // calcul des grandeurs spécifiques aux opérations Umat
107        const Umat_cont& Calcul_grandeurs_Umat(bool premier_calcul);
108        // mise à jour de grandeur via les infos d'umat
109        // il s'agit des grandeurs suivantes: giB à 0 t et tdt,
110        // les routines types Calcul_giB_t (etc..) ne calculent donc plus rien, ici
111        void Mise_a_jour(const UmatAbaqus& umat);
112        // ----- fin spécifique aux opérations Umat -----
113
114
115        protected :
116        // indique si l'on a affaire à une métrique imposée ou non
117        bool metrique_imposee;
118        // METHODES protegees:
119        //==calcul des points, identique a Met_abstraite
120
121
122        // --- calcul de la base naturel et dual ---
123        // dans tous les cas on ne fait rien, mais cela permet de surcharger le calcul de Met_abstraite
124        // si metrique_imposee est vrai, dans ce cas les gi sont imposés
125        // si metrique_imposee est faux, dans ce cas les gi sont unitaires et orthonormées
126
127        // calcul de la base naturel a t0
128        void Calcul_giB_0(const Tableau<Noeud *>& ,const Mat_pleine& , int ,const Vecteur& ) {};
129        // calcul de la base naturel a t
130        void Calcul_giB_t(const Tableau<Noeud *>& ,const Mat_pleine& , int ,const Vecteur& ) {};
131        // calcul de la base naturel a t+dt
132        void Calcul_giB_tdt(const Tableau<Noeud *>& ,const Mat_pleine& , int ,const Vecteur& ) {};
133        // Calcul des composantes covariantes de la metrique de la base naturelle a 0 :
134        // virtual void Calcul_gijBB_0() {};
135        // Calcul des composantes covariantes de la metrique de la base naturelle a t :
136        // virtual void Calcul_gijBB_t() {};
137        // Calcul des composantes covariantes de la metrique de la base naturelle a tdt :
138        // virtual void Calcul_gijBB_tdt() {};
139        // Calcul des composantes contravariantes de la metrique de la base naturelle a 0 :
140        // virtual void Calcul_gijHH_0() {};
141        // Calcul des composantes contravariantes de la metrique de la base naturelle a t :
142        // virtual void Calcul_gijHH_t() {};
143        // Calcul des composantes contravariantes de la metrique de la base naturelle a tdt :
144        // virtual void Calcul_gijHH_tdt() {};
145
146 };
147 /// @} // end of group

```

```

148
149
150 #endif
151
152

```

## 7.152 UmatAbaqus.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      25/02/2005
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *
38 *      BUT:      échange de structures de données: fortran Umat -> C++
39 *              La classe est principalement de type conteneur avec
40 *              accès directe aux informations.
41 *      NB: on reste systématiquement en 3D, car cela permet de stocker
42 *          toutes les informations: par exemple en CP, il nous faut
43 *          epsilon33, et en CP2, il nous faudrait eps33 et 22
44 *
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *
50 *      ! date !   auteur !           but
51 *      -----
52 *      !           !           !
53 *
54 *      *****
55 *      MODIFICATIONS:
56 *      ! date !   auteur !           but
57 *      -----
58 *
59 *      *****/
60 #ifndef UMAT_ABAQUS_H
61 #define UMAT_ABAQUS_H
62
63 #include "Tenseur.h"
64 #include "Vecteur.h"
65 #include "TenseurQ.h"
66 #include <string.h>
67 #include <string>
68 #include "ParaGlob.h"
69 #include "Tableau2_T.h"
70 #include "PtIntegMecaInterne.h"
71
72 // définition d'une union qui lie les réels, les entiers et les caractères
73 union Tab_car_double_int
74 { char tampon[928];
75   double x[116];
76   int n[232];
77 };

```

```

78
79
80
81 class UmatAbaqus
82 {
83     public :
84
85         // CONSTRUCTEURS :
86         // par défaut
87         UmatAbaqus(int dimension_espace = 3, int nb_vecteur = 3);
88         // constructeur de copie
89         UmatAbaqus(const UmatAbaqus& a);
90
91         // DESTRUCTEUR :
92         ~UmatAbaqus();
93
94         // METHODES PUBLIQUES :
95
96         // Surcharge de l'operateur =
97         UmatAbaqus& operator=( UmatAbaqus& a);
98         // >>>> 1) >>>> contexte: utilisation d'herezh comme umat
99         // si utilisation_umat_interne: alors il n'y a pas d'utilisation des pipes
100        // tout se passe en interne
101        // lecture des données Umat
102        void LectureDonneesUmat(bool utilisation_umat_interne,const int niveau);
103        // écriture des données Umat
104        void EcritureDonneesUmat(bool utilisation_umat_interne,const int niveau);
105        // <<<< 2) <<<< contexte: utilisation par herezh d'une umat extérieure
106        // écriture des données pour l'umat
107        void EcritureDonneesPourUmat(bool utilisation_umat_interne,const int niveau);
108        // Puis lecture des résultats de l'umat
109        void LectureResultatUmat(bool utilisation_umat_interne,const int niveau);
110
111        // changement du nom des pipes
112        void Change_nom_pipe_envoi(const string& env) {envoi=env;};
113        void Change_nom_pipe_reception(const string& recep) {reception=recep;};
114
115        // initialisation des données d'entrée tous à zéro
116        // sauf: nom_materiau: qui reste inchangé
117        //      giB_t et BaseB: qui sont initialisée à une base absolue identité
118        //      pnewdt: très grand
119        // sinon on a:
120        //      *eps_meca, *delta_eps_meca : nulles, temps_t=0, temps_tdt et delta_t=0;
121        //      temper_t et delta_temper: nulles,
122        //      pas de variables aux noeuds, pas de proprietes_materiau
123        //      coordonnées à 0,
124        //      longueur caractéristique=0,
125        // et tous les indices=0:
126        //      nb_elem nb_pt_integ nb_du_plis nb_pt_dans_le_plis nb_step nb_increment
127        void Init_zero();
128
129        // initialisation des données d'entrée à un démarrage correcte
130        // *t_sigma=0, v_statev de taille nulle, les énergies nulles, pnewdt très grand
131        //      *eps_meca, *delta_eps_meca : nulles, temps_t=0, temps_tdt et delta_t=1;
132        //      temper_t et delta_temper: nulles,
133        //      pas de variables aux noeuds, pas de proprietes_materiau
134        //      coordonnées à 0, matrice de rotation identité
135        //      longueur caractéristique=1, giB_t et giB_tdt: base identité en absolue
136        // et tous les indices=1:
137        //      nb_elem nb_pt_integ nb_du_plis nb_pt_dans_le_plis nb_step nb_increment
138        // par défaut la dimension des tenseurs est 3, mais pour les CP par exemple
139        // la dimension réelle utilisée est 2
140        // de même pour nb_reel_tau_ij qui indique le nombre de contrainte de cisaillement
141        // réellement utilisé: qui en CP = 1
142        void Init_un(int dim_reel_tens = 3,int nb_reel_tau_ij=3);
143
144        // transfert de l'Umat vers le point d'intégration à t
145        void Transfert_Umat_ptInteg_t(PtIntegMecaInterne& ptIntegMeca);
146        // transfert de l'Umat vers le point d'intégration à tdt
147        void Transfert_Umat_ptInteg_tdt(PtIntegMecaInterne& ptIntegMeca);
148
149        // affichage en fonction d'un niveau
150        void Affiche(ofstream& sort,const int niveau);
151
152        //----- lecture écriture de restart -----
153        // cas donne le niveau de la récupération
154        // = 1 : on récupère tout
155        // = 2 : on récupère uniquement les données variables (supposées comme telles)
156        void Lecture_base_info(ifstream& ent,const int cas);
157        // cas donne le niveau de sauvegarde
158        // = 1 : on sauvegarde tout
159        // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
160        void Ecriture_base_info(ofstream& sort,const int cas);
161
162        // -----
163        // ---                               conteneur VARIABLES PUBLIQUES                               ---
164        // -----

```

```

165
166 TenseurHH* t_sigma; // contrainte: en entrée à t, puis en sortie à tdt
167 Vecteur v_statev; // variables d'état attaché, en entrée à t en sortie doivent être mise à tdt
168 TenseurHHH* d_sigma_deps; // variation de sigma par rapport aux déformations
169 double energie_elastique; // différentes énergie: en entrée à t, en sortie
170 double dissipation_plastique; double dissipation_visqueuse; // à tdt
171 double ener_therm_vol_par_utemps; // énergie thermique volumique générée par
172 // le travail mécanique, par unité de temps
173 TenseurHH* d_sigma_d_tempera; //variation de sigma par rapport à la thermique
174 TenseurHH* d_ener_therm_vol_deps; // variation de l'énergie thermique générée
175 // par rapport à la déformation
176 double d_ener_therm_dtemper; // variation de l'énergie thermique générée
177 // par rapport à la température
178 // -- variables qui peuvent être updated
179 double pnewdt; // pilotage de delta t: delta_t_new *= pnewdt;
180 // en entrée pnewdt est initialisée à une grande valeur
181 // si l'on veut diminuer le pas de temps-> on met en sortie pnewdt < 1
182 // -- variables passées pour information
183 TenseurBB* eps_meca; // déformation mécanique
184 TenseurBB* delta_eps_meca; // incrément de déformation mécanique
185 double temps_t; double temps_tdt; // temps
186 double delta_t; // incrément de temps
187 double temper_t; // température initiale à t
188 double delta_temper; // incrément de température
189 Vecteur ddlNoeA_ptinteg; // variables aux noeuds interpolées au point
190 Vecteur delta_ddlNoeA_ptinteg; // incréments des variables aux noeuds interpolées
191 string nom_materiau; // nom du matériau
192 int dim_tens; // nombre de sigma_ij
193 int nb_tau_ij; // nombre de sigma_ij avec i différent de j
194 int nb_ij_tenseur; // nombre de composantes du tableau stress
195 int nb_variables_etat; // nombre de variables d'état attaché
196 Vecteur proprietes_materiau; // propriétés du matériau;
197 int nb_proprietes_materiau; // nombre de propriétés du matériau
198 Coordonnee coord_pt; // coordonnee du point en absolue pour abaqus
199 Mat_pleine mat_corota; // matrice de rotation du corotationnelle
200 double longueur_caracteristique;
201 BaseB giB_t; // base naturelle à t
202 BaseB giB_tdt; // base naturelle à tdt
203 int nb_elem; // numéro de l'élément
204 int nb_pt_integ; // numéro du point d'intégration
205 int nb_du_plis; // numéro du plis pour les multicouches
206 int nb_pt_dans_le_plis; // numéro du point dans le plis
207 int nb_step; // numéro du step
208 int nb_increment; // numéro de l'incrément
209
210 // --- vecteur normal dans le cas CP
211 CoordonneeB N_t,N_tdt;
212
213 // -----
214 // --- fin conteneur VARIABLES PUBLIQUES -----
215 // -----
216
217
218 private :
219 // VARIABLES PROTEGEES :
220 // -- variables devant être définies
221 double* STRESS; // TenseurHH* t_sigma; // contrainte: en entrée à t, puis en sortie à tdt
222 double* STATEV; // Vecteur v_statev; // variables d'état attaché, en entrée à t
223 // en sortie doivent être mise à tdt
224 double * DDSDD; // TenseurHHBB* d_sigma_deps; // variation de sigma par rapport aux déformations
225 double* SSE,* SPD,* SCD; // double energie_elastique; // différentes énergie: en entrée à t, en
sortie
// double dissipation_plastique; double dissipation_visqueuse; // à tdt
226 double* RPL; // double ener_therm_vol_par_utemps; // énergie thermique volumique générée par
// le travail mécanique, par unité de temps
227 double* DDSDDT; // TenseurHH* d_sigma_d_tempera; //variation de sigma par rapport à la thermique
228 double* DRPLDE; // TenseurHH* d_ener_therm_vol_deps; // variation de l'énergie thermique générée
// par rapport à la déformation
229 double* DRPLDT; // double d_ener_therm_dtemper; // variation de l'énergie thermique générée
// par rapport à la température
230
231 // -- variables qui peuvent être updated
232 double* PNEWDT; // double pnewdt; // pilotage de delta t: delta_t_new *= pnewdt;
233 // en entrée pnewdt est initialisée à une grande valeur
234 // si l'on veut diminuer le pas de temps-> on met en sortie pnewdt < 1
235 // -- variables passées pour information
236 double* STRAN; // TenseurBB* eps_meca; // déformation mécanique
237 double* DSTRAN; // TenseurBB* delta_eps_meca; // incrément de déformation mécanique
238 double* TIME; // step time (ne sert pas pour herezh) puis double temps_t; // temps
239 double* DTIME; // double delta_t; // incrément de temps
240 double* Temp; // double temper; // température initiale à t
241 double* DTEMP; // double delta_temper; // incrément de température
242 double* PREDEF; // Vecteur ddlNoeA_ptinteg; // variables aux noeuds interpolées au point
243 double* DPRED; // Vecteur delta_ddlNoeA_ptinteg; // incréments des variables aux noeuds interpolées
244 char* CNAME; // string nom_materiau; // nom du matériau
245 int* NDI; // int dim_tens; // nombre de sigma_ij
246 int* NSHR; // int nb_tau_ij; // nombre de sigma_ij avec i différent de j
247 int* NTENS; // int nb_ij_tenseur; // nombre de composantes du tableau stress

```

```

251 int* NSTATV; // int nb_variabels_etat; // nombre de variables d'état attaché
252 double* PROPS; // Vecteur proprietes_materiau; // propriétés du matériau;
253 int* NPROPS; // int nb_proprietes_materiau; // nombre de propriétés du matériau
254 double* COORDS; // Coordonnee coord_pt; // coordonnee du point
255 double* DROT; // Mat_pleine mat_corota; // matrice de rotation du corotationnelle
256 double* CELENT; // double longueur_caracteristique;
257 double* DFGRD0; // BaseB giB_t; // base naturelle à t
258 double* DFGRD1; // BaseB giB_tdt; // base naturelle à tdt
259 int* NOEL; // int nb_elem; // numéro de l'élément
260 int* NPT; // int nb_pt_integ; // numéro du point d'intégration
261 int* LAYER; // int nb_du_plis; // numéro du plis pour les multicouches
262 int* KSPT; // int nb_pt_dans_le_plis; // numéro du point dans le plis
263 int* KSTEP; // int nb_step; // numéro du step
264 int* KINC; // int nb_increment; // numéro de l'incrément
265
266 // -- variables pour les tubes nommés -----
267
268 // nom du tube nommé pour l'envoi des données
269 string envoi;
270 // nom du tube nommé pour la reception des données
271 string reception;
272 int passage; // test pour nombre de passage dans l'appel
273 Tab_car_double_int t_car_x_n; // tableau de caractères, réels et entiers
274
275 // --- gestion d'index 3D classique ----
276 class ChangementIndex
277 { public:
278     ChangementIndex();
279     // passage pour les index de la forme vecteur à la forme i,j
280     Tableau <int> idx_i,idx_j;
281     // passage pour les index de la forme i,j à la forme vecteur
282     Tableau2 <int> odVect;
283 };
284 static const ChangementIndex cdex;
285
286 // // --- gestion d'index 2D CP en 3D ----
287 class ChangementIndexCP
288 { public:
289     ChangementIndexCP();
290     // passage pour les index de la forme vecteur à la forme i,j
291     Tableau <int> idx_i,idx_j;
292     // passage pour les index de la forme i,j à la forme vecteur
293     Tableau2 <int> odVect;
294 };
295 static const ChangementIndexCP cdexCP;
296
297
298 // METHODES PROTEGEES :
299 // 1)-- utilisation en tant qu'umat
300 // copie des grandeurs évoluées dans les grandeurs de base
301 // uniquement celles qui varient
302 void Copie_evolue_vers_base(const int niveau);
303 // copie des grandeurs de base vers les grandeurs évoluées
304 // uniquement les grandeurs d'entrée (supposées à lire)
305 void Copie_base_vers_evolue(const int niveau);
306 // 2) -- utilisation en tant qu'utilisation d'une umat extérieure
307 // copie des grandeurs évoluées dans les grandeurs de base
308 // uniquement celles qui sont des données
309 void Copie_evolue_vers_base_PourUmat(const int niveau);
310 // copie des grandeurs de base vers les grandeurs évoluées
311 // uniquement celles qui sont des résultats
312 void Copie_base_vers_evolue_PourUmat(const int niveau);
313
314
315 };
316
317 #endif

```

## 7.153 ExceptionsElemMeca.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify

```

```

16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29 //
30 /*****
31 *      DATE:      1/05/2004
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Définir des classes d'exception pour la gestion d'erreur
39 *                concernant les éléments mécaniques.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date !   auteur !           but
45 *      -----
46 *      !           !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      -----
52 *
53 *****/
54 #ifndef EXCEPTIONSELEMMECA_H
55 #define EXCEPTIONSELEMMECA_H
56
57
58 /// @addtogroup groupe_des_elements_finis
59 /// @{
60 ///
61
62 // cas d'une erreur survenue à cause d'un jacobien négatif
63
64 class ErrJacobienNegatif_ElemMeca {};
65 /// @} // end of group
66
67
68 /// @addtogroup groupe_des_elements_finis
69 /// @{
70 ///
71
72 // cas d'une erreur survenue à cause d'une variation de jacobien trop grande
73
74 class ErrVarJacobienMini_ElemMeca {};
75 /// @} // end of group
76
77
78 /// @addtogroup groupe_des_elements_finis
79 /// @{
80 ///
81
82 // cas d'une erreur inconnue au niveau de l'élément méca
83
84 class Err_inconnue_ElemMeca {};
85 /// @} // end of group
86
87
88 #endif

```

## 7.154 Hexa.h

```

1 // FICHER : Hexa.h
2 // CLASSE : Hexa
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)

```

```

9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31 //
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 *      BUT:      La classe Hexa permet de declarer des elements
41 *      Hexaedriques Trilineaires et de realiser
42 *      le calcul du residu local et de la raideur locale pour une loi de
43 *      comportement donnee. La dimension de l'espace pour un tel element
44 *      est 3.
45 *
46 *      l'interpolation est trilineaire a 8 noeuds, le nombre de
47 *      point d'integration est de 8.
48 *      *****
49 *
50 *      VERIFICATION:
51 *
52 *      ! date ! auteur ! but
53 *      -----
54 *      ! ! ! !
55 *      *****
56 *      MODIFICATIONS:
57 *      ! date ! auteur ! but
58 *      -----
59 *      $
60 *****/
61 // -----classe pour un calcul de mecanique-----
62
63
64
65
66 #ifndef HEXA_H
67 #define HEXA_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontQuadLine.h"
80 #include "FrontSegLine.h"
81
82
83 /// @addtogroup groupe_des_elements_finis
84 /// @{
85 ///
86
87
88 class Hexa : public HexaMemb
89 {
90
91     public :
92
93         // CONSTRUCTEURS :
94

```

```

95     // Constructeur par défaut
96     Hexa ();
97
98     // Constructeur fonction d'un numero
99     // d'identification
100    Hexa (int num_mail,int num_id);
101
102    // Constructeur fonction d'un numero de maillage et d'identification et
103    // du tableau de connexite des noeuds
104    Hexa (int num_mail,int num_id,const Tableau<Noeud *>& tab);
105
106    // Constructeur de copie
107    Hexa (const Hexa& hexa);
108
109    // DESTRUCTEUR :
110    ~Hexa ();
111
112    // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113    // méthode virtuelle
114    Element* Nevez_copie() const { Element * el= new Hexa(*this); return el;};
115
116    // Surcharge de l'operateur = : realise l'egalite entre deux instances de Hexa
117    Hexa& operator= (Hexa& hexa);
118
119    // METHODES :
120    // 1) derivant des virtuelles pures
121
122    // affichage dans la sortie transmise, des variables duales "nom"
123    // aux differents points d'integration
124    // dans le cas ou nom est vide, affichage de "toute" les variables
125    void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127    // 2) derivant des virtuelles
128    // 3) methodes propres a l'element
129
130    protected :
131
132    // adressage des frontières linéiques et surfacique
133    // définit dans les classes dérivées, et utilisées pour la construction des frontières
134    ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135    { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
136    ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137    { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
138
139    // VARIABLES PRIVEES :
140    // place memoire commune a tous les elements TriaMembl1
141    static HexaMemb::DonnComHexa * doCoHexa;
142    // idem mais pour les indicateurs qui servent pour l'initialisation
143    static HexaMemb::UneFois uneFois;
144
145    class NombresConstruireHexa : public NombresConstruire
146    { public: NombresConstruireHexa ();
147      };
148    static NombresConstruireHexa nombre_V; // les nombres propres à l'élément
149
150    // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
151    //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
152    class ConsHexa : public ConstrucElement
153    { public : ConsHexa ()
154      { NouvelleTypeElement nouv (HEXAEDRE, LINEAIRE, MECA_SOLIDE_DEFORMABLE, this);
155        if (ParaGlob::NiveauImpression() >= 4)
156          cout << "\n initialisation Hexa" << endl;
157        Element::listTypeElement.push_back (nouv);
158      };
159      Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
160      {Element * pt;
161        pt = new Hexa (num_maill,num) ;
162        return pt;};
163      // ramene true si la construction de l'element est possible en fonction
164      // des variables globales actuelles: ex en fonction de la dimension
165      bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
166    };
167    static ConsHexa consHexa;
168    static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
169 };
170 /// @} // end of group
171 #endif
172
173
174
175

```

## 7.155 Hexa\_cm1pti.h

```
1 // FICHER : Hexa_cm1pti.h
```



```

2 // CLASSE : Hexa_cm1pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           11/01/2003
34 *
35 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:         Herezh++
38 *
39 *
40 *   BUT:            Idem Hexa mais avec 1 pt d'intégration
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !           but
47 *   !-----!-----!-----!-----!
48 *   !           !           !           !
49 *   !           !           !           !
50 *   !           !           !           !
51 *   !           !           !           !
52 *   !           !           !           !
53 *   !           !           !           !
54 *   *****/
55
56 // -----classe pour un calcul de mecanique-----
57
58
59
60 #ifndef HEXA_CM1PTI_H
61 #define HEXA_CM1PTI_H
62
63 #include "ParaGlob.h"
64 #include "ElemMeca.h"
65 #include "Met_abstraite.h"
66 #include "GeomQuadrangle.h"
67 #include "Noeud.h"
68 #include "UtilLecture.h"
69 #include "Tenseur.h"
70 #include "NevezTenseur.h"
71 #include "Deformation.h"
72 #include "HexaMemb.h"
73 #include "FrontQuadLine.h"
74 #include "FrontSegLine.h"
75
76
77 /// @addtogroup groupe_des_elements_finis
78 /// @{
79 ///
80
81
82 class Hexa_cm1pti : public HexaMemb
83 {
84
85     public :
86
87         // CONSTRUCTEURS :

```

```

88
89 // Constructeur par défaut
90 Hexa_cmlpti ();
91
92 // Constructeur fonction d'un numero
93 // d'identification
94 Hexa_cmlpti (int num_maill,int num_id);
95
96 // Constructeur fonction d'un numero de maillage et d'identification et
97 // du tableau de connexite des noeuds
98 Hexa_cmlpti (int num_maill,int num_id,const Tableau<Noeud *>& tab);
99
100 // Constructeur de copie
101 Hexa_cmlpti (const Hexa_cmlpti& hexa_cmlpti);
102
103
104 // DESTRUCTEUR :
105 ~Hexa_cmlpti ();
106
107 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
108 // méthode virtuelle
109 Element* Nevez_copie() const { Element * el= new Hexa_cmlpti(*this); return el;};
110
111 // Surcharge de l'operateur = : realise l'egalite entre deux instances de Hexa_cmlpti
112 Hexa_cmlpti& operator= (Hexa_cmlpti& hexa_cmlpti);
113
114 // METHODES :
115 // 1) derivant des virtuelles pures
116
117 // affichage dans la sortie transmise, des variables duales "nom"
118 // aux differents points d'integration
119 // dans le cas ou nom est vide, affichage de "toute" les variables
120 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
121
122 // 2) derivant des virtuelles
123 // 3) methodes propres a l'element
124
125 protected :
126
127 // adressage des frontieres linéiques et surfacique
128 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
129 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
130 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
131 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132 { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
133
134 // VARIABLES PRIVEES :
135 // place memoire commune a tous les elements TriaMembL1
136 static HexaMemb::DonnComHexa * doCoHexa_cmlpti;
137 // idem mais pour les indicateurs qui servent pour l'initialisation
138 static HexaMemb::UneFois uneFois;
139
140 class NombresConstruireHexa_cmlpti : public NombresConstruire
141 { public: NombresConstruireHexa_cmlpti();
142 };
143 static NombresConstruireHexa_cmlpti nombre_V; // les nombres propres à l'élément
144
145 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
146 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
147 class ConsHexa_cmlpti : public ConstrucElement
148 { public : ConsHexa_cmlpti ()
149 { NouvelleTypeElement nouv (HEXAEDRE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this,"_cmlpti");
150 if (ParaGlob::NiveauImpression() >= 4)
151 cout << "\n initialisation Hexa_cmlpti" << endl;
152 Element::listTypeElement.push_back(nouv);
153 };
154 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
155 {Element * pt;
156 pt = new Hexa_cmlpti (num_maill,num) ;
157 return pt;};
158 // ramene true si la construction de l'element est possible en fonction
159 // des variables globales actuelles: ex en fonction de la dimension
160 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
161 };
162 static ConsHexa_cmlpti consHexa_cmlpti;
163 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
164 };
165 /// @} // end of group
166 #endif
167

```

## 7.156 Hexa\_cm27pti.h

```

1 // FICHER : Hexa.h
2 // CLASSE : Hexa

```

```

3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      11/01/2003
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *
40 *   BUT:      Idem Hexa mais avec 27 pt d'intégration
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !           but
47 *   -----
48 *   !           !           !
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date !   auteur !           but
52 *   -----
53 *   $
54 *   *****/
55
56 // -----classe pour un calcul de mecanique-----
57
58
59
60 #ifndef HEXA_CM27PTI_H
61 #define HEXA_CM27PTI_H
62
63 #include "ParaGlob.h"
64 #include "ElemMeca.h"
65 #include "Met_abstraite.h"
66 #include "GeomQuadrangle.h"
67 #include "Noeud.h"
68 #include "UtilLecture.h"
69 #include "Tenseur.h"
70 #include "NevezTenseur.h"
71 #include "Deformation.h"
72 #include "HexaMemb.h"
73 #include "FrontQuadLine.h"
74 #include "FrontSegLine.h"
75
76
77 /// @addtogroup groupe_des_elements_finis
78 /// @{
79 ///
80
81
82 class Hexa_cm27pti : public HexaMemb
83 {
84
85     public :
86
87         // CONSTRUCTEURS :
88

```

```

89     // Constructeur par défaut
90     Hexa_cm27pti ();
91
92     // Constructeur fonction d'un numero
93     // d'identification
94     Hexa_cm27pti (int num_mail,int num_id);
95
96     // Constructeur fonction d'un numero de maillage et d'identification et
97     // du tableau de connexite des noeuds
98     Hexa_cm27pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
99
100    // Constructeur de copie
101    Hexa_cm27pti (const Hexa_cm27pti& Hexa_cm27pti);
102
103
104    // DESTRUCTEUR :
105    ~Hexa_cm27pti ();
106
107    // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
108    // méthode virtuelle
109    Element* Nevez_copie() const { Element * el= new Hexa_cm27pti(*this); return el;};
110
111    // Surcharge de l'operateur = : realise l'egalite entre deux instances de Hexa_cm27pti
112    Hexa_cm27pti& operator= (Hexa_cm27pti& Hexa_cm27pti);
113
114    // METHODES :
115    // 1) derivant des virtuelles pures
116
117    // affichage dans la sortie transmise, des variables duales "nom"
118    // aux differents points d'integration
119    // dans le cas ou nom est vide, affichage de "toute" les variables
120    void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
121
122    // 2) derivant des virtuelles
123    // 3) methodes propres a l'element
124
125    protected :
126
127    // adressage des frontières linéiques et surfacique
128    // définit dans les classes dérivées, et utilisées pour la construction des frontières
129    ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
130    { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
131    ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132    { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
133
134    // VARIABLES PRIVEES :
135    // place memoire commune a tous les elements TriaMembL1
136    static HexaMemb::DonnComHexa * doCoHexa_cm27pti;
137    // idem mais pour les indicateurs qui servent pour l'initialisation
138    static HexaMemb::UneFois uneFois;
139
140    class NombresConstruireHexa_cm27pti : public NombresConstruire
141    { public: NombresConstruireHexa_cm27pti();
142    };
143    static NombresConstruireHexa_cm27pti nombre_V; // les nombres propres à l'élément
144
145    // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
146    //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
147    class ConsHexa_cm27pti : public ConstrucElement
148    { public : ConsHexa_cm27pti ()
149    { NouvelleTypeElement nouv (HEXAEDRE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this,"_cm27pti");
150    if (ParaGlob::NiveauImpression() >= 4)
151    cout << "\n initialisation Hexa_cm27pti" << endl;
152    Element::listTypeElement.push_back(nouv);
153    };
154    Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
155    {Element * pt;
156    pt = new Hexa_cm27pti (num_maill,num) ;
157    return pt;};
158    // ramene true si la construction de l'element est possible en fonction
159    // des variables globales actuelles: ex en fonction de la dimension
160    bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
161    };
162    static ConsHexa_cm27pti consHexa_cm27pti;
163    static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
164 };
165 /// @} // end of group
166 #endif
167

```

## 7.157 Hexa\_cm64pti.h

```

1 // FICHER : Hexa.h
2 // CLASSE : Hexa
3

```

```

4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      11/01/2003
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      Idem Hexa mais avec 64 pt d'intégration
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *   ! date !   auteur !       but
46 *   -----
47 *   !       !       !
48 *
49 *   *****
50 *
51 *   MODIFICATIONS:
52 *   ! date !   auteur !       but
53 *   -----
54 *
55 *****/
56 // -----classe pour un calcul de mecanique-----
57
58
59
60 #ifndef HEXA_CM64PTI_H
61 #define HEXA_CM64PTI_H
62
63 #include "ParaGlob.h"
64 #include "ElemMeca.h"
65 #include "Met_abstraite.h"
66 #include "GeomQuadrangle.h"
67 #include "Noeud.h"
68 #include "UtilLecture.h"
69 #include "Tenseur.h"
70 #include "NevezTenseur.h"
71 #include "Deformation.h"
72 #include "HexaMemb.h"
73 #include "FrontQuadLine.h"
74 #include "FrontSegLine.h"
75
76
77 /// @addtogroup groupe_des_elements_finis
78 /// @{
79 ///
80
81
82 class Hexa_cm64pti : public HexaMemb
83 {
84
85     public :
86
87         // CONSTRUCTEURS :
88
89         // Constructeur par defaut

```

```

90     Hexa_cm64pti ();
91
92     // Constructeur fonction d'un numero
93     // d'identification
94     Hexa_cm64pti (int num_maill,int num_id);
95
96     // Constructeur fonction d'un numero de maillage et d'identification et
97     // du tableau de connexite des noeuds
98     Hexa_cm64pti (int num_maill,int num_id,const Tableau<Noeud *>& tab);
99
100    // Constructeur de copie
101    Hexa_cm64pti (const Hexa_cm64pti& Hexa_cm64pti);
102
103
104    // DESTRUCTEUR :
105    ~Hexa_cm64pti ();
106
107    // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
108    // méthode virtuelle
109    Element* Nevez_copie() const { Element * el= new Hexa_cm64pti(*this); return el;};
110
111    // Surcharge de l'operateur = : realise l'egalite entre deux instances de Hexa_cm64pti
112    Hexa_cm64pti& operator= (Hexa_cm64pti& Hexa_cm64pti);
113
114    // METHODES :
115    // 1) derivant des virtuelles pures
116
117    // affichage dans la sortie transmise, des variables duales "nom"
118    // aux differents points d'integration
119    // dans le cas ou nom est vide, affichage de "toute" les variables
120    void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
121
122    // 2) derivant des virtuelles
123    // 3) methodes propres a l'element
124
125    protected :
126
127    // adressage des frontières linéiques et surfacique
128    // définit dans les classes dérivées, et utilisées pour la construction des frontières
129    ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
130    { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
131    ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132    { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
133
134    // VARIABLES PRIVEES :
135    // place memoire commune a tous les elements TriaMembl1
136    static HexaMemb::DonnComHexa * doCoHexa_cm64pti;
137    // idem mais pour les indicateurs qui servent pour l'initialisation
138    static HexaMemb::UneFois uneFois;
139
140    class NombresConstruireHexa_cm64pti : public NombresConstruire
141    { public: NombresConstruireHexa_cm64pti();
142    };
143    static NombresConstruireHexa_cm64pti nombre_V; // les nombres propres à l'élément
144
145    // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
146    //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
147    class ConsHexa_cm64pti : public ConstrucElement
148    { public : ConsHexa_cm64pti ()
149    { NouvelleTypeElement nouv (HEXAEDRE, LINEAIRE,MECA_SOLIDE_DEFORMABLE,this,"_cm64pti");
150      if (ParaGlob::NiveauImpression() >= 4)
151        cout << "\n initialisation Hexa_cm64pti" << endl;
152      Element::listTypeElement.push_back (nouv);
153    };
154      Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
155      {Element * pt;
156        pt = new Hexa_cm64pti (num_maill,num) ;
157        return pt;};
158      // ramene true si la construction de l'element est possible en fonction
159      // des variables globales actuelles: ex en fonction de la dimension
160      bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
161    };
162    static ConsHexa_cm64pti consHexa_cm64pti;
163    static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
164 };
165 /// @} // end of group
166 #endif
167

```

## 7.158 HexaLMemb.h

```

1 // FICHER : Hexa.h
2 // CLASSE : Hexa
3
4 // This file is part of the Herezh++ application.

```

```

5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           09/02/2003
34 *
35 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:        Herezh++
38 *
39 *
40 *   *****
41 *   BUT:   La classe HexalMemb est une classe générique.
42 *   Elle permet de declarer des elements
43 *   Hexaedriques Trilineaires et de realiser
44 *   le calcul du residu local et de la raideur locale pour une loi de
45 *   comportement donnee. La dimension de l'espace pour un tel element
46 *   est 3.
47 *
48 *   l'interpolation est trilineaire a 8 noeuds, le nombre de
49 *   point d'integration est défini dans les classes dérivées.
50 *
51 *   *****
52 *
53 *   VERIFICATION:
54 *
55 *   ! date ! auteur ! but
56 *   -----
57 *   ! ! !
58 *   $
59 *
60 *   *****
61 *
62 *   MODIFICATIONS:
63 *
64 *   ! date ! auteur ! but
65 *   -----
66 *   $
67 *
68 *   *****/
69
70 // -----classe pour un calcul de mecanique-----
71
72 #ifndef HEXALMEMB_H
73 #define HEXALMEMB_H
74
75 #include "ParaGlob.h"
76 #include "ElemMeca.h"
77 #include "Met_abstraite.h"
78 #include "GeomQuadrangle.h"
79 #include "Noeud.h"
80 #include "UtilLecture.h"
81 #include "Tenseur.h"
82 #include "NevezTenseur.h"
83 #include "Deformation.h"
84 #include "HexaMemb.h"
85 #include "FrontQuadLine.h"
86 #include "FrontSegLine.h"
87
88 /// @addtogroup groupe_des_elements_finis
89 /// @{
90 ///
91
92 class HexalMemb : public HexaMemb
93 {

```

```

91
92     public :
93
94         // CONSTRUCTEURS :
95
96         // Constructeur par défaut
97         HexaLMemb ();
98
99         // Constructeur fonction d'un numero
100        // d'identification
101        HexaLMemb (int num_id);
102
103        // Constructeur fonction d'un numero d'identification et
104        // du tableau de connexite des noeuds
105        HexaLMemb (int num_id, const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        HexaLMemb (const HexaLMemb& hexa);
109
110        // DESTRUCTEUR :
111        virtual ~HexaLMemb ();
112
113        // Surcharge de l'operateur = : realise l'egalite entre deux instances de Hexa
114        Hexa& operator= (Hexa& hexa);
115
116        // METHODES :
117        // 1) derivant des virtuelles pures
118
119        // affichage dans la sortie transmise, des variables duales "nom"
120        // aux differents points d'integration
121        // dans le cas ou nom est vide, affichage de "toute" les variables
122        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
123
124        // 2) derivant des virtuelles
125        // 3) methodes propres a l'element
126
127        protected :
128
129        // adressage des frontieres linéiques et surfacique
130        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
131        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132        { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
133        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
134        { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
135
136        // VARIABLES PRIVEES :
137        // place memoire commune a tous les elements TriaMembL1
138        static HexaMemb::DonnComHexa * doCoHexa;
139        // idem mais pour les indicateurs qui servent pour l'initialisation
140        static HexaMemb::UneFois uneFois;
141
142        // fonction définissant les nombres (de noeud, de point d'integ ..)
143        // utilisé dans la construction des éléments
144        NombresConstruire ConstruireLesNombres() ;
145        class NombresConstruireHexa : public NombresConstruire
146        { public: NombresConstruireHexa();
147          };
148        static NombresConstruireHexa nombre_V; // les nombres propres à l'élément
149
150 };
151 /// @} // end of group
152 #endif
153
154
155
156

```

## 7.159 HexaMemb.h

```

1 // FICHER : HexaMemb.h
2 // CLASSE : HexaMemb
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //

```



```

17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *
35 *   DATE:      15/01/97
36 *
37 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:    Herezh++
40 *
41 * *****/
42 * La classe HexaMemb permet de declarer des elements hexahedriques et de realiser
43 * le calcul du residu local et de la raideur locale pour une loi de comportement
44 * donnee. La dimension de l'espace pour un tel element est 3
45 *
46 * l'interpolation le nombre de point d'integration sont definit dans les classes derivees
47 *
48 * l'element est virtuel
49 *
50 *   *****
51 *
52 * VERIFICATION:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *   !       !           !
57 *
58 *   *****
59 *
60 * MODIFICATIONS:
61 *
62 *   ! date !   auteur !           but
63 *   -----
64 *
65 *
66 *****/
67
68 // -----classe pour un calcul de mecanique-----
69
70 // La classe HexaMemb permet de declarer des elements hexahedriques et de realiser
71 // le calcul du residu local et de la raideur locale pour une loi de comportement
72 // donnee. La dimension de l'espace pour un tel element est 3
73 //
74 // l'interpolation le nombre de point d'integration sont definit dans les classes derivees
75 //
76 // l'element est virtuel
77
78
79 #ifndef HEXAMEMB_H
80 #define HEXAMEMB_H
81
82 #include "ParaGlob.h"
83 #include "ElemMeca.h"
84 #include "Met_abstraite.h"
85 #include "ElemGeomC0.h"
86 #include "GeomSeg.h"
87 #include "GeomQuadrangle.h"
88 #include "Noeud.h"
89 #include "UtilLecture.h"
90 #include "Tenseur.h"
91 #include "NevezTenseur.h"
92 #include "Deformation.h"
93
94 /// @addtogroup groupe_des_elements_finis
95 /// @{
96 ///
97
98
99 class HexaMemb : public ElemMeca
100 {
101
102 public :

```

```

103
104 // CONSTRUCTEURS :
105 // Constructeur par defaut
106 HexaMemb ();
107
108 // Constructeur fonction d'un numero
109 // d'identification , d'identificateur d'interpolation et de geometrie et éventuellement un string
d'information annexe
110 HexaMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string info="");
111
112 // Constructeur fonction d'un numero de maillage et d'identification,
113 // du tableau de connexite des noeuds, d'identificateur d'interpolation et de geometrie
114 // et éventuellement un string d'information annexe
115 HexaMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,
116           const Tableau<Noeud *>& tab,string info="");
117
118 // Constructeur de copie
119 HexaMemb (const HexaMemb& hexaMem);
120
121
122 // DESTRUCTEUR :
123 ~HexaMemb ();
124
125
126 // Surcharge de l'operateur = : realise l'egalite entre deux instances de HexaMemb
127 HexaMemb& operator= (HexaMemb& hexaMem);
128
129 // METHODES :
130 // 1) derivant des virtuelles pures
131
132 // Lecture des donnees de la classe sur fichier
133 void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
134
135 // affichage d'info en fonction de ordre
136 // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
137 void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> * tabMaillageNoeud)
138 { return Element::Info_com_El (nombre->nbne,entreePrinc,ordre,tabMaillageNoeud);};
139
140 // ramene l'element geometrique
141 ElemGeomC0& ElementGeometrique() const { return *(unefois->doCoMemb->hexaed);};
142 // ramene l'element geometrique en constant
143 const ElemGeomC0& ElementGeometrique_const() const { return *(unefois->doCoMemb->hexaed);};
144
145 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
associé
146 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
147 // 1) cas où l'on utilise la place passée en argument
148 Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
149 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
150 void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
151
152 // inactive les ddl du problème primaire de mécanique
153 inline void Inactive_ddl_primaire()
154 {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};
155 // active les ddl du problème primaire de mécanique
156 inline void Active_ddl_primaire()
157 {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl);};
158 // ajout des ddl de contraintes pour les noeuds de l'élément
159 inline void Plus_ddl_Sigma()
160 {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
161 // inactive les ddl du problème de recherche d'erreur : les contraintes
162 inline void Inactive_ddl_Sigma()
163 {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
164 // active les ddl du problème de recherche d'erreur : les contraintes
165 inline void Active_ddl_Sigma()
166 {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
167 // active le premier ddl du problème de recherche d'erreur : SIGMA11
168 inline void Active_premier_ddl_Sigma()
169 {ElemMeca::Act_premier_ddl_Sigma();};
170
171 // lecture de données diverses sur le flot d'entrée
172 void LectureContraintes (UtilLecture * entreePrinc)
173 { if (unefois->CalResPrem_t == 1)
174     ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
175     else
176     { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
177       unefois->CalResPrem_t = 1;
178     }
179 };
180
181 // retour des contraintes en absolu retour true si elle existe sinon false
182 bool ContraintesAbsolues (Tableau <Vecteur>& tabSig)
183 { if (unefois->CalResPrem_t == 1)
184     ElemMeca::ContraintesEnAbsolues (false,lesPtMecaInt.TabSigHH_t(),tabSig);
185     else
186     { ElemMeca::ContraintesEnAbsolues (true,lesPtMecaInt.TabSigHH_t(),tabSig);

```

```

187         unefois->CalResPrem_t = 1;
188     }
189     return true; }
190
191     // Libere la place occupee par le residu et eventuellement la raideur
192     // par l'appel de Libere de la classe mere et libere les differents tenseurs
193     // intermediaires cree pour le calcul et les grandeurs pointee
194     // de la raideur et du residu
195     void Libere ();
196
197     // acquisition d'une loi de comportement
198     void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
199
200     // test si l'element est complet
201     // = 1 tout est ok, =0 element incomplet
202     int TestComplet();
203
204     // procedure permettant de completer l'element apres
205     // sa creation avec les donnees du bloc transmis
206     // peut etre appeler plusieurs fois
207     // ici pour l'instant ne fait rien
208     Element* Complete(BlocGen & bloc, LesFonctions_nd* lesFonctionsND);
209     // Compléter pour la mise en place de la gestion de l'hourglass
210     Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
211
212     // ramene vrai si la surface numéro ns existe pour l'élément
213     // dans le cas des éléments Hexaedrique il peut y avoir de 1 à 6 surfaces
214     bool SurfExiste(int ns) const
215     { if ((ns>=1)&&(ns<=6)) return true; else return false;};
216
217     // ramene vrai si l'arête numéro na existe pour l'élément
218     bool AreteExiste(int na) const {if ((na <= 12) || (na>= 1)) return true; else return false;};
219
220
221     // retourne les tableaux de ddl associés aux noeuds, gere par l'element
222     // ce tableau est specifique a l'element
223     const DdlElement & TableauDdl() const
224     { return unefois->doCoMemb->tab_ddl; };
225
226     // Calcul du residu local et de la raideur locale,
227     // pour le schema implicite
228     Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
229
230     // Calcul du residu local a t
231     // pour le schema explicite par exemple
232     Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
233     { return HexaMemb::CalculResidu(false,pa);};
234
235     // Calcul du residu local a tdt
236     // pour le schema explicite par exemple
237     Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
238     { return HexaMemb::CalculResidu(true,pa);};
239
240     // Calcul de la matrice masse pour l'élément
241     Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
242
243     // ----- calcul dynamique -----
244     // calcul de la longueur d'arrête de l'élément minimal
245     // divisé par la célérité la plus rapide dans le matériau
246     double Long_arrete_mini_sur_c(Enum_dure temps)
247     { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
248
249     //----- calcul d'erreur, remontée des contraintes -----
250     // 1) calcul du résidu et de la matrice de raideur pour le calcul d'erreur
251     Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
252     // 2) remontée aux erreurs aux noeuds
253     Element::Er_ResRaid ErreurAuNoeud_ResRaid();
254
255     // ----- affichage ou récupération d'informations -----
256     // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
257     // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
258     // par exemple CoordPtInteg, ou Valeur_a_diff_temps
259     // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
260     // temps: dit si c'est à 0 ou t ou tdt
261     int PointLePlusPres(Enum_dure temps, Enum_ddl enu, const Coordonnee& M)
262     { return PtLePlusPres(temps,enu,M);};
263
264     // recuperation des coordonnées du point de numéro d'ordre iteg pour
265     // la grandeur enu
266     // temps: dit si c'est à 0 ou t ou tdt
267     // si erreur retourne erreur à true
268     Coordonnee CoordPtInteg(Enum_dure temps, Enum_ddl enu, int iteg, bool& erreur)
269     { return CoordPtInt(temps,enu,iteg,erreur);};
270
271     // récupération des valeurs au numéro d'ordre = iteg pour
272     // les grandeur enu
273     Tableau <double> Valeur_a_diff_temps(bool absolue, Enum_dure enu_t, const List_io<Ddl_enum_etendu>&

```

```

enu,int iteg);
274 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
275 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
276 // de tenseurs quelconque associée
277 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int
iteg);
278
279 //===== lecture écriture dans base info =====
280
281 // cas donne le niveau de la récupération
282 // = 1 : on récupère tout
283 // = 2 : on récupère uniquement les données variables (supposées comme telles)
284 void Lecture_base_info
285 (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
286 // cas donne le niveau de sauvegarde
287 // = 1 : on sauvegarde tout
288 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
289 void Ecriture_base_info(ofstream& sort,const int cas) ;
290
291 // 2) derivant des virtuelles
292
293 // retourne un tableau de ddl element, correspondant à la
294 // composante de sigma -> SIG11, pour chaque noeud qui contient
295 // des ddl de contrainte
296 // -> utilisé pour l'assemblage de la raideur d'erreur
297 DdlElement& Tableau_de_Sig11() const
298 {return unefois->doCoMemb->tab_Err1Sig11;} ;
299
300 // actualisation des ddl et des grandeurs actives de t+dt vers t
301 void TdtversT();
302 // actualisation des ddl et des grandeurs actives de t vers tdt
303 void TversTdt();
304
305 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
306 // qu'une fois la remontée aux contraintes effectuées sinon aucune
307 // action. En retour la valeur de l'erreur sur l'élément
308 // type indique le type de calcul d'erreur :
309 void ErreurElement(int type,double& errElemRelative
310 ,double& numerateur, double& denominateur);
311
312 // calcul des seconds membres suivant les chargements
313 // cas d'un chargement volumique,
314 // force indique la force volumique appliquée
315 // retourne le second membre résultant
316 // ici on l'épaisseur de l'élément pour constituer le volume
317 // -> explicite à t
318 Vecteur SM_charge_volumique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle
& pa,bool sur_volume_finale_)
319 { return HexaMemb::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_);} ;
320 // -> explicite à tdt
321 Vecteur SM_charge_volumique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,const
ParaAlgoControle & pa,bool sur_volume_finale_)
322 { return HexaMemb::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_);} ;
323 // -> implicite,
324 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
325 // retourne le second membre et la matrice de raideur correspondant
326 ResRaid SMR_charge_volumique_I(const Coordonnee& force,Fonction_nD* pt_fonct,const
ParaAlgoControle & pa,bool sur_volume_finale_) ;
327
328 // cas d'un chargement surfacique, sur les frontières des éléments
329 // force indique la force surfacique appliquée
330 // numface indique le numéro de la face chargée
331 // retourne le second membre résultant
332 // -> version explicite à t
333 Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
334 { return HexaMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa);} ;
335 // -> version explicite à tdt
336 Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
337 { return HexaMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa);} ;
338 // -> implicite,
339 // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
340 // retourne le second membre et la matrice de raideur correspondant
341 ResRaid SMR_charge_surfacique_I
342 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const ParaAlgoControle & pa)
;
343
344 // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
345 // presUniDir indique le vecteur appliquée
346 // numface indique le numéro de la face chargée
347 // retourne le second membre résultant
348 // -> explicite à t
349 Vecteur SM_charge_presUniDir_E_t(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
350 { return HexaMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,false,pa);} ;
351 // -> explicite à tdt

```

```

352     Vecteur SM_charge_presUniDir_E_tdt(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
353     { return HexaMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,true,pa); } ;
354     // -> implicite,
355     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
356     // retourne le second membre et la matrice de raideur correspondant
357     ResRaid SMR_charge_presUniDir_I(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa);
358
359     // cas d'un chargement lineique, sur les aretes frontieres des éléments
360     // force indique la force lineique appliquée
361     // numarete indique le numéro de l'arete chargée
362     // retourne le second membre résultant
363     // NB: il y a une définition par défaut pour les éléments qui n'ont pas
364     // d'arete externe -> message d'erreur d'où le virtuel et non virtuel pur
365     // -> explicite à t
366     Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
367     { return HexaMemb::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); } ;
368     // -> explicite à tdt
369     Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
370     { return HexaMemb::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); } ;
371     // -> implicite,
372     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
373     // retourne le second membre et la matrice de raideur correspondant
374     ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
375
376     // cas d'un chargement de type pression, sur les frontieres des éléments
377     // pression indique la pression appliquée
378     // numface indique le numéro de la face chargée
379     // retourne le second membre résultant
380     // NB: il y a une définition par défaut pour les éléments qui n'ont pas de
381     // surface externe -> message d'erreur d'où le virtuel et non virtuel pur
382     // -> explicite à t
383     Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
384     { return HexaMemb::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa); } ;
385     // -> explicite à tdt
386     Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
387     { return HexaMemb::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa); } ;
388     // -> implicite,
389     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
390     // retourne le second membre et la matrice de raideur correspondant
391     ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
392
393     // cas d'un chargement surfacique hydrostatique,
394     // poidvol: indique le poids volumique du liquide
395     // M_liquide : un point de la surface libre
396     // dir_normal_liquide : direction normale à la surface libre
397     // sans_limitation : indique s'il y a une limitation du calcul pour les seuls positions négatives
398     // retourne le second membre résultant
399     // -> explicite à t
400     Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
401     ,const ParaAlgoControle & pa,bool sans_limitation)
402     { return
403     HexaMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation); } ;
404     // -> explicite à tdt
405     Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
406     ,const ParaAlgoControle & pa,bool sans_limitation)
407     { return
408     HexaMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation); } ;
409     // -> implicite,
410     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
411     // retourne le second membre et la matrice de raideur correspondant
412     ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
413     ,const ParaAlgoControle & pa,bool sans_limitation) ;
414
415     // cas d'un chargement surfacique hydro-dynamique,
416     // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
417     // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc unitaire)
418     // une suivant la direction normale à la vitesse de type portance
419     // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
420     // une suivant la vitesse tangente de type frottement visqueux
421     // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
422     // coef_mul: est un coefficient multiplicateur global (de tout)
423     // retourne le second membre résultant
424     // -> explicite à t
425     Vecteur SM_charge_hydrodynamique_E_t( CourbelD* frot_fluid,const double& poidvol
426     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul

```

```

428                                     , CourbelD* coef_aero_t
429                                     ,const ParaAlgoControle & pa)
430     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
431 // -> explicite à tdt
432     Vecteur SM_charge_hydrodynamique_E_tdt( CourbelD* frot_fluid,const double& poidvol
433                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
434                                     , CourbelD* coef_aero_t
435                                     ,const ParaAlgoControle & pa)
436     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};
437 // -> implicite,
438 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
439 // retourne le second membre et la matrice de raideur correspondant
440     ResRaid SMR_charge_hydrodynamique_I( CourbelD* frot_fluid,const double& poidvol
441                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
442                                     , CourbelD* coef_aero_t
443                                     ,const ParaAlgoControle & pa) ;
444
445 // ===== définition et/ou construction des frontières =====
446
447 // Calcul des frontieres de l'element
448 // creation des elements frontieres et retour du tableau de ces elements
449 // la création n'a lieu qu'au premier appel
450 // ou lorsque l'on force le paramètre force a true
451 // dans ce dernier cas seul les frontière effacées sont recréée
452     Tableau <ElFrontiere*> const & Frontiere(bool force = false);
453
454 // ramène la frontière point
455 // éventuellement création des frontieres points de l'element et stockage dans l'element
456 // si c'est la première fois sinon il y a seulement retour de l'elements
457 // a moins que le paramètre force est mis a true
458 // dans ce dernier cas la frontière effacée est recréée
459 // num indique le numéro du point à créer (numérotation EF)
460 // ElFrontiere* const Frontiere_points(int num,bool force = false);
461
462 // ramène la frontière linéique
463 // éventuellement création des frontieres linéique de l'element et stockage dans l'element
464 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
465 // a moins que le paramètre force est mis a true
466 // dans ce dernier cas la frontière effacée est recréée
467 // num indique le numéro de l'arête à créer (numérotation EF)
468 // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
469
470 // ramène la frontière surfacique
471 // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
472 // si c'est la première fois sinon il y a seulement retour de l'elements
473 // a moins que le paramètre force est mis a true
474 // dans ce dernier cas la frontière effacée est recréée
475 // num indique le numéro de la surface à créer (numérotation EF)
476 // ElFrontiere* const Frontiere_surfacique(int num,bool force = false);
477
478 // 3) methodes propres a l'element
479
480 // ajout du tableau specific de ddl des noeuds
481 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
482 // des noeuds constituant l'element
483     void ConstTabDdl();
484
485 public :
486 // ----- definition de la classe conteneur de donnees communes -----
487     class DonnComHexa
488     { public :
489         // hexal,hexaeEr et hexaeMas doivent pointer sur des elements deja existants
490         DonnComHexa (ElemGeomC0* hexal,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
491                     Met_abstraite& met_gene,
492                     Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,ElemGeomC0* hexaeEr
493                     ,GeomQuadrangle& quadS,GeomSeg& seggS,
494                     Vecteur& residu_int,Mat_pleine& raideur_int,
495                     Tableau <Vecteur* > & residus_extN,Tableau <Mat_pleine* > & raideurs_extN,
496                     Tableau <Vecteur* > & residus_extA,Tableau <Mat_pleine* > & raideurs_extA,
497                     Tableau <Vecteur* > & residus_extS,Tableau <Mat_pleine* > & raideurs_extS,
498                     Mat_pleine& mat_masse,ElemGeomC0* hexaeMas,int nbi,
499                     ElemGeomC0* hexaeHourg
500                 ) ;
501         DonnComHexa (DonnComHexa& a) ;
502         ~DonnComHexa () ;
503         // variables
504         ElemGeomC0* hexae; // contient les fonctions d'interpolation et
505                             // les derivees
506         DdlElement tab_ddl; // tableau des degres
507                             //de liberte des noeuds de l'element commun a tous les
508                             // elements
509         Met_abstraite metrique;
510         Mat_pleine matGeom ; // matrice géométrique

```

```

511     Mat_pleine matInit ; // matrice initiale
512     Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
513     Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
514     Tableau < Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
515     // ---- concernant les frontières et particulièrement le calcul de second membre
516     GeomQuadrangle quadraS; // contient les fonctions d'interpolation et les derivees pour les
surfaces
517     GeomSeg segS; // " " "
518     // calcul d'erreur
519     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
520     // d'erreur : contraintes
521     DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud, servant pour le
calcul
522     // d'erreur : contraintes, en fait pour l'assemblage
523
524     Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
525     Mat_pleine raidErr; // raideur pour le calcul d'erreur
526     ElemGeomC0* hexaedEr; // contient les fonctions d'interpolation et
527     // les derivees pour le calcul du hessien dans
528     // la résolution de la fonctionnelle d'erreur
529     // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
-----
530     // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
531     Vecteur residu_interne;
532     Mat_pleine raideur_interne;
533     Tableau <Vecteur* > residus_externesN; // pour les noeuds
534     Tableau <Mat_pleine* > raideurs_externesN; // pour les noeuds
535     Tableau <Vecteur* > residus_externesA; // pour les aretes
536     Tableau <Mat_pleine* > raideurs_externesA; // pour les aretes
537     Tableau <Vecteur* > residus_externesS; // pour les surfaces
538     Tableau <Mat_pleine* > raideurs_externesS; // pour les surfaces
539     // ----- données concernant la dynamique -----
540     Mat_pleine matrice_masse;
541     ElemGeomC0* hexaedMas; // contient les fonctions d'interpolation et ...
542     // pour les calculs relatifs à la masse
543     // ----- blocage éventuel d'hourglass
544     // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
Cal_explici_hourglass
545     ElemGeomC0* hexaedHourg; // contient les fonctions d'interpolation
546     };
547
548     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
549     // et un pointeur sur les données statiques communes
550     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
551     // classe est défini. Son allocation est effectuée dans les classes dérivées
552     class UneFois
553     { public :
554         UneFois () ; // constructeur par défaut
555         ~UneFois () ; // destructeur
556
557         // VARIABLES :
558     public :
559         HexaMemb::DonnComHexa * doCoMemb;
560
561         // indicateurs permettant de dimensionner seulement au premier passage
562         // utilise dans "CalculResidu" et "Calcul_implicit"
563         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
564         int CalimpPrem;
565         int dualSortHexa; // pour la sortie des valeurs au pt d'integ
566         int CalSMlin_t; // pour les seconds membres concernant les arretes
567         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
568         int CalSMRlin; // pour les seconds membres concernant les arretes
569         int CalSMsurf_t; // pour les seconds membres concernant les surfaces
570         int CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
571         int CalSMRsurf; // pour les seconds membres concernant les surfaces
572         int CalSMvol_t; // pour les seconds membres concernant les volumes
573         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
574         int CalSMvol; // pour les seconds membres concernant les volumes
575         int CalDynamique; // pour le calcul de la matrice de masse
576         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
577         // ----- sauvegarde du nombre d'élément en cours -----
578         int nbelem_in_Prog;
579     };
580
581     // -----
582
583     protected :
584
585     // VARIABLES PRIVEES :
586     UneFois * unefois; // pointeur défini dans la classe dérivée
587
588     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
589     LesPtIntegMecaInterne lesPtMecaInt;
590
591
592     // type structuré et pointeur pour construire les éléments
593     // le pointeur est défini dans le type dérivé

```

```

594     class NombresConstruire
595     { public:
596         NombresConstruire() : nbne(0), nbneS(0), nbneA(0), nbi(0)
597             , nbiEr(0), nbiV(0), nbiS(0), nbiA(0), nbiMas(0), nbiHour(0) {};
598         int nbne; // le nombre de noeud de l'élément
599         int nbneS; // le nombre de noeud des facettes
600         int nbneA; // le nombre de noeud des aretes
601         int nbi; // le nombre de point d'intégration pour le calcul mécanique
602         int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
603         int nbiV; // le nombre de point d'intégration pour le calcul de second membre volumique
604         int nbiS; // le nombre de point d'intégration pour le calcul de second membre surfacique
605         int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
606             int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse
607             int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
608     };
609     NombresConstruire * nombre; // le pointeur défini dans la classe dérivée d'hexamemb
610
611     // =====>> methodes appelees par les classes derivees <<=====
612     // Fonction d'initialisation servant dans les classes derivant
613     // au niveau du constructeur
614     // les pointeurs d'éléments géométriques sont non nul uniquement lorsque doCoMemb est null
615     // c'est-à-dire pour l'initialisation
616     HexaMemb::DonnComHexa* Init
617         (ElemGeomC0* hexa, ElemGeomC0* hexaEr, ElemGeomC0* hexaMas
618             , ElemGeomC0* hexaeHourg, bool sans_init_noeud = false);
619     // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
620     void Destruction();
621
622     // adressage des frontières linéiques et surfacique
623     // définit dans les classes dérivées, et utilisées pour la construction des frontières
624     virtual ElFrontiere* new_frontiere_lin(int , Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
625     virtual ElFrontiere* new_frontiere_surf(int , Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
626
627     // =====>> methodes virtuelles dérivant d'ElemMeca =====
628     // ramene la dimension des tenseurs contraintes et déformations de l'élément
629     int Dim_sig_eps() const {return 3;};
630
631     //----- fonctions uniquement a usage interne -----
632     private :
633     // definition des données communes : doCoHexa
634     HexaMemb::DonnComHexa* Def_DonneeCommune(ElemGeomC0* hexa, ElemGeomC0* hexaEr
635         , ElemGeomC0* hexaMas, ElemGeomC0* hexaeHourg);
636     // Calcul du residu local a t ou tdt en fonction du booleen
637     Vecteur* CalculResidu (bool atdt, const ParaAlgoControle & pa);
638     // cas d'un chargement de type pression, sur les frontières des éléments
639     // pression indique la pression appliquée
640     // numface indique le numéro de la face chargée
641     // retourne le second membre résultant
642     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
643     Vecteur SM_charge_pression_E(double pression, Fonction_nD* pt_fonct, int numFace, bool atdt, const
644     ParaAlgoControle & pa);
645     // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
646     // presUniDir indique le vecteur appliquée
647     // numface indique le numéro de la face chargée
648     // retourne le second membre résultant
649     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
650     Vecteur SM_charge_presUniDir_E
651     (const Coordonnee& presUniDir, Fonction_nD* pt_fonct, int numFace, bool atdt, const
652     ParaAlgoControle & pa) ;
653     // cas d'un chargement lineique, sur les aretes frontières des éléments
654     // force indique la force lineique appliquée
655     // numarete indique le numéro de l'arete chargée
656     // retourne le second membre résultant
657     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
658     Vecteur SM_charge_lineique_E
659     (const Coordonnee& force, Fonction_nD* pt_fonct, int numArete, bool atdt, const
660     ParaAlgoControle & pa);
661     // cas d'un chargement surfacique, sur les frontières des éléments
662     // force indique la force surfacique appliquée
663     // numface indique le numéro de la face chargée
664     // retourne le second membre résultant
665     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
666     Vecteur SM_charge_surfacique_E
667     (const Coordonnee& force, Fonction_nD* pt_fonct, int numFace, bool atdt, const
668     ParaAlgoControle & pa);
669     // calcul des seconds membres suivant les chargements
670     // cas d'un chargement volumique,
671     // force indique la force volumique appliquée
672     // retourne le second membre résultant
673     // ici on l'épaisseur de l'élément pour constituer le volume
674     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
675     Vecteur SM_charge_volumique_E
676     (const Coordonnee& force, Fonction_nD* pt_fonct, bool atdt, const ParaAlgoControle & pa, bool
677     sur_volume_finale_);
678     // cas d'un chargement surfacique hydrostatique,
679     // poidvol: indique le poids volumique du liquide
680     // M_liquide : un point de la surface libre

```



```

676 // dir_normal_liquide : direction normale à la surface libre
677 // sans_limitation : indique s'il y a une limitation du calcul pour les seuls positions négatives
678 // retourne le second membre résultant
679 // -> explicite à t
680 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
681 ,int numFace,const Coordonnee& M_liquide,bool atdt
682 ,const ParaAlgoControle & pa,bool sans_limitation);
683 // cas d'un chargement surfacique hydro-dynamique,
684 // voir méthode explicite plus haut, pour les arguments
685 // retourne le second membre résultant
686 // bool atdt : permet de spécifier à t ou a t+dt
687 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
688 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
689 , CourbelD* coef_aero_t,bool atdt
690 ,const ParaAlgoControle & pa) ;
691 };
692
693 /// @} // end of group
694
695 #endif
696
697
698
699

```

## 7.160 HexaQ.h

```

1 // FICHER : Hexa.h
2 // CLASSE : Hexa
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 * DATE: 23/01/97 *
34 * * $ *
35 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
36 * * $ *
37 * PROJET: Herezh++ *
38 * * $ *
39 *****/
40 * BUT: La classe Hexa permet de declarer des elements *
41 * Hexaedriques Triquadratiques incomplets et de realiser *
42 * le calcul du residu local et de la raideur locale pour une loi de *
43 * comportement donnee. La dimension de l'espace pour un tel element *
44 * est 3. *
45 * *
46 * l'interpolation est Triquadratiques incomplet a 20 noeuds,le nombre *
47 * de point d'integration est de 8. *
48 * ***** *
49 * *
49 * VERIFICATION: *
50 * *
51 * ! date ! auteur ! but ! *
52 * ----- *
53 * ! ! ! ! *
54 * * $ *
55 * ***** *
56 * MODIFICATIONS: *

```

```

57 *      ! date !      auteur !      but      !      *
58 *      -----
59 *      $      *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef HEXAQ_H
67 #define HEXAQ_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQuad.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class HexaQ : public HexaMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93
94         // Constructeur par default
95         HexaQ ();
96
97         // Constructeur fonction d'un numero
98         // d'identification
99         HexaQ (int num_mail,int num_id);
100
101         // Constructeur fonction d'un numero de maillage et d'identification et
102         // du tableau de connexite des noeuds
103         HexaQ (int num_mail,int num_id,const Tableau<Noeud *>& tab);
104
105         // Constructeur de copie
106         HexaQ (const HexaQ& hexa);
107
108
109         // DESTRUCTEUR :
110         ~HexaQ ();
111
112
113         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
114         // méthode virtuelle
115         Element* Nevez_copie() const { Element * el= new HexaQ(*this); return el;};
116
117         // Surcharge de l'operateur = : realise l'egalite entre deux instances de HexaQ
118         HexaQ& operator= (HexaQ& hexa);
119
120         // METHODES :
121         // 1) derivant des virtuelles pures
122
123         // affichage dans la sortie transmise, des variables duales "nom"
124         // aux differents points d'integration
125         // dans le cas ou nom est vide, affichage de "toute" les variables
126         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
127
128         // 2) derivant des virtuelles
129         // renseignement d'un élément quadratique incomplet à partir d'un élément linéaire de même type
130         // retourne les nouveaux noeuds construits à partir de l'interpolation linéaire.
131         // dans le cas où l'élément n'est pas concerné, retourne une liste vide
132         // ramène également une liste de même dimension contenant les bornes en numéros de noeuds
133         // entre lesquelles il faut définir les nouveaux numéros de noeuds si l'on veut conserver
134         // une largeur de bande optimisée du même type
135         // nbnt+1: est le premier numéro de noeud utilisable pour les nouveaux noeuds
136         list <Noeud *> Construct_from_lineaire(const Element & elem,list <DeuxEntiers> & li_bornes, int
nbnt);
137         // 3) methodes propres a l'element
138
139         protected :
140
141         // adressage des frontieres linéiques et surfacique
142         // définit dans les classes dérivées, et utilisées pour la construction des frontieres

```

```

143     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
144     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
145     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
146     { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
147 // VARIABLES PRIVEES :
148 // place memoire commune a tous les elements TriaMembl1
149 static HexaMemb::DonnComHexa * doCoHexa;
150 // idem mais pour les indicateurs qui servent pour l'initialisation
151 static HexaMemb::UneFois  uneFois;
152
153 class NombresConstruireHexaQ : public NombresConstruire
154 { public: NombresConstruireHexaQ();
155   };
156 static NombresConstruireHexaQ nombre_V; // les nombres propres à l'élément
157
158 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
160 class ConsHexaQ : public ConstrucElement
161 { public : ConsHexaQ ()
162   { NouvelleTypeElement nouv (HEXAEDRE,QUADRATIQUE,MECA_SOLIDE_DEFORMABLE,this);
163   if (ParaGlob::NiveauImpression() >= 4)
164     cout << "\n initialisation HexaQ" << endl;
165     Element::listTypeElement.push_back(nouv);
166   };
167   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168   {Element * pt;
169   pt = new HexaQ (num_maill,num) ;
170   return pt;};
171   // ramene true si la construction de l'element est possible en fonction
172   // des variables globales actuelles: ex en fonction de la dimension
173   bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174   };
175 static ConsHexaQ consHexaQ;
176 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @} // end of group
179 #endif
180
181
182
183

```

## 7.161 HexaQ\_cm1pti.h

```

1 // FICHER : Hexa.h
2 // CLASSE : Hexa
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      09/07/2010
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      La classe Hexa_cm1pti permet de declarer des elements
41 * Hexaedriques Triquadratiques incomplets et de realiser

```

```

42 * le calcul du residu local et de la raideur locale pour une loi de *
43 * comportement donnee. La dimension de l'espace pour un tel element *
44 * est 3. *
45 * *
46 * l'interpolation est Triquadratiques incomplet a 20 noeuds, le nombre *
47 * de point d'integration est de 1. *
48 * ***** *
49 * VERIFICATION: *
50 * *
51 * ! date ! auteur ! but ! *
52 * ----- *
53 * ! ! ! ! *
54 * * $ *
55 * ***** *
56 * MODIFICATIONS: *
57 * ! date ! auteur ! but ! *
58 * ----- *
59 * * $ *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef HEXAQ_CMLPTI_H
67 #define HEXAQ_CMLPTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQuad.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class HexaQ_cmlpti : public HexaMemb
88 {
89 public :
90
91 // CONSTRUCTEURS :
92
93 // Constructeur par default
94 HexaQ_cmlpti ();
95
96 // Constructeur fonction d'un numero
97 // d'identification
98 HexaQ_cmlpti (int num_mail,int num_id);
99
100 // Constructeur fonction d'un numero de maillage et d'identification et
101 // du tableau de connexite des noeuds
102 HexaQ_cmlpti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
103
104 // Constructeur de copie
105 HexaQ_cmlpti (const HexaQ_cmlpti& hexa);
106
107
108 // DESTRUCTEUR :
109 ~HexaQ_cmlpti ();
110
111 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
112 // méthode virtuelle
113 Element* Nevez_copie() const { Element * el= new HexaQ_cmlpti(*this); return el;};
114
115 // Surcharge de l'operateur = : realise l'egalite entre deux instances de HexaQ_cmlpti
116 HexaQ_cmlpti& operator= (HexaQ_cmlpti& hexa);
117
118 // METHODES :
119 // 1) derivant des virtuelles pures
120
121 // affichage dans la sortie transmise, des variables duales "nom"
122 // aux differents points d'integration
123 // dans le cas ou nom est vide, affichage de "toute" les variables
124 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
125
126
127 // 2) derivant des virtuelles

```

```

128 // 3) methodes propres a l'element
129
130 protected :
131
132 // adressage des frontieres lineiques et surfacique
133 // definit dans les classes derivees, et utilisees pour la construction des frontieres
134 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
136 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137 { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
138 // VARIABLES PRIVEES :
139 // place memoire commune a tous les elements HexaQ_cmlpti
140 static HexaMemb::DonnComHexa * doCoHexa;
141 // idem mais pour les indicateurs qui servent pour l'initialisation
142 static HexaMemb::UneFois uneFois;
143
144 class NombresConstruireHexaQ_cmlpti : public NombresConstruire
145 { public: NombresConstruireHexaQ_cmlpti();
146 };
147 static NombresConstruireHexaQ_cmlpti nombre_V; // les nombres propres à l'élément
148
149 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
150 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
151 class ConsHexaQ_cmlpti : public ConstrucElement
152 { public : ConsHexaQ_cmlpti ()
153 { NouvelleTypeElement nouv (HEXAEDRE,QUADRATIQUE,MECA_SOLIDE_DEFORMABLE,this,"_cmlpti");
154 if (ParaGlob::NiveauImpression() >= 4)
155 cout << "\n initialisation HexaQ_cmlpti" << endl;
156 Element::listTypeElement.push_back(nouv);
157 };
158 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
159 {Element * pt;
160 pt = new HexaQ_cmlpti (num_maill,num) ;
161 return pt;};
162 // ramene true si la construction de l'element est possible en fonction
163 // des variables globales actuelles: ex en fonction de la dimension
164 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
165 };
166 static ConsHexaQ_cmlpti consHexaQ_cmlpti;
167 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
168 };
169 /// @} // end of group
170 #endif
171
172
173
174

```

## 7.162 HexaQ\_cm27pti.h

```

1 // FICHER : Hexa.h
2 // CLASSE : Hexa
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 * DATE: 23/01/97 *
34 * * *
35 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *

```

```

36 *                                     $ *
37 *   PROJET:      Herezh++                                     *
38 *                                     $ *
39 * *****
40 *   BUT:   La classe Hexa_cm27pti permet de declarer des elements *
41 * Hexaedriques Triquadratiques incomplets et de realiser      *
42 * le calcul du residu local et de la raideur locale pour une loi de *
43 * comportement donnee. La dimension de l'espace pour un tel element *
44 * est 3.                                                         *
45 *                                                                 *
46 * l'interpolation est Triquadratiques incomplet a 20 noeuds,le nombre *
47 * de point d'integration est de 27.                             *
48 *   ***** *
49 *   *
49 *   VERIFICATION:                                             *
50 *                                                                 *
51 *   ! date !   auteur !           but                       ! *
52 *   ----- *
53 *   !           !           !                               ! *
54 *                                                                 *
55 *   ***** *
56 *   MODIFICATIONS:                                           *
57 *   ! date !   auteur !           but                       ! *
58 *   ----- *
59 *                                                                 *
60 * *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef HEXAQ_CM27PTI_H
67 #define HEXAQ_CM27PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQuad.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class HexaQ_cm27pti : public HexaMemb
88 {
89     public :
90
91         // CONSTRUCTEURS :
92
93         // Constructeur par default
94         HexaQ_cm27pti ();
95
96         // Constructeur fonction d'un numero
97         // d'identification
98         HexaQ_cm27pti (int num_mail,int num_id);
99
100        // Constructeur fonction d'un numero de maillage et d'identification et
101        // du tableau de connexite des noeuds
102        HexaQ_cm27pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
103
104        // Constructeur de copie
105        HexaQ_cm27pti (const HexaQ_cm27pti& hexa);
106
107
108
109        // DESTRUCTEUR :
110        ~HexaQ_cm27pti ();
111
112        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113        // méthode virtuelle
114        Element* Nevez_copie() const { Element * el= new HexaQ_cm27pti(*this); return el;};
115
116        // Surcharge de l'operateur = : realise l'egalite entre deux instances de HexaQ_cm27pti
117        HexaQ_cm27pti& operator= (HexaQ_cm27pti& hexa);
118
119        // METHODES :
120 // 1) derivant des virtuelles pures
121

```

```

122     // affichage dans la sortie transmise, des variables duales "nom"
123     // aux differents points d'integration
124     // dans le cas ou nom est vide, affichage de "toute" les variables
125     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127 // 2) derivant des virtuelles
128 // 3) methodes propres a l'element
129
130 protected :
131
132     // adressage des frontieres linéiques et surfacique
133     // définit dans les classes dérivées, et utilisées pour la construction des frontieres
134     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
136     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137     { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
138 // VARIABLES PRIVEES :
139 // place memoire commune a tous les elements HexaQ_cm27pti
140 static HexaMemb::DonnComHexa * doCoHexa;
141 // idem mais pour les indicateurs qui servent pour l'initialisation
142 static HexaMemb::UneFois uneFois;
143
144 class NombresConstruireHexaQ_cm27pti : public NombresConstruire
145 { public: NombresConstruireHexaQ_cm27pti();
146 };
147 static NombresConstruireHexaQ_cm27pti nombre_V; // les nombres propres à l'élément
148
149 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
150 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
151 class ConsHexaQ_cm27pti : public ConstrucElement
152 { public : ConsHexaQ_cm27pti ()
153     { NouvelleTypeElement nouv(HEXAEDRE,QUADRATIQUE,MECA_SOLIDE_DEFORMABLE,this,"_cm27pti");
154     if (ParaGlob::NiveauImpression() >= 4)
155     cout << "\n initialisation HexaQ_cm27pti" << endl;
156     Element::listTypeElement.push_back(nouv);
157     };
158     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
159     {Element * pt;
160     pt = new HexaQ_cm27pti (num_maill,num) ;
161     return pt;};
162     // ramene true si la construction de l'element est possible en fonction
163     // des variables globales actuelles: ex en fonction de la dimension
164     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
165 };
166 static ConsHexaQ_cm27pti consHexaQ_cm27pti;
167 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
168 };
169 /// @} // end of group
170 #endif
171
172
173
174

```

## 7.163 HexaQ\_cm64pti.h

```

1 // FICHER : Hexa.h
2 // CLASSE : Hexa
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //

```

```

30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      23/01/97
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      La classe Hexa_cm64pti permet de declarer des elements
41 * Hexaedriques Triquadratiques incomplets et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 3.
45 *
46 * l'interpolation est Triquadratiques incomplet a 20 noeuds, le nombre
47 * de point d'integration est de 64.
48 *   *****/
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !
55 *   *****/
56 *
57 *   MODIFICATIONS:
58 *
59 *   ! date !   auteur !           but
60 *   -----
61 *   *****/
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef HEXAQ_CM64PTI_H
67 #define HEXAQ_CM64PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQuad.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class HexaQ_cm64pti : public HexaMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93
94         // Constructeur par defaut
95         HexaQ_cm64pti ();
96
97         // Constructeur fonction d'un numero
98         // d'identification
99         HexaQ_cm64pti (int num_mail,int num_id);
100
101         // Constructeur fonction d'un numero de maillage et d'identification et
102         // du tableau de connexite des noeuds
103         HexaQ_cm64pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
104
105         // Constructeur de copie
106         HexaQ_cm64pti (const HexaQ_cm64pti& hexa);
107
108
109         // DESTRUCTEUR :
110         ~HexaQ_cm64pti ();
111
112         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113         // méthode virtuelle
114         Element* Nevez_copie() const { Element * el= new HexaQ_cm64pti(*this); return el;};
115

```



```

116         // Surcharge de l'operateur = : realise l'egalite entre deux instances de HexaQ_cm64pti
117         HexaQ_cm64pti& operator= (HexaQ_cm64pti& hexa);
118
119         // METHODES :
120 // 1) derivant des virtuelles pures
121
122         // affichage dans la sortie transmise, des variables duales "nom"
123         // aux differents points d'integration
124         // dans le cas ou nom est vide, affichage de "toute" les variables
125         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127 // 2) derivant des virtuelles
128 // 3) methodes propres a l'element
129
130     protected :
131
132         // adressage des frontieres linéiques et surfacique
133         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
134         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
136         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137         { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
138 // VARIABLES PRIVEES :
139 // place memoire commune a tous les elements HexaQ_cm64pti
140 static HexaMemb::DonnComHexa * doCoHexa;
141 // idem mais pour les indicateurs qui servent pour l'initialisation
142 static HexaMemb::UneFois  uneFois;
143
144 class NombresConstruireHexaQ_cm64pti : public NombresConstruire
145 { public: NombresConstruireHexaQ_cm64pti();
146 };
147 static NombresConstruireHexaQ_cm64pti nombre_V; // les nombres propres à l'élément
148
149 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
150 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
151 class ConsHexaQ_cm64pti : public ConstrucElement
152 { public : ConsHexaQ_cm64pti ()
153     { NouvelleTypeElement nouv(HEXAEDRE,QUADRATIQUE,MECA_SOLIDE_DEFORMABLE,this,"_cm64pti");
154     if (ParaGlob::NiveauImpression() >= 4)
155         cout << "\n initialisation HexaQ_cm64pti" << endl;
156         Element::listTypeElement.push_back(nouv);
157     };
158     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
159     {Element * pt;
160     pt = new HexaQ_cm64pti (num_maill,num) ;
161     return pt;};
162     // ramene true si la construction de l'element est possible en fonction
163     // des variables globales actuelles: ex en fonction de la dimension
164     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
165 };
166 static ConsHexaQ_cm64pti consHexaQ_cm64pti;
167 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
168 };
169 /// @} // end of group
170 #endif
171
172
173
174

```

## 7.164 HexaQComp.h

```

1 // FICHER : HexaQComp.h
2 // CLASSE : HexaQComp
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty

```

```

24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      17/03/2003
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      La classe HexaQComp permet de declarer des elements
41 * Hexaedriques Triquadratiques complets et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 3.
45 *
46 * l'interpolation est Triquadratiques complet a 27 noeuds, le nombre
47 * de point d'integration par défaut est de 8.
48 *
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur   !           but
53 *   -----
54 *   !           !           !
55 *   $
56 *
57 *   MODIFICATIONS:
58 *   ! date !   auteur   !           but
59 *   -----
60 *   $
61 *****/
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef HEXAQCOMP_H
67 #define HEXAQCOMP_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQC.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class HexaQComp : public HexaMemb
88 {
89     public :
90
91         // CONSTRUCTEURS :
92
93         // Constructeur par default
94         HexaQComp ();
95
96         // Constructeur fonction d'un numero
97         // d'identification
98         HexaQComp (int num_mail,int num_id);
99
100
101         // Constructeur fonction d'un numero de maillage et d'identification et
102         // du tableau de connexite des noeuds
103         HexaQComp (int num_mail,int num_id,const Tableau<Noeud *>& tab);
104
105         // Constructeur de copie
106         HexaQComp (const HexaQComp& hexa);
107
108
109         // DESTRUCTEUR :

```

```

110     ~HexaQComp ();
111
112     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113     // méthode virtuelle
114     Element* Nevez_copie() const { Element * el= new HexaQComp(*this); return el;};
115
116     // Surcharge de l'opérateur = : realise l'egalite entre deux instances de HexaQComp
117     HexaQComp& operator= (HexaQComp& hexa);
118
119     // METHODES :
120 // 1) derivant des virtuelles pures
121
122     // renseignement d'un élément complet à partir d'un élément incomplet de même type
123     // retourne les nouveaux noeuds construit à partir de l'interpolation incomplète.
124     // dans le cas l'élément n'est pas concerné, retourne une liste vide
125     // ramène également une liste de même dimension contenant les bornes en numéros de noeuds
126     // entre lesquelles il faut définir les nouveaux numéros de noeuds si l'on veut conserver
127     // une largeur de bande optimisée du même type
128     // nbnt+1: est le premier numéro de noeud utilisable pour les nouveaux noeuds
129     list <Noeud *> Construct_from_imcomplet(const Element & elem,list <DeuxEntiers> & li_bornes,int
nbnt);
130
131     // affichage dans la sortie transmise, des variables duales "nom"
132     // aux differents points d'integration
133     // dans le cas ou nom est vide, affichage de "toute" les variables
134     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
135
136 // 2) derivant des virtuelles
137 // 3) methodes propres a l'element
138
139     protected :
140
141     // adressage des frontières linéiques et surfacique
142     // définit dans les classes dérivées, et utilisées pour la construction des frontières
143     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
144     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
145     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
146     { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
147 // VARIABLES PRIVEES :
148 // place memoire commune a tous les elements TriaMemBl1
149 static HexaMemb::DonnComHexa * doCoHexa;
150 // idem mais pour les indicateurs qui servent pour l'initialisation
151 static HexaMemb::UneFois uneFois;
152
153 class NombresConstruireHexaQComp : public NombresConstruire
154 { public: NombresConstruireHexaQComp();
155 };
156 static NombresConstruireHexaQComp nombre_V; // les nombres propres à l'élément
157
158 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
160 class ConsHexaQComp : public ConstrucElement
161 { public : ConsHexaQComp ()
162 { NouvelleTypeElement nouv (HEXAEDRE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this);
163   if (ParaGlob::NiveauImpression() >= 4)
164     cout << "\n initialisation HexaQComp" << endl;
165   Element::listTypeElement.push_back(nouv);
166   };
167   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168   {Element * pt;
169   pt = new HexaQComp (num_maill,num) ;
170   return pt;};
171   // ramene true si la construction de l'element est possible en fonction
172   // des variables globales actuelles: ex en fonction de la dimension
173   bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174   };
175 static ConsHexaQComp consHexaQComp;
176 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @// end of group
179 #endif
180
181
182
183

```

## 7.165 HexaQComp\_cm1pti.h

```

1 // FICHER : HexaQComp_cm1pti.h
2 // CLASSE : HexaQComp_cm1pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.

```

```

8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           17/03/2003
34 *
35 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:         Herezh++
38 *
39 *
40 *   *****
41 *   BUT: La classe HexaQComp_cmlpti permet de declarer des elements
42 *   Hexaedriques Triquadratiques complets et de realiser
43 *   le calcul du residu local et de la raideur locale pour une loi de
44 *   comportement donnee. La dimension de l'espace pour un tel element
45 *   est 3.
46 *
47 *   l'interpolation est Triquadratiques complet a 27 noeuds, le nombre
48 *   de point d'integration par defaut est de 1.
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !           $
55 *
56 *   MODIFICATIONS:
57 *
58 *   ! date !   auteur !           but
59 *   -----
60 *   $
61 *   *****/
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef HEXAQCOMP_CM1PTI_H
67 #define HEXAQCOMP_CM1PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQC.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class HexaQComp_cmlpti : public HexaMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93

```

```

94     // Constructeur par défaut
95     HexaQComp_cmlpti ();
96
97     // Constructeur fonction d'un numero
98     // d'identification
99     HexaQComp_cmlpti (int num_mail,int num_id);
100
101     // Constructeur fonction d'un numero de maillage et d'identification et
102     // du tableau de connexite des noeuds
103     HexaQComp_cmlpti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
104
105     // Constructeur de copie
106     HexaQComp_cmlpti (const HexaQComp_cmlpti& hexa);
107
108
109     // DESTRUCTEUR :
110     ~HexaQComp_cmlpti ();
111
112     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113     // méthode virtuelle
114     Element* Nevez_copie() const { Element * el= new HexaQComp_cmlpti(*this); return el;};
115
116     // Surcharge de l'operateur = : realise l'egalite entre deux instances de HexaQComp_cmlpti
117     HexaQComp_cmlpti& operator= (HexaQComp_cmlpti& hexa);
118
119     // METHODES :
120 // 1) derivant des virtuelles pures
121
122     // affichage dans la sortie transmise, des variables duales "nom"
123     // aux differents points d'integration
124     // dans le cas ou nom est vide, affichage de "toute" les variables
125     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127 // 2) derivant des virtuelles
128 // 3) methodes propres a l'element
129
130     protected :
131
132     // adressage des frontières linéiques et surfacique
133     // définit dans les classes dérivées, et utilisées pour la construction des frontières
134     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
136     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137     { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
138 // VARIABLES PRIVEES :
139 // place memoire commune a tous les elements TriaMemBl1
140 static HexaMemb::DonnComHexa * doCoHexa;
141 // idem mais pour les indicateurs qui servent pour l'initialisation
142 static HexaMemb::UneFois uneFois;
143
144 class NombresConstruireHexaQComp_cmlpti : public NombresConstruire
145 { public: NombresConstruireHexaQComp_cmlpti();
146 };
147 static NombresConstruireHexaQComp_cmlpti nombre_V; // les nombres propres à l'élément
148
149 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
150 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
151 class ConsHexaQComp_cmlpti : public ConstrucElement
152 { public : ConsHexaQComp_cmlpti ()
153 { NouvelleTypeElement nouv (HEXAEDRE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cmlpti");
154   if (ParaGlob::NiveauImpression() >= 4)
155     cout << "\n initialisation HexaQComp_cmlpti" << endl;
156   Element::listTypeElement.push_back(nouv);
157   };
158   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
159   {Element * pt;
160     pt = new HexaQComp_cmlpti (num_maill,num) ;
161     return pt;};
162   // ramene true si la construction de l'element est possible en fonction
163   // des variables globales actuelles: ex en fonction de la dimension
164   bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
165   };
166   static ConsHexaQComp_cmlpti consHexaQComp_cmlpti;
167   static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
168 };
169 /// @} // end of group
170 #endif
171
172
173
174

```

## 7.166 HexaQComp\_cm27pti.h

```
1 // FICHER : HexaQComp_cm27pti.h
```

```

2 // CLASSE : HexaQComp_cm27pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          17/03/2003
34 *
35 *      AUTEUR:        G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *
40 *      *****
41 *      BUT: La classe HexaQComp_cm27pti permet de declarer des elements
42 *      Hexaedriques Triquadratiques complets et de realiser
43 *      le calcul du residu local et de la raideur locale pour une loi de
44 *      comportement donnee. La dimension de l'espace pour un tel element
45 *      est 3.
46 *      l'interpolation est Triquadratiques complet a 27 noeuds, le nombre
47 *      de point d'integration par défaut est de 27.
48 *      *****
49 *
50 *      VERIFICATION:
51 *
52 *      ! date !   auteur !           but
53 *      -----
54 *      !           !           !           $
55 *      *****
56 *      MODIFICATIONS:
57 *
58 *      ! date !   auteur !           but
59 *      -----
60 *      $
61 *      *****/
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef HEXAQCOMP_CM27PTI_H
67 #define HEXAQCOMP_CM27PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQC.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class HexaQComp_cm27pti : public HexaMemb

```

```

88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93
94         // Constructeur par défaut
95         HexaQComp_cm27pti ();
96
97         // Constructeur fonction d'un numero
98         // d'identification
99         HexaQComp_cm27pti (int num_maill,int num_id);
100
101         // Constructeur fonction d'un numero de maillage et d'identification et
102         // du tableau de connexite des noeuds
103         HexaQComp_cm27pti (int num_maill,int num_id,const Tableau<Noeud *>& tab);
104
105         // Constructeur de copie
106         HexaQComp_cm27pti (const HexaQComp_cm27pti& hexa);
107
108
109         // DESTRUCTEUR :
110         ~HexaQComp_cm27pti ();
111
112         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113         // méthode virtuelle
114         Element* Nevez_copie() const { Element * el= new HexaQComp_cm27pti(*this); return el;};
115
116         // Surchage de l'operateur = : realise l'egalite entre deux instances de HexaQComp_cm27pti
117         HexaQComp_cm27pti& operator= (HexaQComp_cm27pti& hexa);
118
119         // METHODES :
120 // 1) derivant des virtuelles pures
121
122         // affichage dans la sortie transmise, des variables duales "nom"
123         // aux differents points d'integration
124         // dans le cas ou nom est vide, affichage de "toute" les variables
125         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127 // 2) derivant des virtuelles
128 // 3) methodes propres a l'element
129
130     protected :
131
132         // adressage des frontières linéiques et surfacique
133         // définit dans les classes dérivées, et utilisées pour la construction des frontières
134         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
136         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137         { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
138     // VARIABLES PRIVEES :
139     // place memoire commune a tous les elements TriaMemBl1
140     static HexaMemb::DonnComHexa * doCoHexa;
141     // idem mais pour les indicateurs qui servent pour l'initialisation
142     static HexaMemb::UneFois uneFois;
143
144     class NombresConstruireHexaQComp_cm27pti : public NombresConstruire
145     { public: NombresConstruireHexaQComp_cm27pti ();
146     };
147     static NombresConstruireHexaQComp_cm27pti nombre_V; // les nombres propres à l'élément
148
149     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
150     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
151     class ConsHexaQComp_cm27pti : public ConstrucElement
152     { public : ConsHexaQComp_cm27pti ()
153     { NouvelleTypeElement nouv (HEXAEDRE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cm27pti");
154       if (ParaGlob::NiveauImpression() >= 4)
155         cout << "\n initialisation HexaQComp_cm27pti" << endl;
156       Element::listTypeElement.push_back(nouv);
157     };
158     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
159     {Element * pt;
160       pt = new HexaQComp_cm27pti (num_maill,num) ;
161       return pt;};
162     // ramene true si la construction de l'element est possible en fonction
163     // des variables globales actuelles: ex en fonction de la dimension
164     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
165     };
166     static ConsHexaQComp_cm27pti consHexaQComp_cm27pti;
167     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
168 };
169 /// @} // end of group
170 #endif
171
172
173
174

```

## 7.167 HexaQComp\_cm64pti.h

```

1 // FICHER : HexaQComp_cm64pti.h
2 // CLASSE : HexaQComp_cm64pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      17/03/2003
34 *
35 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      BUT:      La classe HexaQComp_cm64pti permet de declarer des elements
41 * Hexaedriques Triquadratiques complets et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 3.
45 *
46 * l'interpolation est Triquadratiques complet a 27 noeuds,le nombre
47 * de point d'integration par défaut est de 8.
48 *
49 *
50 *
51 *      VERIFICATION:
52 *
53 *      ! date !      auteur !      but
54 *      -----
55 *
56 *      MODIFICATIONS:
57 *      ! date !      auteur !      but
58 *      -----
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef HEXAQCOMP_CM64PTI_H
67 #define HEXAQCOMP_CM64PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "HexaMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQC.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///

```



```

85
86
87 class HexaQComp_cm64pti : public HexaMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93
94         // Constructeur par défaut
95         HexaQComp_cm64pti ();
96
97         // Constructeur fonction d'un numero
98         // d'identification
99         HexaQComp_cm64pti (int num_mail,int num_id);
100
101         // Constructeur fonction d'un numero de maillage et d'identification et
102         // du tableau de connexite des noeuds
103         HexaQComp_cm64pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
104
105         // Constructeur de copie
106         HexaQComp_cm64pti (const HexaQComp_cm64pti& hexa);
107
108
109         // DESTRUCTEUR :
110         ~HexaQComp_cm64pti ();
111
112         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113         // méthode virtuelle
114         Element* Nevez_copie() const { Element * el= new HexaQComp_cm64pti(*this); return el;};
115
116         // Surcharge de l'operateur = : realise l'egalite entre deux instances de HexaQComp_cm64pti
117         HexaQComp_cm64pti& operator= (HexaQComp_cm64pti& hexa);
118
119         // METHODES :
120 // 1) derivant des virtuelles pures
121
122         // affichage dans la sortie transmise, des variables duales "nom"
123         // aux differents points d'integration
124         // dans le cas ou nom est vide, affichage de "toute" les variables
125         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127 // 2) derivant des virtuelles
128 // 3) methodes propres a l'element
129
130     protected :
131
132         // adressage des frontières linéiques et surfacique
133         // définit dans les classes dérivées, et utilisées pour la construction des frontières
134         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
136         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137         { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
138     // VARIABLES PRIVEES :
139     // place memoire commune a tous les elements TriaMembL1
140     static HexaMemb::DonnComHexa * doCoHexa;
141     // idem mais pour les indicateurs qui servent pour l'initialisation
142     static HexaMemb::UneFois uneFois;
143
144     class NombresConstruireHexaQComp_cm64pti : public NombresConstruire
145     { public: NombresConstruireHexaQComp_cm64pti ();
146     };
147     static NombresConstruireHexaQComp_cm64pti nombre_V; // les nombres propres à l'élément
148
149     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
150     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
151     class ConsHexaQComp_cm64pti : public ConstrucElement
152     { public : ConsHexaQComp_cm64pti ()
153     { NouvelleTypeElement nouv (HEXAEDRE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cm64pti");
154       if (ParaGlob::NiveauImpression() >= 4)
155         cout << "\n initialisation HexaQComp_cm64pti" << endl;
156       Element::listTypeElement.push_back(nouv);
157     };
158     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
159     {Element * pt;
160       pt = new HexaQComp_cm64pti (num_maill,num) ;
161       return pt;};
162     // ramene true si la construction de l'element est possible en fonction
163     // des variables globales actuelles: ex en fonction de la dimension
164     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
165     };
166     static ConsHexaQComp_cm64pti consHexaQComp_cm64pti;
167     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
168 };
169 /// @} // end of group
170 #endif
171

```

172  
173  
174

## 7.168 LesChargeExtSurElement.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           06/03/2013
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *****/
38 *   BUT: Classe pour stocker l'ensemble des informations concernant
39 *         les chargements externes sur l'élément
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date ! auteur ! but
46 *   -----
47 *   ! ! !
48 *   *****
49 *
50 *   MODIFICATIONS:
51 *
52 *   ! date ! auteur ! but
53 *   -----
54 *   $
55 *****/
56
57 #ifndef LESCHARGEEXTSURELEMENT_H
58 #define LESCHARGEEXTSURELEMENT_H
59
60 #include <iostream>
61 using namespace std; //introduces namespace std
62 #include <stdlib.h>
63 #include "Tableau_T.h"
64 #include "ParaGlob.h"
65 #include "Coordonnee.h"
66
67 /// @addtogroup Groupe_concernant_le_chargement
68 /// @{
69
70 class Pression_appliquee
71 { public: double press;Coordonnee P;
72   Pression_appliquee(): press(0),P(ParaGlob::Dimension()) {};
73   Pression_appliquee(const Pression_appliquee & a):
74     press(a.press),P(a.P) {};
75   Pression_appliquee& operator= (const Pression_appliquee& a)
76     {press=a.press;P=a.P;return (*this);};
77   void Zero() {press=0.;P.Zero();};
78   // surcharge de l'operator de lecture
79   friend istream & operator >> (istream & ent, Pression_appliquee & a)

```

```

79         { string nom; ent » nom » a.press » nom » a.P; return ent;
80     };
81     // surcharge de l'operator d'écriture
82     friend ostream & operator « (ostream & sort , const Pression_appliquee & a)
83     { sort « " Press " « a.press « " " « " P " « a.P « " "; return sort;};
84     };
85 /// @} // end of group
86
87
88     /// @addtogroup Groupe_concernant_le_chargement
89     /// @{
90     ///
91
92     class Force_hydroDyna
93     { public: Coordonnee F_n,F_t,T;//trainée, portance, visqueux
94     Force_hydroDyna():
95     F_n(ParaGlob::Dimension()),F_t(ParaGlob::Dimension()),T(ParaGlob::Dimension()) {};
96     Force_hydroDyna(const Force_hydroDyna & a):
97     F_n(a.F_n),F_t(a.F_t),T(a.T) {};
98     Force_hydroDyna& operator= (const Force_hydroDyna& a)
99     {F_n=a.F_n;F_t=a.F_t;T=a.T;return (*this);};
100     void Zero() {F_n.Zero();F_t.Zero();T.Zero();};
101     // surcharge de l'operator de lecture
102     friend istream & operator » (istream & ent, Force_hydroDyna & a)
103     { string nom; ent » nom » a.F_n » nom » a.F_t » nom » a.T ; return ent;
104     // on ne fait pas a.F_n=a.F_t; ceci pour garder une valeur à t différentes éventuellement
105     };
106     // surcharge de l'operator d'écriture
107     friend ostream & operator « (ostream & sort , const Force_hydroDyna & a)
108     { sort « " F_n " « a.F_n « " F_t "« a.F_t « " T "« a.T « " "; return sort;};
109     };
110 /// @} // end of group
111
112 /// @addtogroup Groupe_concernant_le_chargement
113 /// @{
114 ///
115
116 class LesChargeExtSurElement
117 {
118 // surcharge de l'operator de lecture
119 friend istream & operator » (istream &, LesChargeExtSurElement &);
120 // surcharge de l'operator d'écriture
121 friend ostream & operator « (ostream &, const LesChargeExtSurElement &);
122
123 public :
124 // CONSTRUCTEURS :
125 // constructeur par défaut
126 LesChargeExtSurElement();
127 // constructeur fonction du nombre de points d'intégration et de la dimension de tenseurs
128 LesChargeExtSurElement(int nbpti, int dimtens);
129 // constructeur de copie
130 LesChargeExtSurElement(const LesChargeExtSurElement& lespti);
131
132 // DESTRUCTEUR :
133 ~LesChargeExtSurElement();
134
135 // METHODES PUBLIQUES :
136 // Surcharge de l'operateur =
137 LesChargeExtSurElement& operator= ( const LesChargeExtSurElement& lespti);
138
139 // initialisation à 0 de tous les conteneurs existants
140 void Zero();
141
142 // fonction d'accès
143 // les pressions éventuellement exercées sur les faces de l'élément
144 // lesPressionsExternes(i)(j) : pression au points d'intégration de surface j, pour la face i
145 Tableau <Tableau <Pression_appliquee> >* LesPressionsExternes() {return lesPressionsExternes;};
146 void LesPressionsExternes_Change_taille(int n);
147 // les forces volumiques qui s'exercent éventuellement sur l'élément
148 // force_volume(i) = la force de volume au pti I
149 Tableau < Coordonnee >* Force_volume() {return force_volume;};
150 void Force_volume_Change_taille(int n);
151 // les densité d'effort dont la direction reste fixe éventuellement exercées sur les faces de
152 l'élément
153 // lesEffortsDirFixe(i)(j) : effort au points d'intégration de surface j, pour la face i
154 Tableau <Tableau <Coordonnee> >* LesEffortsDirFixe() {return lesEffortsDirFixe;};
155 void LesEffortsDirFixe_Change_taille(int n);
156 // les densités éventuelles d'effort de surface dont la direction suit la face face de l'élément
157 // lesPressDir(i)(j) : effort au points d'intégration de surface j, pour la face i
158 Tableau <Tableau <Coordonnee> >* LesPressDir() {return lesPressDir;};
159 void LesPressDir_Change_taille(int n);
160 // les densités éventuelles d'effort de surface d'origine hydrodynamique
161 // lesHydroDyna(i)(j) : effort au points d'intégration de surface j, pour la face i
162 Tableau <Tableau <Force_hydroDyna> >* LesHydroDyna() {return lesHydroDyna;};
163 void LesHydroDyna_Change_taille(int n);

```

```

164 // charge linéique
165 // lesLineique(i)(j) : effort au points d'intégration de surface j, pour la face i
166 Tableau <Tableau <Coordonnee> >* LesLineique(){return lesLineique;};
167 void LesLineique_Change_taille(int n);
168 // charge linéique suivieuse
169 // lesLineiqueSuiveuse(i)(j) : effort au points d'intégration de surface j, pour la face i
170 Tableau <Tableau <Coordonnee> >* LesLineiqueSuiveuse(){return lesLineiqueSuiveuse;};
171 void LesLineiqueSuiveuse_Change_taille(int n);
172
173 //===== lecture écriture dans base info =====
174 // cas donne le niveau de la récupération
175 // = 1 : on récupère tout
176 // = 2 : on récupère uniquement les données variables (supposées comme telles)
177 void Lecture_base_info (ifstream& ent,const int cas);
178 // cas donne le niveau de sauvegarde
179 // = 1 : on sauvegarde tout
180 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
181 void Ecriture_base_info(ofstream& sort,const int cas);
182
183 protected:
184 // données protégées
185 // les deux premières sont historiques et l'enregistrement est sans doute sur-abondant
186 // pour l'instant on laisse tel quel, ensuite on pourra faire : Force_de_volume -> Coordonnee
idem pour pression appliquée
187
188 // les pressions éventuellement exercées sur les faces de l'élément
189 // lesPressionsExternes(i)(j) : pression au points d'intégration de surface j, pour la face i
190 Tableau <Tableau <Pression_appliquee> >* lesPressionsExternes;
191 // les forces volumiques qui s'exercent éventuellement sur l'élément
192 // force_volume(i) = la force de volume au pti I
193 Tableau < Coordonnee >* force_volume;
194 // les densité d'effort dont la direction reste fixe éventuellement exercées sur les faces de
l'élément
195 // lesEffortsDirFixe(i)(j) : effort au points d'intégration de surface j, pour la face i
196 Tableau <Tableau <Coordonnee> >* lesEffortsDirFixe;
197 // les densités éventuelles d'effort de surface dont la direction suit la face face de l'élément
198 // lesPressDir(i)(j) : effort au points d'intégration de surface j, pour la face i
199 Tableau <Tableau <Coordonnee> >* lesPressDir;
200 // les densités éventuelles d'effort de surface d'origine hydrodynamique
201 // lesHydroDyna(i)(j) : effort au points d'intégration de surface j, pour la face i
202 Tableau <Tableau <Force_hydroDyna> >* lesHydroDyna;
203 // charge linéique
204 // lesLineique(i)(j) : effort au points d'intégration de surface j, pour la face i
205 Tableau <Tableau <Coordonnee> >* lesLineique;
206 // charge linéique suivieuse
207 // lesLineiqueSuiveuse(i)(j) : effort au points d'intégration de surface j, pour la face i
208 Tableau <Tableau <Coordonnee> >* lesLineiqueSuiveuse;
209
210 };
211 /// @} // end of group
212
213 #endif

```

## 7.169 LesPtIntegMecalInterne.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****

```

```

31 *      DATE:          26/11/2006
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *****
38 *      BUT: Classe pour stocker l'ensemble des informations aux points
39 *          d'intégration mécanique (plutôt puissance interne)
40 *      Les données sont totalement, et volontairement non encapsulées,
41 *      l'accès est direct, l'encapsulation s'effectue à l'étage supérieur.*
42 *
43 *      $
44 *
45 *      VERIFICATION:
46 *
47 *      ! date ! auteur ! but
48 *      -----
49 *      ! ! !
50 *      $
51 *
52 *      MODIFICATIONS:
53 *      ! date ! auteur ! but
54 *      -----
55 *      $
56 *****/
57 #ifndef LESPTINTEGMECAINTERNE_H
58 #define LESPTINTEGMECAINTERNE_H
59 #include "PtIntegMecaInterne.h"
60
61 /// @addtogroup Groupe_concernant_les_points_integration
62 /// @{
63 ///
64
65 class LesPtIntegMecaInterne
66 {
67     // surcharge de l'operator de lecture
68     friend istream & operator » (istream &, LesPtIntegMecaInterne &);
69     // surcharge de l'operator d'écriture
70     friend ostream & operator « (ostream &, const LesPtIntegMecaInterne &);
71
72 public :
73     // CONSTRUCTEURS :
74     // constructeur par défaut
75     LesPtIntegMecaInterne();
76     // constructeur fonction du nombre de points d'intégration et de la dimension de tenseurs
77     LesPtIntegMecaInterne(int nbpti, int dimtens);
78     // constructeur de copie
79     LesPtIntegMecaInterne(const LesPtIntegMecaInterne& lespti);
80
81     // DESTRUCTEUR :
82     ~LesPtIntegMecaInterne();
83
84     // METHODES PUBLIQUES :
85     // Surcharge de l'operateur =
86     LesPtIntegMecaInterne& operator= ( const LesPtIntegMecaInterne& lespti);
87
88     // le tableau des grandeurs aux points d'intégration
89     // en lecture écriture
90     Tableau <PtIntegMecaInterne>& TabPtIntMeca() {return tabPtInt;};
91     // l'élément PtIntegMecaInterne de numéro i
92     PtIntegMecaInterne& operator () (int i) {return tabPtInt(i);};
93
94     // les tableaux des contraintes
95     Tableau <TenseurHH *>& TabSigHH() {return tabSigHH;}; // contrainte finale
96     Tableau <TenseurHH *>& TabSigHH_t() {return tabSigHH_t;}; // contrainte au début de l'incrément
97     // nombre de points d'intégration
98     int NbPti() const {return tabPtInt.Taille();};
99     // changement de taille donc de nombre de points d'intégration
100    // fonction du nombre de points d'intégration et de la dimension de tenseurs
101    // attention: il s'agit d'un dimensionnement pas défaut (les activations diverses
102    // sont ensuite à faire: par exemple pour les invariants)
103    void Change_taille_PtIntegMeca(int nbpti, int dimtens);
104    // idem, mais les instances ajoutées ou retirées ont la même dimension de tenseur que celles
105    // qui existent déjà
106    void Change_taille_PtIntegMeca(int nbpti);
107    // retour la dimension des tenseurs gérés
108    int DimTens() const;
109    // actualisation des grandeurs actives de t+dt vers t, pour celles qui existent
110    // sous ces deux formes
111    void TdtversT();
112    // actualisation des grandeurs actives de t vers tdt, pour celles qui existent
113    // sous ces deux formes
114    void TversTdt();

```

```

117
118 // ramène la compressibilité moyenne sur tous les points d'intégration
119 double CompressibiliteMoyenne() const;
120
121 //===== lecture écriture dans base info =====
122 // cas donne le niveau de la récupération
123 // = 1 : on récupère tout
124 // = 2 : on récupère uniquement les données variables (supposées comme telles)
125 void Lecture_base_info (ifstream& ent,const int cas);
126 // cas donne le niveau de sauvegarde
127 // = 1 : on sauvegarde tout
128 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
129 void Ecriture_base_info(ofstream& sort,const int cas);
130
131 protected:
132 // données protégées
133 // grandeurs aux points d'intégration
134 Tableau <PtIntegMecaInterne> tabPtInt;
135 // contraintes groupées sous forme de tableau, qui pointent sur celles de tabPtMecaInt
136 Tableau <TenseurHH *> tabSigHH; // contrainte finale
137 Tableau <TenseurHH *> tabSigHH_t; // contrainte au début de l'incrément
138
139 };
140 /// @} // end of group
141
142 #endif

```

## 7.170 PentaL.h

```

1 // FICHER : PentaL.h
2 // CLASSE : PentaL
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:     Herezh++
38 *
39 *****/
40 *      BUT:      La classe PentaL permet de declarer des elements
41 * Pentaedriques Trilineaires et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 3.
45 *
46 * l'interpolation est trilineaire a 6 noeuds, le nombre de
47 * point d'integration est par défaut de 2.
48 *
49 *
50 *
51 *      VERIFICATION:
52 *
53 *      ! date ! auteur ! but
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```

```

56 *      MODIFICATIONS:
57 *      ! date ! auteur ! but
58 *      -----
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef PENTAL_H
67 #define PENTAL_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "PentaMemb.h"
79 #include "FrontTriaLine.h"
80 #include "FrontQuadLine.h"
81 #include "FrontSegLine.h"
82
83
84 /// @addtogroup groupe_des_elements_finis
85 /// @{
86 ///
87
88
89 class Pental : public PentaMemb
90 {
91
92     public :
93
94         // CONSTRUCTEURS :
95
96         // Constructeur par default
97         Pental ();
98
99         // Constructeur fonction d'un numero
100        // d'identification
101        Pental (int num_mail,int num_id);
102
103        // Constructeur fonction d'un numero de maillage et d'identification et
104        // du tableau de connexite des noeuds
105        Pental (int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        Pental (const Pental& Penta);
109
110
111        // DESTRUCTEUR :
112        ~Pental ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new Pental(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de Pental
119        Pental& operator= (Pental& Penta);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // affichage dans la sortie transmise, des variables duales "nom"
125        // aux differents points d'integration
126        // dans le cas ou nom est vide, affichage de "toute" les variables
127        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129        // 2) derives
130        // 3) methodes propres a l'element
131
132        protected :
133
134        // adressage des frontieres linéiques et surfacique
135        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
136        // frontière linéique verticale (rectangle)
137        ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
138        { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
139        // frontière linéique horizontale (triangle)
140        ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
141        { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
142        // frontière surfacique verticale (rectangle)

```

```

143     ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
144     { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
145     // frontiere surfacique horizontale (triangle)
146     ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
147     { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
148
149     // VARIABLES PRIVEES :
150     // place memoire commune a tous les elements PentaL
151     static PentaMemb::DonnComPenta * doCoPentaL;
152     // idem mais pour les indicateurs qui servent pour l'initialisation
153     static PentaMemb::UneFois uneFois;
154
155     class NombresConstruirePentaL : public NombresConstruire
156     { public: NombresConstruirePentaL();
157       };
158     static NombresConstruirePentaL nombre_V; // les nombres propres à l'élément
159
160     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
161     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
162     class ConsPentaL : public ConstrucElement
163     { public : ConsPentaL ()
164       { NouvelleTypeElement nouv (PENTAEDRE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this);
165         if (ParaGlob::NiveauImpression() >= 4)
166           cout << "\n initialisation PentaL" << endl;
167         Element::listTypeElement.push_back(nouv);
168       };
169       Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
170       {Element * pt;
171         pt = new PentaL (num_maill,num) ;
172         return pt;};
173       // ramene true si la construction de l'element est possible en fonction
174       // des variables globales actuelles: ex en fonction de la dimension
175       bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
176     };
177     static ConsPentaL consPentaL;
178     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
179 };
180 /// @} // end of group
181 #endif
182
183
184
185

```

## 7.171 PentaL\_cm1pti.h

```

1 // FICHER : PentaL_cmlpti.h
2 // CLASSE : PentaL_cmlpti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      23/01/97
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 * *****/

```



```

40 *      BUT:  La classe Pental_cmlpti permet de declarer des elements *
41 * Pentaedriques Trilineaires et de realiser *
42 * le calcul du residu local et de la raideur locale pour une loi de *
43 * comportement donnee. La dimension de l'espace pour un tel element *
44 * est 3. *
45 * *
46 * l'interpolation est trilineaire a 6 noeuds, le nombre de *
47 * point d'integration est par défaut de 2. *
48 *      ***** *
49 *      *
49 *      VERIFICATION: *
50 * *
51 *      ! date !      auteur !      but *
52 *      ----- *
53 *      !      !      ! *
54 *      * *
55 *      ***** *
56 *      MODIFICATIONS: *
57 *      ! date !      auteur !      but *
58 *      ----- *
59 *      *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef PENTAL_CM1PTI_H
67 #define PENTAL_CM1PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "PentaMemb.h"
79 #include "FrontTriaLine.h"
80 #include "FrontQuadLine.h"
81 #include "FrontSegLine.h"
82
83 /// @addtogroup groupe_des_elements_finis
84 /// @{
85 ///
86
87
88
89 class Pental_cmlpti : public PentaMemb
90 {
91
92     public :
93
94         // CONSTRUCTEURS :
95
96         // Constructeur par default
97         Pental_cmlpti ();
98
99         // Constructeur fonction d'un numero
100        // d'identification
101        Pental_cmlpti (int num_mail,int num_id);
102
103        // Constructeur fonction d'un numero de maillage et d'identification et
104        // du tableau de connexite des noeuds
105        Pental_cmlpti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        Pental_cmlpti (const Pental_cmlpti& Penta);
109
110
111        // DESTRUCTEUR :
112        ~Pental_cmlpti ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new Pental_cmlpti(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de Pental_cmlpti
119        Pental_cmlpti& operator= (Pental_cmlpti& Penta);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // affichage dans la sortie transmise, des variables duales "nom"
125        // aux differents points d'integration

```

```

126         // dans le cas ou nom est vide, affichage de "toute" les variables
127         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129 // 2) derivant des virtuelles
130 // 3) methodes propres a l'element
131
132     protected :
133
134         // adressage des frontieres linéiques et surfacique
135         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
136         // frontière linéique verticale (rectangle)
137         ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
138         { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
139         // frontière linéique horizontale (triangle)
140         ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
141         { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
142         // frontière surfacique verticale (rectangle)
143         ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
144         { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
145         // frontière surfacique horizontale (triangle)
146         ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
147         { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
148
149     // VARIABLES PRIVEES :
150     // place memoire commune a tous les elements PentaL_cmlpti
151     static PentaMemb::DonnComPenta * doCoPentaL_cmlpti;
152     // idem mais pour les indicateurs qui servent pour l'initialisation
153     static PentaMemb::UneFois uneFois;
154
155     class NombresConstruirePentaL_cmlpti : public NombresConstruire
156     { public: NombresConstruirePentaL_cmlpti();
157       };
158     static NombresConstruirePentaL_cmlpti nombre_V; // les nombres propres à l'élément
159
160     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
161     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
162     class ConsPentaL_cmlpti : public ConstrucElement
163     { public : ConsPentaL_cmlpti ()
164       { NouvelleTypeElement nouv(PENTAEDRE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this,"_cmlpti");
165         if (ParaGlob::NiveauImpression() >= 4)
166           cout << "\n initialisation PentaL_cmlpti" << endl;
167         Element::listTypeElement.push_back(nouv);
168       };
169       Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
170       {Element * pt;
171         pt = new PentaL_cmlpti (num_maill,num) ;
172         return pt;};
173       // ramene true si la construction de l'element est possible en fonction
174       // des variables globales actuelles: ex en fonction de la dimension
175       bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
176     };
177     static ConsPentaL_cmlpti consPentaL_cmlpti;
178     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
179 };
180 /// @} // end of group
181 #endif
182
183
184
185

```

## 7.172 PentaL\_cm2pti.h

```

1 // FICHER : PentaL_cm2pti.h
2 // CLASSE : PentaL_cm2pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,

```

```

23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      23/01/97
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 * *****/
40 *   BUT:   La classe Pental_cm2pti permet de declarer des elements
41 * Pentaedriques Trilineaires et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 3.
45 *
46 * l'interpolation est trilineaire a 6 noeuds, le nombre de
47 * point d'integration est par defaut de 2.
48 *
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !           !
55 *   *****
56 *   MODIFICATIONS:
57 *   ! date !   auteur !           but
58 *   -----
59 *   !           !           !           !
60 *   *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef PENTAL_CM2PTI_H
67 #define PENTAL_CM2PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "PentaMemb.h"
79 #include "FrontTriaLine.h"
80 #include "FrontQuadLine.h"
81 #include "FrontSegLine.h"
82
83
84 /// @addtogroup groupe_des_elements_finis
85 /// @{
86 ///
87
88
89 class Pental_cm2pti : public PentaMemb
90 {
91
92     public :
93
94         // CONSTRUCTEURS :
95
96         // Constructeur par defaut
97         Pental_cm2pti ();
98
99         // Constructeur fonction d'un numero
100        // d'identification
101        Pental_cm2pti (int num_mail,int num_id);
102
103        // Constructeur fonction d'un numero de maillage et d'identification et
104        // du tableau de connexite des noeuds
105        Pental_cm2pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        Pental_cm2pti (const Pental_cm2pti& Penta);

```

```

109
110
111     // DESTRUCTEUR :
112     ~PentaL_cm2pti ();
113
114     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115     // méthode virtuelle
116     Element* Nevez_copie() const { Element * el= new PentaL_cm2pti(*this); return el;};
117
118     // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaL_cm2pti
119     PentaL_cm2pti& operator= (PentaL_cm2pti& Penta);
120
121     // METHODES :
122     // 1) derivant des virtuelles pures
123
124     // affichage dans la sortie transmise, des variables duales "nom"
125     // aux differents points d'integration
126     // dans le cas ou nom est vide, affichage de "toute" les variables
127     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129     // 2) derivant des virtuelles
130     // 3) methodes propres a l'element
131
132     protected :
133
134     // adressage des frontières linéiques et surfacique
135     // définit dans les classes dérivées, et utilisées pour la construction des frontières
136     // frontière linéique verticale (rectangle)
137     ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
138     { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
139     // frontière linéique horizontale (triangle)
140     ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
141     { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
142     // frontière surfacique verticale (rectangle)
143     ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
144     { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
145     // frontière surfacique horizontale (triangle)
146     ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
147     { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
148
149     // VARIABLES PRIVEES :
150     // place memoire commune a tous les elements PentaL_cm2pti
151     static PentaMemb::DonnComPenta * doCoPentaL_cm2pti;
152     // idem mais pour les indicateurs qui servent pour l'initialisation
153     static PentaMemb::UneFois uneFois;
154
155     class NombresConstruirePentaL_cm2pti : public NombresConstruire
156     { public: NombresConstruirePentaL_cm2pti();
157       };
158     static NombresConstruirePentaL_cm2pti nombre_V; // les nombres propres à l'élément
159
160     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
161     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
162     class ConsPentaL_cm2pti : public ConstrucElement
163     { public : ConsPentaL_cm2pti ()
164       { NouvelleTypeElement nouv (PENTAEDRE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this,"_cm2pti");
165         if (ParaGlob::NiveauImpression() >= 4)
166           cout << "\n initialisation PentaL_cm2pti" << endl;
167         Element::listTypeElement.push_back(nouv);
168       };
169       Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
170       {Element * pt;
171         pt = new PentaL_cm2pti (num_maill,num) ;
172         return pt;};
173       // ramene true si la construction de l'element est possible en fonction
174       // des variables globales actuelles: ex en fonction de la dimension
175       bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
176     };
177     static ConsPentaL_cm2pti consPentaL_cm2pti;
178     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
179 };
180 /// @} // end of group
181 #endif
182
183
184
185

```

## 7.173 PentaL\_cm6pti.h

```

1 // FICHER : PentaL_cm6pti.h
2 // CLASSE : PentaL_cm6pti
3
4 // This file is part of the Herezh++ application.
5 //

```

```

6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31 //
32 /*****
33 *   DATE:      23/01/97
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *   *****
40 *   BUT:      La classe Pental_cm6pti permet de declarer des elements
41 *   Pentaedriques Trilineaires et de realiser
42 *   le calcul du residu local et de la raideur locale pour une loi de
43 *   comportement donnee. La dimension de l'espace pour un tel element
44 *   est 3.
45 *
46 *   l'interpolation est trilineaire a 6 noeuds, le nombre de
47 *   point d'integration est par defaut de 2.
48 *   *****
49 *
50 *   VERIFICATION:
51 *
52 *   ! date ! auteur ! but
53 *   -----
54 *   ! ! ! !
55 *   *****
56 *   MODIFICATIONS:
57 *
58 *   ! date ! auteur ! but
59 *   -----
60 *   $
61 *   *****/
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef PENTAL_CM6PTI_H
67 #define PENTAL_CM6PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "PentaMemb.h"
79 #include "FrontTriaLine.h"
80 #include "FrontQuadLine.h"
81 #include "FrontSegLine.h"
82
83
84 /// @addtogroup groupe_des_elements_finis
85 /// @{
86 ///
87
88
89 class Pental_cm6pti : public PentaMemb
90 {
91

```

```

92     public :
93
94         // CONSTRUCTEURS :
95
96         // Constructeur par défaut
97         PentaL_cm6pti ();
98
99         // Constructeur fonction d'un numero
100        // d'identification
101        PentaL_cm6pti (int num_mail,int num_id);
102
103        // Constructeur fonction d'un numero de maillage et d'identification et
104        // du tableau de connexite des noeuds
105        PentaL_cm6pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        PentaL_cm6pti (const PentaL_cm6pti& Penta);
109
110
111        // DESTRUCTEUR :
112        ~PentaL_cm6pti ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new PentaL_cm6pti(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaL_cm6pti
119        PentaL_cm6pti& operator= (PentaL_cm6pti& Penta);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // affichage dans la sortie transmise, des variables duales "nom"
125        // aux differents points d'integration
126        // dans le cas ou nom est vide, affichage de "toute" les variables
127        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129        // 2) derivant des virtuelles
130        // 3) methodes propres a l'element
131
132     protected :
133
134        // adressage des frontieres linéiques et surfacique
135        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
136        // frontière linéique verticale (rectangle)
137        ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
138        { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
139        // frontière linéique horizontale (triangle)
140        ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
141        { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
142        // frontière surfacique verticale (rectangle)
143        ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
144        { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
145        // frontière surfacique horizontale (triangle)
146        ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
147        { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
148
149        // VARIABLES PRIVEES :
150        // place memoire commune a tous les elements PentaL_cm6pti
151        static PentaMemb::DonnComPenta * doCoPentaL_cm6pti;
152        // idem mais pour les indicateurs qui servent pour l'initialisation
153        static PentaMemb::UneFois uneFois;
154
155        class NombresConstruirePentaL_cm6pti : public NombresConstruire
156        { public: NombresConstruirePentaL_cm6pti();
157          };
158        static NombresConstruirePentaL_cm6pti nombre_V; // les nombres propres à l'élément
159
160        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
161        //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
162        class ConsPentaL_cm6pti : public ConstrucElement
163        { public : ConsPentaL_cm6pti ()
164          { NouvelleTypeElement nouv (PENTAEDRE, LINEAIRE, MECA_SOLIDE_DEFORMABLE, this, "_cm6pti");
165            if (ParaGlob::NiveauImpression() >= 4)
166              cout << "\n initialisation PentaL_cm6pti" << endl;
167              Element::listTypeElement.push_back (nouv);
168          };
169          Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
170          {Element * pt;
171            pt = new PentaL_cm6pti (num_maill,num) ;
172            return pt;};
173          // ramene true si la construction de l'element est possible en fonction
174          // des variables globales actuelles: ex en fonction de la dimension
175          bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
176          };
177        static ConsPentaL_cm6pti consPentaL_cm6pti;
178        static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh

```

```

179 };
180 /// @} // end of group
181 #endif
182
183
184
185

```

## 7.174 PentaMemb.h

```

1 // FICHER : PentaMemb.h
2 // CLASSE : PentaMemb
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *
35 *   DATE:           15/01/97
36 *
37 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:        Herezh++
40 *
41 *
42 * *****
43 * La classe PentaMemb permet de declarer des elements Pentahedriques et de realiser
44 * le calcul du residu local et de la raideur locale pour une loi de comportement
45 * donnee. La dimension de l'espace pour un tel element est 3
46 *
47 * l'interpolation le nombre de point d'integration sont definit dans les classes derivees
48 * l'element est virtuel
49 *
50 *   $
51 *
52 * VERIFICATION:
53 *   ! date !   auteur !   but
54 *   -----
55 *   !       !       !
56 *
57 *   $
58 *
59 * *****
60 * MODIFICATIONS:
61 *   ! date !   auteur !   but
62 *   -----
63 *
64 *
65 *   $
66 *
67 * *****/
68
69 // -----classe pour un calcul de mecanique-----
70
71 // La classe PentaMemb permet de declarer des elements Pentahedriques et de realiser
72 // le calcul du residu local et de la raideur locale pour une loi de comportement
73 // donnee. La dimension de l'espace pour un tel element est 3
74 //

```

```

75 // l'interpolation le nombre de point d'integration sont definit dans les classes derivees
76 //
77 // l'element est virtuel
78
79
80 #ifndef PENTAMEMB_H
81 #define PENTAMEMB_H
82
83 #include "ParaGlob.h"
84 #include "ElemMeca.h"
85 #include "Met_abstraite.h"
86 #include "ElemGeomC0.h"
87 #include "Noeud.h"
88 #include "UtilLecture.h"
89 #include "Tenseur.h"
90 #include "NevezTenseur.h"
91 #include "Deformation.h"
92 #include "GeomSeg.h"
93 #include "GeomQuadrangle.h"
94 #include "GeomTriangle.h"
95
96 /// @addtogroup groupe_des_elements_finis
97 /// @{
98 ///
99
100
101 class PentaMemb : public ElemMeca
102 {
103
104     public :
105
106         // CONSTRUCTEURS :
107         // Constructeur par default
108         PentaMemb ();
109
110         // Constructeur fonction d'un numero
111         // d'identification , d'identificateur d'interpolation et de geometrie et éventuellement un
112         // string d'information annexe
113         PentaMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string
114         info="");
115
116         // Constructeur fonction d'un numero de maillage et d'identification,
117         // du tableau de connexite des noeuds, d'identificateur d'interpolation et de geometrie
118         // et éventuellement un string d'information annexe
119         PentaMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,
120         const Tableau<Noeud *>& tab,string info="") ;
121
122         // Constructeur de copie
123         PentaMemb (const PentaMemb& pentaMem);
124
125         // DESTRUCTEUR :
126         ~PentaMemb ();
127
128         // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaMemb
129         PentaMemb& operator= (PentaMemb& pentaMem);
130
131         // METHODES :
132         // 1) derivant des virtuelles pures
133
134         // Lecture des donnees de la classe sur fichier
135         void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
136
137         // affichage d'info en fonction de ordre
138         // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
139         void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> *
140         tabMaillageNoeud)
141         { return Element::Info_com_El (nombre->nbne,entreePrinc,ordre,tabMaillageNoeud);};
142
143         // ramene l'element geometrique
144         ElemGeomC0& ElementGeometrique() const { return *(unefois->doCoMemb->pentaed);};
145         // ramene l'element geometrique en constant
146         const ElemGeomC0& ElementGeometrique_const() const {return *(unefois->doCoMemb->pentaed);};
147
148         // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
149         // associé
150         // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
151         // 1) cas où l'on utilise la place passée en argument
152         Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
153         // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
154         void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
155
156         // inactive les ddl du problème primaire de mécanique
157         inline void Inactive_ddl_primaire()
158         {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};

```



```

157 // active les ddl du problème primaire de mécanique
158 inline void Active_ddl_primaire()
159     {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl)};
160 // ajout des ddl de contraintes pour les noeuds de l'élément
161 inline void Plus_ddl_Sigma()
162     {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr)};
163 // inactive les ddl du problème de recherche d'erreur : les contraintes
164 inline void Inactive_ddl_Sigma()
165     {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr)};
166 // active les ddl du problème de recherche d'erreur : les contraintes
167 inline void Active_ddl_Sigma()
168     {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr)};
169 // active le premier ddl du problème de recherche d'erreur : SIGMA11
170 inline void Active_premier_ddl_Sigma()
171     {ElemMeca::Act_premier_ddl_Sigma()};
172
173 // lecture de données diverses sur le flot d'entrée
174 void LectureContraintes(UtilLecture * entreePrinc)
175     { if (unefois->CalResPrem_t == 1)
176         ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
177     else
178         { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
179           unefois->CalResPrem_t = 1;
180         }
181     };
182
183 // retour des contraintes en absolu retour true si elle existe sinon false
184 bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
185     { if (unefois->CalResPrem_t == 1)
186         ElemMeca::ContraintesEnAbsolues(false,lesPtMecaInt.TabSigHH_t(),tabSig);
187     else
188         { ElemMeca::ContraintesEnAbsolues(true,lesPtMecaInt.TabSigHH_t(),tabSig);
189           unefois->CalResPrem_t = 1;
190         }
191     return true; }
192
193 // Libere la place occupee par le residu et eventuellement la raideur
194 // par l'appel de Libere de la classe mere et libere les differents tenseurs
195 // intermediaires cree pour le calcul et les grandeurs pointee
196 // de la raideur et du residu
197 void Libere ();
198
199 // acquisition d'une loi de comportement
200 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
201
202 // test si l'element est complet
203 // = 1 tout est ok, =0 element incomplet
204 int TestComplet();
205
206 // procedure permettant de completer l'element apres
207 // sa creation avec les donnees du bloc transmis
208 // peut etre appeler plusieurs fois
209 // ici pour l'instant ne fait rien
210 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
211 // Compléter pour la mise en place de la gestion de l'hourglass
212 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
213
214 // ramene vrai si la surface numéro ns existe pour l'élément
215 // dans le cas des éléments Pentaedrique il peut y avoir de 1 à 5 surfaces
216 bool SurfExiste(int ns) const
217     { if ((ns>=1)&&(ns<=5)) return true; else return false;};
218
219 // ramene vrai si l'arête numéro na existe pour l'élément
220 bool AreteExiste(int na) const {if ((na <= 9) || (na>= 1)) return true; else return false;};
221
222 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
223 // ce tableau et specifique a l'element
224 const DdlElement & TableauDdl() const
225     { return unefois->doCoMemb->tab_ddl; };
226
227 // Calcul du residu local et de la raideur locale,
228 // pour le schema implicite
229 Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
230
231 // Calcul du residu local a t
232 // pour le schema explicit par exemple
233 Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
234     { return PentaMemb::CalculResidu(false,pa);};
235
236 // Calcul du residu local a tdt
237 // pour le schema explicit par exemple
238 Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
239     { return PentaMemb::CalculResidu(true,pa);};
240
241 // Calcul de la matrice masse pour l'élément
242 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
243

```

```

244 // ----- calcul dynamique -----
245 // calcul de la longueur d'arrête de l'élément minimal
246 // divisé par la célérité la plus rapide dans le matériau
247 double Long_arrete_mini_sur_c(Enum_dure temps)
248     { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps); };
249
250 //----- calcul d'erreur, remontée des contraintes -----
251 // 1) calcul du résidu et de la matrice de raideur pour le calcul d'erreur
252 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
253 // 2) remontée aux erreurs aux noeuds
254 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
255
256 // ----- affichage ou récupération d'informations -----
257 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
258 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
259 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
260 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
261 // temps: dit si c'est à 0 ou t ou tdt
262 int PointLePlusPres(Enum_dure temps, Enum_ddl enu, const Coordonnee& M)
263     { return PtLePlusPres(temps, enu, M); };
264
265 // recuperation des coordonnées du point de numéro d'ordre iteg pour
266 // la grandeur enu
267 // temps: dit si c'est à 0 ou t ou tdt
268 // si erreur retourne erreur à true
269 Coordonnee CoordPtInteg(Enum_dure temps, Enum_ddl enu, int iteg, bool& erreur)
270     { return CoordPtInt(temps, enu, iteg, erreur); };
271
272 // récupération des valeurs au numéro d'ordre = iteg pour
273 // les grandeur enu
274 Tableau<double> Valeur_a_diff_temps(bool absolue, Enum_dure enu_t, const
List_io<Ddl_enum_etendu>& enu, int iteg);
275 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
276 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
277 // de conteneurs quelconque associée
278 void ValTensorielle_a_diff_temps(bool absolue, Enum_dure enu_t, List_io<TypeQuelconque>& enu, int
iteg);
279
280 //===== lecture écriture dans base info =====
281
282 // cas donne le niveau de la récupération
283 // = 1 : on récupère tout
284 // = 2 : on récupère uniquement les données variables (supposées comme telles)
285 void Lecture_base_info
286     (ifstream& ent, const Tableau<Noeud *> * tabMaillageNoeud, const int cas);
287 // cas donne le niveau de sauvegarde
288 // = 1 : on sauvegarde tout
289 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
290 void Ecriture_base_info(ofstream& sort, const int cas);
291
292 // 2) derivant des virtuelles
293
294 // retourne un tableau de ddl element, correspondant à la
295 // composante de sigma -> SIG11, pour chaque noeud qui contiend
296 // des ddl de contrainte
297 // -> utilisé pour l'assemblage de la raideur d'erreur
298 DdlElement& Tableau_de_Sig11() const
299     { return unefois->doCoMemb->tab_Err1Sig11; };
300
301 // actualisation des ddl et des grandeurs actives de t+dt vers t
302 void TdtversT();
303 // actualisation des ddl et des grandeurs actives de t vers tdt
304 void TversTdt();
305
306 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
307 // qu'une fois la remontée aux contraintes effectuées sinon aucune
308 // action. En retour la valeur de l'erreur sur l'élément
309 // type indique le type de calcul d'erreur :
310 void ErreurElement(int type, double& errElemRelative
311     , double& numerateur, double& denominateur);
312
313 // calcul des seconds membres suivant les chargements
314 // cas d'un chargement en pression volumique,
315 // force indique la force volumique appliquée
316 // retourne le second membre résultant
317 // ici on l'épaisseur de l'élément pour constituer le volume
318 // -> explicite à t
319 Vecteur SM_charge_volumique_E_t
320     (const Coordonnee& force, Fonction_nd* pt_fonct, const ParaAlgoControle & pa, bool
sur_volume_finale_)
321     { return PentaMemb::SM_charge_volumique_E(force, pt_fonct, false, pa, sur_volume_finale_); };
322 // -> explicite à tdt
323 Vecteur SM_charge_volumique_E_tdt
324     (const Coordonnee& force, Fonction_nd* pt_fonct, const ParaAlgoControle & pa, bool
sur_volume_finale_)
325     { return PentaMemb::SM_charge_volumique_E(force, pt_fonct, true, pa, sur_volume_finale_); };
326 // -> implicite,

```

```

327 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
328 // retourne le second membre et la matrice de raideur correspondant
329 ResRaid SMR_charge_volumique_I
330 (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_);
331
332 // cas d'un chargement surfacique, sur les frontières des éléments
333 // force indique la force surfacique appliquée
334 // numface indique le numéro de la face chargée
335 // retourne le second membre résultant
336 // -> version explicite à t
337 Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct
, int numFace,const ParaAlgoControle & pa)
338 { return PentaMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa); };
339 // -> version explicite à tdt
340 Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
341 { return PentaMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa); };
342 // -> implicite,
343 // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
344 // retourne le second membre et la matrice de raideur correspondant
345 ResRaid SMR_charge_surfacique_I
346 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const ParaAlgoControle &
pa);
347
348 // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
349 // presUniDir indique le vecteur appliquée
350 // numface indique le numéro de la face chargée
351 // retourne le second membre résultant
352 // -> explicite à t
353 Vecteur SM_charge_presUniDir_E_t(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
354 { return PentaMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,false,pa); };
355 // -> explicite à tdt
356 Vecteur SM_charge_presUniDir_E_tdt(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
357 { return PentaMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,true,pa); };
358 // -> implicite,
359 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
360 // retourne le second membre et la matrice de raideur correspondant
361 ResRaid SMR_charge_presUniDir_I(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa);
362
363 // cas d'un chargement lineique, sur les aretes frontières des éléments
364 // force indique la force lineique appliquée
365 // numarete indique le numéro de l'arete chargée
366 // retourne le second membre résultant
367 // NB: il y a une définition par défaut pour les éléments qui n'ont pas
368 // d'arete externe -> message d'erreur d'où le virtuel et non virtuel pur
369 // -> explicite à t
370 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
371 { return PentaMemb::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); };
372 // -> explicite à tdt
373 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
374 { return PentaMemb::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); };
375 // -> implicite,
376 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
377 // retourne le second membre et la matrice de raideur correspondant
378 ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa);
379
380 // cas d'un chargement de type pression, sur les frontières des éléments
381 // pression indique la pression appliquée
382 // numface indique le numéro de la face chargée
383 // retourne le second membre résultant
384 // NB: il y a une définition par défaut pour les éléments qui n'ont pas de
385 // surface externe -> message d'erreur d'où le virtuel et non virtuel pur
386 // -> explicite à t
387 Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
388 { return PentaMemb::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa); };
389 // -> explicite à tdt
390 Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
391 { return PentaMemb::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa); };
392 // -> implicite,
393 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
394 // retourne le second membre et la matrice de raideur correspondant
395 ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa);
396
397 // cas d'un chargement surfacique hydrostatique,
398 // poidvol: indique le poids volumique du liquide
399 // M_liquide : un point de la surface libre
400 // dir_normal_liquide : direction normale à la surface libre
401

```

```

402 // retourne le second membre résultant
403 // -> explicite à t
404 Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
405                                     ,int numFace,const Coordonnee& M_liquide
406                                     ,const ParaAlgoControle & pa
407                                     ,bool sans_limitation)
408 { return
PentaMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation));};
409 // -> explicite à tdt
410 Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
411                                       ,int numFace,const Coordonnee& M_liquide
412                                       ,const ParaAlgoControle & pa
413                                       ,bool sans_limitation)
414 { return
PentaMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation));};
415 // -> implicite,
416 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
417 // retourne le second membre et la matrice de raideur correspondant
418 ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
419                                    ,int numFace,const Coordonnee& M_liquide
420                                    ,const ParaAlgoControle & pa
421                                    ,bool sans_limitation) ;
422
423 // cas d'un chargement surfacique hydro-dynamique,
424 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
425 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc
unitaire)
426 // une suivant la direction normale à la vitesse de type portance
427 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
428 // une suivant la vitesse tangente de type frottement visqueux
429 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
430 // coef_mul: est un coefficient multiplicateur global (de tout)
431 // retourne le second membre résultant
432 // -> explicite à t
433 Vecteur SM_charge_hydrodynamique_E_t( CourbelD* frot_fluid,const double& poidvol
434                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
435                                     , CourbelD* coef_aero_t
436                                     ,const ParaAlgoControle & pa)
437 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa));};
438 // -> explicite à tdt
439 Vecteur SM_charge_hydrodynamique_E_tdt( CourbelD* frot_fluid,const double& poidvol
440                                         , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
441                                         , CourbelD* coef_aero_t,const ParaAlgoControle &
pa)
442 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa));};
443 // -> implicite,
444 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
445 // retourne le second membre et la matrice de raideur correspondant
446 ResRaid SMR_charge_hydrodynamique_I( CourbelD* frot_fluid,const double& poidvol
447                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
448                                     , CourbelD* coef_aero_t,const ParaAlgoControle &
pa) ;
449
450 // ===== définition et/ou construction des frontières =====
451
452 // Calcul des frontieres de l'element
453 // creation des elements frontieres et retour du tableau de ces elements
454 // la création n'a lieu qu'au premier appel
455 // ou lorsque l'on force le paramètre force a true
456 // dans ce dernier cas seul les frontière effacées sont recréée
457 Tableau <ElFrontiere*> const & Frontiere(bool force = false);
458
459 // ramène la frontière point
460 // éventuellement création des frontieres points de l'element et stockage dans l'element
461 // si c'est la première fois sinon il y a seulement retour de l'elements
462 // a moins que le paramètre force est mis a true
463 // dans ce dernier cas la frontière effacée est recréée
464 // num indique le numéro du point à créer (numérotation EF)
465 // ElFrontiere* const Frontiere_points(int num,bool force = false);
466
467 // ramène la frontière linéique
468 // éventuellement création des frontieres linéique de l'element et stockage dans l'element
469 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
470 // a moins que le paramètre force est mis a true
471 // dans ce dernier cas la frontière effacée est recréée
472 // num indique le numéro de l'arête à créer (numérotation EF)
473 // on maintient la fonction ici, car elle fait des choses qui ne sont pas fait
474 // dans la fonction d'ElemMeca
475 // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
476
477 // ramène la frontière surfacique
478 // éventuellement création des frontieres surfacique de l'element et stockage dans l'element

```

```

479 // si c'est la première fois sinon il y a seulement retour de l'elements
480 // a moins que le paramètre force est mis a true
481 // dans ce dernier cas la frontière effacée est recrée
482 // num indique le numéro de la surface à créer (numérotation EF)
483 ElFrontiere* const Frontiere_surfacique(int num,bool force = false);
484
485 // ----- calcul de frontières en protected -----
486
487 // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
particulière à l'élément
488 // adressage des frontières linéiques et surfacique
489 // définit dans les classes dérivées, et utilisées pour la construction des frontières
490 virtual ElFrontiere* new_frontiere_lin(int num,Tableau <Noeud *> & tab, DdlElement& ddelem);
491 virtual ElFrontiere* new_frontiere_surf(int num,Tableau <Noeud *> & tab, DdlElement& ddelem);
492
493 // 3) methodes propres a l'element
494
495 // ajout du tableau specific de ddl des noeuds
496 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
497 // des noeuds constituants l'element
498 void ConstTabDdl();
499
500 public :
501 // ----- definition de la classe conteneur de donnees communes -----
502 class DonnComPenta
503 { public :
504 // pental doit pointer sur un element deja existant via un new
505 DonnComPenta (ElemGeomC0* pental,DdlElement& tab,DdlElement& tabErr,DdlElement&
tab_Err1Sig,
506 Met_abstraite& met_gene,
507 Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,ElemGeomC0* pentaeEr
,GeomQuadrangle quadS,GeomTriangle triS,
508 GeomSeg& seggHS,GeomSeg& seggFS,
509 Vecteur& residu_int,Mat_pleine& raideur_int,
510 Tableau <Vecteur*> & residus_extN,Tableau <Mat_pleine*> & raideurs_extN,
511 Tableau <Vecteur*> & residus_extA,Tableau <Mat_pleine*> & raideurs_extA,
512 Tableau <Vecteur*> & residus_extS,Tableau <Mat_pleine*> & raideurs_extS,
513 Mat_pleine& mat_masse,ElemGeomC0* pentaeMas,int nbi,
514 ElemGeomC0* pentaeHourg
515 );
516
517 DonnComPenta(DonnComPenta& a) ;
518 ~DonnComPenta();
519 // variables
520 ElemGeomC0* pentaed; // contient les fonctions d'interpolation et
521 // les derivees
522 DdlElement tab_ddl; // tableau des degres
523 //de liberte des noeuds de l'element commun a tous les
524 // elements
525 Met_abstraite metrique;
526 Mat_pleine matGeom ; // matrice géométrique
527 Mat_pleine matInit ; // matrice initiale
528 Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
529 Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
530 Tableau < Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
531 // ---- concernant les frontières et particulièrement le calcul de second membre
532 GeomQuadrangle quadraS; // contient les fonctions d'interpolation
533 GeomTriangle triaS; // et les derivees pour les deux types de surfaces
534 GeomSeg segHS; // cas des segments dans la hauteur
535 GeomSeg segFS; // cas des segments des faces
536
537 // calcul d'erreur
538 DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
539 // d'erreur : contraintes
540 DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud, servant pour le
calcul
541 // d'erreur : contraintes, en fait pour l'assemblage
542
543 Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
544 Mat_pleine raidErr; // raideur pour le calcul d'erreur
545 ElemGeomC0* pentaedEr; // contient les fonctions d'interpolation et
546 // les derivees pour le calcul du hessien dans
547 //la résolution de la fonctionnelle d'erreur
548 // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
-----
549 // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
550 Vecteur residu_interne;
551 Mat_pleine raideur_interne;
552 Tableau <Vecteur*> residus_externes; // pour les noeuds
553 Tableau <Mat_pleine*> raideurs_externes; // pour les noeuds
554 Tableau <Vecteur*> residus_externesA; // pour les aretes
555 Tableau <Mat_pleine*> raideurs_externesA; // pour les aretes
556 Tableau <Vecteur*> residus_externesS; // pour les surfaces
557 Tableau <Mat_pleine*> raideurs_externesS; // pour les surfaces
558 // ----- données concernant la dynamique -----
559 Mat_pleine matrice_masse;
560 ElemGeomC0* pentaedMas; // contient les fonctions d'interpolation et ...
561 // pour les calculs relatifs à la masse

```

```

562         // ----- blocage éventuel d'hourglass
563         // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
Cal_explici_hourglass
564         ElemGeomCO* pentaedHourg; // contient les fonctions d'interpolation
565     };
566
567     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
568     // et un pointeur sur les données statiques communes
569     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
570     // classe est défini. Son allocation est effectuée dans les classes dérivées
571     class UneFois
572     { public :
573         UneFois () ; // constructeur par défaut
574         ~UneFois () ; // destructeur
575
576         // VARIABLES :
577         public :
578         PentaMemb::DonnComPenta * doCoMemb;
579
580         // incicateurs permettant de dimensionner seulement au premier passage
581         // utilise dans "CalculResidu" et "Calcul_implicit"
582         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
583         int CalimpPrem;
584         int dualSortPenta; // pour la sortie des valeurs au pt d'integ
585         int CalSMlin_t; // pour les seconds membres concernant les arretes
586         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
587         int CalSMRlin; // pour les seconds membres concernant les arretes
588         Tableau<int> CalSMsurf_t; // pour les seconds membres concernant les surfaces
589         Tableau<int> CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
590         Tableau<int> CalSMRsurf; // pour les seconds membres concernant les surfaces
591         int CalSMvol_t; // pour les seconds membres concernant les volumes
592         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
593         int CalSMvol; // pour les seconds membres concernant les volumes
594         int CalDynamique; // pour le calcul de la matrice de masse
595         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
596         // ----- sauvegarde du nombre d'élément en cours -----
597         int nbelem_in_Prog;
598     };
599
600     // -----
601
602     protected :
603
604     // VARIABLES PRIVEES :
605     UneFois * unefois; // pointeur défini dans la classe dérivée
606
607     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
608     LesPtIntegMecaInterne lesPtMecaInt;
609
610     // type structuré et pointeur pour construire les éléments
611     // le pointeur est défini dans le type dérivé
612     class NombresConstruire
613     { public:
614         NombresConstruire() : nbne(0), nbneSQ(0), nbneST(0), nbneAQ(0), nbneAT(0), nbI(0)
615             , nbIQ(0), nbIT(0), nbIEr(0), nbIV(0), nbISQ(0), nbIST(0), nbIAQ(0)
616             , nbIAT(0), nbIMas(0), nbIHour(0) {};
617         int nbne; // le nombre de noeud de l'élément
618         int nbneSQ ; // le nombre de noeud des facettes quadrangulaires
619         int nbneST ; // le nombre de noeud des facettes triangulaires
620         int nbneAQ ; // le nombre de noeud des aretes entres les faces triangulaires
621         int nbneAT ; // le nombre de noeud des aretes des facettes triangulaires
622         int nbI; // nombre point d'intég pour le calcul méca pour l'élément
623         int nbIQ; // nombre point d'intég pour le calcul méca pour les triangles
624         int nbIT; // nombre point d'intég pour le calcul méca pour les quadrangles
625         int nbIEr; // le nombre de point d'intégration pour le calcul d'erreur
626         int nbIV; // le nombre de point d'intégration pour le calcul de second membre volumique
627         int nbISQ; // nombre point d'intég pour le calcul de second membre surfacique quadrangulaire
628         int nbIST; // nombre point d'intég pour le calcul de second membre surfacique triangulaire
629         int nbIAQ; // nB pt integ pour calcul second membre linéique des arête entre faces triangles
630         int nbIAT; // nB pt integ pour calcul second membre linéique des arête des triangles
631             int nbIMas; // le nombre de point d'intégration pour le calcul de la matrice masse
632             int nbIHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
633     };
634     NombresConstruire * nombre; // le pointeur défini dans la classe dérivée d'hexamemb
635
636     // =====>>> methodes appelees par les classes derivees <<=====
637
638     // fonction d'initialisation servant dans les classes derivant
639     // au niveau du constructeur
640     // les pointeurs d'éléments géométriques sont non nul uniquement lorsque doCoMemb est null
641     // c'est-à-dire pour l'initialisation
642     PentaMemb::DonnComPenta* Init (ElemGeomCO* penta, ElemGeomCO* pentaEr, ElemGeomCO* pentaMas
643         , ElemGeomCO* pentaedHourg, bool sans_init_noeud = false);
644     // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
645     void Destruction();
646
647     // adressage des frontières linéiques et surfacique

```

```

648 // définit dans les classes dérivées, et utilisées pour la construction des frontières
649 // frontière linéique verticale (rectangle)
650 virtual ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
651 // frontière linéique horizontale (triangle)
652 virtual ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
653 // frontière surfacique verticale (rectangle)
654 virtual ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
655 // frontière surfacique horizontale (triangle)
656 virtual ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
657
658 // ==== »» methodes virtuelles dérivant d'ElemMeca =====
659 // ramene la dimension des tenseurs contraintes et déformations de l'élément
660 int Dim_sig_eps() const {return 3;};
661
662 //----- fonctions uniquement a usage interne -----
663 private :
664 // definition des données communes : doCopenta
665 // nbiSQ sont pour le nombre de point d'intégration de surface quadrangle pour le second membre
666 // nbiST idem mais pour les surfaces triangles
667 PentaMemb::DonnComPenta* Def_DonneeCommune(ElemGeomC0* penta,ElemGeomC0* pentaEr
668 ,ElemGeomC0* pentaMas,ElemGeomC0* pentaHourg);
669 // Calcul du residu local a t ou tdt en fonction du booleen
670 Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
671 // cas d'un chargement de type pression, sur les frontières des éléments
672 // pression indique la pression appliquée
673 // numface indique le numéro de la face chargée
674 // retourne le second membre résultant
675 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
676 Vecteur SM_charge_pression_E(double pression,Fonction_nD* pt_fonct,int numFace,bool atdt,const
ParaAlgoControle & pa);
677 // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
678 // presUniDir indique le vecteur appliquée
679 // numface indique le numéro de la face chargée
680 // retourne le second membre résultant
681 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
682 Vecteur SM_charge_presUniDir_E
683 (const Coordonnee& presUniDir,Fonction_nD* pt_fonct
684 ,int numFace,bool atdt,const ParaAlgoControle & pa) ;
685 // cas d'un chargement lineique, sur les aretes frontières des éléments
686 // force indique la force lineique appliquée
687 // numarete indique le numéro de l'arete chargée
688 // retourne le second membre résultant
689 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
690 Vecteur SM_charge_lineique_E
691 (const Coordonnee& force,Fonction_nD* pt_fonct
692 ,int numArete,bool atdt,const ParaAlgoControle & pa);
693 // cas d'un chargement surfacique, sur les frontières des éléments
694 // force indique la force surfacique appliquée
695 // numface indique le numéro de la face chargée
696 // retourne le second membre résultant
697 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
698 Vecteur SM_charge_surfacique_E
699 (const Coordonnee& force,Fonction_nD* pt_fonct
700 ,int numFace,bool atdt,const ParaAlgoControle & pa);
701 // calcul des seconds membres suivant les chargements
702 // cas d'un chargement volumique,
703 // force indique la force volumique appliquée
704 // retourne le second membre résultant
705 // ici on l'épaisseur de l'élément pour constituer le volume
706 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
707 Vecteur SM_charge_volumique_E
708 (const Coordonnee& force,Fonction_nD* pt_fonct
709 ,bool atdt,const ParaAlgoControle & pa,bool sur_volume_finale_);
710 // cas d'un chargement surfacique hydrostatique,
711 // poidvol: indique le poids volumique du liquide
712 // M_liquide : un point de la surface libre
713 // dir_normal_liquide : direction normale à la surface libre
714 // retourne le second membre résultant
715 // -> explicite à t
716 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
717 ,int numFace,const Coordonnee& M_liquide,bool atdt
718 ,const ParaAlgoControle & pa
719 ,bool sans_limitation);
720 // cas d'un chargement surfacique hydro-dynamique,
721 // voir méthode explicite plus haut, pour les arguments
722 // retourne le second membre résultant
723 // bool atdt : permet de spécifier à t ou a t+dt
724 Vecteur SM_charge_hydrodynamique_E( Courbeld* frot_fluid,const double& poidvol
725 , Courbeld* coef_aero_n,int numFace,const double&
coef_mul
726 , Courbeld* coef_aero_t,bool atdt
727 ,const ParaAlgoControle & pa) ;
728 };
729 /// @} // end of group
730
731
732 #endif

```



733  
734  
735  
736

## 7.175 PentaQ.h

```

1 // FICHER : PentaQ.h
2 // CLASSE : PentaQ
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      23/01/97
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:   La classe PentaQ permet de declarer des elements
41 * Pentaedriques quadratiques incomplets et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 3.
45 *
46 * l'interpolation est triquadratique a 5 noeuds, le nombre de
47 * point d'integration par défaut est de 6.
48 *
49 *
50 *
51 *   VERIFICATION:
52 *
53 *   ! date !   auteur !           but
54 *   -----
55 *
56 *   MODIFICATIONS:
57 *   ! date !   auteur !           but
58 *   -----
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef PENTAQ_H
67 #define PENTAQ_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"

```



```

78 #include "PentaMemb.h"
79 #include "FrontTriaQuad.h"
80 #include "FrontQuadQuad.h"
81 #include "FrontSegQuad.h"
82
83
84 /// @addtogroup groupe_des_elements_finis
85 /// @{
86 ///
87
88
89 class PentaQ : public PentaMemb
90 {
91     public :
92
93         // CONSTRUCTEURS :
94
95         // Constructeur par défaut
96         PentaQ ();
97
98         // Constructeur fonction d'un numero
99         // d'identification
100        PentaQ (int num_mail,int num_id);
101
102
103        // Constructeur fonction d'un numero de maillage et d'identification et
104        // du tableau de connexite des noeuds
105        PentaQ (int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        PentaQ (const PentaQ& Penta);
109
110
111        // DESTRUCTEUR :
112        ~PentaQ ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new PentaQ(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQ
119        PentaQ& operator= (PentaQ& Penta);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // affichage dans la sortie transmise, des variables duales "nom"
125        // aux differents points d'integration
126        // dans le cas ou nom est vide, affichage de "toute" les variables
127        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129    protected :
130
131        // adressage des frontières linéiques et surfacique
132        // définit dans les classes dérivées, et utilisées pour la construction des frontières
133        // frontière linéique verticale (rectangle)
134        ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
135        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
136        // frontière linéique horizontale (triangle)
137        ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
138        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
139        // frontière surfacique verticale (rectangle)
140        ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
141        { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
142        // frontière surfacique horizontale (triangle)
143        ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
144        { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
145
146        // VARIABLES PRIVEES :
147        // place memoire commune a tous les elements PentaQ
148        static PentaMemb::DonnComPenta * doCoPentaQ;
149        // idem mais pour les indicateurs qui servent pour l'initialisation
150        static PentaMemb::UneFois uneFois;
151
152        class NombresConstruirePentaQ : public NombresConstruire
153        { public: NombresConstruirePentaQ();
154          };
155        static NombresConstruirePentaQ nombre_V; // les nombres propres à l'élément
156
157        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
158        //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
159        class ConsPentaQ : public ConstrucElement
160        { public : ConsPentaQ ()
161          { NouvelleTypeElement nouv(PENTAEDRE,QUADRATIQUE,MECA_SOLIDE_DEFORMABLE,this);
162            if (ParaGlob::NiveauImpression() >= 4)
163              cout << "\n initialisation PentaQ" << endl;
164            Element::listTypeElement.push_back(nouv);

```

```

165         };
166         Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
167         {Element * pt;
168         pt = new PentaQ (num_maill,num) ;
169         return pt;};
170         // ramene true si la construction de l'element est possible en fonction
171         // des variables globales actuelles: ex en fonction de la dimension
172         bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
173     };
174     static ConsPentaQ consPentaQ;
175     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
176 };
177 /// @} // end of group
178 #endif
179
180
181
182

```

## 7.176 PentaQ\_cm12pti.h

```

1 // FICHER : PentaQ_cml2pti.h
2 // CLASSE : PentaQ_cml2pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      *****
41 *      BUT:      La classe PentaQ_cml2pti permet de declarer des elements
42 *      Pentaedriques quadratiques incomplets et de realiser
43 *      le calcul du residu local et de la raideur locale pour une loi de
44 *      comportement donnee. La dimension de l'espace pour un tel element
45 *      est 3.
46 *
47 *      l'interpolation est triquadratique a 15 noeuds, le nombre de
48 *      point d'integration est de 12, 4 dans le triangle et 3 dans
49 *      l'épaisseur.
50 *
51 *      *****
52 *
53 *      VERIFICATION:
54 *
55 *      ! date !   auteur !           but
56 *      -----
57 *      !           !           !
58 *
59 *      *****
60 *
61 *      MODIFICATIONS:
62 *
63 *      ! date !   auteur !           but
64 *      -----
65 *
66 *      *****
67
68 // -----classe pour un calcul de mecanique-----

```

```

64
65
66
67 #ifndef PENTAQ_CM12PTI_H
68 #define PENTAQ_CM12PTI_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 #include "Met_abstraite.h"
73 #include "GeomQuadrangle.h"
74 #include "Noeud.h"
75 #include "UtilLecture.h"
76 #include "Tenseur.h"
77 #include "NevezTenseur.h"
78 #include "Deformation.h"
79 #include "PentaMemb.h"
80 #include "FrontTriaQuad.h"
81 #include "FrontQuadQuad.h"
82 #include "FrontSegQuad.h"
83
84
85 /// @addtogroup groupe_des_elements_finis
86 /// @{
87 ///
88
89
90 class PentaQ_cm12pti : public PentaMemb
91 {
92
93     public :
94
95         // CONSTRUCTEURS :
96
97         // Constructeur par defaut
98         PentaQ_cm12pti ();
99
100        // Constructeur fonction d'un numero
101        // d'identification
102        PentaQ_cm12pti (int num_mail,int num_id);
103
104        // Constructeur fonction d'un numero de maillage et d'identification et
105        // du tableau de connexite des noeuds
106        PentaQ_cm12pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
107
108        // Constructeur de copie
109        PentaQ_cm12pti (const PentaQ_cm12pti & Penta);
110
111
112        // DESTRUCTEUR :
113        ~PentaQ_cm12pti ();
114
115        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
116        // méthode virtuelle
117        Element* Nevez_copie() const { Element * el= new PentaQ_cm12pti(*this); return el;};
118
119        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQ_cm12pti
120        PentaQ_cm12pti& operator= (PentaQ_cm12pti& Penta);
121
122        // METHODES :
123        // 1) derivant des virtuelles pures
124
125        // affichage dans la sortie transmise, des variables duales "nom"
126        // aux differents points d'integration
127        // dans le cas ou nom est vide, affichage de "toute" les variables
128        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
129
130    protected :
131
132        // adressage des frontières linéiques et surfacique
133        // définit dans les classes dérivées, et utilisées pour la construction des frontières
134        // frontière linéique verticale (rectangle)
135        ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
136        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
137        // frontière linéique horizontale (triangle)
138        ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
139        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
140        // frontière surfacique verticale (rectangle)
141        ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
142        { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
143        // frontière surfacique horizontale (triangle)
144        ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
145        { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
146
147        // VARIABLES PRIVEES :
148        // place memoire commune a tous les elements PentaQ_cm12pti
149        static PentaMemb::DonnComPenta * doCoPentaQ_cm12pti;
150        // idem mais pour les indicateurs qui servent pour l'initialisation

```

```

151     static PentaMemb::UneFois  uneFois;
152
153     class NombresConstruirePentaQ_cml2pti : public NombresConstruire
154     { public: NombresConstruirePentaQ_cml2pti();
155     };
156     static NombresConstruirePentaQ_cml2pti nombre_V; // les nombres propres à l'élément
157
158     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
160     class ConsPentaQ_cml2pti : public ConstrucElement
161     { public : ConsPentaQ_cml2pti ()
162     { NouvelleTypeElement nouv (PENTAEDRE, QUADRATIQUE, MECA_SOLIDE_DEFORMABLE, this, "_cml2pti");
163     if (ParaGlob::NiveauImpression() >= 4)
164     cout << "\n initialisation PentaQ_cml2pti" << endl;
165     Element::listTypeElement.push_back(nouv);
166     };
167     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168     {Element * pt;
169     pt = new PentaQ_cml2pti (num_maill,num) ;
170     return pt;};
171     // ramene true si la construction de l'element est possible en fonction
172     // des variables globales actuelles: ex en fonction de la dimension
173     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174     };
175     static ConsPentaQ_cml2pti consPentaQ_cml2pti;
176     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @} // end of group
179 #endif
180
181
182
183

```

## 7.177 PentaQ\_cml18pti.h

```

1 // FICHER : PentaQ_cml18pti.h
2 // CLASSE : PentaQ_cml18pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      14/03/2003
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *
40 *   BUT:      La classe PentaQ_cml18pti permet de declarer des elements
41 *   Pentaedriques quadratiques incomplets et de realiser
42 *   le calcul du residu local et de la raideur locale pour une loi de
43 *   comportement donnee. La dimension de l'espace pour un tel element
44 *   est 3.
45 *
46 *   l'interpolation est triquadratique a 15 noeuds, le nombre de
47 *   point d'integration est de 18, 6 dans le triangle et 3 dans
48 *   l'épaisseur.

```

```

49 *      *
50 *      VERIFICATION:
51 *
52 *      ! date ! auteur ! but !
53 *      -----
54 *      ! ! ! !
55 *      $
56 *      *
57 *      MODIFICATIONS:
58 *      ! date ! auteur ! but !
59 *      -----
60 *      $
61 *      *****/
62
63 // -----classe pour un calcul de mecanique-----
64
65
66
67 #ifndef PENTAQ_CM18PTI_H
68 #define PENTAQ_CM18PTI_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 #include "Met_abstraite.h"
73 #include "GeomQuadrangle.h"
74 #include "Noeud.h"
75 #include "UtilLecture.h"
76 #include "Tenseur.h"
77 #include "NevezTenseur.h"
78 #include "Deformation.h"
79 #include "PentaMemb.h"
80 #include "FrontTriaQuad.h"
81 #include "FrontQuadQuad.h"
82 #include "FrontSegQuad.h"
83
84
85 /// @addtogroup groupe_des_elements_finis
86 /// @{
87 ///
88
89
90 class PentaQ_cm18pti : public PentaMemb
91 {
92
93     public :
94
95         // CONSTRUCTEURS :
96
97         // Constructeur par défaut
98         PentaQ_cm18pti ();
99
100        // Constructeur fonction d'un numero
101        // d'identification
102        PentaQ_cm18pti (int num_mail,int num_id);
103
104        // Constructeur fonction d'un numero de maillage et d'identification et
105        // du tableau de connexite des noeuds
106        PentaQ_cm18pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
107
108        // Constructeur de copie
109        PentaQ_cm18pti (const PentaQ_cm18pti & Penta);
110
111
112        // DESTRUCTEUR :
113        ~PentaQ_cm18pti ();
114
115        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
116        // méthode virtuelle
117        Element* Nevez_copie() const { Element * el= new PentaQ_cm18pti(*this); return el;};
118
119        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQ_cm18pti
120        PentaQ_cm18pti & operator= (PentaQ_cm18pti & Penta);
121
122        // METHODES :
123        // 1) derivant des virtuelles pures
124
125        // affichage dans la sortie transmise, des variables duales "nom"
126        // aux differents points d'integration
127        // dans le cas ou nom est vide, affichage de "toute" les variables
128        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
129
130    protected :
131
132        // adressage des frontières linéiques et surfacique
133        // définit dans les classes dérivées, et utilisées pour la construction des frontières
134        // frontière linéique verticale (rectangle)

```

```

135     ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
136     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
137     // frontière linéique horizontale (triangle)
138     ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
139     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
140     // frontière surfacique verticale (rectangle)
141     ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
142     { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
143     // frontière surfacique horizontale (triangle)
144     ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
145     { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
146
147     // VARIABLES PRIVEES :
148     // place memoire commune a tous les elements PentaQ_cm18pti
149     static PentaMemb::DonnComPenta * doCoPentaQ_cm18pti;
150     // idem mais pour les indicateurs qui servent pour l'initialisation
151     static PentaMemb::UneFois uneFois;
152
153     class NombresConstruirePentaQ_cm18pti : public NombresConstruire
154     { public: NombresConstruirePentaQ_cm18pti();
155     };
156     static NombresConstruirePentaQ_cm18pti nombre_V; // les nombres propres à l'élément
157
158     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
160     class ConsPentaQ_cm18pti : public ConstrucElement
161     { public : ConsPentaQ_cm18pti ()
162     { NouvelleTypeElement nouv (PENTAEDRE, QUADRATIQUE, MECA_SOLIDE_DEFORMABLE, this, "_cm18pti");
163     if (ParaGlob::NiveauImpression() >= 4)
164     cout << "\n initialisation PentaQ_cm18pti" << endl;
165     Element::listTypeElement.push_back(nouv);
166     };
167     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168     {Element * pt;
169     pt = new PentaQ_cm18pti (num_maill,num) ;
170     return pt;};
171     // ramene true si la construction de l'element est possible en fonction
172     // des variables globales actuelles: ex en fonction de la dimension
173     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174     };
175     static ConsPentaQ_cm18pti consPentaQ_cm18pti;
176     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @} // end of group
179 #endif
180
181
182
183

```

## 7.178 PentaQ\_cm3pti.h

```

1 // FICHER : PentaQ_cm3pti.h
2 // CLASSE : PentaQ_cm3pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      14/03/2003      *

```

```

34 *
35 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)      $   *
36 *
37 *   PROJET:     Herezh++                                $   *
38 *
39 * *****
40 *   BUT:   La classe PentaQ_cm3pti permet de declarer des elements *
41 * Pentaedriques quadratiques incomplets et de realiser          *
42 * le calcul du residu local et de la raideur locale pour une loi de *
43 * comportement donnee. La dimension de l'espace pour un tel element *
44 * est 3.
45 *
46 * l'interpolation est triquadratique a 15 noeuds, le nombre de    *
47 * point d'integration est de 3, 3 dans le triangle et 1 dans      *
48 * l'epaisseur.
49 *   *****
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but                               !   *
53 *   -----
54 *   !           !           !           !                           !   *
55 *   $
56 *   *****
57 *   MODIFICATIONS:
58 *   ! date !   auteur !           but                               !   *
59 *   -----
60 *   $
61 * *****/
62
63 // -----classe pour un calcul de mecanique-----
64
65
66
67 #ifndef PENTAQ_CM3PTI_H
68 #define PENTAQ_CM3PTI_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 #include "Met_abstraite.h"
73 #include "GeomQuadrangle.h"
74 #include "Noeud.h"
75 #include "UtilLecture.h"
76 #include "Tenseur.h"
77 #include "NevezTenseur.h"
78 #include "Deformation.h"
79 #include "PentaMemb.h"
80 #include "FrontTriaQuad.h"
81 #include "FrontQuadQuad.h"
82 #include "FrontSegQuad.h"
83
84
85 /// @addtogroup groupe_des_elements_finis
86 /// @{
87 ///
88
89
90 class PentaQ_cm3pti : public PentaMemb
91 {
92
93     public :
94
95         // CONSTRUCTEURS :
96
97         // Constructeur par defaut
98         PentaQ_cm3pti ();
99
100        // Constructeur fonction d'un numero
101        // d'identification
102        PentaQ_cm3pti (int num_mail,int num_id);
103
104        // Constructeur fonction d'un numero d'identification et
105        // du tableau de connexite des noeuds
106        PentaQ_cm3pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
107
108        // Constructeur de copie
109        PentaQ_cm3pti (const PentaQ_cm3pti& Penta);
110
111
112        // DESTRUCTEUR :
113        ~PentaQ_cm3pti ();
114
115        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
116        // méthode virtuelle
117        Element* Nevez_copie() const { Element * el= new PentaQ_cm3pti(*this); return el;};
118
119        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQ_cm3pti

```

```

120     PentaQ_cm3pti& operator= (PentaQ_cm3pti& Penta);
121
122     // METHODES :
123 // 1) derivant des virtuelles pures
124
125     // affichage dans la sortie transmise, des variables duales "nom"
126     // aux differents points d'integration
127     // dans le cas ou nom est vide, affichage de "toute" les variables
128     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
129
130 protected :
131
132     // adressage des frontieres linéiques et surfacique
133     // définit dans les classes dérivées, et utilisées pour la construction des frontieres
134     // frontière linéique verticale (rectangle)
135     ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
136     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
137     // frontière linéique horizontale (triangle)
138     ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
139     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
140     // frontière surfacique verticale (rectangle)
141     ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
142     { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
143     // frontière surfacique horizontale (triangle)
144     ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
145     { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
146
147 // VARIABLES PRIVEES :
148 // place memoire commune a tous les elements PentaQ_cm3pti
149 static PentaMemb::DonnComPenta * doCoPentaQ_cm3pti;
150 // idem mais pour les indicateurs qui servent pour l'initialisation
151 static PentaMemb::UneFois uneFois;
152
153 class NombresConstruirePentaQ_cm3pti : public NombresConstruire
154 { public: NombresConstruirePentaQ_cm3pti();
155 };
156 static NombresConstruirePentaQ_cm3pti nombre_V; // les nombres propres à l'élément
157
158 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
160 class ConsPentaQ_cm3pti : public ConstrucElement
161 { public : ConsPentaQ_cm3pti ()
162 { NouvelleTypeElement nouv(PENTAEDRE, QUADRATIQUE, MECA_SOLIDE_DEFORMABLE, this, "_cm3pti");
163   if (ParaGlob::NiveauImpression() >= 4)
164     cout << "\n initialisation PentaQ_cm3pti" << endl;
165     Element::listTypeElement.push_back(nouv);
166   };
167   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168   {Element * pt;
169     pt = new PentaQ_cm3pti (num_maill,num) ;
170     return pt;};
171   // ramene true si la construction de l'element est possible en fonction
172   // des variables globales actuelles: ex en fonction de la dimension
173   bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174   };
175 static ConsPentaQ_cm3pti consPentaQ_cm3pti;
176 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @} // end of group
179 #endif
180
181
182
183

```

## 7.179 PentaQ\_cm9pti.h

```

1 // FICHER : PentaQ_cm9pti.h
2 // CLASSE : PentaQ_cm9pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by

```



```
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      14/03/2003
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 * *****/
40 *   BUT:      La classe PentaQ_cm9pti permet de declarer des elements
41 * Pentaedriques quadratiques incomplets et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 3.
45 *
46 * l'interpolation est triquadratique a 15 noeuds, le nombre de
47 * point d'integration est de 9, 3 dans le triangle et 3 dans
48 * l'epaisseur.
49 *   *****
50 *
51 *   VERIFICATION:
52 *
53 *   ! date ! auteur ! but
54 *   -----
55 *   !           !           !
56 *   *****
57 *   MODIFICATIONS:
58 *   ! date ! auteur ! but
59 *   -----
60 *   $
61 * *****/
62 // -----classe pour un calcul de mecanique-----
63
64
65
66
67 #ifndef PENTAQ_CM9PTI_H
68 #define PENTAQ_CM9PTI_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 #include "Met_abstraite.h"
73 #include "GeomQuadrangle.h"
74 #include "Noeud.h"
75 #include "UtilLecture.h"
76 #include "Tenseur.h"
77 #include "NevezTenseur.h"
78 #include "Deformation.h"
79 #include "PentaMemb.h"
80 #include "FrontTriaQuad.h"
81 #include "FrontQuadQuad.h"
82 #include "FrontSegQuad.h"
83
84
85 /// @addtogroup groupe_des_elements_finis
86 /// @{
87 ///
88
89
90 class PentaQ_cm9pti : public PentaMemb
91 {
92
93 public :
94
95     // CONSTRUCTEURS :
96
97     // Constructeur par défaut
98     PentaQ_cm9pti ();
99
100     // Constructeur fonction d'un numero
101     // d'identification
102     PentaQ_cm9pti (int num_mail,int num_id);
103
104     // Constructeur fonction d'un numero d'identification et
```

```

105 // du tableau de connexite des noeuds
106 PentaQ_cm9pti (int num_maill,int num_id,const Tableau<Noeud *>& tab);
107
108 // Constructeur de copie
109 PentaQ_cm9pti (const PentaQ_cm9pti& Penta);
110
111
112 // DESTRUCTEUR :
113 ~PentaQ_cm9pti ();
114
115 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
116 // méthode virtuelle
117 Element* Nevez_copie() const { Element * el= new PentaQ_cm9pti(*this); return el;};
118
119 // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQ_cm9pti
120 PentaQ_cm9pti& operator= (PentaQ_cm9pti& Penta);
121
122 // METHODES :
123 // 1) derivant des virtuelles pures
124
125 // affichage dans la sortie transmise, des variables duales "nom"
126 // aux differents points d'integration
127 // dans le cas ou nom est vide, affichage de "toute" les variables
128 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
129
130 protected :
131
132 // adressage des frontieres linéiques et surfacique
133 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
134 // frontière linéique verticale (rectangle)
135 ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
136 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
137 // frontière linéique horizontale (triangle)
138 ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
139 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
140 // frontière surfacique verticale (rectangle)
141 ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
142 { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
143 // frontière surfacique horizontale (triangle)
144 ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
145 { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
146
147 // VARIABLES PRIVEES :
148 // place memoire commune a tous les elements PentaQ_cm9pti
149 static PentaMemb::DonnComPenta * doCoPentaQ_cm9pti;
150 // idem mais pour les indicateurs qui servent pour l'initialisation
151 static PentaMemb::UneFois uneFois;
152
153 class NombresConstruirePentaQ_cm9pti : public NombresConstruire
154 { public: NombresConstruirePentaQ_cm9pti();
155 };
156 static NombresConstruirePentaQ_cm9pti nombre_V; // les nombres propres à l'élément
157
158 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
160 class ConsPentaQ_cm9pti : public ConstrucElement
161 { public : ConsPentaQ_cm9pti ()
162 { NouvelleTypeElement nouv (PENTAEDRE,QUADRATIQUE,MECA_SOLIDE_DEFORMABLE,this,"_cm9pti");
163 if (ParaGlob::NiveauImpression() >= 4)
164 cout << "\n initialisation PentaQ_cm9pti" << endl;
165 Element::listTypeElement.push_back(nouv);
166 };
167 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168 {Element * pt;
169 pt = new PentaQ_cm9pti (num_maill,num) ;
170 return pt;};
171 // ramene true si la construction de l'element est possible en fonction
172 // des variables globales actuelles: ex en fonction de la dimension
173 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174 };
175 static ConsPentaQ_cm9pti consPentaQ_cm9pti;
176 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @} // end of group
179 #endif
180
181
182
183

```

## 7.180 PentaQComp.h

```

1 // FICHER : PentaQComp.h
2 // CLASSE : PentaQComp
3

```

```

4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           14/03/2003
34 *
35 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:        Herezh++
38 *
39 *****/
40 *   BUT:   La classe PentaQ permet de declarer des elements
41 *   Pentaedriques quadratiques complets et de realiser
42 *   le calcul du residu local et de la raideur locale pour une loi de
43 *   comportement donnee. La dimension de l'espace pour un tel element
44 *   est 3.
45 *
46 *   l'interpolation par défaut est triquadratique, le nombre
47 *   de point d'integration par défaut est de 6.
48 *
49 *   VERIFICATION:
50 *
51 *   ! date !   auteur !   but
52 *   -----
53 *   !       !       !
54 *
55 *   *****
56 *   MODIFICATIONS:
57 *   ! date !   auteur !   but
58 *   -----
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef PENTAQCOMP_H
67 #define PENTAQCOMP_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "PentaMemb.h"
79 #include "FrontTriaQuad.h"
80 #include "FrontQuadQC.h"
81 #include "FrontSegQuad.h"
82
83
84 /// @addtogroup groupe_des_elements_finis
85 /// @{
86 ///
87
88
89 class PentaQComp : public PentaMemb

```

```

90 {
91
92     public :
93
94         // CONSTRUCTEURS :
95
96         // Constructeur par défaut
97         PentaQComp ();
98
99         // Constructeur fonction d'un numero
100        // d'identification
101        PentaQComp (int num_mail,int num_id);
102
103        // Constructeur fonction d'un numero de maillage et d'identification et
104        // du tableau de connexite des noeuds
105        PentaQComp (int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        PentaQComp (const PentaQComp& Penta);
109
110
111        // DESTRUCTEUR :
112        ~PentaQComp ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new PentaQComp(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQComp
119        PentaQComp& operator= (PentaQComp& Penta);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // renseignement d'un élément complet à partir d'un élément incomplet de même type
125        // retourne les nouveaux noeuds construit à partir de l'interpolation incomplète.
126        // dans le cas l'élément n'est pas concerné, retourne une liste vide
127        // ramène également une liste de même dimension contenant les bornes en numéros de noeuds
128        // entre lesquelles il faut définir les nouveaux numéros de noeuds si l'on veut conserver
129        // une largeur de bande optimisée du même type
130        // nbnt+1: est le premier numéro de noeud utilisable pour les nouveaux noeuds
131        list <Noeud *> Construct_from_imcomplet(const Element & elem,list <DeuxEntiers> & li_bornes,int
nbnt);
132
133        // affichage dans la sortie transmise, des variables duales "nom"
134        // aux differents points d'integration
135        // dans le cas ou nom est vide, affichage de "toute" les variables
136        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
137
138    protected :
139
140        // adressage des frontières linéiques et surfacique
141        // définit dans les classes dérivées, et utilisées pour la construction des frontières
142        // frontière linéique verticale (rectangle)
143        ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
144        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
145        // frontière linéique horizontale (triangle)
146        ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
147        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
148        // frontière surfacique verticale (rectangle)
149        ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
150        { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
151        // frontière surfacique horizontale (triangle)
152        ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
153        { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
154
155        // VARIABLES PRIVEES :
156        // place memoire commune a tous les elements PentaQComp
157        static PentaMemb::DonnComPenta * doCoPentaQComp;
158        // idem mais pour les indicateurs qui servent pour l'initialisation
159        static PentaMemb::UneFois uneFois;
160
161        class NombresConstruirePentaQComp : public NombresConstruire
162        { public: NombresConstruirePentaQComp();
163          };
164        static NombresConstruirePentaQComp nombre_V; // les nombres propres à l'élément
165
166        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
167        //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
168        class ConsPentaQComp : public ConstrucElement
169        { public : ConsPentaQComp ()
170          { NouvelleTypeElement nouv (PENTAEDRE, QUADRACOMPL, MECA_SOLIDE_DEFORMABLE, this);
171            if (ParaGlob::NiveauImpression() >= 4)
172              cout << "\n initialisation PentaQComp" << endl;
173              Element::listTypeElement.push_back (nouv);
174          };
175        Element * NouvelElement(int num_mail,int num) // un nouvel élément sans rien

```

```

176         {Element * pt;
177         pt = new PentaQComp (num_maill,num) ;
178         return pt;};
179         // ramene true si la construction de l'element est possible en fonction
180         // des variables globales actuelles: ex en fonction de la dimension
181         bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
182     };
183     static ConsPentaQComp consPentaQComp;
184     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
185 };
186 /// @} // end of group
187 #endif
188
189
190
191

```

## 7.181 PentaQComp\_cm12pti.h

```

1 // FICHER : PentaQComp_cm12pti.h
2 // CLASSE : PentaQComp_cm12pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      14/03/2003
34 *
35 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      *****
41 *      BUT:      La classe PentaQ permet de declarer des elements
42 *      Pentaedriques quadratiques complets et de realiser
43 *      le calcul du residu local et de la raideur locale pour une loi de
44 *      comportement donnee. La dimension de l'espace pour un tel element
45 *      est 3.
46 *
47 *      l'interpolation par défaut est triquadratique, le nombre
48 *      de point d'integration est de 12, 4 dans la surface du triangle et
49 *      3 dans l'épaisseur.
50 *
51 *      *****
52 *
53 *      VERIFICATION:
54 *
55 *      ! date ! auteur ! but
56 *      -----
57 *      ! ! !
58 *
59 *      *****
60 *
61 *      MODIFICATIONS:
62 *
63 *      ! date ! auteur ! but
64 *      -----
65 *
66 *      *****/
67 // -----classe pour un calcul de mecanique-----
68
69

```

```

66
67 #ifndef PENTAQCOMP_CML2PTI_H
68 #define PENTAQCOMP_CML2PTI_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 #include "Met_abstraite.h"
73 #include "GeomQuadrangle.h"
74 #include "Noeud.h"
75 #include "UtilLecture.h"
76 #include "Tenseur.h"
77 #include "NevezTenseur.h"
78 #include "Deformation.h"
79 #include "PentaMemb.h"
80 #include "FrontTriaQuad.h"
81 #include "FrontQuadQC.h"
82 #include "FrontSegQuad.h"
83
84
85 /// @addtogroup groupe_des_elements_finis
86 /// @{
87 ///
88
89
90 class PentaQComp_cml2pti : public PentaMemb
91 {
92
93     public :
94
95         // CONSTRUCTEURS :
96
97         // Constructeur par défaut
98         PentaQComp_cml2pti ();
99
100        // Constructeur fonction d'un numero
101        // d'identification
102        PentaQComp_cml2pti (int num_mail,int num_id);
103
104        // Constructeur fonction d'un numero de maillage et d'identification et
105        // du tableau de connexite des noeuds
106        PentaQComp_cml2pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
107
108        // Constructeur de copie
109        PentaQComp_cml2pti (const PentaQComp_cml2pti& Penta);
110
111
112        // DESTRUCTEUR :
113        ~PentaQComp_cml2pti ();
114
115        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
116        // méthode virtuelle
117        Element* Nevez_copie() const { Element * el= new PentaQComp_cml2pti(*this); return el;};
118
119        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQComp_cml2pti
120        PentaQComp_cml2pti& operator= (PentaQComp_cml2pti& Penta);
121
122        // METHODES :
123        // 1) derivant des virtuelles pures
124
125        // affichage dans la sortie transmise, des variables duales "nom"
126        // aux differents points d'integration
127        // dans le cas ou nom est vide, affichage de "toute" les variables
128        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
129
130    protected :
131
132        // adressage des frontieres linéiques et surfacique
133        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
134        // frontière linéique verticale (rectangle)
135        ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
136        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
137        // frontière linéique horizontale (triangle)
138        ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
139        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
140        // frontière surfacique verticale (rectangle)
141        ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
142        { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
143        // frontière surfacique horizontale (triangle)
144        ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
145        { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
146
147        // VARIABLES PRIVEES :
148        // place memoire commune a tous les elements PentaQComp_cml2pti
149        static PentaMemb::DonnComPenta * doCoPentaQComp_cml2pti;
150        // idem mais pour les indicateurs qui servent pour l'initialisation
151        static PentaMemb::UneFois uneFois;
152

```

```

153     class NombresConstruirePentaQComp_cm12pti : public NombresConstruire
154     { public: NombresConstruirePentaQComp_cm12pti();
155     };
156     static NombresConstruirePentaQComp_cm12pti nombre_V; // les nombres propres à l'élément
157
158     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
160     class ConsPentaQComp_cm12pti : public ConstrucElement
161     { public : ConsPentaQComp_cm12pti ()
162     { NouvelleTypeElement nouv(PENTAEDRE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cm12pti");
163     if (ParaGlob::NiveauImpression() >= 4)
164     cout << "\n initialisation PentaQComp_cm12pti" << endl;
165     Element::listTypeElement.push_back(nouv);
166     };
167     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168     {Element * pt;
169     pt = new PentaQComp_cm12pti (num_maill,num) ;
170     return pt;};
171     // ramene true si la construction de l'element est possible en fonction
172     // des variables globales actuelles: ex en fonction de la dimension
173     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174     };
175     static ConsPentaQComp_cm12pti consPentaQComp_cm12pti;
176     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @} // end of group
179 #endif
180
181
182
183

```

## 7.182 PentaQComp\_cm18pti.h

```

1 // FICHER : PentaQComp_cm18pti.h
2 // CLASSE : PentaQComp_cm18pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      14/03/2003
34 *
35 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      *****
41 *      BUT:      La classe PentaQ permet de declarer des elements
42 *      Pentaedriques quadratiques complets et de realiser
43 *      le calcul du residu local et de la raideur locale pour une loi de
44 *      comportement donnee. La dimension de l'espace pour un tel element
45 *      est 3.
46 *
47 *      l'interpolation par défaut est triquadratique, le nombre
48 *      de point d'integration par défaut est de 18, 6 dans la surface du
49 *      triangle et 3 dans l'épaisseur.
50 *
51 *      *****
52 *
53 *      VERIFICATION:
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```

```

51 *
52 * ! date ! auteur ! but ! *
53 * ----- *
54 * ! ! ! ! *
55 * $ *
56 * //***** *
57 * MODIFICATIONS: *
58 * ! date ! auteur ! but ! *
59 * ----- *
60 * $ *
61 *****/
62
63 // -----classe pour un calcul de mecanique-----
64
65
66
67 #ifndef PENTAQCOMP_CM18PTI_H
68 #define PENTAQCOMP_CM18PTI_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 #include "Met_abstraite.h"
73 #include "GeomQuadrangle.h"
74 #include "Noeud.h"
75 #include "UtilLecture.h"
76 #include "Tenseur.h"
77 #include "NevezTenseur.h"
78 #include "Deformation.h"
79 #include "PentaMemb.h"
80 #include "FrontTriaQuad.h"
81 #include "FrontQuadQC.h"
82 #include "FrontSegQuad.h"
83
84
85 /// @addtogroup groupe_des_elements_finis
86 /// @{
87 ///
88
89
90 class PentaQComp_cm18pti : public PentaMemb
91 {
92
93     public :
94
95         // CONSTRUCTEURS :
96
97         // Constructeur par défaut
98         PentaQComp_cm18pti ();
99
100        // Constructeur fonction d'un numero
101        // d'identification
102        PentaQComp_cm18pti (int num_mail,int num_id);
103
104        // Constructeur fonction d'un numero de maillage et d'identification et
105        // du tableau de connexite des noeuds
106        PentaQComp_cm18pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
107
108        // Constructeur de copie
109        PentaQComp_cm18pti (const PentaQComp_cm18pti& Penta);
110
111
112        // DESTRUCTEUR :
113        ~PentaQComp_cm18pti ();
114
115        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
116        // méthode virtuelle
117        Element* Nevez_copie() const { Element * el= new PentaQComp_cm18pti(*this); return el;};
118
119        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQComp_cm18pti
120        PentaQComp_cm18pti& operator= (PentaQComp_cm18pti& Penta);
121
122        // METHODES :
123        // 1) derivant des virtuelles pures
124
125        // affichage dans la sortie transmise, des variables duales "nom"
126        // aux differents points d'integration
127        // dans le cas ou nom est vide, affichage de "toute" les variables
128        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
129
130    protected :
131
132        // adressage des frontieres linéiques et surfacique
133        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
134        // frontière linéique verticale (rectangle)
135        ElFrontiere* new_frontiere_lin_rec(Tableau<Noeud *> & tab, DdlElement& ddelem)
136        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
137        // frontière linéique horizontale (triangle)

```



```

138     ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
139     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
140     // frontiere surfacique verticale (rectangle)
141     ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
142     { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
143     // frontiere surfacique horizontale (triangle)
144     ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
145     { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
146
147     // VARIABLES PRIVEES :
148     // place memoire commune a tous les elements PentaQComp_cm18pti
149     static PentaMemb::DonnComPenta * doCoPentaQComp_cm18pti;
150     // idem mais pour les indicateurs qui servent pour l'initialisation
151     static PentaMemb::UneFois uneFois;
152
153     class NombresConstruirePentaQComp_cm18pti : public NombresConstruire
154     { public: NombresConstruirePentaQComp_cm18pti();
155       };
156     static NombresConstruirePentaQComp_cm18pti nombre_V; // les nombres propres à l'élément
157
158     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
160     class ConsPentaQComp_cm18pti : public ConstrucElement
161     { public : ConsPentaQComp_cm18pti ()
162       { NouvelleTypeElement nouv (PENTAEDRE, QUADRACOMPL, MECA_SOLIDE_DEFORMABLE, this, "_cm18pti");
163         if (ParaGlob::NiveauImpression() >= 4)
164           cout << "\n initialisation PentaQComp_cm18pti" << endl;
165         Element::listTypeElement.push_back(nouv);
166       };
167       Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168       {Element * pt;
169         pt = new PentaQComp_cm18pti (num_maill,num) ;
170         return pt;};
171       // ramene true si la construction de l'element est possible en fonction
172       // des variables globales actuelles: ex en fonction de la dimension
173       bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174     };
175     static ConsPentaQComp_cm18pti consPentaQComp_cm18pti;
176     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @} // end of group
179 #endif
180
181
182
183

```

## 7.183 PentaQComp\_cm9pti.h

```

1 // FICHER : PentaQComp_cm9pti.h
2 // CLASSE : PentaQComp_cm9pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      14/03/2003
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *

```

```

37 *   PROJET:   Herezh++   *
38 *                                           $   *
39 *****
40 *   BUT:   La classe PentaQ permet de declarer des elements   *
41 * Pentaedriques quadratiques complets et de realiser   *
42 * le calcul du residu local et de la raideur locale pour une loi de   *
43 * comportement donnee. La dimension de l'espace pour un tel element   *
44 * est 3.   *
45 *   *   *
46 * l'interpolation par defaut est triquadratique, le nombre   *
47 * de point d'integration est de 9, 3 dans la surface du triangle et   *
48 * 3 dans l'epaisseur.   *
49 *   *****   *
50 *   VERIFICATION:   *
51 *   *   *
52 *   ! date !   auteur !   but   !   *
53 *   -----   *
54 *   !   !   !   !   *
55 *   *   $   *
56 *   *****   *
57 *   MODIFICATIONS:   *
58 *   ! date !   auteur !   but   !   *
59 *   -----   *
60 *   *   $   *
61 *****/
62
63 // -----classe pour un calcul de mecanique-----
64
65
66
67 #ifndef PENTAQCOMP_CM9PTI_H
68 #define PENTAQCOMP_CM9PTI_H
69
70 #include "ParaGlob.h"
71 #include "ElemMeca.h"
72 #include "Met_abstraite.h"
73 #include "GeomQuadrangle.h"
74 #include "Noeud.h"
75 #include "UtilLecture.h"
76 #include "Tenseur.h"
77 #include "NevezTenseur.h"
78 #include "Deformation.h"
79 #include "PentaMemb.h"
80 #include "FrontTriaQuad.h"
81 #include "FrontQuadQC.h"
82 #include "FrontSegQuad.h"
83
84
85 /// @addtogroup groupe_des_elements_finis
86 /// @{
87 ///
88
89
90 class PentaQComp_cm9pti : public PentaMemb
91 {
92
93     public :
94
95         // CONSTRUCTEURS :
96
97         // Constructeur par defaut
98         PentaQComp_cm9pti ();
99
100        // Constructeur fonction d'un numero
101        // d'identification
102        PentaQComp_cm9pti (int num_mail,int num_id);
103
104        // Constructeur fonction d'un numero de maillage et d'identification et
105        // du tableau de connexite des noeuds
106        PentaQComp_cm9pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
107
108        // Constructeur de copie
109        PentaQComp_cm9pti (const PentaQComp_cm9pti& Penta);
110
111
112        // DESTRUCTEUR :
113        ~PentaQComp_cm9pti ();
114
115        // creation d'un element de copie: utilisation de l'operateur new et du constructeur de copie
116        // methode virtuelle
117        Element* Nevez_copie() const { Element * el= new PentaQComp_cm9pti(*this); return el;};
118
119        // Surcharge de l'operateur = : realise l'egalite entre deux instances de PentaQComp_cm9pti
120        PentaQComp_cm9pti& operator= (PentaQComp_cm9pti& Penta);
121
122        // METHODES :

```

```

123 // 1) derivant des virtuelles pures
124
125     // affichage dans la sortie transmise, des variables duales "nom"
126     // aux differents points d'integration
127     // dans le cas ou nom est vide, affichage de "toute" les variables
128     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
129
130 protected :
131
132     // adressage des frontieres linéiques et surfacique
133     // définit dans les classes dérivées, et utilisées pour la construction des frontieres
134     // frontière linéique verticale (rectangle)
135     ElFrontiere* new_frontiere_lin_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
136     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
137     // frontière linéique horizontale (triangle)
138     ElFrontiere* new_frontiere_lin_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
139     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
140     // frontière surfacique verticale (rectangle)
141     ElFrontiere* new_frontiere_surf_rec(Tableau <Noeud *> & tab, DdlElement& ddelem)
142     { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
143     // frontière surfacique horizontale (triangle)
144     ElFrontiere* new_frontiere_surf_tri(Tableau <Noeud *> & tab, DdlElement& ddelem)
145     { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
146
147 // VARIABLES PRIVEES :
148 // place memoire commune a tous les elements PentaQComp_cm9pti
149 static PentaMemb::DonnComPenta * doCoPentaQComp_cm9pti;
150 // idem mais pour les indicateurs qui servent pour l'initialisation
151 static PentaMemb::UneFois uneFois;
152
153 class NombresConstruirePentaQComp_cm9pti : public NombresConstruire
154 { public: NombresConstruirePentaQComp_cm9pti();
155 };
156 static NombresConstruirePentaQComp_cm9pti nombre_V; // les nombres propres à l'élément
157
158 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
159 //ajout de l'element dans la liste : listTypeElement, geree par la class Element
160 class ConsPentaQComp_cm9pti : public ConstrucElement
161 { public : ConsPentaQComp_cm9pti ()
162 { NouvelleTypeElement nouv(PENTAEDRE, QUADRACOMPL, MECA_SOLIDE_DEFORMABLE, this, "_cm9pti");
163   if (ParaGlob::NiveauImpression() >= 4)
164     cout << "\n initialisation PentaQComp_cm9pti" << endl;
165     Element::listTypeElement.push_back(nouv);
166   };
167   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
168   {Element * pt;
169     pt = new PentaQComp_cm9pti (num_maill,num) ;
170     return pt;};
171   // ramene true si la construction de l'element est possible en fonction
172   // des variables globales actuelles: ex en fonction de la dimension
173   bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
174   };
175 static ConsPentaQComp_cm9pti consPentaQComp_cm9pti;
176 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
177 };
178 /// @} // end of group
179 #endif
180
181
182
183

```

## 7.184 PtIntegMecalInterne.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty

```

```

22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      26/11/2006
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: Classe pour stocker les informations aux points
39 *         d'intégration mécanique (plutôt puissance interne)
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *****/
54 #ifndef PTINTEGMECAINTERNE_H
55 #define PTINTEGMECAINTERNE_H
56
57 #include "Tenseur.h"
58 #include "Vecteur.h"
59 #include "Temps_CPU_HZpp.h"
60
61
62 /** @defgroup Groupe_concernant_les_points_integration
63 *
64 *   BUT: groupe relatif aux points d'intégration
65 *
66 *
67 * \author Gérard Rio
68 * \version 1.0
69 * \date 26/11/2006
70 * \brief groupe relatif aux points d'intégration
71 *
72 */
73
74 /// @addtogroup Groupe_concernant_les_points_integration
75 /// @{
76 ///
77
78
79 class PtIntegMecaInterne
80
81 {
82     // surcharge de l'operator de lecture
83     friend istream & operator » (istream &, PtIntegMecaInterne &);
84     // surcharge de l'operator d'écriture
85     friend ostream & operator « (ostream &, const PtIntegMecaInterne &);
86
87 public :
88     // CONSTRUCTEURS :
89     // constructeur par défaut
90     PtIntegMecaInterne();
91     // constructeur fonction de la dimension de tenseurs
92     PtIntegMecaInterne(int dimtens);
93     // constructeur de copie
94     PtIntegMecaInterne(const PtIntegMecaInterne& pti);
95
96     // DESTRUCTEUR :
97     ~PtIntegMecaInterne();
98
99     // METHODES PUBLIQUES :
100    // Surcharge de l'opérateur =
101    PtIntegMecaInterne& operator= ( const PtIntegMecaInterne& pti);
102
103    // deformation finale
104    TenseurBB * EpsBB() {return epsBB;};
105    // vitesse finale de deformation
106    TenseurBB * DepsBB() {return depsBB;};
107    // variation de deformation entre t et t + delta t
108    TenseurBB * DeltaEpsBB() {return deltaEpsBB;};

```

```

108 // contrainte finale
109 TenseurHH * SigHH() {return sigHH;};
110 // contrainte en début d'incrément
111 TenseurHH * SigHH_t() {return sigHH_t;};
112 // module de compressibilité
113 // si = -1, cela veut dire que le module n'a pas été mis à jour
114 double& ModuleCompressibilite() {return module_compressibilite;};
115 // module de cisaillement
116 // si = -1, cela veut dire que le module n'a pas été mis à jour
117 double& ModuleCisaillement() { return module_cisaillement;};
118 // volume élémentaire au pti
119 // si = -1, cela veut dire que le volume élémentaire n'a pas été mis à jour
120 double& Volume_pti() { return volume_pti;};
121 // tableau relatif aux différentes grandeurs de type def scalaires équivalentes
122 // def_equi(1) = deformation cumulée = somme des sqrt(2./3. * (delta_eps_barre_BH &&
delta_eps_barre_BH) );
123 // def_equi(2) = deformation duale de la contrainte de mises = sqrt(2./3. * (eps_barre_BH &&
eps_barre_BH) );
124 // def_equi(3) = niveau maxi atteint par def_equi(2)
125 // def_equi(4) = delta def cumulée = sqrt(2./3. * (delta_eps_barre_BH && delta_eps_barre_BH));
126 Tableau <double>& Deformation_equi() { return def_equi;};
127 Tableau <double>& Deformation_equi_t() { return def_equi_t;};
128
129 // idem coté contrainte
130 // (1) = contrainte_mises, (2) = contrainte_tresca
131 Tableau <double>& Sig_equi() { return sig_equi;};
132 Tableau <double>& Sig_equi_t() { return sig_equi_t;};
133
134 // les positions modifiables
135 Coordonnee& M_0() {return M0;};
136 Coordonnee& M_t() {return Mt;};
137 Coordonnee& M_tdt() {return Mtdt;};
138 // repérage dans maillage
139 int Nb_mail() const {return mail;};
140 int Nb_ele() const {return nele;};
141 int Nb_pti() const {return npti;};
142 void Change_Nb_mail(int nb) {mail=nb;};
143 void Change_Nb_ele(int nb) {nele=nb;};
144 void Change_Nb_pti(int nb) {npti=nb;};
145 void Signature() const
146 {cout << " mail: " << mail << ", ele= " << nele << ", pti=" << npti << " "};
147
148 // --- temps cpu
149 // tps cpu relatif à la métrique uniquement
150 Temps_CPU_HZpp& TpsMetrrique() {return tpsMetrrique;};
151 // temps cumulé relatif à la loi de comportement
152 Temps_CPU_HZpp& Tps_cpu_loi_comp() {return tps_cpu_loi_comp;};
153
154 // ---- acces idem en constants
155 // deformation finale
156 const TenseurBB & EpsBB_const() const {return *epsBB;};
157 // vitesse finale de deformation
158 const TenseurBB & DepsBB_const() const {return *depsBB;};
159 // variation de deformation entre t et t + delta t
160 const TenseurBB & DeltaEpsBB_const() const {return *deltaEpsBB;};
161 // contrainte finale
162 const TenseurHH & SigHH_const() const {return *sigHH;};
163 // contrainte en début d'incrément
164 const TenseurHH & SigHH_t_const() const {return *sigHH_t;};
165 // module de compressibilité
166 // si = -1, cela veut dire que le module n'a pas été mis à jour
167 const double& ModuleCompressibilite_const() const {return module_compressibilite;};
168 // module de cisaillement
169 // si = -1, cela veut dire que le module n'a pas été mis à jour
170 const double& ModuleCisaillement_const() const { return module_cisaillement;};
171 // volume élémentaire au pti
172 // si = -1, cela veut dire que le volume élémentaire n'a pas été mis à jour
173 const double& Volume_pti_const() { return volume_pti;};
174
175 // tableau constant relatif aux différentes grandeurs de type def scalaires équivalentes
176 // def_equi(1) = deformation cumulée = somme des sqrt(2./3. * (delta_eps_barre_BH &&
delta_eps_barre_BH) );
177 // def_equi(2) = deformation duale de la contrainte de mises = sqrt(2./3. * (eps_barre_BH &&
eps_barre_BH) );
178 // def_equi(3) = niveau maxi atteint par def_equi(2)
179 // def_equi(4) = delta def cumulée = sqrt(2./3. * (delta_eps_barre_BH && delta_eps_barre_BH));
180 const Tableau <double>& Deformation_equi_const () const { return def_equi;};
181 const Tableau <double>& Deformation_equi_t_const () const { return def_equi_t;};
182
183 // idem coté contrainte
184 // (1) = contrainte_mises, (2) = contrainte_tresca
185 const Tableau <double>& Sig_equi_const() const { return sig_equi;};
186 const Tableau <double>& Sig_equi_t_const() const { return sig_equi_t;};
187
188 // les positions en const
189 const Coordonnee& M0_const() const {return M0;};
190 const Coordonnee& Mt_const() const {return Mt;};

```

```

191     const Coordonnee& Mtdt_const() const {return Mtdt;};
192     // --- temps cpu
193     // tps cpu relatif à la métrique uniquement
194     const Temps_CPU_HZpp& TpsMetrrique_const() const {return tpsMetrrique;};
195     // temps cumulé relatif à la loi de comportement
196     const Temps_CPU_HZpp& Tps_cpu_loi_comp_const() const {return tps_cpu_loi_comp;};
197
198     // --- cas des invariants: a priori non-activé, pour les utiliser il faut les activer avant
199     // le statut: activé ou pas activé
200     bool Statut_Invariants_deformation () const {return ((epsInvar!=NULL) ? true :false);};
201     bool Statut_Invariants_vitesseDeformation () const {return ((depsInvar!=NULL) ? true :false);};
202     bool Statut_Invariants_contrainte () const {return ((sigInvar!=NULL) ? true :false);};
203     // le changement de statut, si c'est déjà ok, on ne fait rien
204     // nevez_statut: indique si l'on veut activer ou au contraire désactiver
205     void Change_statut_Invariants_deformation (bool nevez_statut);
206     void Change_statut_Invariants_vitesseDeformation (bool nevez_statut);
207     void Change_statut_Invariants_contrainte (bool nevez_statut);
208
209     // la récupération des invariants (s'ils sont actifs !!) en lecture écriture
210     // rappel sur la signification des différentes composantes des vecteurs "invariant":
211     // (3) ==> Det();
212     // (2) ==> II() = A:A;
213     // (1) ==> Trace();
214
215     // invariants de déformations
216     Vecteur& EpsInvar() {return *epsInvar;};
217     const Vecteur& EpsInvar_const() const {return *epsInvar;};
218     // invariants des vitesses de déformations
219     Vecteur& DepsInvar() {return *depsInvar;};
220     const Vecteur& DepsInvar_const() const {return *depsInvar;};
221     // invariants des contraintes
222     Vecteur& SigInvar() {return *sigInvar;};
223     const Vecteur& SigInvar_const() const {return *sigInvar;};
224
225     // actualisation des grandeurs actives de t+dt vers t, pour celles qui existent
226     // sous ces deux formes
227     void TdtversT();
228     // actualisation des grandeurs actives de t vers tdt, pour celles qui existent
229     // sous ces deux formes
230     void TversTdt();
231
232     //==== méthode particulière pour un passage de l'ordre 2D à 3D des tenseurs et l'inverse
233     //====
234     // plusZero: = true: indique qu'il faut compléter les grandeurs manquantes avec des 0
235     //           = false: on ne complète pas
236     // il faut que ptintmec comporte des tenseurs d'ordre 2 et this des tenseurs 3D
237     void Affectation_2D_a_3D(const PtIntegMecaInterne& ptintmec,bool plusZero);
238     // l'inverse: comme le conteneur d'arrivée est plus petit, il n'y a pas de complétion
239     // il faut que ptintmec comporte des tenseurs d'ordre 3 et this des tenseurs 2D
240     void Affectation_3D_a_2D(const PtIntegMecaInterne& ptintmec);
241
242     //==== méthode particulière pour un passage de l'ordre 1D à 3D des tenseurs et l'inverse
243     //====
244     // plusZero: = true: indique qu'il faut compléter les grandeurs manquantes avec des 0
245     //           = false: on ne complète pas
246     // il faut que ptintmec comporte des tenseurs d'ordre 1 et this des tenseurs 3D
247     void Affectation_1D_a_3D(const PtIntegMecaInterne& ptintmec,bool plusZero);
248     // l'inverse: comme le conteneur d'arrivée est plus petit, il n'y a pas de complétion
249     // il faut que ptintmec comporte des tenseurs d'ordre 3 et this des tenseurs 1D
250     void Affectation_3D_a_1D(const PtIntegMecaInterne& ptintmec);
251
252     //==== lecture écriture dans base info =====
253     // cas donne le niveau de la récupération
254     // = 1 : on récupère tout
255     // = 2 : on récupère uniquement les données variables (supposées comme telles)
256     void Lecture_base_info (ifstream& ent,const int cas);
257     // cas donne le niveau de sauvegarde
258     // = 1 : on sauvegarde tout
259     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
260     void Ecriture_base_info(ofstream& sort,const int cas);
261
262     protected :
263
264     // VARIABLES PROTÉGÉES :
265
266     TenseurBB * epsBB; // deformation finale
267     TenseurBB * depsBB; // vitesse finale de deformation
268     TenseurBB * deltaEpsBB; // variation de deformation entre t et t + delta t
269     TenseurHH * sigHH; // contrainte finale
270     TenseurHH * sigHH_t; // contrainte en début d'incrément
271     double module_compressibilite, module_cisaillement;
272     double volume_pti; // volume élémentaire au pti
273     // tableau relatif aux différentes grandeurs de type def scalaires équivalentes
274     // def_equi(1) = deformation cumulée = somme des sqrt(2./3. * (delta_eps_barre_BH &&
275     // delta_eps_barre_BH) );
276     // def_equi(2) = deformation duale de la contrainte de mises = sqrt(2./3. * (eps_barre_BH &&

```

```

    eps_barre_BH) ;
275 // def_equi(3) = niveau maxi atteint par def_equi(2)
276 // def_equi(4) = delta def cumulée = sqrt(2./3. * (delta_eps_barre_BH && delta_eps_barre_BH));
277 Tableau <double> def_equi_t, def_equi;
278
279 // idem coté contrainte
280 // (1) = contrainte_mises, (2) = contrainte_tresca
281 Tableau <double> sig_equi_t, sig_equi;
282
283 // positions du point d'intégration
284 Coordonnee M0,Mt,Mtdt;
285 // repérage dans éléments, maillage, nb pti
286 int mail,nele,npti;
287
288 //peut contenir éventuellement un tableau de base rattachée
289
290 // Tableau < BaseB_0_t_tdt>* tab_baseB_0_t_tdt;
291
292
293 // ---- les invariants éventuels (cas où les pointeurs ne sont pas nuls)
294 // rappel sur la signification des différentes composantes des vecteurs "invariant":
295 // (3) ==> Det();
296 // (2) ==> II() = A:A;
297 // (1) ==> Trace();
298 Vecteur * epsInvar ; // invariants de déformations
299 Vecteur * depsInvar ; // invariants de vitesses de déformations
300 Vecteur * sigInvar ; // invariants des contraintes
301 // --- temps cpu
302 Temps_CPU_HZpp tpsMetricque; // tps cpu relatif à la métrique uniquement
303 Temps_CPU_HZpp tps_cpu_loi_comp; // temps cumulé relatif à la loi de comportement
304
305 };
306 /// @} // end of group
307
308 #endif

```

## 7.185 QuadAxiCCom.h

```

1 // FICHER : QuadAxiCCom.h
2 // CLASSE : QuadAxiCCom
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      10/11/01
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      La classe QuadAxiCCom : éléments quadrangle cubiques
41 *   complet, 16 noeuds, 16 pt d'integ
42 *
43 *
44 *   *****
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but

```

```

48 * ----- *
49 * ! ! ! ! *
50 * $ *
51 * ***** *
52 * MODIFICATIONS: *
53 * ! date ! auteur ! but ! *
54 * ----- *
55 * $ *
56 *****/
57
58 // -----classe pour un calcul de mecanique-----
59
60
61
62 #ifndef QUADAXICCOM_H
63 #define QUADAXICCOM_H
64
65 #include "ParaGlob.h"
66 #include "ElemMeca.h"
67 #include "Met_abstraite.h"
68 #include "GeomQuadrangle.h"
69 #include "Noeud.h"
70 #include "UtilLecture.h"
71 #include "Tenseur.h"
72 #include "NevezTenseur.h"
73 #include "Deformation.h"
74 #include "QuadAxiMemb.h"
75 #include "FrontQuadCC.h"
76 #include "FrontSegCub.h"
77
78 /// @addtogroup groupe_des_elements_finis
79 /// @{
80 ///
81
82
83 class QuadAxiCCom : public QuadAxiMemb
84 {
85
86     public :
87
88         // CONSTRUCTEURS :
89         // Constructeur par default
90         QuadAxiCCom ();
91
92         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
93         // d'identification
94         QuadAxiCCom (double epaiss,int num_mail=0,int num_id=-3);
95
96         // Constructeur fonction d'un numero de maillage et d'identification
97         QuadAxiCCom (int num_mail,int num_id);
98
99         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
100         // du tableau de connexite des noeuds
101         QuadAxiCCom (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
102
103         // Constructeur de copie
104         QuadAxiCCom (const QuadAxiCCom& quadra);
105
106
107         // DESTRUCTEUR :
108         ~QuadAxiCCom ();
109
110         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
111         // méthode virtuelle
112         Element* Nevez_copie() const { Element * el= new QuadAxiCCom(*this); return el;};
113
114         // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadAxiCCom
115         QuadAxiCCom& operator= (QuadAxiCCom& quadra);
116
117         // METHODES :
118         // 1) derivant des virtuelles pures
119
120         // affichage dans la sortie transmise, des variables duales "nom"
121         // aux differents points d'integration
122         // dans le cas ou nom est vide, affichage de "toute" les variables
123         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
124
125         // 2) derivant des virtuelles
126         // 3) methodes propres a l'element
127
128     protected :
129
130         // adressage des frontieres linéiques et surfacique
131         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
132         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
133         { return ((ElFrontiere*) (new FrontSegCub(tab,ddelem)));};
134         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)

```



```

135         { return ((ElFrontiere*) (new FrontQuadCC(tab,ddelem)));};
136
137 // VARIABLES PRIVEES :
138 // place memoire commune a tous les elements
139 static QuadAxiMemb::DonnComQuad * doCoQuadAxiCCom;
140 // idem mais pour les indicateurs qui servent pour l'initialisation
141 static QuadAxiMemb::UneFois uneFoisQCom;
142
143 class NombresConstruireQuadAxiCCom : public NombresConstruire
144 { public: NombresConstruireQuadAxiCCom();
145 };
146 static NombresConstruireQuadAxiCCom nombre_V; // les nombres propres à l'élément
147
148 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
149 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
150 class ConsQuadAxiCCom : public ConstrucElement
151 { public : ConsQuadAxiCCom ()
152 { NouvelleTypeElement nouv (QUAD_AXI,CUBIQUE,MECA_SOLIDE_DEFORMABLE,this);
153   if (ParaGlob::NiveauImpression() >= 4)
154     cout << "\n initialisation QuadAxiCCom" << endl;
155     Element::listTypeElement.push_back(nouv);
156   };
157   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
158   {Element * pt;
159     pt = new QuadAxiCCom (num_maill,num) ;
160     return pt;};
161   // ramene true si la construction de l'element est possible en fonction
162   // des variables globales actuelles: ex en fonction de la dimension
163   bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
164   };
165   static ConsQuadAxiCCom consQuadAxiCCom;
166 };
167 /// @} // end of group
168 #endif
169
170
171
172

```

## 7.186 QuadAxiCCom\_cm9pti.h

```

1 // FICHER : QuadAxiCCom_cm9pti.h
2 // CLASSE : QuadAxiCCom_cm9pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      10/11/01
34 *
35 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 *      BUT: La classe QuadAxiCCom_cm9pti : éléments quadrangle cubiques
41 *      complet, 16 noeuds, 9 pt d'integ + gestion hourglass
42 *
43 *
44 *      *****

```

```

45 *      VERIFICATION:                                     *
46 *      ! date !   auteur !           but                ! *
47 *      -----
48 *      !           !           !           !           ! *
49 *      !           !           !           !           ! *
50 *      !           !           !           !           ! *
51 *      !           !           !           !           ! *
52 *      !           !           !           !           ! *
53 *      ! date !   auteur !           but                ! *
54 *      -----
55 *      !           !           !           !           ! *
56 *      !           !           !           !           ! *
57 *      !           !           !           !           ! *
58 *      !           !           !           !           ! *
59 *      !           !           !           !           ! *
60 *      !           !           !           !           ! *
61 *      !           !           !           !           ! *
62 *      !           !           !           !           ! *
63 *      !           !           !           !           ! *
64 *      !           !           !           !           ! *
65 *      !           !           !           !           ! *
66 *      !           !           !           !           ! *
67 *      !           !           !           !           ! *
68 *      !           !           !           !           ! *
69 *      !           !           !           !           ! *
70 *      !           !           !           !           ! *
71 *      !           !           !           !           ! *
72 *      !           !           !           !           ! *
73 *      !           !           !           !           ! *
74 *      !           !           !           !           ! *
75 *      !           !           !           !           ! *
76 *      !           !           !           !           ! *
77 *      !           !           !           !           ! *
78 *      !           !           !           !           ! *
79 *      !           !           !           !           ! *
80 *      !           !           !           !           ! *
81 *      !           !           !           !           ! *
82 *      !           !           !           !           ! *
83 *      !           !           !           !           ! *
84 *      !           !           !           !           ! *
85 *      !           !           !           !           ! *
86 *      !           !           !           !           ! *
87 *      !           !           !           !           ! *
88 *      !           !           !           !           ! *
89 *      !           !           !           !           ! *
90 *      !           !           !           !           ! *
91 *      !           !           !           !           ! *
92 *      !           !           !           !           ! *
93 *      !           !           !           !           ! *
94 *      !           !           !           !           ! *
95 *      !           !           !           !           ! *
96 *      !           !           !           !           ! *
97 *      !           !           !           !           ! *
98 *      !           !           !           !           ! *
99 *      !           !           !           !           ! *
100 *      !           !           !           !           ! *
101 *      !           !           !           !           ! *
102 *      !           !           !           !           ! *
103 *      !           !           !           !           ! *
104 *      !           !           !           !           ! *
105 *      !           !           !           !           ! *
106 *      !           !           !           !           ! *
107 *      !           !           !           !           ! *
108 *      !           !           !           !           ! *
109 *      !           !           !           !           ! *
110 *      !           !           !           !           ! *
111 *      !           !           !           !           ! *
112 *      !           !           !           !           ! *
113 *      !           !           !           !           ! *
114 *      !           !           !           !           ! *
115 *      !           !           !           !           ! *
116 *      !           !           !           !           ! *
117 *      !           !           !           !           ! *
118 *      !           !           !           !           ! *
119 *      !           !           !           !           ! *
120 *      !           !           !           !           ! *
121 *      !           !           !           !           ! *
122 *      !           !           !           !           ! *
123 *      !           !           !           !           ! *
124 *      !           !           !           !           ! *
125 *      !           !           !           !           ! *
126 *      !           !           !           !           ! *
127 *      !           !           !           !           ! *
128 *      !           !           !           !           ! *
129 *      !           !           !           !           ! *
130 *      !           !           !           !           ! *
131 *      !           !           !           !           ! *

```

```

132     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
133     { return ((ElFrontiere*) (new FrontSegCub(tab,ddelem)));};
134     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135     { return ((ElFrontiere*) (new FrontQuadCC(tab,ddelem)));};
136
137     // VARIABLES PRIVEES :
138     // place memoire commune a tous les elements
139     static QuadAxiMemb::DonnComQuad * doCoQuadAxiCCom_cm9pti;
140     // idem mais pour les indicateurs qui servent pour l'initialisation
141     static QuadAxiMemb::UneFois uneFoisQCom;
142
143     class NombresConstruireQuadAxiCCom_cm9pti : public NombresConstruire
144     { public: NombresConstruireQuadAxiCCom_cm9pti();
145     };
146     static NombresConstruireQuadAxiCCom_cm9pti nombre_V; // les nombres propres à l'élément
147
148     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
149     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
150     class ConsQuadAxiCCom_cm9pti : public ConstrucElement
151     { public : ConsQuadAxiCCom_cm9pti ()
152     { NouvelleTypeElement nouv (QUAD_AXI,CUBIQUE,MECA_SOLIDE_DEFORMABLE,this,"_cm9pti");
153     if (ParaGlob::NiveauImpression() >= 4)
154     cout << "\n initialisation QuadAxiCCom_cm9pti" << endl;
155     Element::listTypeElement.push_back(nouv);
156     };
157     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
158     {Element * pt;
159     pt = new QuadAxiCCom_cm9pti (num_maill,num) ;
160     return pt;};
161     // ramene true si la construction de l'element est possible en fonction
162     // des variables globales actuelles: ex en fonction de la dimension
163     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
164     };
165     static ConsQuadAxiCCom_cm9pti consQuadAxiCCom_cm9pti;
166 };
167 /// @} // end of group
168 #endif
169
170
171
172

```

## 7.187 QuadAxiL1.h

```

1 // FICHER : QuadAxiL1.h
2 // CLASSE : QuadAxiL1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      15/01/2006
34 *
35 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 *      BUT:      La classe QuadAxiL1 permet de declarer des éléments
41 *               quadrangulaires axisymétriques. La dimension de l'espace

```

```

42 *          est 3 avec z qui est hors service.          *
43 *          Les éléments finis sont ainsi de dimensions 3. *
44 *          L'interpolation est ici linéaire avec 4pti.   *
45 *                                                     $ *
46 *          ***** *
47 *          VERIFICATION: *
48 * *
49 *          ! date ! auteur ! but ! *
50 *          ----- *
51 *          ! ! ! ! *
52 *          $ *
53 *          ***** *
54 *          MODIFICATIONS: *
55 *          ! date ! auteur ! but ! *
56 *          ----- *
57 *          $ *
58 *          *****/
59
60 // -----classe pour un calcul de mecanique-----
61
62
63
64 #ifndef QUAD_AXI_L1_H
65 #define QUAD_AXI_L1_H
66
67 #include "ParaGlob.h"
68 #include "ElemMeca.h"
69 #include "Met_abstraite.h"
70 #include "GeomQuadrangle.h"
71 #include "Noeud.h"
72 #include "UtilLecture.h"
73 #include "Tenseur.h"
74 #include "NevezTenseur.h"
75 #include "Deformation.h"
76 #include "QuadAxiMemb.h"
77 #include "ElFrontiere.h"
78 #include "FrontSegLine.h"
79 #include "FrontQuadLine.h"
80
81 /// @addtogroup groupe_des_elements_finis
82 /// @{
83 ///
84
85
86 class QuadAxiL1 : public QuadAxiMemb
87 {
88
89     public :
90
91         // CONSTRUCTEURS :
92         // Constructeur par défaut
93         QuadAxiL1 ();
94
95         // Constructeur fonction d'un numero de maillage et d'identification
96         QuadAxiL1 (int num_mail,int num_id);
97
98         // Constructeur fonction d'un numero de maillage et d'identification,
99         // du tableau de connexite des noeuds
100        QuadAxiL1 (int num_mail,int num_id,const Tableau<Noeud *>& tab);
101
102        // Constructeur de copie
103        QuadAxiL1 (const QuadAxiL1& quad);
104
105
106        // DESTRUCTEUR :
107        ~QuadAxiL1 ();
108
109        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
110        // méthode virtuelle
111        Element* Nevez_copie() const { Element * el= new QuadAxiL1(*this); return el;};
112
113        // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadAxiL1
114        QuadAxiL1& operator= (QuadAxiL1& quad);
115
116        // METHODES :
117        // 1) derivant des virtuelles pures
118
119        // affichage dans la sortie transmise, des variables duales "nom"
120        // aux differents points d'integration
121        // dans le cas ou nom est vide, affichage de "toute" les variables
122        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
123
124        // 2) derivant des virtuelles
125        // 3) methodes propres a l'element
126
127     protected :
128

```

```

129 // adressage des frontières linéiques et surfacique
130 // définit dans les classes dérivées, et utilisées pour la construction des frontières
131 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
133 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
134 { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
135
136 // VARIABLES PRIVEES :
137 // place memoire commune a tous les elements QuadAxiL1
138 static QuadAxiMemb::DonnComQuad * doCoMembL1;
139 // idem mais pour les indicateurs qui servent pour l'initialisation
140 static QuadAxiMemb::UneFois uneFoisL1;
141
142 class NombresConstruireQuadAxiL1 : public NombresConstruire
143 { public: NombresConstruireQuadAxiL1();
144 };
145 static NombresConstruireQuadAxiL1 nombre_V; // les nombres propres à l'élément
146
147 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
148 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
149 class ConsQuadAxiL1 : public ConstrucElement
150 { public : ConsQuadAxiL1 ()
151 { NouvelleTypeElement nouv (QUAD_AXI, LINEAIRE, MECA_SOLIDE_DEFORMABLE, this);
152 if (ParaGlob::NiveauImpression() >= 4)
153 cout << "\n initialisation QuadAxiL1" << endl;
154 Element::listTypeElement.push_back(nouv);
155 };
156 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
157 {Element * pt;
158 pt = new QuadAxiL1 (num_maill,num) ;
159 return pt;};
160 // ramene true si la construction de l'element est possible en fonction
161 // des variables globales actuelles: ex en fonction de la dimension
162 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
163 };
164 static ConsQuadAxiL1 consQuadAxiL1;
165 };
166 /// @} // end of group
167 #endif
168
169
170
171

```

## 7.188 QuadAxiL1\_cm1pti.h

```

1 // FICHER : QuadAxiL1_cm1pti.h
2 // CLASSE : QuadAxiL1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      03/05/2011
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 * *****/

```

```

40 *      BUT: La classe QuadAxiLl_cmlpti permet de declarer des éléments *
41 * linéaire quadrangulaires axisymétriques. La dimension de l'espace *
42 * est 3 avec z qui est hors service. *
43 * Les éléments finis sont ainsi de dimensions 3. *
44 * L'interpolation est ici linéaire, lpti, avec gestion. *
45 * d'hourglass *
46 * * *
47 * ***** *
48 * VERIFICATION: *
49 * * *
50 * ! date ! auteur ! but ! *
51 * ----- *
52 * ! ! ! ! *
53 * * *
54 * ***** *
55 * MODIFICATIONS: *
56 * ! date ! auteur ! but ! *
57 * ----- *
58 * * *
59 *****/
60
61 // -----classe pour un calcul de mecanique-----
62
63
64
65 #ifndef QUAD_AXI_LL_CMLPTI_H
66 #define QUAD_AXI_LL_CMLPTI_H
67
68 #include "ParaGlob.h"
69 #include "ElemMeca.h"
70 #include "Met_abstraite.h"
71 #include "GeomQuadrangle.h"
72 #include "Noeud.h"
73 #include "UtilLecture.h"
74 #include "Tenseur.h"
75 #include "NevezTenseur.h"
76 #include "Deformation.h"
77 #include "QuadAxiMemb.h"
78 #include "ElFrontiere.h"
79 #include "FrontSegLine.h"
80 #include "FrontQuadLine.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class QuadAxiLl_cmlpti : public QuadAxiMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93         // Constructeur par défaut
94         QuadAxiLl_cmlpti ();
95
96         // Constructeur fonction d'un numero de maillage et d'identification
97         QuadAxiLl_cmlpti (int num_mail,int num_id);
98
99         // Constructeur fonction d'un numero de maillage et d'identification,
100         // du tableau de connexite des noeuds
101         QuadAxiLl_cmlpti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
102
103         // Constructeur de copie
104         QuadAxiLl_cmlpti (const QuadAxiLl_cmlpti& quad);
105
106
107         // DESTRUCTEUR :
108         ~QuadAxiLl_cmlpti ();
109
110         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
111         // méthode virtuelle
112         Element* Nevez_copie() const { Element * el= new QuadAxiLl_cmlpti(*this); return el;};
113
114         // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadAxiLl_cmlpti
115         QuadAxiLl_cmlpti& operator= (QuadAxiLl_cmlpti& quad);
116
117         // METHODES :
118         // 1) derivant des virtuelles pures
119
120         // affichage dans la sortie transmise, des variables duales "nom"
121         // aux differents points d'integration
122         // dans le cas ou nom est vide, affichage de "toute" les variables
123         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
124
125         // 2) derivant des virtuelles
126         // 3) methodes propres a l'element

```

```

127
128     protected :
129
130         // adressage des frontières linéiques et surfacique
131         // définit dans les classes dérivées, et utilisées pour la construction des frontières
132         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
133         { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
134         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135         { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
136
137         // VARIABLES PRIVEES :
138         // place memoire commune a tous les elements QuadAxiLl_cmlpti
139         static QuadAxiMemb::DonnComQuad * doCoMembLl;
140         // idem mais pour les indicateurs qui servent pour l'initialisation
141         static QuadAxiMemb::UneFois uneFoisLl;
142
143         class NombresConstruireQuadAxiLl_cmlpti : public NombresConstruire
144         { public: NombresConstruireQuadAxiLl_cmlpti();
145           };
146         static NombresConstruireQuadAxiLl_cmlpti nombre_V; // les nombres propres à l'élément
147
148         // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
149         //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
150         class ConsQuadAxiLl_cmlpti : public ConstrucElement
151         { public : ConsQuadAxiLl_cmlpti ()
152           { NouvelleTypeElement nouv(QUAD_AXI,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this,"_cmlpti");
153             if (ParaGlob::NiveauImpression() >= 4)
154               cout << "\n initialisation QuadAxiLl_cmlpti" << endl;
155             Element::listTypeElement.push_back(nouv);
156           };
157           Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
158           {Element * pt;
159             pt = new QuadAxiLl_cmlpti (num_maill,num) ;
160             return pt;};
161           // ramene true si la construction de l'element est possible en fonction
162           // des variables globales actuelles: ex en fonction de la dimension
163           bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
164         };
165         static ConsQuadAxiLl_cmlpti consQuadAxiLl_cmlpti;
166 };
167 /// @} // end of group
168 #endif
169
170
171
172

```

## 7.189 QuadAxiMemb.h

```

1 // FICHER : QuadAxiMemb.h
2 // CLASSE : QuadAxiMemb
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      15/01/2006
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *

```

```

37 *   PROJET:      Herezh++
38 *
39 *
40 *   BUT:   La classe QuadAxiMemb permet de declarer des éléments
41 *         quadrangulaires axisymétriques. La dimension de l'espace
42 *         est 3 avec z qui est hors service.
43 *         Les éléments finis sont ainsi de dimensions 3.
44 *         L'interpolation le nombre de point d'integration sont
45 *         definit dans les classes derivees.
46 *         La classe est virtuelle.
47 *
48 *   *****
49 *
50 *   VERIFICATION:
51 *   ! date !   auteur !       but
52 *   -----
53 *   !       !       !
54 *   $
55 *   *****
56 *   MODIFICATIONS:
57 *   ! date !   auteur !       but
58 *   -----
59 *   $
60 *****/
61
62 #ifndef QUAD_AXI_MEMB_H
63 #define QUAD_AXI_MEMB_H
64
65 #include "ParaGlob.h"
66 #include "ElemMeca.h"
67 #include "MetAxisymetrique3D.h"
68 #include "GeomQuadrangle.h"
69 #include "GeomSeg.h"
70 #include "Noeud.h"
71 #include "UtilLecture.h"
72 #include "Tenseur.h"
73 #include "NevezTenseur.h"
74 #include "Deformation.h"
75
76 /// @addtogroup groupe_des_elements_finis
77 /// @{
78 ///
79
80
81 class QuadAxiMemb : public ElemMeca
82 {
83
84     public :
85
86         // CONSTRUCTEURS :
87         // Constructeur par default
88         QuadAxiMemb ();
89
90         // Constructeur fonction d'un numero
91         // d'identification , d'identificateur d'interpolation et de geometrie
92         // et éventuellement un string d'information annexe
93         QuadAxiMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string
info="");
94
95         // Constructeur fonction d'un numero de maillage et d'identification,
96         // du tableau de connexite des noeuds, d'identificateur d'interpolation et de geometrie
97         // et éventuellement un string d'information annexe
98         QuadAxiMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,
99                     const Tableau<Noeud *>& tab,string info="") ;
100
101         // Constructeur de copie
102         QuadAxiMemb (const QuadAxiMemb& quad);
103
104
105         // DESTRUCTEUR :
106         ~QuadAxiMemb ();
107
108
109         // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadAxiMemb
110         QuadAxiMemb& operator= (const QuadAxiMemb& quad);
111
112         // METHODES :
113 // 1) derivant des virtuelles pures
114
115         // Lecture des donnees de la classe sur fichier
116         void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
117
118         // affichage d'info en fonction de ordre
119         // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
120         void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> *
tabMaillageNoeud)

```



```

121         { return Element::Info_com_El (nombre->nbne,entreePrinc,ordre,tabMaillageNoeud);};
122
123         // ramene l'element geometrique
124         ElemGeomC0& ElementGeometrique() const { return unefois->doCoMemb->quad};
125         // ramene l'element geometrique en constant
126         const ElemGeomC0& ElementGeometrique_const() const { return unefois->doCoMemb->quad};
127
128         // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
129         // associé
130         // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
131         // 1) cas où l'on utilise la place passée en argument
132         Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
133         // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
134         void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
135
136         // inactive les ddl du problème primaire de mécanique
137         inline void Inactive_ddl_primaire()
138         {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};
139         // active les ddl du problème primaire de mécanique
140         inline void Active_ddl_primaire()
141         {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl);};
142         // ajout des ddl de contraintes pour les noeuds de l'élément
143         inline void Plus_ddl_Sigma()
144         {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
145         // inactive les ddl du problème de recherche d'erreur : les contraintes
146         inline void Inactive_ddl_Sigma()
147         {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
148         // active les ddl du problème de recherche d'erreur : les contraintes
149         inline void Active_ddl_Sigma()
150         {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
151         // active le premier ddl du problème de recherche d'erreur : SIGMA11
152         inline void Active_premier_ddl_Sigma()
153         {ElemMeca::Act_premier_ddl_Sigma();};
154
155         // lecture de données diverses sur le flot d'entrée
156         void LectureContraintes(UtilLecture * entreePrinc)
157         { if (unefois->CalResPrem_t == 1)
158           ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
159         else
160           { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
161             unefois->CalResPrem_t = 1;
162           }
163         };
164
165         // retour des contraintes en absolu retour true si elle existe sinon false
166         bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
167         { if (unefois->CalResPrem_t == 1)
168           ElemMeca::ContraintesEnAbsolues (false,lesPtMecaInt.TabSigHH_t(),tabSig);
169         else
170           { ElemMeca::ContraintesEnAbsolues (true,lesPtMecaInt.TabSigHH_t(),tabSig);
171             unefois->CalResPrem_t = 1;
172           }
173         return true; }
174
175         // Libere la place occupee par le residu et eventuellement la raideur
176         // par l'appel de Libere de la classe mere et libere les differents tenseurs
177         // intermediaires cree pour le calcul et les grandeurs pointee
178         // de la raideur et du residu
179         void Libere ();
180
181         // acquisition d'une loi de comportement
182         void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
183
184         // test si l'element est complet
185         // = 1 tout est ok, =0 element incomplet
186         int TestComplet();
187
188         // procedure permettant de completer l'element apres
189         // sa creation avec les donnees du bloc transmis
190         // peut etre appeler plusieurs fois
191         Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
192         // Compléter pour la mise en place de la gestion de l'hourglass
193         Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
194
195         // ramene vrai si la surface numéro ns existe pour l'élément
196         // dans le cas des éléments quadrangulaires il ne peut y avoir qu'une
197         // seule surface
198         bool SurfExiste(int ns) const
199         { if ((ns==1)&&(ParaGlob::Dimension()) >= 2) return true; else return false;};
200
201         // ramene vrai si l'arête numéro na existe pour l'élément
202         bool AreteExiste(int na) const
203         {if ((na <= 3) || (na>= 1)) return true; else return false;};
204
205         // retourne les tableaux de ddl associés aux noeuds, gere par l'element

```

```

206 // ce tableau et specifique a l'element
207 const DdlElement & TableauDdl() const
208 { return unefois->doCoMemb->tab_ddl; };
209
210 // Calcul du residu local et de la raideur locale,
211 // pour le schema implicite
212 Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
213
214 // Calcul du residu local a t
215 // pour le schema explicit par exemple
216 Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
217 { return QuadAxiMemb::CalculResidu(false,pa);};
218
219 // Calcul du residu local a tdt
220 // pour le schema explicit par exemple
221 Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
222 { return QuadAxiMemb::CalculResidu(true,pa);};
223
224 // Calcul de la matrice masse pour l'élément
225 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
226
227 // ----- calcul dynamique -----
228 // calcul de la longueur d'arrête de l'élément minimal
229 // divisé par la célérité la plus rapide dans le matériau
230 double Long_arrete_mini_sur_c(Enum_dure temps)
231 { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
232
233 //----- calcul d'erreur, remontée des contraintes -----
234 // 1) calcul du résidu et de la matrice de raideur pour le calcul d'erreur
235 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
236 // 2) remontée aux erreurs aux noeuds
237 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
238
239 // ----- affichage ou récupération d'informations -----
240 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
241 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
242 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
243 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
244 // temps: dit si c'est à 0 ou t ou tdt
245 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
246 { return PtLePlusPres(temps,enu,M);};
247
248 // recuperation des coordonnées du point de numéro d'ordre iteg pour
249 // la grandeur enu
250 // temps: dit si c'est à 0 ou t ou tdt
251 // si erreur retourne erreur à true
252 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
253 { return CoordPtInt(temps,enu,iteg,erreur);};
254
255 // récupération des valeurs au numéro d'ordre = iteg pour
256 // les grandeur enu
257 Tableau<double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const
List_io<Ddl_enum_etendu>& enu,int iteg);
258 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
259 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
260 // de conteneurs quelconque associée
261 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int
iteg);
262
263 //===== lecture écriture dans base info =====
264
265 // cas donne le niveau de la récupération
266 // = 1 : on récupère tout
267 // = 2 : on récupère uniquement les données variables (supposées comme telles)
268 void Lecture_base_info
269 (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
270 // cas donne le niveau de sauvegarde
271 // = 1 : on sauvegarde tout
272 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
273 void Ecriture_base_info(ofstream& sort,const int cas) ;
274
275 // 2) derivant des virtuelles
276
277 // retourne un tableau de ddl element, correspondant à la
278 // composante de sigma -> SIG11, pour chaque noeud qui contiend
279 // des ddl de contrainte
280 // -> utilisé pour l'assemblage de la raideur d'erreur
281 DdlElement& Tableau_de_Sigl() const
282 {return unefois->doCoMemb->tab_Err1Sig11;};
283
284 // actualisation des ddl et des grandeurs actives de t+dt vers t
285 void TdtversT();
286 // actualisation des ddl et des grandeurs actives de t vers tdt
287 void TversTdt();
288
289 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
290 // qu'une fois la remontée aux contraintes effectuées sinon aucune

```

```

291 // action. En retour la valeur de l'erreur sur l'élément
292 // type indique le type de calcul d'erreur :
293 void ErreurElement(int type,double& errElemRelative
294                 ,double& numerateur, double& denominateur);
295
296 // calcul des seconds membres suivant les chargements
297 // cas d'un chargement volumique,
298 // force indique la force volumique appliquée
299 // retourne le second membre résultant
300 // ici on l'épaisseur de l'élément pour constituer le volume
301 // -> explicite à t
302 Vecteur SM_charge_volumique_E_t
303 (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_)
304 { return QuadAxiMemb::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_); } ;
305 // -> explicite à tdt
306 Vecteur SM_charge_volumique_E_tdt
307 (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_)
308 { return QuadAxiMemb::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_); } ;
309 // -> implicite,
310 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
311 // retourne le second membre et la matrice de raideur correspondant
312 ResRaid SMR_charge_volumique_I
313 (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_) ;
314
315 // cas d'un chargement surfacique, sur les frontières des éléments
316 // force indique la force surfacique appliquée
317 // numface indique le numéro de la face chargée
318 // retourne le second membre résultant
319 // -> version explicite à t
320 Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct
321                               ,int numFace,const ParaAlgoControle & pa)
322 { return QuadAxiMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa); } ;
323 // -> version explicite à tdt
324 Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
325 { return QuadAxiMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa); } ;
326 // -> implicite,
327 // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
328 // retourne le second membre et la matrice de raideur correspondant
329 ResRaid SMR_charge_surfacique_I
330 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const ParaAlgoControle & pa)
;
331
332 // cas d'un chargement lineique, sur les aretes frontières des éléments
333 // force indique la force lineique appliquée
334 // numarete indique le numéro de l'arete chargée
335 // retourne le second membre résultant
336 // -> explicite à t
337 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
338 { return QuadAxiMemb::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); } ;
339 // -> explicite à tdt
340 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
341 { return QuadAxiMemb::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); } ;
342 // -> implicite,
343 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
344 // retourne le second membre et la matrice de raideur correspondant
345 ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
346
347 // cas d'un chargement lineique suiveuse, sur les aretes frontières des éléments 2D (uniquement)
348 // force indique la force lineique appliquée
349 // numarete indique le numéro de l'arete chargée
350 // retourne le second membre résultant
351 // -> explicite à t
352 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
353 { return QuadAxiMemb::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa); } ;
354 // -> explicite à tdt
355 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa)
356 { return QuadAxiMemb::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa); } ;
357 // -> implicite,
358 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
359 // retourne le second membre et la matrice de raideur correspondant
360 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int
numArete,const ParaAlgoControle & pa) ;
361
362 // cas d'un chargement de type pression, sur les frontières des éléments
363 // pression indique la pression appliquée
364 // numface indique le numéro de la face chargée
365 // retourne le second membre résultant
366 // -> explicite à t

```

```

367     Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
368     { return QuadAxiMemb::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa);};
369     // -> explicite à tdt
370     Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
371     { return QuadAxiMemb::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa);};
372     // -> implicite,
373     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
374     // retourne le second membre et la matrice de raideur correspondant
375     ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
376
377     // cas d'un chargement surfacique hydrostatique,
378     // poidvol: indique le poids volumique du liquide
379     // M_liquide : un point de la surface libre
380     // dir_normal_liquide : direction normale à la surface libre
381     // retourne le second membre résultant
382     // -> explicite à t
383     Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
384     ,int numFace,const Coordonnee& M_liquide
385     ,const ParaAlgoControle & pa
386     ,bool sans_limitation)
387     { return
QuadAxiMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation);};
388     // -> explicite à tdt
389     Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
390     ,int numFace,const Coordonnee& M_liquide
391     ,const ParaAlgoControle & pa
392     ,bool sans_limitation)
393     { return
QuadAxiMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation);};
394     // -> implicite,
395     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
396     // retourne le second membre et la matrice de raideur correspondant
397     ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
398     ,int numFace,const Coordonnee& M_liquide
399     ,const ParaAlgoControle & pa
400     ,bool sans_limitation) ;
401
402     // cas d'un chargement surfacique hydro-dynamique,
403     // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
404     // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc
unitaire)
405     // une suivant la direction normale à la vitesse de type portance
406     // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
407     // une suivant la vitesse tangente de type frottement visqueux
408     // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
409     // coef_mul: est un coefficient multiplicateur global (de tout)
410     // retourne le second membre résultant
411     // -> explicite à t
412     Vecteur SM_charge_hydrodynamique_E_t( CourbeID* frot_fluid,const double& poidvol
413     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
414     , CourbeID* coef_aero_t
415     ,const ParaAlgoControle & pa)
416     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
417     // -> explicite à tdt
418     Vecteur SM_charge_hydrodynamique_E_tdt( CourbeID* frot_fluid,const double& poidvol
419     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
420     , CourbeID* coef_aero_t
421     ,const ParaAlgoControle & pa)
422     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};
423     // -> implicite,
424     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
425     // retourne le second membre et la matrice de raideur correspondant
426     ResRaid SMR_charge_hydrodynamique_I( CourbeID* frot_fluid,const double& poidvol
427     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
428     , CourbeID* coef_aero_t,const ParaAlgoControle &
pa) ;
429
430     // ===== définition et/ou construction des frontières =====
431
432     // Calcul des frontieres de l'element
433     // creation des elements frontieres et retour du tableau de ces elements
434     // la création n'a lieu qu'au premier appel
435     // ou lorsque l'on force le paramètre force a true
436     // dans ce dernier cas seul les frontière effacées sont recréée
437     Tableau <ElFrontiere*> const & Frontiere(bool force = false);
438
439     // ramène la frontière point
440     // éventuellement création des frontieres points de l'element et stockage dans l'element
441     // si c'est la première fois sinon il y a seulement retour de l'elements

```

```

442 // a moins que le paramètre force est mis a true
443 // dans ce dernier cas la frontière effacée est recréée
444 // num indique le numéro du point à créer (numérotation EF)
445 // ElFrontiere* const Frontiere_points(int num,bool force = false);
446
447 // ramène la frontière linéique
448 // éventuellement création des frontieres linéique de l'element et stockage dans l'element
449 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
450 // a moins que le paramètre force est mis a true
451 // dans ce dernier cas la frontière effacée est recréée
452 // num indique le numéro de l'arête à créer (numérotation EF)
453 // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
454
455 // ramène la frontière surfacique
456 // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
457 // si c'est la première fois sinon il y a seulement retour de l'elements
458 // a moins que le paramètre force est mis a true
459 // dans ce dernier cas la frontière effacée est recréée
460 // num indique le numéro de la surface à créer (numérotation EF)
461 // ici normalement uniquement 1 possible
462 // ElFrontiere* const Frontiere_surfacique(int num,bool force = false);
463
464
465 // 3) methodes propres a l'element
466
467 // ajout du tableau specific de ddl des noeuds
468 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
469 // des noeuds constituant l'element
470 void ConstTabDdl();
471
472 // ----- definition de la classe conteneur de donnees communes -----
473 class DonnComQuad
474 { public :
475     DonnComQuad (GeomQuadrangle& tri,DdlElement& tab,DdlElement& tabErr,DdlElement&
476     tab_Err1Sig,
477     MetAxisymetrique3D& met_gene, Tableau <Vecteur * > & resEr,Mat_pleine&
478     raidEr,
479     GeomQuadrangle& quadEr,GeomQuadrangle& quadS,
480     GeomSeg& segmS,Vecteur& residu_int,Mat_pleine& raideur_int,
481     Tableau <Vecteur* > & residus_extN,Tableau <Mat_pleine* >& raideurs_extN,
482     Tableau <Vecteur* > & residus_extA,Tableau <Mat_pleine* >& raideurs_extA,
483     Tableau <Vecteur* >& residus_extS,Tableau <Mat_pleine* >& raideurs_extS,
484     Mat_pleine& mat_masse,GeomQuadrangle& quadMas,int nbi,GeomQuadrangle&
485     quadHourg
486     ) ;
487     DonnComQuad(DonnComQuad& a) ;
488     ~DonnComQuad();
489 // variables
490 GeomQuadrangle quad; // contient les fonctions d'interpolation et
491 // les derivees pour le calcul d'équilibre en déplacement
492 DdlElement tab_ddl; // tableau des degres
493 //de liberte des noeuds de l'element commun a tous les
494 // elements. Il s'agit des ddl primaires pour le problème de mécanique
495 MetAxisymetrique3D met_QuadAxiMemb;
496 Mat_pleine matGeom ; // matrice géométrique
497 Mat_pleine matInit ; // matrice initiale
498 Tableau <TenseurBB * > d_epsBB; // place pour la variation des def
499 Tableau <TenseurHH * > d_sigHH; // place pour la variation des contraintes
500 Tableau <Tableau2 <TenseurBB * > > d2_epsBB; // variation seconde des déformations
501 // ---- concernant les frontières et particulièrement le calcul de second membre
502 GeomQuadrangle quadS; // contient les fonctions d'interpolation et les derivees
503 GeomSeg segS; // " " "
504 //----- calcul d'erreur -----
505 DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
506 // d'erreur : contraintes
507 DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud, servant pour le
508 calcul
509 // d'erreur : contraintes, en fait pour l'assemblage
510 Tableau <Vecteur* > resErr; // residu pour le calcul d'erreur
511 Mat_pleine raidErr; // raideur pour le calcul d'erreur
512 GeomQuadrangle quadEr; // contient les fonctions d'interpolation et
513 // les derivees pour le calcul du hessien dans
514 //la résolution de la fonctionnelle d'erreur
515 // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
516 -----
517 // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
518 Vecteur residu_interne;
519 Mat_pleine raideur_interne;
520 Tableau <Vecteur* > residus_externen; // pour les noeuds
521 Tableau <Mat_pleine* > raideurs_externen; // pour les noeuds
522 Tableau <Vecteur* > residus_externea; // pour les aretes
523 Tableau <Mat_pleine* > raideurs_externea; // pour les aretes
524 Tableau <Vecteur* > residus_externes; // pour les surfaces
525 Tableau <Mat_pleine* > raideurs_externes; // pour les surfaces
526 // ----- données concernant la dynamique -----
527 Mat_pleine matrice_masse;

```

```

524         GeomQuadrangle quadMas; // contient les fonctions d'interpolation et ...
525             // pour les calculs relatifs à la masse
526         // ----- blocage éventuel d'hourglass
527         // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
Cal_explici_hourglass
528         GeomQuadrangle* quadraHourg; // contient les fonctions d'interpolation
529     };
530
531     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
532     // et un pointeur sur les données statiques communes
533     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
534     // classe est défini. Son allocation est effectuée dans les classes dérivées
535     class UneFois
536     { public :
537         UneFois () ; // constructeur par défaut
538         ~UneFois () ; // destructeur
539
540     // VARIABLES :
541     public :
542         QuadAxiMemb::DonnComQuad * doCoMemb;
543
544         // incicateurs permettant de dimensionner seulement au premier passage
545         // utilise dans "CalculResidu" et "Calcul_implicit"
546         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
547         int CalimpPrem;
548         int dualSortQuad; // pour la sortie des valeurs au pt d'integ
549         int CalSMlin_t; // pour les seconds membres concernant les arretes
550         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
551         int CalSMRlin; // pour les seconds membres concernant les arretes
552         int CalSMsurf_t; // pour les seconds membres concernant les surfaces
553         int CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
554         int CalSMRsurf; // pour les seconds membres concernant les surfaces
555         int CalSMvol_t; // pour les seconds membres concernant les volumes
556         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
557         int CalSMvol; // pour les seconds membres concernant les volumes
558         int CalDynamique; // pour le calcul de la matrice de masse
559         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
560         // ----- sauvegarde du nombre d'élément en cours -----
561         int nbelem_in_Prog;
562     };
563
564     // -----
565
566 protected :
567     // VARIABLES PROTÉGÉES :
568     UneFois * unefois; // pointeur défini dans la classe dérivée
569     // les données spécifiques sont groupées dans une structure pour sécuriser
570     // le passage de paramètre dans init par exemple
571     class Donnee_specif{ public :
572         Donnee_specif() {};
573         Donnee_specif(const Donnee_specif& ) {};
574         Donnee_specif & operator = ( const Donnee_specif & )
575         {return *this;};
576         // data // rien pour l'instant contrairement au cas des quadrangles 2D
577         };
578     Donnee_specif donnee_specif;
579
580     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
581     LesPtIntegMecaInterne lesPtMecaInt;
582
583     // type structuré et fonction virtuelle pour construire les éléments
584     // la fonction est défini dans le type dérivé
585     class NombresConstruire
586     { public:
587         NombresConstruire():nbne(0),nbneA(0),nbi(0)
588             ,nbiEr(0),nbiS(0),nbiA(0),nbiMas(0),nbiHour(0) {};
589         int nbne; // le nombre de noeud de l'élément
590         int nbneA ; // le nombre de noeud des aretes
591         int nbis; // le nombre de point d'intégration pour le calcul mécanique
592         int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
593         int nbiS; // le nombre de point d'intégration pour le calcul de second membre surfacique
594         int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
595         int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse
596         int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
597     };
598     NombresConstruire * nombre; // le pointeur défini dans la classe dérivée d'hexamemb
599
600     // METHODES
601     // =====>>> methodes appelees par les classes derivees <<<=====
602
603     // fonction d'initialisation servant dans les classes derivant
604     // au niveau du constructeur, si rien initialisation par défaut
605     QuadAxiMemb::DonnComQuad* Init (Donnee_specif donnee_specif = Donnee_specif()
606         ,bool sans_init_noeud = false);
607     // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
608     void Destruction();
609

```

```

610
611 // ==== »» methodes virtuelles derivant d'ElemMeca =====
612 // ramene la dimension des tenseurs contraintes et deformations de l'element
613 int Dim_sig_eps() const {return 3;};
614
615 //----- fonctions uniquement a usage interne -----
616 private :
617 // definition des donnees communes : CoQuad
618 QuadAxiMemb::DonnComQuad* Def_DonneeCommune();
619 // Calcul du residu local a t ou tdt en fonction du boolean
620 Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
621 // calcul des seconds membres suivant les chargements
622 // cas d'un chargement volumique,
623 // force indique la force volumique appliquee
624 // retourne le second membre resultant
625 // ici on l'epaisseur de l'element pour constituer le volume
626 // -> explicite a t ou tdt en fonction de la variable booleenne atdt
627 Vecteur SM_charge_volumique_E
628 (const Coordonnee& force,Fonction_nD* pt_fonct,bool atdt,const ParaAlgoControle & pa,bool
sur_volume_finale_);
629 // cas d'un chargement surfacique, sur les frontieres des elements
630 // force indique la force surfacique appliquee
631 // numface indique le numero de la face chargee
632 // retourne le second membre resultant
633 // -> explicite a t ou tdt en fonction de la variable booleenne atdt
634 Vecteur SM_charge_surfacique_E
635 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,bool atdt,const
ParaAlgoControle & pa);
636 // cas d'un chargement lineique, sur les aretes frontieres des elements
637 // force indique la force lineique appliquee
638 // numarete indique le numero de l'arete chargee
639 // retourne le second membre resultant
640 // -> explicite a t ou tdt en fonction de la variable booleenne atdt
641 Vecteur SM_charge_lineique_E
642 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
643 // cas d'un chargement lineique suiveuse, sur les aretes frontieres des elements 2D (uniquement)
644 // force indique la force lineique appliquee
645 // numarete indique le numero de l'arete chargee
646 // retourne le second membre resultant
647 // -> explicite a t
648 Vecteur SM_charge_lineique_Suiv_E
649 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
650 // cas d'un chargement de type pression, sur les frontieres des elements
651 // pression indique la pression appliquee
652 // numface indique le numero de la face chargee
653 // retourne le second membre resultant
654 // -> explicite a t ou tdt en fonction de la variable booleenne atdt
655 Vecteur SM_charge_pression_E
656 (double pression,Fonction_nD* pt_fonct,int numFace,bool atdt,const ParaAlgoControle
& pa);
657 // cas d'un chargement surfacique hydrostatique,
658 // poidvol: indique le poids volumique du liquide
659 // M_liquide : un point de la surface libre
660 // dir_normal_liquide : direction normale a la surface libre
661 // retourne le second membre resultant
662 // -> explicite a t
663 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
, int numFace,const Coordonnee& M_liquide,bool atdt
, const ParaAlgoControle & pa
, bool sans_limitation);
664 // cas d'un chargement surfacique hydro-dynamique,
665 // voir methode explicite plus haut, pour les arguments
666 // retourne le second membre resultant
667 // bool atdt : permet de specifier a t ou a t+dt
668 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
, CourbelD* coef_aero_n,int numFace,const double&
coef_mul
, CourbelD* coef_aero_t,bool atdt
,const ParaAlgoControle & pa) ;
675 };
676 /// @} // end of group
677 #endif
678
679
680
681

```

## 7.190 QuadAxiQ.h

```

1 // FICHER : QuadAxiQ.h
2 // CLASSE : QuadAxiQ
3
4 // This file is part of the Herezh++ application.

```

```

5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           15/01/2006
34 *
35 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:        Herezh++
38 *
39 * *****/
40 *   BUT:           Element quadrangulaire, quadratique 4 pt d'integ.
41 *                 8 noeuds -> quadratique incomplet.
42 *   *****/
43 *
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !           but
47 *   -----
48 *
49 *
50 *   *****/
51 *
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *
58 * *****/
59
60 // -----classe pour un calcul de mecanique-----
61
62
63
64 #ifndef QUAD_AXI_Q3_H
65 #define QUAD_AXI_Q3_H
66
67 #include "ParaGlob.h"
68 #include "ElemMeca.h"
69 #include "Met_abstraite.h"
70 #include "GeomQuadrangle.h"
71 #include "Noeud.h"
72 #include "UtilLecture.h"
73 #include "Tenseur.h"
74 #include "NevezTenseur.h"
75 #include "Deformation.h"
76 #include "QuadAxiMemb.h"
77 #include "FrontSegQuad.h"
78 #include "FrontQuadQuad.h"
79
80 /// @addtogroup groupe_des_elements_finis
81 /// @{
82 ///
83
84
85 class QuadAxiQ : public QuadAxiMemb
86 {
87
88     public :
89
90     // CONSTRUCTEURS :

```



```

91     // Constructeur par défaut
92     QuadAxiQ ();
93
94     // Constructeur fonction d'un numero de maillage et d'identification
95     QuadAxiQ (int num_maill,int num_id);
96
97     // Constructeur fonction d'un numero d'identification,
98     // du tableau de connexite des noeuds
99     QuadAxiQ (int num_maill,int num_id,const Tableau<Noeud *>& tab);
100
101     // Constructeur de copie
102     QuadAxiQ (const QuadAxiQ& quad);
103
104
105     // DESTRUCTEUR :
106     ~QuadAxiQ ();
107
108     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
109     // méthode virtuelle
110     Element* Nevez_copie() const { Element * el= new QuadAxiQ(*this); return el;};
111
112     // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadAxiQ
113     QuadAxiQ& operator= (QuadAxiQ& quad);
114
115     // METHODES :
116 // 1) derivant des virtuelles pures
117
118     // affichage dans la sortie transmise, des variables duales "nom"
119     // aux differents points d'integration
120     // dans le cas ou nom est vide, affichage de "toute" les variables
121     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
122
123 protected :
124
125     // adressage des frontières linéiques et surfacique
126     // définit dans les classes dérivées, et utilisées pour la construction des frontières
127     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
128     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
129     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
130     { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
131
132 // VARIABLES PRIVEES :
133 // place memoire commune a tous les elements QuadAxiQ
134 static QuadAxiMemb::DonnComQuad * doCoMembQ3;
135 // idem mais pour les indicateurs qui servent pour l'initialisation
136 static QuadAxiMemb::UneFois uneFoisQ3;
137
138 class NombresConstruireQuadAxiQ : public NombresConstruire
139 { public: NombresConstruireQuadAxiQ();
140 };
141 static NombresConstruireQuadAxiQ nombre_V; // les nombres propres à l'élément
142
143 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
144 //ajout de l'element dans la liste : listTypeElement, geree par la class Element
145 class ConsQuadAxiQ : public ConstrucElement
146 { public : ConsQuadAxiQ ()
147 { NouvelleTypeElement nouv (QUAD_AXI, QUADRATIQUE, MECA_SOLIDE_DEFORMABLE, this);
148   if (ParaGlob::NiveauImpression() >= 4)
149     cout << "\n initialisation QuadAxiQ" << endl;
150   Element::listTypeElement.push_back (nouv);
151   };
152   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
153   {Element * pt;
154     pt = new QuadAxiQ (num_maill,num) ;
155     return pt;};
156   // ramene true si la construction de l'element est possible en fonction
157   // des variables globales actuelles: ex en fonction de la dimension
158   bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
159   };
160   static ConsQuadAxiQ consQuadAxiQ;
161 };
162 /// @} // end of group
163 #endif
164
165
166
167

```

## 7.191 QuadAxiQComp.h

```

1 // FICHER : QuadAxiQComp.h
2 // CLASSE : QuadAxiQComp
3
4 // This file is part of the Herezh++ application.
5 //

```

```

6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31 //
32 /*****
33 *   DATE:      15/01/2006
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *   ****
40 *   BUT:      Element quadrangulaire, axi-symétrique,
41 *   quadratique 9 pt d'integ. 9 noeuds -> quadratique complet.
42 *
43 *   *****
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !           but
47 *   -----
48 *   !       !           !
49 *
50 *   *****
51 *   MODIFICATIONS:
52 *
53 *   ! date !   auteur !           but
54 *   -----
55 *
56 *
57 *
58 *
59 *****/
60
61 // -----classe pour un calcul de mecanique-----
62
63
64
65 #ifndef QUAD_AXI_Q_COMP_H
66 #define QUAD_AXI_Q_COMP_H
67
68 #include "ParaGlob.h"
69 #include "ElemMeca.h"
70 #include "Met_abstraite.h"
71 #include "GeomQuadrangle.h"
72 #include "Noeud.h"
73 #include "UtilLecture.h"
74 #include "Tenseur.h"
75 #include "NevezTenseur.h"
76 #include "Deformation.h"
77 #include "QuadAxiMemb.h"
78 #include "FrontSegQuad.h"
79 #include "FrontQuadQC.h"
80
81 /// @addtogroup groupe_des_elements_finis
82 /// @{
83 ///
84
85
86 class QuadAxiQComp : public QuadAxiMemb
87 {
88
89     public :
90
91         // CONSTRUCTEURS :
92         // Constructeur par default

```

```

93     QuadAxiQComp ();
94
95     // Constructeur fonction d'un numero de maillage et d'identification
96     QuadAxiQComp (int num_mail,int num_id);
97
98     // Constructeur fonction d'un numero d'identification,
99     // du tableau de connexite des noeuds
100     QuadAxiQComp (int num_mail,int num_id,const Tableau<Noeud *>& tab);
101
102     // Constructeur de copie
103     QuadAxiQComp (const QuadAxiQComp& quad);
104
105
106     // DESTRUCTEUR :
107     ~QuadAxiQComp ();
108
109     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
110     // méthode virtuelle
111     Element* Nevez_copie() const { Element * el= new QuadAxiQComp(*this); return el;};
112
113     // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadAxiQComp
114     QuadAxiQComp& operator= (QuadAxiQComp& quad);
115
116     // METHODES :
117     // 1) derivant des virtuelles pures
118
119     // affichage dans la sortie transmise, des variables duales "nom"
120     // aux differents points d'integration
121     // dans le cas ou nom est vide, affichage de "toute" les variables
122     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
123
124 protected :
125
126     // adressage des frontières linéiques et surfacique
127     // définit dans les classes dérivées, et utilisées pour la construction des frontières
128     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
129     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
130     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
131     { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
132
133     // VARIABLES PRIVEES :
134     // place memoire commune a tous les elements QuadAxiQComp
135     static QuadAxiMemb::DonnComQuad * doCoMembQ3;
136     // idem mais pour les indicateurs qui servent pour l'initialisation
137     static QuadAxiMemb::UneFois uneFoisQ3;
138
139     class NombresConstruireQuadAxiQComp : public NombresConstruire
140     { public: NombresConstruireQuadAxiQComp();
141       };
142     static NombresConstruireQuadAxiQComp nombre_V; // les nombres propres à l'élément
143
144     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
145     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
146     class ConsQuadAxiQComp : public ConstrucElement
147     { public : ConsQuadAxiQComp ()
148       { NouvelleTypeElement nouv (QUAD_AXI,QUADCOMPL,MECA_SOLIDE_DEFORMABLE,this);
149         if (ParaGlob::NiveauImpression() >= 4)
150           cout << "\n initialisation QuadAxiQComp" << endl;
151         Element::listTypeElement.push_back(nouv);
152       };
153       Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
154       {Element * pt;
155         pt = new QuadAxiQComp (num_maill,num) ;
156         return pt;};
157       // ramene true si la construction de l'element est possible en fonction
158       // des variables globales actuelles: ex en fonction de la dimension
159       bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
160     };
161     static ConsQuadAxiQComp consQuadAxiQComp;
162 };
163 /// @} // end of group
164 #endif
165
166
167
168

```

## 7.192 QuadAxiQComp\_cm4pti.h

```

1 // FICHER : QuadAxiQComp_cm4pti.h
2 // CLASSE : QuadAxiQComp_cm4pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field

```

```

7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           3/05/2011
34 *
35 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:         Herezh++
38 *
39 *****/
40 *   BUT:           Element quadrangulaire, axi-symétrique,
41 *   quadratique 4 pt d'integ. 9 noeuds -> quadratique complet.
42 *   gestion des modes d'hourglass.
43 *
44 *   *****
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   -----
49 *   !       !           !
50 *
51 *   *****
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *
58 *
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef QUAD_AXI_Q_COMP_CM4PTI_H
67 #define QUAD_AXI_Q_COMP_CM4PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "QuadAxiMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQC.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class QuadAxiQComp_cm4pti : public QuadAxiMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93         // Constructeur par default

```

```

94     QuadAxiQComp_cm4pti ();
95
96     // Constructeur fonction d'un numero de maillage et d'identification
97     QuadAxiQComp_cm4pti (int num_maill,int num_id);
98
99     // Constructeur fonction d'un numero d'identification,
100    // du tableau de connexite des noeuds
101    QuadAxiQComp_cm4pti (int num_maill,int num_id,const Tableau<Noeud *>& tab);
102
103    // Constructeur de copie
104    QuadAxiQComp_cm4pti (const QuadAxiQComp_cm4pti& quad);
105
106
107    // DESTRUCTEUR :
108    ~QuadAxiQComp_cm4pti ();
109
110    // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
111    // méthode virtuelle
112    Element* Nevez_copie() const { Element * el= new QuadAxiQComp_cm4pti(*this); return el;};
113
114    // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadAxiQComp_cm4pti
115    QuadAxiQComp_cm4pti& operator= (QuadAxiQComp_cm4pti& quad);
116
117    // METHODES :
118    // 1) derivant des virtuelles pures
119
120    // affichage dans la sortie transmise, des variables duales "nom"
121    // aux differents points d'integration
122    // dans le cas ou nom est vide, affichage de "toute" les variables
123    void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
124
125    protected :
126
127    // adressage des frontieres linéiques et surfacique
128    // définit dans les classes dérivées, et utilisées pour la construction des frontieres
129    ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
130    { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
131    ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132    { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
133
134    // VARIABLES PRIVEES :
135    // place memoire commune a tous les elements QuadAxiQComp_cm4pti
136    static QuadAxiMemb::DonnComQuad * doCoMembQ3;
137    // idem mais pour les indicateurs qui servent pour l'initialisation
138    static QuadAxiMemb::UneFois uneFoisQ3;
139
140    class NombresConstruireQuadAxiQComp_cm4pti : public NombresConstruire
141    { public: NombresConstruireQuadAxiQComp_cm4pti();
142      };
143    static NombresConstruireQuadAxiQComp_cm4pti nombre_V; // les nombres propres à l'élément
144
145    // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
146    //ajout de l'element dans la liste : listTypeElement, geree par la class Element
147    class ConsQuadAxiQComp_cm4pti : public ConstrucElement
148    { public : ConsQuadAxiQComp_cm4pti ()
149      { NouvelleTypeElement nouv (QUAD_AXI,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cm4pti");
150        if (ParaGlob::NiveauImpression() >= 4)
151          cout << "\n initialisation QuadAxiQComp_cm4pti" << endl;
152          Element::listTypeElement.push_back(nouv);
153      };
154      Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
155      {Element * pt;
156        pt = new QuadAxiQComp_cm4pti (num_maill,num) ;
157        return pt;};
158      // ramene true si la construction de l'element est possible en fonction
159      // des variables globales actuelles: ex en fonction de la dimension
160      bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
161    };
162    static ConsQuadAxiQComp_cm4pti consQuadAxiQComp_cm4pti;
163 };
164 /// @} // end of group
165 #endif
166
167
168
169

```

## 7.193 Quad.h

```

1 // FICHER : Quad.h
2 // CLASSE : Quad
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field

```

```

7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      23/01/97
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:   La classe Quad permet de declarer des elements
41 *   Quadrangulaire membrane et de realiser
42 *   le calcul du residu local et de la raideur locale pour une loi de
43 *   comportement donnee. La dimension de l'espace pour un tel element
44 *   est 2.
45 *
46 *   l'interpolation est lineaire a 4 noeuds, le nombre de
47 *   point d'integration est de 4.
48 *   *****
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !   but
53 *   -----
54 *   !       !       !
55 *   *****
56 *
57 *   MODIFICATIONS:
58 *
59 *   ! date !   auteur !   but
60 *   -----
61 *   $
62 *****/
63 // -----classe pour un calcul de mecanique-----
64
65
66 #ifndef QUAD_H
67 #define QUAD_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "QuadraMemb.h"
79 #include "FrontSegLine.h"
80 #include "FrontQuadLine.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class Quad : public QuadraMemb
88 {
89
90     public :
91
92     // CONSTRUCTEURS :

```

```

93     // Constructeur par défaut
94     Quad ();
95
96     // Constructeur fonction d'une epaisseur et eventuellement d'un numero
97     // d'identification
98     Quad (double epaiss,int num_mail=0,int num_id=-3);
99
100    // Constructeur fonction d'un numero de maillage et d'identification
101    Quad (int num_mail,int num_id);
102
103    // Constructeur fonction d'une epaisseur, d'un numero d'identification,
104    // du tableau de connexite des noeuds
105    Quad (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107    // Constructeur de copie
108    Quad (const Quad& quadra);
109
110
111    // DESTRUCTEUR :
112    ~Quad ();
113
114    // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115    // méthode virtuelle
116    Element* Nevez_copie() const { Element * el= new Quad(*this); return el;};
117
118    // Surcharge de l'operateur = : realise l'egalite entre deux instances de Quad
119    Quad& operator= (Quad& quadra);
120
121    // METHODES :
122    // 1) derivant des virtuelles pures
123
124    // affichage dans la sortie transmise, des variables duales "nom"
125    // aux differents points d'integration
126    // dans le cas ou nom est vide, affichage de "toute" les variables
127    void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129    // 2) derivant des virtuelles
130
131    protected :
132
133    // adressage des frontières linéiques et surfacique
134    // définit dans les classes dérivées, et utilisées pour la construction des frontières
135    ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
136    { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
137    ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
138    { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
139
140    // VARIABLES PRIVEES :
141    // place memoire commune a tous les elements TriaMembl1
142    static QuadraMemb::DonnComQuad * doCoQuad;
143    // idem mais pour les indicateurs qui servent pour l'initialisation
144    static QuadraMemb::UneFois uneFoisL1;
145
146    class NombresConstruireQuad : public NombresConstruire
147    { public: NombresConstruireQuad();
148      };
149    static NombresConstruireQuad nombre_V; // les nombres propres à l'élément
150
151    // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
152    //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
153    class ConsQuad : public ConstrucElement
154    { public : ConsQuad ()
155      { NouvelleTypeElement nouv (QUADRANGLE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this);
156        if (ParaGlob::NiveauImpression() >= 4)
157          cout << "\n initialisation Quad" << endl;
158          Element::listTypeElement.push_back(nouv);
159        };
160      Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
161      {Element * pt;
162        pt = new Quad (num_maill,num) ;
163        return pt;};
164      // ramene true si la construction de l'element est possible en fonction
165      // des variables globales actuelles: ex en fonction de la dimension
166      bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
167    };
168    static ConsQuad consQuad;
169 };
170 /// @} // end of group
171 #endif
172
173
174
175

```

## 7.194 Quad\_cm1pti.h

```

1 // FICHER : Quad_cm1pti.h
2 // CLASSE : Quad
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          23/01/97
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *
40 *      BUT:           La classe Quad_cm1pti permet de declarer des elements
41 * Quadrangulaire membrane et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 2.
45 *
46 * l'interpolation est lineaire a 4 noeuds, le nombre de
47 * point d'integration est de 1 + gestion d'hourglass
48 *
49 *      VERIFICATION:
50 *
51 *      ! date !   auteur !           but
52 *      -----
53 *      !           !           !
54 *
55 *
56 *      MODIFICATIONS:
57 *      ! date !   auteur !           but
58 *      -----
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef QUAD_CM1PTI_H
67 #define QUAD_CM1PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "QuadraMemb.h"
79 #include "FrontSegLine.h"
80 #include "FrontQuadLine.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85

```



```

86
87 class Quad_cm1pti : public QuadraMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93         // Constructeur par défaut
94         Quad_cm1pti ();
95
96         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
97         // d'identification
98         Quad_cm1pti (double epaiss,int num_mail=0,int num_id=-3);
99
100        // Constructeur fonction d'un numero de maillage et d'identification
101        Quad_cm1pti (int num_mail,int num_id);
102
103        // Constructeur fonction d'une epaisseur, d'un numero d'identification,
104        // du tableau de connexite des noeuds
105        Quad_cm1pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        Quad_cm1pti (const Quad_cm1pti& quadra);
109
110
111        // DESTRUCTEUR :
112        ~Quad_cm1pti ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new Quad_cm1pti(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de Quad_cm1pti
119        Quad_cm1pti& operator= (Quad_cm1pti& quadra);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // affichage dans la sortie transmise, des variables duales "nom"
125        // aux differents points d'integration
126        // dans le cas ou nom est vide, affichage de "toute" les variables
127        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129        // 2) derivant des virtuelles
130
131        protected :
132
133        // adressage des frontieres linéiques et surfacique
134        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
135        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
136        { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
137        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
138        { return ((ElFrontiere*) (new FrontQuadLine(tab,ddelem)));};
139
140        // VARIABLES PRIVEES :
141        // place memoire commune a tous les elements
142        static QuadraMemb::DonnComQuad * doCoQuad;
143        // idem mais pour les indicateurs qui servent pour l'initialisation
144        static QuadraMemb::UneFois uneFoisL1;
145
146        class NombresConstruireQuad_cm1pti : public NombresConstruire
147        { public: NombresConstruireQuad_cm1pti();
148          };
149        static NombresConstruireQuad_cm1pti nombre_V; // les nombres propres à l'élément
150
151        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
152        //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
153        class ConsQuad_cm1pti : public ConstrucElement
154        { public : ConsQuad_cm1pti ()
155          { NouvelleTypeElement nouv (QUADRANGLE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this,"_cm1pti");
156            if (ParaGlob::NiveauImpression() >= 4)
157              cout << "\n initialisation Quad_cm1pti" << endl;
158              Element::listTypeElement.push_back(nouv);
159          };
160          Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
161          {Element * pt;
162            pt = new Quad_cm1pti (num_maill,num) ;
163            return pt;};
164          // ramene true si la construction de l'element est possible en fonction
165          // des variables globales actuelles: ex en fonction de la dimension
166          bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
167        };
168        static ConsQuad_cm1pti consQuad_cm1pti;
169    };
170    /// @} // end of group
171 #endif
172

```

173  
174  
175

## 7.195 QuadCCom.h

```

1 // FICHER : QuadCCom.h
2 // CLASSE : QuadCCom
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      10/11/01
34 *
35 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 *      BUT:      La classe QuadCCom permet de declarer des elements
41 *      Quadrangulaire membranne et de realiser
42 *      le calcul du residu local et de la raideur locale pour une loi de
43 *      comportement donnee. La dimension de l'espace pour un tel element
44 *      est 2.
45 *
46 *      l'interpolation est cubique complet 16 noeuds, le nombre de
47 *      point d'integration est de 16.
48 *      *****
49 *      VERIFICATION:
50 *
51 *      ! date ! auteur ! but
52 *      -----
53 *      ! ! !
54 *      $
55 *      *****
56 *      MODIFICATIONS:
57 *      ! date ! auteur ! but
58 *      -----
59 *      $
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef QUADCCOM_H
67 #define QUADCCOM_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "QuadraMemb.h"
79 #include "FrontQuadCC.h"

```

```

80 #include "FrontSegCub.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class QuadCCom : public QuadraMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93         // Constructeur par default
94         QuadCCom ();
95
96         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
97         // d'identification
98         QuadCCom (double epaiss,int num_mail=0,int num_id=-3);
99
100        // Constructeur fonction d'un numero de maillage et d'identification
101        QuadCCom (int num_mail,int num_id);
102
103        // Constructeur fonction d'une epaisseur, d'un numero d'identification,
104        // du tableau de connexite des noeuds
105        QuadCCom (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        QuadCCom (const QuadCCom& quadra);
109
110
111        // DESTRUCTEUR :
112        ~QuadCCom ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new QuadCCom(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadCCom
119        QuadCCom & operator= (QuadCCom& quadra);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // affichage dans la sortie transmise, des variables duales "nom"
125        // aux differents points d'integration
126        // dans le cas ou nom est vide, affichage de "toute" les variables
127        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129        // 2) derivant des virtuelles
130        // 3) methodes propres a l'element
131
132        protected :
133
134        // adressage des frontieres linéiques et surfacique
135        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
136        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137        { return ((ElFrontiere*) (new FrontSegCub(tab,ddelem)));};
138        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
139        { return ((ElFrontiere*) (new FrontQuadCC(tab,ddelem)));};
140
141        // VARIABLES PRIVEES :
142        // place memoire commune a tous les elements TriaMembL1
143        static QuadraMemb::DonnComQuad * doCoQuadCCom;
144        // idem mais pour les indicateurs qui servent pour l'initialisation
145        static QuadraMemb::UneFois uneFoisQCom;
146
147        class NombresConstruireQuadCCom : public NombresConstruire
148        { public: NombresConstruireQuadCCom();
149          };
150        static NombresConstruireQuadCCom nombre_V; // les nombres propres à l'élément
151
152        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
153        //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
154        class ConsQuadCCom : public ConstrucElement
155        { public : ConsQuadCCom ()
156          { NouvelleTypeElement nouv (QUADRANGLE,CUBIQUE,MECA_SOLIDE_DEFORMABLE,this);
157            if (ParaGlob::NiveauImpression() >= 4)
158              cout << "\n initialisation QuadCCom" << endl;
159            Element::listTypeElement.push_back(nouv);
160          };
161
162          Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
163          {Element * pt;
164            pt = new QuadCCom (num_maill,num) ;
165            return pt;};
166
167        // ramene true si la construction de l'element est possible en fonction
168        // des variables globales actuelles: ex en fonction de la dimension

```

```

167         bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
168     };
169     static ConsQuadCCom consQuadCCom;
170 };
171 /// @} // end of group
172 #endif
173
174
175
176

```

## 7.196 QuadCCom\_cm9pti.h

```

1 // FICHER : QuadCCom_cm9pti.h
2 // CLASSE : QuadCCom
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           3/05/2011
34 *
35 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:         Herezh++
38 *
39 *   $
40 *
41 *   *****
42 *   BUT:   La classe QuadQCom_cm9pti permet de declarer des elements
43 *   * Quadrangulaire membrane et de realiser
44 *   * le calcul du residu local et de la raideur locale pour une loi de
45 *   * comportement donnee. La dimension de l'espace pour un tel element
46 *   * est 2.
47 *
48 *   *
49 *   l'interpolation est cubique complet 16 noeuds, le nombre de
50 *   * point d'integration est de 9 avec gestion d'hourglass.
51 *
52 *   *****
53 *   VERIFICATION:
54 *
55 *   ! date ! auteur ! but
56 *
57 *   -----
58 *   !           !           !
59 *
60 *   $
61 *
62 *   *****
63 *   MODIFICATIONS:
64 *
65 *   ! date ! auteur ! but
66 *
67 *   -----
68 *
69 *   $
70 *
71 *   *****/
72
73 // -----classe pour un calcul de mecanique-----
74
75
76
77
78 #ifndef QUADCCOM_CM9PTI_H
79 #define QUADCCOM_CM9PTI_H
80
81 #include "ParaGlob.h"
82 #include "ElemMeca.h"
83 #include "Met_abstraite.h"
84 #include "GeomQuadrangle.h"

```

```

73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "QuadraMemb.h"
79 #include "FrontQuadCC.h"
80 #include "FrontSegCub.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class QuadCCom_cm9pti : public QuadraMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93         // Constructeur par default
94         QuadCCom_cm9pti ();
95
96         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
97         // d'identification
98         QuadCCom_cm9pti (double epaiss,int num_mail=0,int num_id=-3);
99
100        // Constructeur fonction d'un numero de maillage et d'identification
101        QuadCCom_cm9pti (int num_mail,int num_id);
102
103        // Constructeur fonction d'une epaisseur, d'un numero d'identification,
104        // du tableau de connexite des noeuds
105        QuadCCom_cm9pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        QuadCCom_cm9pti (const QuadCCom_cm9pti& quadra);
109
110
111        // DESTRUCTEUR :
112        ~QuadCCom_cm9pti ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new QuadCCom_cm9pti(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadCCom_cm9pti
119        QuadCCom_cm9pti& operator= (QuadCCom_cm9pti& quadra);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // affichage dans la sortie transmise, des variables duales "nom"
125        // aux differents points d'integration
126        // dans le cas ou nom est vide, affichage de "toute" les variables
127        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129        // 2) derivant des virtuelles
130        // 3) methodes propres a l'element
131
132        protected :
133
134        // adressage des frontieres linéiques et surfacique
135        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
136        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137        { return ((ElFrontiere*) (new FrontSegCub(tab,ddelem)));};
138        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
139        { return ((ElFrontiere*) (new FrontQuadCC(tab,ddelem)));};
140
141        // VARIABLES PRIVEES :
142        // place memoire commune a tous les elements TriaMembL1
143        static QuadraMemb::DonnComQuad * doCoQuadCCom;
144        // idem mais pour les indicateurs qui servent pour l'initialisation
145        static QuadraMemb::UneFois uneFoisQCom;
146
147        class NombresConstruireQuadCCom_cm9pti : public NombresConstruire
148        { public: NombresConstruireQuadCCom_cm9pti();
149        };
150        static NombresConstruireQuadCCom_cm9pti nombre_V; // les nombres propres à l'élément
151
152        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
153        //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
154        class ConsQuadCCom_cm9pti : public ConstrucElement
155        { public : ConsQuadCCom_cm9pti ()
156        { NouvelleTypeElement nouv (QUADRANGLE,CUBIQUE,MECA_SOLIDE_DEFORMABLE,this,"_cm9pti");
157        if (ParaGlob::NiveauImpression() >= 4)
158        cout << "\n initialisation QuadCCom_cm9pti" << endl;
159        Element::listTypeElement.push_back (nouv);

```

```

160         };
161     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
162     {Element * pt;
163     pt = new QuadCCom_cm9pti (num_maill,num) ;
164     return pt;};
165     // ramene true si la construction de l'element est possible en fonction
166     // des variables globales actuelles: ex en fonction de la dimension
167     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
168     };
169     static ConsQuadCCom_cm9pti consQuadCCom_cm9pti;
170 };
171 /// @} // end of group
172 #endif
173
174
175
176

```

## 7.197 QuadQ.h

```

1 // FICHER : QuadQ.h
2 // CLASSE : QuadQ
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:      G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:      Herezh++
38 *
39 *****/
40 *      BUT:      La classe QuadQ permet de declarer des elements
41 *      Quadrangulaire membrane et de realiser
42 *      le calcul du residu local et de la raideur locale pour une loi de
43 *      comportement donnee. La dimension de l'espace pour un tel element
44 *      est 2.
45 *
46 *      l'interpolation est quadratique incomplete 8 noeuds, le nombre de
47 *      point d'integration est de 4.
48 *
49 *
50 *      VERIFICATION:
51 *
52 *      ! date ! auteur ! but
53 *      -----
54 *      ! ! ! !
55 *      $
56 *
57 *      MODIFICATIONS:
58 *
59 *      ! date ! auteur ! but
60 *      -----
61 *      $
62 // -----classe pour un calcul de mecanique-----
63
64

```

```

65
66 #ifndef QUADQ_H
67 #define QUADQ_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "QuadraMemb.h"
79 #include "FrontSegQuad.h"
80 #include "FrontQuadQuad.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class QuadQ : public QuadraMemb
88 {
89     public :
90
91         // CONSTRUCTEURS :
92         // Constructeur par default
93         QuadQ ();
94
95         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
96         // d'identification
97         QuadQ (double epaiss,int num_mail=0,int num_id=-3);
98
99         // Constructeur fonction d'un numero de maillage et d'identification
100        QuadQ (int num_mail,int num_id);
101
102        // Constructeur fonction d'une epaisseur, d'un numero d'identification,
103        // du tableau de connexite des noeuds
104        QuadQ (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
105
106        // Constructeur de copie
107        QuadQ (const QuadQ& quadra);
108
109
110        // DESTRUCTEUR :
111        ~QuadQ ();
112
113        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
114        // méthode virtuelle
115        Element* Nevez_copie() const { Element * el= new QuadQ(*this); return el;};
116
117        // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadQ
118        QuadQ& operator= (QuadQ& quadra);
119
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // affichage dans la sortie transmise, des variables duales "nom"
125        // aux differents points d'integration
126        // dans le cas ou nom est vide, affichage de "toute" les variables
127        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
128
129        // 2) derivant des virtuelles
130        // 3) methodes propres a l'element
131
132        protected :
133
134        // adressage des frontieres linéiques et surfacique
135        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
136        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
138        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
139        { return ((ElFrontiere*) (new FrontQuadQuad(tab,ddelem)));};
140
141        // VARIABLES PRIVEES :
142        // place memoire commune a tous les elements TriaMembL1
143        static QuadraMemb::DonnComQuad * doCoQuadQ;
144        // idem mais pour les indicateurs qui servent pour l'initialisation
145        static QuadraMemb::UneFois uneFoisQ;
146
147        class NombresConstruireQuadQ : public NombresConstruire
148        { public: NombresConstruireQuadQ();
149          };
150        static NombresConstruireQuadQ nombre_V; // les nombres propres à l'élément
151

```

```

152 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
153 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
154 class ConsQuadQ : public ConstrucElement
155 { public : ConsQuadQ ()
156   { NouvelleTypeElement nouv (QUADRANGLE,QUADRATIQUE,MECA_SOLIDE_DEFORMABLE,this);
157   if (ParaGlob::NiveauImpression() >= 4)
158     cout << "\n initialisation QuadQ" << endl;
159     Element::listTypeElement.push_back(nouv);
160   };
161   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
162   {Element * pt;
163     pt = new QuadQ (num_maill,num) ;
164     return pt;};
165   // ramene true si la construction de l'element est possible en fonction
166   // des variables globales actuelles: ex en fonction de la dimension
167   bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
168   };
169   static ConsQuadQ consQuadQ;
170 };
171 /// @} // end of group
172 #endif
173
174
175
176

```

## 7.198 QuadQCom.h

```

1 // FICHER : QuadQCom.h
2 // CLASSE : QuadQCom
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      *****
41 *      BUT:      La classe QuadQCom permet de declarer des elements
42 *      Quadrangulaire membrane et de realiser
43 *      le calcul du residu local et de la raideur locale pour une loi de
44 *      comportement donnee. La dimension de l'espace pour un tel element
45 *      est 2.
46 *
47 *      l'interpolation est quadratique complete 9 noeuds, le nombre de
48 *      point d'integration est de 9.
49 *
50 *      *****
51 *
52 *      VERIFICATION:
53 *
54 *      ! date ! auteur ! but
55 *
56 *      ! ! !
57 *
58 *      *****
59 *
60 *      MODIFICATIONS:
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```



```

57 *      ! date !   auteur !           but           !      *
58 *      -----
59 *                                     $      *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef QUADQCOM_H
67 #define QUADQCOM_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "QuadraMemb.h"
79 #include "FrontQuadQC.h"
80 #include "FrontSegQuad.h"
81
82
83 /// @addtogroup groupe_des_elements_finis
84 /// @{
85 ///
86
87
88 class QuadQCom : public QuadraMemb
89 {
90
91     public :
92
93         // CONSTRUCTEURS :
94         // Constructeur par default
95         QuadQCom ();
96
97         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
98         // d'identification
99         QuadQCom (double epaiss,int num_mail=0,int num_id=-3);
100
101         // Constructeur fonction d'un numero de maillage et d'identification
102         QuadQCom (int num_mail,int num_id);
103
104         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
105         // du tableau de connexite des noeuds
106         QuadQCom (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
107
108         // Constructeur de copie
109         QuadQCom (const QuadQCom& quadra);
110
111
112         // DESTRUCTEUR :
113         ~QuadQCom ();
114
115         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
116         // méthode virtuelle
117         Element* Nevez_copie() const { Element * el= new QuadQCom(*this); return el;};
118
119         // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadQCom
120         QuadQCom& operator= (QuadQCom& quadra);
121
122         // METHODES :
123         // 1) derivant des virtuelles pures
124
125         // renseignement d'un élément complet à partir d'un élément incomplet de même type
126         // retourne les nouveaux noeuds construit à partir de l'interpolation incomplète.
127         // dans le cas l'élément n'est pas concerné, retourne une liste vide
128         // ramène également une liste de même dimension contenant les bornes en numéros de noeuds
129         // entre lesquelles il faut définir les nouveaux numéros de noeuds si l'on veut conserver
130         // une largeur de bande optimisée du même type
131         // nbnt+1: est le premier numéro de noeud utilisable pour les nouveaux noeuds
132         virtual list <Noeud *> Construct_from_incomplet(const Element & elem,list <DeuxEntiers> &
li_bornes,int nbnt);
133
134         // affichage dans la sortie transmise, des variables duales "nom"
135         // aux differents points d'integration
136         // dans le cas ou nom est vide, affichage de "toute" les variables
137         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
138
139         // 2) derivant des virtuelles
140         // 3) methodes propres a l'element
141
142     protected :

```

```

143
144 // adressage des frontières linéiques et surfacique
145 // définit dans les classes dérivées, et utilisées pour la construction des frontières
146 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
147 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
148 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
149 { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
150
151 // VARIABLES PRIVEES :
152 // place memoire commune a tous les elements TriaMembL1
153 static QuadraMemb::DonnComQuad * doCoQuadQCom;
154 // idem mais pour les indicateurs qui servent pour l'initialisation
155 static QuadraMemb::UneFois uneFoisQCom;
156
157 class NombresConstruireQuadQCom : public NombresConstruire
158 { public: NombresConstruireQuadQCom();
159 };
160 static NombresConstruireQuadQCom nombre_V; // les nombres propres à l'élément
161
162 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
163 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
164 class ConsQuadQCom : public ConstrucElement
165 { public : ConsQuadQCom ()
166 { NouvelleTypeElement nouv (QUADRANGLE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this);
167 if (ParaGlob::NiveauImpression() >= 4)
168 cout << "\n initialisation QuadQCom" << endl;
169 Element::listTypeElement.push_back(nouv);
170 };
171 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
172 {Element * pt;
173 pt = new QuadQCom (num_maill,num) ;
174 return pt;};
175 // ramene true si la construction de l'element est possible en fonction
176 // des variables globales actuelles: ex en fonction de la dimension
177 bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
178 };
179 static ConsQuadQCom consQuadQCom;
180 };
181 /// @} // end of group
182 #endif
183
184
185
186

```

## 7.199 QuadQCom\_cm4pti.h

```

1 // FICHER : QuadQCom_cm4pti.h
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:      23/01/97
33 *
34 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:    Herezh++
37 *
38 * *****/

```

```

39 *      BUT:  La classe QuadQCom permet de declarer des elements      *
40 * Quadrangulaire membrane et de realiser                          *
41 * le calcul du residu local et de la raideur locale pour une loi de *
42 * comportement donnee. La dimension de l'espace pour un tel element *
43 * est 2.                                                            *
44 *                                                                    *
45 * l'interpolation est quadratique complete 9 noeuds, le nombre de  *
46 * point d'integration est de 4 + gestion d'hourglass.            *
47 *      ***** *
48 *      VERIFICATION:                                               *
49 *                                                                    *
50 *      ! date !      auteur !      but                               ! *
51 *      ----- *
52 *      !      !      !      !                                     ! *
53 *      *      *      *      *                                     $ *
54 *      ***** *
55 *      MODIFICATIONS:                                              *
56 *      ! date !      auteur !      but                               ! *
57 *      ----- *
58 *      *      *      *      *                                     $ *
59 *      *****/
60
61 // -----classe pour un calcul de mecanique-----
62
63
64
65 #ifndef QUADQCOM_CM4PTI_H
66 #define QUADQCOM_CM4PTI_H
67
68 #include "ParaGlob.h"
69 #include "ElemMeca.h"
70 #include "Met_abstraite.h"
71 #include "GeomQuadrangle.h"
72 #include "Noeud.h"
73 #include "UtilLecture.h"
74 #include "Tenseur.h"
75 #include "NevezTenseur.h"
76 #include "Deformation.h"
77 #include "QuadraMemb.h"
78 #include "FrontQuadQC.h"
79 #include "FrontSegQuad.h"
80
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class QuadQCom_cm4pti : public QuadraMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93         // Constructeur par defaut
94         QuadQCom_cm4pti ();
95
96         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
97         // d'identification
98         QuadQCom_cm4pti (double epaiss,int num_mail=0,int num_id=-3);
99
100        // Constructeur fonction d'un numero de maillage et d'identification
101        QuadQCom_cm4pti (int num_mail,int num_id);
102
103        // Constructeur fonction d'une epaisseur, d'un numero d'identification,
104        // du tableau de connexite des noeuds
105        QuadQCom_cm4pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        QuadQCom_cm4pti (const QuadQCom_cm4pti& quadra);
109
110
111        // DESTRUCTEUR :
112        ~QuadQCom_cm4pti ();
113
114        // creation d'un element de copie: utilisation de l'operateur new et du constructeur de copie
115        // methode virtuelle
116        Element* Nevez_copie() const { Element * el= new QuadQCom_cm4pti(*this); return el;};
117
118        // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadQCom_cm4pti
119        QuadQCom_cm4pti& operator= (QuadQCom_cm4pti& quadra);
120
121        // METHODES :
122        // 1) derivant des virtuelles pures
123
124        // renseignement d'un element complet a partir d'un element incomplet de meme type
125        // retourne les nouveaux noeuds construit a partir de l'interpolation incomplete.

```

```

126 // dans le cas l'élément n'est pas concerné, retourne une liste vide
127 // ramène également une liste de même dimension contenant les bornes en numéros de noeuds
128 // entre lesquelles il faut définir les nouveaux numéros de noeuds si l'on veut conserver
129 // une largeur de bande optimisée du même type
130 // nbnt+1: est le premier numéro de noeud utilisable pour les nouveaux noeuds
131 virtual list <Noeud *> Construct_from_imcomplet(const Element & elem,list <DeuxEntiers> &
    li_bornes,int nbnt);
132
133 // affichage dans la sortie transmise, des variables duales "nom"
134 // aux différents points d'intégration
135 // dans le cas ou nom est vide, affichage de "toute" les variables
136 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
137
138 // 2) derivant des virtuelles
139 // 3) methodes propres a l'element
140
141 protected :
142
143 // adressage des frontières linéiques et surfacique
144 // définit dans les classes dérivées, et utilisées pour la construction des frontières
145 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
146 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
147 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
148 { return ((ElFrontiere*) (new FrontQuadQC(tab,ddelem)));};
149
150 // VARIABLES PRIVEES :
151 // place memoire commune a tous les elements TriaMemBL1
152 static QuadraMemb::DonnComQuad * doCoQuadQCom;
153 // idem mais pour les indicateurs qui servent pour l'initialisation
154 static QuadraMemb::UneFois uneFoisQCom;
155
156 class NombresConstruireQuadQCom_cm4pti : public NombresConstruire
157 { public: NombresConstruireQuadQCom_cm4pti();
158 };
159 static NombresConstruireQuadQCom_cm4pti nombre_V; // les nombres propres à l'élément
160
161 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
162 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
163 class ConsQuadQCom_cm4pti : public ConstrucElement
164 { public : ConsQuadQCom_cm4pti ()
165     { NouvelleTypeElement nouv(QUADRANGLE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cm4pti");
166     if (ParaGlob::NiveauImpression() >= 4)
167     cout << "\n initialisation QuadQCom_cm4pti" << endl;
168     Element::listTypeElement.push_back(nouv);
169     };
170     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
171     {Element * pt;
172     pt = new QuadQCom_cm4pti (num_maill,num) ;
173     return pt;};
174     // ramene true si la construction de l'element est possible en fonction
175     // des variables globales actuelles: ex en fonction de la dimension
176     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
177 };
178 static ConsQuadQCom_cm4pti consQuadQCom_cm4pti;
179 };
180 /// @} // end of group
181 #endif
182
183
184
185

```

## 7.200 QuadraMemb.h

```

1 // FICHER : QuadraMemb.h
2 // CLASSE : QuadraMemb
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //

```

```

22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *
34 *   DATE:      15/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *****/
41 * La classe QuadraMemb permet de declarer des elements quadrangumaire membrane et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de comportement
43 * donnee. La dimension de l'espace pour un tel element est 2
44 *
45 * l'interpolation le nombre de point d'integration sont definit dans les classes derivees
46 *
47 * l'element est virtuel
48 *
49 *
50 *
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !       !           !
55 *
56 *
57 *
58 *
59 *
60 *   ! date !   auteur !           but
61 *   -----
62 *
63 *
64 *
65 *****/
66
67 // -----classe pour un calcul de mecanique-----
68
69 // La classe QuadraMemb permet de declarer des elements quadrangumaire membrane et de realiser
70 // le calcul du residu local et de la raideur locale pour une loi de comportement
71 // donnee. La dimension de l'espace pour un tel element est 2
72 //
73 // l'interpolation le nombre de point d'integration sont definit dans les classes derivees
74 //
75 // l'element est virtuel
76
77
78 #ifndef QUADRAMEMB_H
79 #define QUADRAMEMB_H
80
81 #include "ParaGlob.h"
82 #include "ElemMeca.h"
83 #include "Met_abstraite.h"
84 #include "GeomQuadrangle.h"
85 #include "GeomSeg.h"
86 #include "Noeud.h"
87 #include "Utillecture.h"
88 #include "Tenseur.h"
89 #include "NevezTenseur.h"
90 #include "Deformation.h"
91 #include "Epai.h"
92
93 /// @addtogroup groupe_des_elements_finis
94 /// @
95 ///
96
97
98 class QuadraMemb : public ElemMeca
99 {
100
101     public :
102
103         // CONSTRUCTEURS :
104         // Constructeur par default
105         QuadraMemb ();
106
107         // Constructeur fonction d'un numero

```

```

108 // d'identification , d'identificateur d'interpolation et de geometrie
109 // et éventuellement un string d'information annexe
110 QuadraMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string
info="");
111
112 // Constructeur fonction d'un numero de maillage et d'identification,
113 // du tableau de connexite des noeuds, d'identificateur d'interpolation et de geometrie
114 // et éventuellement un string d'information annexe
115 QuadraMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,
116 const Tableau<Noeud *>& tab,string info="") ;
117
118 // Constructeur de copie
119 QuadraMemb (const QuadraMemb& quadra);
120
121
122 // DESTRUCTEUR :
123 ~QuadraMemb ();
124
125
126 // Surcharge de l'operateur = : realise l'egalite entre deux instances de QuadraMemb
127 QuadraMemb& operator= (QuadraMemb& quadra);
128
129 // METHODES :
130 // 1) derivant des virtuelles pures
131
132 // Lecture des donnees de la classe sur fichier
133 void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
134
135 // affichage d'info en fonction de ordre
136 // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
137 void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> *
tabMaillageNoeud)
138 { return Element::Info_com_El (nombre->nbne,entreePrinc,ordre,tabMaillageNoeud);};
139
140 // ramene l'element geometrique
141 ElemGeomC0& ElementGeometrique() const { return unefois->doCoMemb->quadra;};
142 // ramene l'element geometrique en constant
143 const ElemGeomC0& ElementGeometrique_const() const {return unefois->doCoMemb->quadra;};
144
145 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
associé
146 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
147 // 1) cas où l'on utilise la place passée en argument
148 Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
149
150 // -- connaissances particulières sur l'élément
151 // ramène l'épaisseur de l'élément
152 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
153 // ici par défaut l'épaisseur est constante
154 virtual double Epaisseurs(Enum_dure enu, const Coordonnee& ) {return H(enu);};
155 // ramène l'épaisseur moyenne de l'élément (indépendante du point)
156 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
157 virtual double EpaisseurMoyenne(Enum_dure enu ) {return H(enu);};
158
159 // retourne la liste des données particulières actuellement utilisés
160 // par l'élément (actif ou non), sont exclu de cette liste les données particulières des noeuds
161 // reliés à l'élément
162 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
163 List_io <TypeQuelconque> Les_types_particuliers_internes (bool absolue) const;
164
165 // récupération de grandeurs particulières au numéro d'ordre = iteg
166 // celles-ci peuvent être quelconques
167 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
168 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
169 void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);
170
171 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
172 void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
173
174 // inactive les ddl du problème primaire de mécanique
175 inline void Inactive_ddl_primaire()
176 {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};
177 // active les ddl du problème primaire de mécanique
178 inline void Active_ddl_primaire()
179 {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl);};
180 // ajout des ddl de contraintes pour les noeuds de l'élément
181 inline void Plus_ddl_Sigma()
182 {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
183 // inactive les ddl du problème de recherche d'erreur : les contraintes
184 inline void Inactive_ddl_Sigma()
185 {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
186 // active les ddl du problème de recherche d'erreur : les contraintes
187 inline void Active_ddl_Sigma()
188 {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
189 // active le premier ddl du problème de recherche d'erreur : SIGMA11
190 inline void Active_premier_ddl_Sigma()

```

```

191     {ElemMeca::Act_premier_ddl_Sigma()};
192
193 // lecture de données diverses sur le flot d'entrée
194 void LectureContraintes(UtilLecture * entreePrinc)
195 { if (unefois->CalResPrem_t == 1)
196     ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
197     else
198     { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
199       unefois->CalResPrem_t = 1;
200     }
201 };
202
203 // retour des contraintes en absolu retour true si elle existe sinon false
204 bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
205 { if (unefois->CalResPrem_t == 1)
206     ElemMeca::ContraintesEnAbsolues(false,lesPtMecaInt.TabSigHH_t(),tabSig);
207     else
208     { ElemMeca::ContraintesEnAbsolues(true,lesPtMecaInt.TabSigHH_t(),tabSig);
209       unefois->CalResPrem_t = 1;
210     }
211     return true;
212 };
213
214 // Libere la place occupee par le residu et eventuellement la raideur
215 // par l'appel de Libere de la classe mere et libere les differents tenseurs
216 // intermediaires cree pour le calcul et les grandeurs pointee
217 // de la raideur et du residu
218 void Libere ();
219
220 // acquisition d'une loi de comportement
221 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
222
223 // test si l'element est complet
224 // = 1 tout est ok, =0 element incomplet
225 int TestComplet();
226
227 // procedure permettant de completer l'element apres
228 // sa creation avec les donnees du bloc transmis
229 // peut etre appeler plusieurs fois
230 // ici il s'agit de l'epaisseur
231 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
232 // Compléter pour la mise en place de la gestion de l'hourglass
233 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc) ;
234
235 // ramene l'epaisseur au point d'integration ni
236 // dans le cas où l'élément n'est pas totalement construit -> retour 0.
237 inline double H(int ni, Enum_dure enu = TEMPS_tdt )
238 {if (donnee_specif.epais.Taille())
239     {switch (enu)
240         { case TEMPS_0: return donnee_specif.epais(ni).epaisseur0; break;
241           case TEMPS_t: return donnee_specif.epais(ni).epaisseur_t; break;
242           case TEMPS_tdt: return donnee_specif.epais(ni).epaisseur_tdt; break;
243         };
244     }
245     else
246     return 0.; // cas épaisseur pas défini
247 };
248
249 // ramene l'epaisseur moyenne de tous les pti
250 // dans le cas où l'élément n'est pas totalement construit -> retour 0.
251 inline double H_moy(Enum_dure enu)
252 {double retour=0.;
253     int nb_epais_actuel = donnee_specif.epais.Taille();
254     for (int i=1; i<= nb_epais_actuel;i++)
255     {switch (enu)
256         { case TEMPS_0: retour += donnee_specif.epais(i).epaisseur0; break;
257           case TEMPS_t: retour += donnee_specif.epais(i).epaisseur_t; break;
258           case TEMPS_tdt: retour += donnee_specif.epais(i).epaisseur_tdt; break;
259           default: cout << "\n erreur dans le calcul de l'epaisseur moyenne "
260                     << "\n H_moy(...)";
261                     Sortie(1);
262         };
263     };
264     return retour/nombre->nbi; // retour de la moyenne
265 };
266
267 // // modifie l'epaisseur Moyenne à tdt
268 // virtual void Modifie_epaisseur_moyenne_tdt(const double& h_t)
269 // {donnee_specif.epais.epaisseur_tdt = h_t;};
270
271 // ramene vrai si la surface numéro ns existe pour l'élément
272 // dans le cas des éléments quadrangulaires il ne peut y avoir qu'une
273 // seule surface
274 bool SurfExiste(int ns) const
275 { if ((ns==1)&&(ParaGlob::Dimension() >= 2)) return true; else return false;};
276
277 // ramene vrai si l'arête numéro na existe pour l'élément

```

```

278     bool AreteExiste(int na) const
279         { if ((na <= 4) || (na >= 1)) return true; else return false; };
280
281     // retourne les tableaux de ddl associés aux noeuds, gere par l'element
282     // ce tableau et specifique a l'element
283     const DdlElement & TableauDdl() const
284         { return unefois->doCoMemb->tab_ddl; };
285
286     // Calcul du residu local et de la raideur locale,
287     // pour le schema implicite
288     Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
289
290     // Calcul du residu local a t
291     // pour le schema explicite par exemple
292     Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
293         { return QuadraMemb::CalculResidu(false,pa); };
294
295     // Calcul du residu local a tdt
296     // pour le schema explicite par exemple
297     Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
298         { return QuadraMemb::CalculResidu(true,pa); };
299
300 // Calcul de la matrice masse pour l'élément
301 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
302
303 // ----- calcul dynamique -----
304 // calcul de la longueur d'arrête de l'élément minimal
305 // divisé par la célérité la plus rapide dans le matériau
306 double Long_arrete_mini_sur_c(Enum_dure temps)
307     { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps); };
308
309 //----- calcul d'erreur, remontée des contraintes -----
310 // 1) calcul du résidu et de la matrice de raideur pour le calcul d'erreur
311 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
312 // 2) remontée aux erreurs aux noeuds
313 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
314
315 // ----- affichage ou récupération d'informations -----
316 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
317 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
318 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
319 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
320 // temps: dit si c'est à 0 ou t ou tdt
321 int PointLePlusPres(Enum_dure temps, Enum_ddl enu, const Coordonnee& M)
322     { return PtLePlusPres(temps, enu, M); };
323
324 // recuperation des coordonnées du point de numéro d'ordre iteg pour
325 // la grandeur enu
326 // temps: dit si c'est à 0 ou t ou tdt
327 // si erreur retourne erreur à true
328 Coordonnee CoordPtInteg(Enum_dure temps, Enum_ddl enu, int iteg, bool& erreur)
329     { return CoordPtInt(temps, enu, iteg, erreur); };
330
331 // récupération des valeurs au numéro d'ordre = iteg pour
332 // les grandeur enu
333 Tableau <double> Valeur_a_diff_temps(bool absolue, Enum_dure enu_t, const List_io<Ddl_enum_etendu>&
    enu, int iteg);
334 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
335 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
336 // de conteneurs quelconque associée
337 void ValTensorielle_a_diff_temps(bool absolue, Enum_dure enu_t, List_io<TypeQuelconque>& enu, int iteg);
338
339 //===== lecture écriture dans base info =====
340
341 // cas donne le niveau de la récupération
342 // = 1 : on récupère tout
343 // = 2 : on récupère uniquement les données variables (supposées comme telles)
344 void Lecture_base_info
345     (ifstream& ent, const Tableau<Noeud *> * tabMaillageNoeud, const int cas) ;
346 // cas donne le niveau de sauvegarde
347 // = 1 : on sauvegarde tout
348 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
349 void Ecriture_base_info(ofstream& sort, const int cas) ;
350
351 // 2) derivant des virtuelles
352
353 // retourne un tableau de ddl element, correspondant à la
354 // composante de sigma -> SIG11, pour chaque noeud qui contient
355 // des ddl de contrainte
356 // -> utilisé pour l'assemblage de la raideur d'erreur
357 DdlElement& Tableau_de_Sig1() const
358     { return unefois->doCoMemb->tab_Err1Sig11; };
359
360 // actualisation des ddl et des grandeurs actives de t+dt vers t
361 void TdtversT();

```



```

363 // actualisation des ddl et des grandeurs actives de t vers tdt
364 void TversTdt();
365
366 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
367 // qu'une fois la remontée aux contraintes effectuées sinon aucune
368 // action. En retour la valeur de l'erreur sur l'élément
369 // type indique le type de calcul d'erreur :
370 void ErreurElement(int type,double& errElemRelative
371 ,double& numerateur, double& denominateur);
372
373 // mise à jour de la boite d'encombrement de l'élément, suivant les axes I_a globales
374 // en retour coordonnées du point mini dans retour.Premier() et du point maxi dans .Second()
375 // la méthode est différente de la méthode générale car il faut prendre en compte l'épaisseur de
l'élément
376 virtual const DeuxCoordonnees& Boite_encombre_element(Enum_dure temps);
377
378 // --- calcul des seconds membres suivant les chargements
379
380 // calcul des seconds membres suivant les chargements
381 // cas d'un chargement volumique,
382 // force indique la force volumique appliquée
383 // retourne le second membre résultant
384 // ici on l'épaisseur de l'élément pour constituer le volume
385 // -> explicite à t
386 Vecteur SM_charge_volumique_E_t
387 (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_)
388 { return QuadraMemb::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_); };
389 // -> explicite à tdt
390 Vecteur SM_charge_volumique_E_tdt
391 (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_)
392 { return QuadraMemb::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_); };
393 // -> implicite,
394 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
395 // retourne le second membre et la matrice de raideur correspondant
396 ResRaid SMR_charge_volumique_I
397 (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_);
398
399 // cas d'un chargement surfacique, sur les frontières des éléments
400 // force indique la force surfacique appliquée
401 // numface indique le numéro de la face chargée
402 // retourne le second membre résultant
403 // -> version explicite à t
404 Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
405 { return QuadraMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa); };
406 // -> version explicite à tdt
407 Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
408 { return QuadraMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa); };
409 // -> implicite,
410 // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
411 // retourne le second membre et la matrice de raideur correspondant
412 ResRaid SMR_charge_surfacique_I
413 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const ParaAlgoControle &
pa);
414
415 // cas d'un chargement lineique, sur les aretes frontières des éléments
416 // force indique la force lineique appliquée
417 // numarete indique le numéro de l'arete chargée
418 // retourne le second membre résultant
419 // NB: il y a une définition par défaut pour les éléments qui n'ont pas
420 // d'arete externe -> message d'erreur d'où le virtuel et non virtuel pur
421 // -> explicite à t
422 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
423 { return QuadraMemb::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); };
424 // -> explicite à tdt
425 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
426 { return QuadraMemb::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); };
427 // -> implicite,
428 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
429 // retourne le second membre et la matrice de raideur correspondant
430 ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa);
431
432 // cas d'un chargement lineique suiveuse, sur les aretes frontières des éléments 2D (uniquement)
433 // force indique la force lineique appliquée
434 // numarete indique le numéro de l'arete chargée
435 // retourne le second membre résultant
436 // -> explicite à t
437 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
438 { return QuadraMemb::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa); };

```

```

439 // -> explicite à tdt
440 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nd* pt_fonct,int
numArete,const ParaAlgoControle & pa)
441 { return QuadraMemb::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa);};
442 // -> implicite,
443 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
444 // retourne le second membre et la matrice de raideur correspondant
445 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nd* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
446
447 // cas d'un chargement de type pression, sur les frontières des éléments
448 // pression indique la pression appliquée
449 // numface indique le numéro de la face chargée
450 // retourne le second membre résultant
451 // NB: il y a une définition par défaut pour les éléments qui n'ont pas de
452 // surface externe -> message d'erreur d'où le virtuel et non virtuel pur
453 // -> explicite à t
454 Vecteur SM_charge_pression_E_t(double pression,Fonction_nd* pt_fonct,int numFace,const
ParaAlgoControle & pa)
455 { return QuadraMemb::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa);};
456 // -> explicite à tdt
457 Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nd* pt_fonct,int numFace,const
ParaAlgoControle & pa)
458 { return QuadraMemb::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa);};
459 // -> implicite,
460 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
461 // retourne le second membre et la matrice de raideur correspondant
462 ResRaid SMR_charge_pression_I(double pression,Fonction_nd* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
463
464 // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
465 // presUniDir indique le vecteur appliquée
466 // numface indique le numéro de la face chargée
467 // retourne le second membre résultant
468 // -> explicite à t
469 Vecteur SM_charge_presUniDir_E_t(const Coordonnee& presUniDir,Fonction_nd* pt_fonct,int
numFace,const ParaAlgoControle & pa)
470 { return QuadraMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,false,pa);};
471 // -> explicite à tdt
472 Vecteur SM_charge_presUniDir_E_tdt(const Coordonnee& presUniDir,Fonction_nd* pt_fonct,int
numFace,const ParaAlgoControle & pa)
473 { return QuadraMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,true,pa);};
474 // -> implicite,
475 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
476 // retourne le second membre et la matrice de raideur correspondant
477 ResRaid SMR_charge_presUniDir_I(const Coordonnee& presUniDir,Fonction_nd* pt_fonct,int
numFace,const ParaAlgoControle & pa);
478
479 // cas d'un chargement surfacique hydrostatique,
480 // poidvol: indique le poids volumique du liquide
481 // M_liquide : un point de la surface libre
482 // dir_normal_liquide : direction normale à la surface libre
483 // retourne le second membre résultant
484 // -> explicite à t
485 Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
486 ,const ParaAlgoControle & pa
487 ,bool sans_limitation)
488 { return
489 QuadraMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation);};
490 // -> explicite à tdt
491 Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
492 ,const ParaAlgoControle & pa
493 ,bool sans_limitation)
494 { return
495 QuadraMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation);};
496 // -> implicite,
497 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
498 // retourne le second membre et la matrice de raideur correspondant
499 ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
500 ,const ParaAlgoControle & pa
501 ,bool sans_limitation) ;
502
503
504 // cas d'un chargement surfacique hydro-dynamique,
505 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
506 //  $F_n = \text{poids\_volu} * f_n(V) * S * (\text{normale} * u) * u$ , u étant le vecteur directeur de V (donc
unitaire)
507 // une suivant la direction normale à la vitesse de type portance
508 //  $F_t = \text{poids\_volu} * f_t(V) * S * (\text{normale} * u) * w$ , w unitaire, normal à V, et dans le plan n et V
509 // une suivant la vitesse tangente de type frottement visqueux
510 //  $T = \text{to}(Vt) * S * ut$ , Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
511 // coef_mul: est un coefficient multiplicateur global (de tout)
512 // retourne le second membre résultant
513 // -> explicite à t
514 Vecteur SM_charge_hydrodynamique_E_t( CourbelD* frot_fluid,const double& poidvol

```

```

515         , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
516         , CourbelD* coef_aero_t
517         ,const ParaAlgoControle & pa)
518     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
519     // -> explicite à tdt
520     Vecteur SM_charge_hydrodynamique_E_tdt( CourbelD* frot_fluid,const double& poidvol
521         , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
522         , CourbelD* coef_aero_t
523         ,const ParaAlgoControle & pa)
524     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};
525     // -> implicite,
526     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
527     // retourne le second membre et la matrice de raideur correspondant
528     ResRaid SMR_charge_hydrodynamique_I( CourbelD* frot_fluid,const double& poidvol
529         , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
530         , CourbelD* coef_aero_t
531         ,const ParaAlgoControle & pa) ;
532
533     // ===== définition et/ou construction des frontières =====
534
535     // Calcul des frontieres de l'element
536     // creation des elements frontieres et retour du tableau de ces elements
537     // la création n'a lieu qu'au premier appel
538     // ou lorsque l'on force le paramètre force a true
539     // dans ce dernier cas seul les frontière effacées sont recréées
540     Tableau <ElFrontiere*> const & Frontiere(bool force = false);
541
542     // ramène la frontière point
543     // éventuellement création des frontieres points de l'element et stockage dans l'element
544     // si c'est la première fois sinon il y a seulement retour de l'elements
545     // a moins que le paramètre force est mis a true
546     // dans ce dernier cas la frontière effacée est recréée
547     // num indique le numéro du point à créer (numérotation EF)
548     // ElFrontiere* const Frontiere_points(int num,bool force = false);
549
550     // ramène la frontière linéique
551     // éventuellement création des frontieres linéique de l'element et stockage dans l'element
552     // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
553     // a moins que le paramètre force est mis a true
554     // dans ce dernier cas la frontière effacée est recréée
555     // num indique le numéro de l'arête à créer (numérotation EF)
556     // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
557
558     // ramène la frontière surfacique
559     // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
560     // si c'est la première fois sinon il y a seulement retour de l'elements
561     // a moins que le paramètre force est mis a true
562     // dans ce dernier cas la frontière effacée est recréée
563     // num indique le numéro de la surface à créer (numérotation EF)
564     // ici normalement uniquement 1 possible
565     // ElFrontiere* const Frontiere_surfacique(int num,bool force = false);
566
567
568     // 3) methodes propres a l'element
569
570     // ajout du tableau specific de ddl des noeuds
571     // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
572     // des noeuds constituant l'element
573     void ConstTabDdl();
574
575     public :
576
577     // ----- definition de la classe conteneur de donnees communes -----
578     class DonnComQuad
579     { public :
580         DonnComQuad (GeomQuadrangle& quad,DdlElement& tab,
581             DdlElement& tabErr,DdlElement& tab_ErrlSig,
582             Met_abstraite& met_gene
583             , Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
584             GeomQuadrangle& quadEr,GeomQuadrangle& quadS,
585             GeomSeg& segmS,Vecteur& residu_int,Mat_pleine& raideur_int,
586             Tableau <Vecteur* > & residu_extN,Tableau <Mat_pleine* >& raideurs_extN,
587             Tableau <Vecteur* > & residu_extA,Tableau <Mat_pleine* >& raideurs_extA,
588             Tableau <Vecteur* >& residu_extS,Tableau <Mat_pleine* >& raideurs_extS,
589             Mat_pleine& mat_masse,GeomQuadrangle& quadMas,int nbi,GeomQuadrangle&
quadHourg,
590             Mat_pleine* quadmatD,Vecteur* quadresD
591             ) ;
592         DonnComQuad(DonnComQuad& a);
593         ~DonnComQuad();
594         // variables
595         GeomQuadrangle quadra; // contient les fonctions d'interpolation et

```

```

596                                     // les derivees
597         DdlElement tab_ddl; // tableau des degres
598         //de liberte des noeuds de l'element commun a tous les
599         // elements
600     Met_abstraite met_quadraMemb;
601         Mat_pleine matGeom ; // matrice géométrique
602     Mat_pleine matInit ; // matrice initiale
603     Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
604     Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
605     Tableau < Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
606     // ---- concernant les frontières et particulièrement le calcul de second membre
607     GeomQuadrangle quadraS; // contient les fonctions d'interpolation et les derivees
608     GeomSeg          segS; // " " "
609
610         //----- calcul d'erreur -----
611     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
612     // d'erreur : contraintes
613     DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud, servant pour le
calcul
614         // d'erreur : contraintes, en fait pour l'assemblage
615     Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
616     Mat_pleine raidErr; // raideur pour le calcul d'erreur
617     GeomQuadrangle quadraEr; // contient les fonctions d'interpolation et
618         // les derivees pour le calcul du hessien dans
619         //la résolution de la fonctionnelle d'erreur
620     // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
-----
621         // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
622     Vecteur residu_interne;
623     Mat_pleine raideur_interne;
624     Tableau <Vecteur* > residus_externeN; // pour les noeuds
625     Tableau <Mat_pleine* > raideurs_externeN; // pour les noeuds
626     Tableau <Vecteur* > residus_externeA; // pour les aretes
627     Tableau <Mat_pleine* > raideurs_externeA; // pour les aretes
628     Tableau <Vecteur* > residus_externeS; // pour les surfaces
629     Tableau <Mat_pleine* > raideurs_externeS; // pour les surfaces
630     // ----- données concernant la dynamique -----
631     Mat_pleine matrice_masse;
632     GeomQuadrangle quadraMas; // contient les fonctions d'interpolation et ...
633         // pour les calculs relatifs à la masse
634         // ----- blocage éventuel d'hourglass
635         // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
Cal_explici_hourglass
636     GeomQuadrangle* quadraHourg; // contient les fonctions d'interpolation
637
638         // --- blocage éventuel de stabilisation normale à la membrane
639         // utilisé dans: ElemMeca::Cal_implicit_StabMembBiel, ElemMeca::Cal_explicit_StabMembBiel
640     Mat_pleine* quadramatD; // raideur éventuelle,
641     Vecteur* quadraresD; // résidu éventuelle,
642
643     };
644
645     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
646     // et un pointeur sur les données statiques communes
647     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
648     // classe est défini. Son allocation est effectuée dans les classes dérivées
649     class UneFois
650     { public :
651         UneFois () ; // constructeur par défaut
652         ~UneFois () ; // destructeur
653
654         // VARIABLES :
655     public :
656         QuadraMemb::DonnComQuad * doCoMemb;
657
658         // incicateurs permettant de dimensionner seulement au premier passage
659         // utilise dans "CalculResidu" et "Calcul_implicit"
660         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
661         int CalimpPrem;
662         int dualSortQuad; // pour la sortie des valeurs au pt d'integ
663         int CalSMLin_t; // pour les seconds membres concernant les arretes
664         int CalSMLin_tdt; // pour les seconds membres concernant les arretes
665         int CalSMRlin; // pour les seconds membres concernant les arretes
666         int CalSMsurf_t; // pour les seconds membres concernant les surfaces
667         int CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
668         int CalSMRsurf; // pour les seconds membres concernant les surfaces
669         int CalSMvol_t; // pour les seconds membres concernant les volumes
670         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
671         int CalSMvol; // pour les seconds membres concernant les volumes
672         int CalDynamique; // pour le calcul de la matrice de masse
673         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
674         // ----- sauvegarde du nombre d'élément en cours -----
675         int nbelem_in_Prog;
676     };
677
678     // -----
679

```

```

680     protected :
681
682         // VARIABLES PRIVEES :
683
684         UneFois * unefois; // pointeur défini dans la classe dérivée
685         // les données spécifiques sont groupées dans une structure pour sécuriser
686         // le passage de paramètre dans init par exemple
687         class Donnee_specif
688         { public :
689             Donnee_specif() :
690                 epais(), use_epais_moyenne(1) {};
691             Donnee_specif(double epai, int nbi) :
692                 epais(nbi, Epai(epai, epai, epai)), use_epais_moyenne(1) {};
693             Donnee_specif(double epai0, double epai_t, double epai_tdt, int nbi) :
694                 epais(nbi, Epai(epai0, epai_t, epai_tdt)), use_epais_moyenne(1) {};
695             Donnee_specif(const Donnee_specif& a) :
696                 epais(a.epais), use_epais_moyenne(a.use_epais_moyenne) {};
697             Donnee_specif & operator = (const Donnee_specif& a)
698             { epais = a.epais; use_epais_moyenne = a.use_epais_moyenne;
699               return *this; };
700             // data
701             Tableau < Epai > epais; // épaisseur à chaque point d'intégration
702             int use_epais_moyenne; // = 1 : (val par défaut) c'est l'épaisseur moyenne qui est mise à
jour
703                                     // = 0 : c'est l'épaisseur à chaque pti qui est mise à jour
704         };
705         Donnee_specif donnee_specif;
706
707         // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
708         LesPtIntegMecaInterne lesPtMecaInt;
709
710         // type structuré et fonction virtuelle pour construire les éléments
711         // la fonction est défini dans le type dérivé
712         class NombresConstruire
713         { public :
714             NombresConstruire() : nbne(0), nbneA(0), nbi(0)
715                 , nbiEr(0), nbiS(0), nbiA(0), nbiMas(0), nbiHour(0) {};
716             int nbne; // le nombre de noeud de l'élément
717             int nbneA; // le nombre de noeud des aretes
718             int nbi; // le nombre de point d'intégration pour le calcul mécanique
719             int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
720             int nbiS; // le nombre de point d'intégration pour le calcul de second membre surfacique
721             int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
722             int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse
723             int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
724         };
725         NombresConstruire * nombre; // le pointeur défini dans la classe dérivée d'hexamemb
726
727         // =====> methodes appelees par les classes derivees <<=====
728
729         // fonction d'initialisation servant dans les classes derivant
730         // au niveau du constructeur, si rien initialisation par défaut
731         QuadraMemb::DonnComQuad* Init (Donnee_specif donnee_specif = Donnee_specif()
732                                     , bool sans_init_noeud = false);
733         // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
734         void Destruction();
735
736         // =====> methodes virtuelles derivant d'ElemMeca =====
737         // ramene la dimension des tenseurs contraintes et déformations de l'élément
738         int Dim_sig_eps() const {return 2;};
739
740         //----- fonctions uniquement a usage interne -----
741         private :
742         // definition des données communes : CoQuadra
743         QuadraMemb::DonnComQuad* Def_DonneeCommune();
744         // Calcul du residu local a t ou tdt en fonction du booleen
745         Vecteur* CalculResidu (bool atdt, const ParaAlgoControle & pa);
746         // calcul des seconds membres suivant les chargements
747         // cas d'un chargement volumique,
748         // force indique la force volumique appliquée
749         // retourne le second membre résultant
750         // ici on l'épaisseur de l'élément pour constituer le volume
751         // -> explicite à t ou tdt en fonction de la variable booleenne atdt
752         Vecteur SM_charge_volumique_E
753         (const Coordonnee& force, Fonction_nD* pt_fonct, bool atdt, const ParaAlgoControle & pa, bool
sur_volume_finale_);
754         // cas d'un chargement surfacique, sur les frontières des éléments
755         // force indique la force surfacique appliquée
756         // numface indique le numéro de la face chargée
757         // retourne le second membre résultant
758         // -> explicite à t ou tdt en fonction de la variable booleenne atdt
759         Vecteur SM_charge_surfacique_E
760         (const Coordonnee& force, Fonction_nD* pt_fonct, int numFace, bool atdt, const
ParaAlgoControle & pa);
761         // cas d'un chargement lineique, sur les aretes frontières des éléments
762         // force indique la force lineique appliquée
763         // numarete indique le numéro de l'arete chargée

```

```

764 // retourne le second membre résultant
765 // NB: il y a une définition par défaut pour les éléments qui n'ont pas
766 // d'arête externe -> message d'erreur d'où le virtuel et non virtuel pur
767 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
768 Vecteur SM_charge_lineique_E
769 (const Coordonnee& force, Fonction_nD* pt_fonct, int numArete, bool atdt, const
ParaAlgoControle & pa);
770 // cas d'un chargement lineique suivieuse, sur les aretes frontières des éléments 2D (uniquement)
771 // force indique la force lineique appliquée
772 // numarete indique le numéro de l'arête chargée
773 // retourne le second membre résultant
774 // -> explicite à t
775 Vecteur SM_charge_lineique_Suiv_E
776 (const Coordonnee& force, Fonction_nD* pt_fonct, int numArete, bool atdt, const
ParaAlgoControle & pa);
777 // cas d'un chargement de type pression, sur les frontières des éléments
778 // pression indique la pression appliquée
779 // numface indique le numéro de la face chargée
780 // retourne le second membre résultant
781 // NB: il y a une définition par défaut pour les éléments qui n'ont pas de
782 // surface externe -> message d'erreur d'où le virtuel et non virtuel pur
783 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
784 Vecteur SM_charge_pression_E
785 (double pression, Fonction_nD* pt_fonct, int numFace, bool atdt, const ParaAlgoControle
& pa);
786 // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
787 // presUniDir indique le vecteur appliquée
788 // numface indique le numéro de la face chargée
789 // retourne le second membre résultant
790 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
791 Vecteur SM_charge_presUniDir_E
792 (const Coordonnee& presUniDir, Fonction_nD* pt_fonct, int numFace, bool atdt, const
ParaAlgoControle & pa);
793 // cas d'un chargement surfacique hydrostatique,
794 // poidvol: indique le poids volumique du liquide
795 // M_liquide : un point de la surface libre
796 // dir_normal_liquide : direction normale à la surface libre
797 // retourne le second membre résultant
798 // -> explicite à t
799 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide, const double& poidvol
, int numFace, const Coordonnee& M_liquide, bool atdt
, const ParaAlgoControle & pa
, bool sans_limitation);
800 // cas d'un chargement surfacique hydro-dynamique,
801 // voir méthode explicite plus haut, pour les arguments
802 // retourne le second membre résultant
803 // bool atdt : permet de spécifier à t ou a t+tdt
804 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid, const double& poidvol
, CourbelD* coef_aero_n, int numFace, const double&
coef_mul
, CourbelD* coef_aero_t, bool atdt
, const ParaAlgoControle & pa);
805 // calcul de la nouvelle épaisseur moyenne finale (sans raideur)
806 // ramène l'épaisseur moyenne calculée à atdt ou t
807 // met à jour les épaisseurs aux pti en fonction de l'épaisseur moyenne calculée
808 // erreur = true en retour -> une erreur que l'on exploite en debug
809 const double CalEpaisseurMoyenne_et_transfert_pti(const bool atdt, bool& erreur);
810 };
811 /// @} // end of group
812 #endif
813
814
815
816
817
818
819
820
821
822

```

## 7.201 DeformationSfe1.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.

```

```

19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: Calcul des differentes grandeurs liee a la deformation
39 *   des elements sfel
40 *   La classe fonctionne comme une boite a outil. On y choisit ce
41 *   dont on a besoin. Bien faire attention a l'ordre d'appel des
42 *   differentes methodes lorsque il faut suivre une chronologie.
43 *   Cette classe calcul mais ne stock pas.
44 *
45 *   *****
46 *
47 *   VERIFICATION:
48 *   ! date !   auteur !           but
49 *   -----
50 *   !           !           !
51 *   *****
52 *
53 *   MODIFICATIONS:
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *****/
58 #ifndef DEFORMATIONSFEL_H
59 #define DEFORMATIONSFEL_H
60
61 #include "Tableau_T.h"
62 #include "Met_abstraite.h"
63 #include "Deformation.h"
64 #include "Met_Sfel.h"
65 #include "EnuTypeCL.h"
66 #include "Epai.h"
67
68 /// @addtogroup groupe_des_deformations
69 /// @{
70 ///
71
72
73 ///   BUT: Calcul des differentes grandeurs liee a la deformation
74 ///   des elements sfel
75 ///   La classe fonctionne comme une boite a outil. On y choisit ce
76 ///   dont on a besoin. Bien faire attention a l'ordre d'appel des
77 ///   differentes methodes lorsque il faut suivre une chronologie.
78 ///   Cette classe calcul mais ne stock pas.
79 ///
80 ///
81 /// \author   Gérard Rio
82 /// \version  1.0
83 /// \date    23/01/97
84
85 class DeformationSfel : public Deformation
86 {
87 public :
88
89
90     // CONSTRUCTEURS :
91     DeformationSfel () ; // par defaut ne doit pas etre utilise -> message
92     // d'erreur en phase de mise au point
93     // constructeur normal dans le cas d'un ou de plusieurs pt d'integration
94     // tabDphi et tabPhi sont relatifs aux fonctions d'interpolation
95     // la terminaison H est relative aux grandeurs d'épaisseur, S pour la surface
96     DeformationSfel (Met_abstraite & ,Tableau<Noeud *>& tabnoeud
97     ,Tableau <Mat_pleine> const & tabDphiH,Tableau <Vecteur> const & tabPhiH
98     ,Tableau <Mat_pleine> const & tabDphiS,Tableau <Vecteur> const & tabPhiS);
99     // constructeur de copie
100     DeformationSfel (const DeformationSfel &);
101     // DESTRUCTEUR :
102     virtual ~DeformationSfel ();
103
104     // METHODES PUBLIQUES :

```



```

105 //-----
106 // //  définition d'une classe conteneur spécifique à chaque point ou la déformation
107 // //  est calculée
108
109     class SaveDefResulSfel : public Deformation::SaveDefResul
110 { friend class Deformation ;friend class DeformationSfel;
111 public :
112     // Constructeurs
113     SaveDefResulSfel(); // constructeur par défaut-> à ne pas utiliser -> message d'erreur
114     SaveDefResulSfel(const Met_abstraite& meta); // le constructeur recevable
115     SaveDefResulSfel(const SaveDefResulSfel& a); // constructeur de copie
116     ~SaveDefResulSfel(); // destructeur
117
118     // surcharge de l'opérateur d'affectation
119     virtual SaveDefResul& operator= (const SaveDefResul& a); // surcharge d'affectation
120
121     // affichage des infos
122     virtual void Affiche();
123     // idem sur un ofstream
124     virtual void Affiche(ofstream& sort);
125     //===== lecture écriture dans base info =====
126     // cas donne le niveau de la récupération
127     // = 1 : on récupère tout
128     // = 2 : on récupère uniquement les données variables (supposées comme telles)
129     virtual void Lecture_base_info (ifstream& ent,const int cas);
130     // cas donne le niveau de sauvegarde
131     // = 1 : on sauvegarde tout
132     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
133     virtual void Ecriture_base_info(ofstream& sort,const int cas);
134
135     // mise à jour des informations transitoires en définitif, ou l'inverse
136     virtual void TdtversT() {meti_a_t = meti_a_tdt;};
137     virtual void TversTdt() {meti_a_tdt = meti_a_t;};
138     // mise à jour des grandeurs meti_00
139     virtual void MiseAJourGrandeurs_a_0(const Met_abstraite * metrique);
140     // mise à jour des grandeurs meti_tdt
141     virtual void MiseAJourGrandeurs_a_tdt(const Met_abstraite * metrique,const TenseurBB& Deps_BB);
142     // il n'y a pas de mise à jour des grandeurs à t, car celles-ci sont mise à jour à l'aide
143     // de la méthode TdtversT !!
144
145     // lecture des infos des métriques
146     const Deformation::Stmet& Meti_a_00() const {return meti_a_00;};
147     const Deformation::Stmet& Meti_a_t() const {return meti_a_t;};
148     const Deformation::Stmet& Meti_a_tdt() const {return meti_a_tdt;};
149
150     protected : // VARIABLES PROTEGEES :
151     // ici il s'agit des données de la facette, et dans SaveDefResul on met les infos
152     // correspondant aux métriques dans l'épaisseur
153     // en fait ne sert à rien d'une manière pratique !! si pour les calculs de variation d'épaisseur
154     !!! Deformation::Stmet meti_a_00,meti_a_t,meti_a_tdt; // les infos à 0 à t et à tdt
155     };
156
157     // création d'une instance de SaveDefResulSfel et initialisation.
158     virtual SaveDefResul * New_et_Initialise() {return (new SaveDefResulSfel(*metrique));};
159     // affichage des donnees particulieres a l'elements
160     // de matiere traite ( c-a-dire au pt calcule)
161     virtual void AfficheDataSpecif(ofstream& sort,SaveDefResul * a) const {a->Affiche(sort);};
162     // met à jour les données spécifiques du point considéré
163     void Mise_a_jour_data_specif(Deformation::SaveDefResul* don) {saveDefResul=don;};
164
165 // //  fin de définition relatif à la classe conteneur spécifique à chaque point
166 // //  ou la déformation est calculée
167 //-----
168 // ===== changement de grandeurs stockees =====
169
170 // définition du déformation du même type, permet d'utiliser des types dérivée surchargé
171 virtual Deformation * Nevez_deformation(Tableau <Noeud *> & tabN) const
172 { DeformationSfel* def = new DeformationSfel(*this);def->PointeurTableauNoeud(tabN);return def; } ;
173
174 // Surcharge de l'opérateur = : realise l'affectation
175 // fonction virtuelle
176 inline Deformation& operator= (const Deformation& def)
177 { return (Deformation::operator=(def));};
178
179 // on renseigne les conditions limites éventuelles
180 void RenseigneCondLim(const Tableau <EnuTypeCL> & arTypeCL,const Tableau <Coordonnee3>& vpla )
181 {tabTypeCL = &arTypeCL; vplan = &vpla;};
182 // ==== affichage ====
183 // affichage des informations
184 virtual void Affiche() const;
185
186 // ===== calcul des raideurs =====
187 // calcul explicit à t : tous les parametres sont des resultats
188 virtual const Met_abstraite::Expli& Cal_explicit_t
189 ( const Tableau <double>& def_equi_t,TenseurBB & epsBB_t,Tableau <TenseurBB *> &

```



```

d_epsBB
190     ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& DeltaEpsBB,bool
premier_calcul);
191 // calcul explicite à tdt : tous les parametres sont des resultats
192 virtual const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt
193     (const Tableau <double>& def_equi_t, TenseurBB & epsBB_tdt,Tableau <TenseurBB *> &
d_epsBB
194     ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB&
delta_epsBB_tdt,bool premier_calcul);
195 // cas implicite
196 const Met_abstraite::Impli& Cal_implicit
197     (const Tableau <double>& def_equi_t,TenseurBB & epsBB_tdt,Tableau <TenseurBB *> & d_epsBB
198     ,Tableau <double>& def_equi,TenseurBB& DepsBB,TenseurBB& delta_epsBB,bool premier_calcul);
199
200 // ----- calcul des variables primaires autre que pour la mécanique -----
201 // ----- donc par de retour relatif aux déformations
202 // calcul explicite à t : tous les parametres sont des resultats
203 virtual const Met_abstraite::Expli& Cal_explicit_t(bool premier_calcul);
204 // calcul explicite à tdt : tous les parametres sont des resultats
205 virtual const Met_abstraite::Expli_t_tdt& Cal_explicit_tdt(bool premier_calcul);
206 // cas implicite
207 const Met_abstraite::Impli& Cal_implicit(bool premier_calcul);
208 // ----- test sur la courbure -----
209 // test si la courbure est anormalement trop grande
210 // inf_normale : indique en entrée le det mini pour la courbure en locale
211 // retour 1: si tout est ok,
212 // 0: une courbure trop grande a été détecté
213 int Test_courbure_anormale(Enum_dure temps,double inf_normale)
214     {return ((Met_Sfel*) metrique)->Test_courbure_anormale(
215         temps,inf_normale,
*tabnoeud,(*tabDphi)(numInteg_surf),nbNoeud,(*tabPhi)(numInteg_surf));
216     };
217
218 // ===== remontee aux informations =====
219 // calcul :
220 // M0 : point d'integration numInteg a t = 0
221 // Mt ou Mtdt : point d'integration final
222 // Aa0 et Aafin : matrice de passage initiale et finale dans un repere ortho tel que
223 // la nouvelle base Aa est calculee par projection de "Ipa" sur Gi
224 // gijHH et gijBB : metrique finale
225 // Aa(i,a) = Aa^i_{.a}, avec g^i = Aa^i_{.a} * Ip^a
226 // tout ce passe comme si Ip^a est la nouvelle base vers laquelle on veut évoluer
227
228 // cas sortie d'un calcul implicite
229 virtual const Met_abstraite::InfoImp RemontImp(bool absolue,Mat_pleine& Aa0,Mat_pleine& Aafin);
230 // idem sans le calcul des matrices de passage
231 virtual const Met_abstraite::InfoImp RemontImp();
232 // cas sortie d'un calcul explicite à t
233 virtual const Met_abstraite::InfoExp_t RemontExp_t(bool absolue,Mat_pleine& Aa0,Mat_pleine& Aafin);
234 // idem sans le calcul des matrices de passage
235 virtual const Met_abstraite::InfoExp_t RemontExp_t();
236 // cas sortie d'un calcul explicite à tdt
237 virtual const Met_abstraite::InfoExp_tdt RemontExp_tdt(bool absolue,Mat_pleine& Aa0,Mat_pleine&
Aafin);
238 // idem sans le calcul des matrices de passage
239 virtual const Met_abstraite::InfoExp_tdt RemontExp_tdt();
240 // cas sortie d'un calcul à 0, t et tdt
241 virtual const Met_abstraite::Info0_t_tdt Remont0_t_tdt(bool absolue,Mat_pleine& Aa0,Mat_pleine&
Aat,Mat_pleine& Aatdt);
242 // idem sans le calcul des matrices de passage
243 virtual const Met_abstraite::Info0_t_tdt Remont0_t_tdt();
244 // récupération des infos relatives à la courbure pour 0, t et tdt
245 virtual const Met_Sfel::Courbure_t_tdt& RecupCourbure0_t_tdt()const
246     {return ((Met_Sfel*) metrique)->RecupCourbure();};
247
248 // change le numero d'integration courant
249 virtual void ChangeNumInteg(int ni);
250 // change les numeros d'integration de surface et d'epaisseur courant
251 // nisurf : nouveau point de surface
252 // niepaiss : nouveau point d'epaisseur
253 void ChangeNumIntegSfel(int nisurf, int niepaiss);
254 // gestion du parcours de tous les points d'integration
255 void PremierPtInteg();
256 bool DernierPtInteg();
257 void NevezPtInteg();
258 // méthode pour revenir au pti précédent
259 virtual void Retour_pti_precedant();
260
261 int Nb_pt_int_surf()const {return (tabPhi->Taille());};
262 int Nb_pt_int_epai()const {return (tabPhiH->Taille());};
263 // change l'épaisseur stockée, et préparation du stockage si nécessaire
264 void Change_epaisseur(const Epai& epaisse) {*epais=epaisse;};
265
266 // calcul de la position dans le repère absolu du point d'intégration
267 virtual const Coordonnee& Position_0() // à t=0
268     {return ((Met_Sfel*) metrique)->PointSfelM_0(*tabnoeud,(*tabPhi)(numInteg_surf)
269         ,(*tabDphi)(numInteg_surf),epais,(*tabPhiH)(numInteg_ep));};

```

```

270 virtual const Coordonnee& Position_t() // à t
271     {return ((Met_Sfel*) metrique)->PointSfelM_t(*tabnoeud,(*tabPhi)(numInteg_surf)
272         ,(*tabDphi)(numInteg_surf),epais,(*tabPhiH)(numInteg_ep));};
273 virtual const Coordonnee& Position_tdt() // à t+dt
274     {return ((Met_Sfel*) metrique)->PointSfelM_tdt(*tabnoeud,(*tabPhi)(numInteg_surf)
275         ,(*tabDphi)(numInteg_surf),epais,(*tabPhiH)(numInteg_ep));};
276 /* // calcul de la variation dans le repère absolu du point d'intégration
277 virtual const Tableau <Coordonnee>& Der_Pos_t() // à t
278     {return metrique->D_PointM_t(*tabnoeud,(*tabPhi)(numInteg));};
279 virtual const Tableau <Coordonnee>& Der_Pos_tdt() // à t+dt
280     {return metrique->D_PointM_tdt(*tabnoeud,(*tabPhi)(numInteg));}; */
281
282 // détermination des bases de passages (seules les données en entrées et sortie
283 // sont utilisées )
284 virtual void BasePassage(bool absolue,const BaseB & giB0,const BaseB & giB,const BaseH & giH0,
285     const BaseH & giH, Mat_pleine& Aa0,Mat_pleine& Aa0fin);
286 // cas où l'on veut les matrices de passages à 0 t et tdt
287 virtual void BasePassage(bool absolue,const BaseB & giB0,const BaseB & giB_t,const BaseB & giB_tdt
288     ,const BaseH & giH0,const BaseH & giH_t,const BaseH & giH_tdt
289     ,Mat_pleine& Aa0,Mat_pleine& Aa_t,Mat_pleine& Aa_tdt);
290
291 protected :
292     // VARIABLES PROTEGEES :
293
294     Tableau <Mat_pleine> const * tabDphiH; // derivees des fonctions d'interpolation suivant H
295     Tableau <Vecteur> const * tabPhiH; // les fonctions d'interpolation suivant H
296     // les tableaux tabDphi et tabPhi définis dans la classe mère Deformation, sont utilisés
297     // pour le stockage de l'interpolation suivant l'axe ou le plan.
298
299     Epai * epais; // != NULL s'il y a des épaisseurs courantes stocké à l'élément et donc
300     // relayées ici, sinon = NULL
301
302     // information concernant des conditions limites éventuelles, qui ont des répercussions sur
303     // le calcul de la métrique
304     Tableau <EnuTypeCL> const * tabTypeCL; // tabTypeCL(i) : si différent de RIEN_TYPE_CL, indique le
type de condition
305     // limite de l'arête i
306     Tableau <Coordonnee3> const * vplan; // util pour des conditions de symétrie ou d'encastrement
307     // si tabTypeCL(i) = TANGENTE_CL , alors vplan(i) contient
308     // un vecteur du plan normal à la tangente,
309     // l'arête donne un second vecteur
310
311
312 // +++++ des variables qui vont varier au cours du calcul: variables transitoires
313 int numInteg_ep; // numero du point d'integration en cours dans l'épaisseur
314 int numInteg_surf; // numero du point d'integration en cours sur la surface
315 // numInteg de la class mère, est égal à numinteg_surf
316 // méthode pour la gestion d'un changement momentané de pti
317 int sauve_numInteg_ep; /// sauvegarde pour un retour au pti précédant avec la méthode
Retour_pti_precedant();
318 int sauve_numInteg_surf;
319
320 // +++++ fin des variables qui vont varier au cours du calcul: variables transitoires
321
322 // METHODES PROTEGEES :
323 // --- méthodes de vérifications ---
324 // méthode de vérification par différences finies du calcul des différentes dérivées
325
326 void VerifCal_def(bool gradV_instantane,const Met_abstraite::Impli & ex,TenseurBB& epsBB_tdt
327     ,Tableau <TenseurBB *> & d_epsBB_tdt);
328 void VerifCal_implicit(bool gradV_instantane,const Met_abstraite::Impli & ex);
329 // indicateur utilisé par Verif_par différences finis
330 static int indic_VerifCal_implicitSfel;
331 };
332 /// @} // end of group
333
334 #endif

```

## 7.202 Met\_Sfel.h

```

1 // FICHER : Met_Sfel.h
2 // CLASSE : Met_Sfel
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio

```

```

15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *   DATE:           04/10/2007
35 *
36 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:        Herezh++
39 *
40 *
41 *   ****
42 *   BUT:   La classe Met_Sfel est une classe derivee de la classe
43 *          Met_abstraite et permet de realiser tous les calculs
44 *          particuliers lies a la metrique d'un element Sfe.
45 *          N.B. : La dimension de l'espace des points est 3,
46 *          et le nombre de vecteur est :
47 *          - uniquement 2, les 2 vecteurs de surfaces, s'il n'y
48 *          pas de degre de liberte d'epaisseur aux noeuds
49 *          - 3 = 2 vecteurs de surfaces + un vecteur parallele
50 *          a la normale : cas ou il
51 *          y a un degre de liberte d'epaisseur.
52 *
53 *   *****
54 *   VERIFICATION:
55 *
56 *   ! date ! auteur ! but
57 *   -----
58 *   ! ! !
59 *   *****
60 *   MODIFICATIONS:
61 *   ! date ! auteur ! but
62 *   -----
63 *   $
64 *   *****/
65
66
67 #ifndef MET_SFEL_H
68 #define MET_SFEL_H
69
70 #include "Met_abstraite.h"
71 #include "TabOper_T.h"
72 #include "Coordonnee3.h"
73 #include "Tenseur2.h"
74 #include "Enum_interpol.h"
75 #include "EnumTypeCL.h"
76 #include "Epai.h"
77 #include "Util.h"
78 #include "MathUtil.h"
79
80 /// @addtogroup groupe_des_metrrique
81 /// @{
82 ///
83
84
85 class Met_Sfel : public Met_abstraite
86 {
87
88
89     public :
90
91
92         // CONSTRUCTEUR :
93
94         // Constructeur par defaut
95         Met_Sfel ();
96
97         // constructeur permettant de dimensionner uniquement certaine variables
98         // dim = dimension de l'espace, ici forcement = 3
99         // nbvec = nb de vecteur des bases, tab = liste des variables a initialiser
100         Met_Sfel ( const DdlElement& tabddl,int nbvec
101                 ,const Tableau<Enum_variable_metrrique>& tabb, Enum_interpol enui
102                 ,int nb_noeud_central);

```

```

102     // constructeur de copie
103     Met_Sfel ( const Met_Sfel&);
104     // DESTRUCTEUR :
105
106     virtual ~Met_Sfel ();
107
108     // METHODES PUBLIQUES :
109
110     // Surcharge de l'operateur = : realise l'affectation
111     // dérivant de virtuel, a ne pas employer -> message d'erreur
112     Met_abstraite& operator= (const Met_abstraite& met);
113     // normale
114     virtual Met_Sfel& operator= (const Met_Sfel& met);
115
116     // affichage minimale des éléments de métrique de base
117     virtual void Affiche() const;
118
119     // attribution de l'indicateur de type de calcul de jacobien:
120     // =1: le jacobien = celui de la facette médiane sans courbure
121     // =2: le jacobien = celui de la facette médiane + la courbure actuelle
122     void ChangeTypeCalculJacobien(int nouveau_type);
123
124     //=====
125     // définition de différentes classes conteneurs, utilisés pour ramener les résultats après calcul |
126     // ici spécifique au cas des éléments sfe (en plus des conteneurs déjà définis dans Met_abstraite.h)
127     #include "Met_Sfel_struct_donnees.h"
128     //=====
129
130     // ----- calculs -----
131     // cas explicite à t, toutes les grandeurs sont a 0 ou t
132     // calcul des termes : gijBB_0, gijBB_t, gijHH_t, d_gijBB_t, jacobien
133     // il y a l'interpolation d'épaisseur en plus par rapport à la fonction originale ds
134     met_abstraite
135     // gradV_instantane : true : calcul du gradient de vitesse instantannée
136     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
137     // false: uniquement les grandeurs à t+dt sont calculées
138     const Met_abstraite::Expli& CalSfel_explicit_t( const Tableau<Noeud *>& tab_noeud, bool
139     gradV_instantane
140     ,Mat_pleine const & tabdphiS, int nombre_noeud, Vecteur const & tabphiS
141     ,bool premier_calcul, Mat_pleine const & tabdphiH, Vecteur const& tabphiH, const Epai*
142     epais
143     ,Tableau <EnumTypeCL> const & tabTypeCL, Tableau <Coordonnee3> const & vplan);
144     // calcul simplifié, a utiliser lorsque l'on se sert des grandeurs
145     // liers a la facette deja calculee, ainsi que la courbure
146     // gradV_instantane : true : calcul du gradient de vitesse instantannée
147     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
148     // false: uniquement les grandeurs à t+dt sont calculées
149     const Met_abstraite::Expli& CalSfel_explicit_simple_t( const Tableau<Noeud *>& tab_noeud, bool
150     gradV_instantane
151     ,Mat_pleine const & tabdphiS, int nombre_noeud, Vecteur const & tabphiS
152     ,bool premier_calcul, Mat_pleine const & tabdphiH, Vecteur const& tabphiH, const Epai*
153     epais);
154     // cas explicite à tdt, toutes les grandeurs sont a 0 ou t
155     // calcul des termes : gijBB_0, gijBB_t, gijHH_t, d_gijBB_t, jacobien
156     // il y a un parametre de plus (phi) que dans la fonction originale ds met_abstraite
157     // gradV_instantane : true : calcul du gradient de vitesse instantannée
158     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
159     // false: uniquement les grandeurs à t+dt sont calculées
160     const Met_abstraite::Expli_t_tdt& CalSfel_explicit_tdt( const Tableau<Noeud *>& tab_noeud, bool
161     gradV_instantane
162     ,Mat_pleine const & tabdphiS, int nombre_noeud, Vecteur const & tabphiS
163     ,bool premier_calcul, Mat_pleine const & tabdphiH, Vecteur const& tabphiH, const Epai*
164     epais
165     ,Tableau <EnumTypeCL> const & tabTypeCL, Tableau <Coordonnee3> const & vplan);
166     // calcul simplifié, a utiliser lorsque l'on se sert des grandeurs
167     // liers a la facette deja calculee, ainsi que la courbure
168     // gradV_instantane : true : calcul du gradient de vitesse instantannée
169     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
170     // false: uniquement les grandeurs à t+dt sont calculées
171     const Met_abstraite::Expli_t_tdt& CalSfel_explicit_simple_tdt( const Tableau<Noeud *>&
172     tab_noeud, bool gradV_instantane
173     ,Mat_pleine const & tabdphiS, int nombre_noeud, Vecteur const & tabphiS
174     ,bool premier_calcul, Mat_pleine const & tabdphiH, Vecteur const& tabphiH, const Epai*
175     epais);
176     // récup du conteneur Expli_t_dtFac : c'est-à-dire des infos de la FACETTE !!!
177     virtual const Expli_t_tdt& Conteneur_Expli_tdtFac() const {return ex_expli_t_tdtFac;};
178     // cas implicite
179     // calcul des termes de la classe impli
180     // gradV_instantane : true : calcul du gradient de vitesse instantannée
181     // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculées
182     // false: uniquement les grandeurs à t+dt sont calculées
183     const Met_abstraite::Impli& CalSfel_implicit( const Tableau<Noeud *>& tab_noeud, bool
184     gradV_instantane
185     ,Mat_pleine const & tabdphiS, int nombre_noeud, Vecteur const & tabphiS
186     ,bool premier_calcul, Mat_pleine const & tabdphiH, Vecteur const& tabphiH, const Epai*
187     epais
188     ,Tableau <EnumTypeCL> const & tabTypeCL, Tableau <Coordonnee3> const & vplan);

```

```

178 // calcul simplifie, a utiliser lorsque l'on se sert des grandeurs
179 // liers a la facette deja calculee, ainsi que la courbure et sa variation
180 // gradV_instantane : true : calcul du gradient de vitesse instantanee
181 // premier_calcul : true : cas d'un premier calcul -> toutes les grandeurs sont calculees
182 // false: uniquement les grandeurs a t+dt sont calculees
183 const Met_abstraite::Impli& CalSfel_implicit_simple( const Tableau<Noeud *>& tab_noeud,bool
gradV_instantane
184 ,Mat_pleine const & tabdphiS,int nombre_noeud,Vecteur const & tabphiS
185 ,bool premier_calcul,Mat_pleine const & tabdphiH,Vecteur const& tabphiH,const Epai*
epais);
186
187 // ----- cas de la dynamique -----
188 // calcul pour la matrice masse
189 virtual const Dynamiq& Cal_pourMatMass(const Tableau<Noeud *>& tab_noeud, const Mat_pleine&
tabDphi
190 ,int nombre_noeud,const Vecteur& phi);
191 // ----- test sur la courbure -----
192 // test si la courbure est anormalement trop grande
193 // inf_normale : indique en entree le det mini pour la courbure en locale
194 // retour 1: si tout est ok,
195 // 0: une courbure trop grande a ete detecte
196 int Test_courbure_anormale(Enum_dure temps,double inf_normale, const Tableau<Noeud *>& tab_noeud
197 ,const Mat_pleine& dphiS,int nombre_noeud,const Vecteur& phiS);
198
199 // ----- remontee aux informations -----
200
201 // cas sortie d'infoImp
202 // calcul des termes de la classe InfoImp
203 const Met_abstraite::InfoImp& CalSfel_InfoImp( const Tableau<Noeud *>& tab_noeud
204 ,Mat_pleine const & tabdphiS,int nombre_noeud,Vecteur const & tabphiS
205 ,Mat_pleine const & tabdphiH,Vecteur const & tabphiH,const Epai* epais
206 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
207 // cas sortie d'infoExp
208 // calcul des termes de la classe InfoExp_t
209 const Met_abstraite::InfoExp_t& CalSfel_InfoExp_t( const Tableau<Noeud *>& tab_noeud
210 ,Mat_pleine const & tabdphiS,int nombre_noeud,Vecteur const & tabphiS
211 ,Mat_pleine const & tabdphiH,Vecteur const & tabphiH,const Epai* epais
212 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
213 // calcul des termes de la classe InfoExp_tdt
214 const Met_abstraite::InfoExp_tdt& CalSfel_InfoExp_tdt( const Tableau<Noeud *>& tab_noeud
215 ,Mat_pleine const & tabdphiS,int nombre_noeud,Vecteur const & tabphiS
216 ,Mat_pleine const & tabdphiH,Vecteur const & tabphiH,const Epai* epais
217 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
218 // calcul des termes de la classe Info0_t_tdt
219 virtual const Info0_t_tdt& CalSfel_Info0_t_tdt( const Tableau<Noeud *>& tab_noeud
220 ,Mat_pleine const & tabdphiS,int nombre_noeud,Vecteur const & tabphiS
221 ,Mat_pleine const & tabdphiH,Vecteur const & tabphiH,const Epai* epais
222 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
223
224 // cas du calcul de la deformation d'Almansi uniquement
225 virtual const Pour_def_Almansi CalSfel_pour_def_Almansi_au_temps
226 (Enum_dure temps,const Tableau<Noeud *>& tab_noeud
227 ,Mat_pleine const & tabdphiS,int nombre_noeud,Vecteur const & tabphiS
228 ,Mat_pleine const & tabdphiH,Vecteur const & tabphiH,const Epai* epais
229 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
230 // cas du calcul de la deformation logarithmique uniquement
231 virtual const Pour_def_log CalSfel_pour_def_log_au_temps
232 (Enum_dure temps,const Tableau<Noeud *>& tab_noeud
233 ,Mat_pleine const & tabdphiS,int nombre_noeud,Vecteur const & tabphiS
234 ,Mat_pleine const & tabdphiH,Vecteur const & tabphiH,const Epai* epais
235 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
236
237 // ===== utilise par le contact et autres =====
238 // calcul d'un point M en fonction des phi et des coordonnees a 0
239 virtual const Coordonnee & PointSfelM_0 // ( stockage a t=0)
240 (const Tableau<Noeud *>& tab_noeud, const Vecteur& phiS,Mat_pleine const & dphiS
241 ,const Epai* epais, const Vecteur& phiH);
242
243 // calcul d'un point M en fonction des phi et des coordonnees a t
244 virtual const Coordonnee & PointSfelM_t // ( stockage a t=t)
245 (const Tableau<Noeud *>& tab_noeud, const Vecteur& phiS,Mat_pleine const & dphiS
246 ,const Epai* epais, const Vecteur& phiH);
247 /*
248 // calcul de la variation d'un point M par rapport aux ddl ,
249 // en fonction des phi et des coordonnees a t
250 virtual const Tableau<Coordonnee> & D_PointSfelM_t // ( stockage a t)
251 ( const Tableau<Noeud *>& tab_noeud, const Vecteur& phiS,Mat_pleine const & dphiS
252 ,const double& epais_t, const Vecteur& phiH)
253 */
254 // calcul d'un point M en fonction des phi et des coordonnees a tdt
255 virtual const Coordonnee & PointSfelM_tdt // ( stockage a tdt)
256 (const Tableau<Noeud *>& tab_noeud, const Vecteur& phiS,Mat_pleine const & dphiS
257 ,const Epai* epais, const Vecteur& phiH);
258 /*
259 // calcul de la variation d'un point M par rapport aux ddl ,
260 // en fonction des phi et des coordonnees a tdt
261 virtual const Tableau<Coordonnee> & D_PointSfelM_tdt // ( stockage a tdt)

```

```

262         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
263
264     // calcul de la vitesse du point M en fonction de phi et des coordonnees a 0
265     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
266     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
267     virtual const Coordonnee & VitesseSfelM_0 // ( stockage a t=0)
268         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
269     // idem mais au noeud passé en paramètre
270     virtual const Coordonnee & VitesseSfelM_0 (const Noeud* noeud);
271
272     // calcul de la vitesse du point M en fonction de phi et des coordonnees a t
273     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
274     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
275     virtual const Coordonnee & VitesseSfelM_t // ( stockage a t=t)
276         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
277     // idem mais au noeud passé en paramètre
278     virtual const Coordonnee & VitesseSfelM_t (const Noeud* noeud);
279
280     // calcul de la variation de la vitesse du point M par rapport aux ddl ,
281     // en fonction de phi et des coordonnees a t
282     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
283     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
284     // ddl_vitesse : indique si oui ou non il s'agit de variation par rapport
285     // aux ddl de vitesse ou au ddl de position
286     virtual const Tableau<Coordonnee> & D_VitesseSfelM_t // ( stockage a t)
287         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi,bool ddl_vitesse);
288     // idem mais au noeud passé en paramètre
289     virtual const Tableau<Coordonnee> & D_VitesseSfelM_t (const Noeud* noeud,bool ddl_vitesse);
290
291     // calcul de la vitesse du point M en fonction de phi et des coordonnees a tdt
292     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
293     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
294     // ddl_vitesse : indique si oui ou non il s'agit de variation par rapport
295     // aux ddl de vitesse ou au ddl de position
296     virtual const Coordonnee & VitesseSfelM_tdt // ( stockage a tdt)
297         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);
298     // idem mais au noeud passé en paramètre
299     virtual const Coordonnee & VitesseSfelM_tdt (const Noeud* noeud);
300
301     // calcul de la variation de la vitesse du point M par rapport aux ddl ,
302     // en fonction de phi et des coordonnees a tdt
303     // dans le cas où les ddl de vitesse existent, ils sont directement interpolés
304     // dans le cas où ils n'existent pas, on utilise les vitesses moyennes : delta M / delta t
305     // ddl_vitesse : indique si oui ou non il s'agit de variation par rapport
306     // aux ddl de vitesse ou au ddl de position
307     virtual const Tableau<Coordonnee> & D_VitesseM_tdt // ( stockage a tdt)
308         (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi,bool ddl_vitesse);
309     // idem mais au noeud passé en paramètre
310     virtual const Tableau<Coordonnee> & D_VitesseSfelM_tdt (const Noeud* noeud,bool ddl_vitesse);
311
312     // calcul de la base naturel , au point correspondant au dphi en fonction des coord a t=0
313     virtual const BaseB& BaseSfelNat_0 // ( stockage a t=0)
314         (const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,const Vecteur& phi);
315
316     // calcul de la base naturel , au point correspondant au dphi en fonction des coord a t
317     virtual const BaseB& BaseSfelNat_t // ( stockage a t)
318         (const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,const Vecteur& phi);
319
320     // calcul de la base naturel , au point correspondant au dphi en fonction des coord a tdt
321     virtual const BaseB& BaseSfelNat_tdt // ( stockage a tdt)
322         (const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,const Vecteur& phi);
323
324     // calcul de la base naturelle et duale ( stockage a t=0) en fonction des coord a 0
325     virtual void BaseSfelND_0(const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,
326                             const Vecteur& phi,BaseB& bB,BaseH& bH);
327
328     // calcul de la base naturelle et duale ( stockage a t) en fonction des coord a t
329     virtual void BaseSfelND_t(const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,
330                             const Vecteur& phi,BaseB& bB,BaseH& bH);
331
332     // calcul de la base naturelle et duale ( stockage a tdt) en fonction des coord a tdt
333     virtual void BaseSfelND_tdt(const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi,
334                             const Vecteur& phi,BaseB& bB,BaseH& bH);
335
336     // ===== informations diverses (mais cohérentes, c-a-d, peuvent être appelées directement sans
337     // init) =====
338     double JacobienInitial(const Tableau<Noeud *>& tab_noeud,const Mat_pleine& tabDphi
339                             ,int nombre_noeud,const Vecteur& phi);
340 */
341     // ----- gestion de la memoire -----
342
343     // Ajout d'initialisation de differentes variables : liste dans tab
344     void PlusInitVariables(Tableau<Enum_variable_métrique>& tab) ;
345
346     // ----- récupération particulière à la courbure -----
347     // doit être utilisé après une méthode qui calcul toutes les grandeurs en questions !!!!!
348     const Met_Sfel::Courbure_t_tdt& RecupCourbure () const {return courbure_t_tdt;};

```

```

348
349 // ===== protege =====
350
351 protected :
352 // METHODES protegees:
353 // calcul des normales a la facette
354 void Calcul_N_0();
355 void Calcul_N_t();
356 void Calcul_N_tdt();
357
358 //== calcul des variations de la normales
359 void Cal_d_N_t(); // il faut avant le calcul des ap_alpha_t et de épaisseur à t
360 void Cal_d_N_tdt(); // il faut avant le calcul des ap_alpha_tdt et de épaisseur à tdt
361
362 //== calcul de N,alpha (sur la surface courbe) pour les différents temps
363 void Cal_N_alpha_0()
364     {N_alpha_0_1.ConstructionAPartirDe_H( -(curb_0(1,1) * (*aiH_0)(1) + curb_0(1,2) * (*aiH_0)(2)));
365
366     N_alpha_0_2.ConstructionAPartirDe_H( -(curb_0(2,1) * (*aiH_0)(1) + curb_0(2,2) *
367     (*aiH_0)(2)));};
368 void Cal_N_alpha_t()
369     {N_alpha_t_1.ConstructionAPartirDe_H( -(curb_t(1,1) * (*aiH_t)(1) + curb_t(1,2) * (*aiH_t)(2)));
370
371     N_alpha_t_2.ConstructionAPartirDe_H( -(curb_t(2,1) * (*aiH_t)(1) + curb_t(2,2) *
372     (*aiH_t)(2)));};
373 void Cal_N_alpha_tdt()
374     {N_alpha_tdt_1.ConstructionAPartirDe_H( -(curb_tdt(1,1) * (*aiH_tdt)(1) + curb_tdt(1,2) *
375     (*aiH_tdt)(2)));
376
377     N_alpha_tdt_2.ConstructionAPartirDe_H( -(curb_tdt(2,1) * (*aiH_tdt)(1) + curb_tdt(2,2) *
378     (*aiH_tdt)(2)));};
379
380 //== calcul de la variation de N,alpha
381 void Cal_d_N_alpha_t(); // il faut la courbure, la variation de la courbure, aH et d_aH
382 void Cal_d_N_alpha_tdt();
383
384 //==calcul des points, identique a Met_abstraite
385 // calcul du point a t0
386 void Calcul_M0 ( const Tableau<Noeud *>& , const Vecteur& phi, int );
387 void Calcul_Mt ( const Tableau<Noeud *>& , const Vecteur& phi, int );
388 void Calcul_Mtdt ( const Tableau<Noeud *>& , const Vecteur& phi, int );
389
390 //== le nombre de vecteur de base pour les Sfel est soit 2: cas historique, sans ddl d'épaisseur
391 // soit 3: cas avec ddl d'épaisseurs à chaque noeud
392 //== certaines fonctions de calcul de base sont donc redefini
393 // -- calcul des bases médianes : tout d'abord la partie a_alpha
394 void Calcul_a_alpha_B_0(const Tableau<Noeud *>& tab_noeud
395     , const Mat_pleine& tabDphi, int nombre_noeud,const Vecteur& phi);
396 void Calcul_a_alpha_B_t(const Tableau<Noeud *>& tab_noeud
397     , const Mat_pleine& tabDphi, int nombre_noeud,const Vecteur& phi);
398 void Calcul_a_alpha_B_tdt(const Tableau<Noeud *>& tab_noeud
399     , const Mat_pleine& tabDphi, int nombre_noeud,const Vecteur& phi);
400 // -- calcul du vecteur a3 des bases médianes
401 // nécessite auparavant d'avoir calculé la normale et l'épaisseur
402 void Calcul_a_3_B_0() {(aiB_0)->CoordoB(3) = N_0 * (epais.epaisseur0 * 0.5)};
403 void Calcul_a_3_B_t() {(aiB_t)->CoordoB(3) = N_t * (epais.epaisseur_t * 0.5)};
404 void Calcul_a_3_B_tdt() {(aiB_tdt)->CoordoB(3) = N_tdt * (epais.epaisseur_tdt * 0.5)};
405
406 //== calcul de la variation des bases sur la surface médiane
407 // ici il ne s'agit uniquement que de la variation de a_alpha et pas a_3
408 void D_a_alpha_B_t( const Mat_pleine& tabDphi,
409     int nbnoeu,const Vecteur & phi); // avant calcul de : a_alpha_B_t
410 void D_a_alpha_B_tdt( const Mat_pleine& tabDphi,
411     int nbnoeu,const Vecteur & phi); // avant calcul de : a_alpha_B_tdt
412 //== calcul de la variation de a_3_B
413 void D_a_3_B_t();
414 void D_a_3_B_tdt();
415
416 // -- calcul des bases dans l'épaisseur
417 // calcul de la base naturel a t0
418 // pendant le traitement il y a calcul de la base aiB et aiH de la facette plane, et éventuellement
419 // les 3ième vecteur (proportionnelle à la normale) s'il y a un ddl d'épaisseur
420 // la fonction contient un argument de plus que la routine dans met_abstraite
421 void Calcul_giB_0 ( const Tableau<Noeud *>& , const Mat_pleine& ,
422     int, const Vecteur& phi);
423 // calcul de la base naturel a t
424 void Calcul_giB_t ( const Tableau<Noeud *>& , const Mat_pleine& ,
425     int, const Vecteur& phi);
426 // calcul de la base naturel a t+dt
427 void Calcul_giB_tdt ( const Tableau<Noeud *>& , const Mat_pleine& ,
428     int, const Vecteur& phi);
429 //== calcul de la variation des bases
430 void D_giB_t( const Mat_pleine& , int , const Vecteur & phi);
431 void D_giB_tdt( const Mat_pleine& , int , const Vecteur & phi);
432
433 // cas des épaisseurs interpolées
434 void Calcul_epaisseur_0(const Tableau<Noeud *>& tab_noeud, const Vecteur& phiS);

```



```

429 void Calcul_epaisseur_t(const Tableau<Noeud *>& tab_noeud, const Vecteur& phiS);
430 void Calcul_epaisseur_tdt(const Tableau<Noeud *>& tab_noeud, const Vecteur& phiS);
431 // calcul de la variation de l'épaisseur par rapport aux ddl
432 void D_epaisseur_t(const Vecteur& phiS);
433 void D_epaisseur_tdt(const Vecteur& phiS);
434
435 //-----// calcul du tenseur de courbure dans la base naturelle
436 // on utilise un tenseur non symétrique, bien qu'en théorie la courbure est symétrique
437 // suivant les différentes méthodes de calcul, le tenseur en sortie est symétrique ou pas
438 // on peut toujours le symétriser, si l'on veut un tenseur symétrique
439 // on a :  $N_t a_i = -b_{i|be} \cdot a^{be}$ 
440 // plusieurs cas sont étudiés suivant l'instant considéré
441 // a l'instant  $t = 0$ 
442 Tenseur_ns2BB& courbure_0 ( const Tableau<Noeud *>& tab_noeud
443                             ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const &
vplan);
444 // a l'instant  $t$ 
445 Tenseur_ns2BB& courbure_t ( const Tableau<Noeud *>& tab_noeud
446                             ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const &
vplan);
447 // a l'instant  $t+dt$ 
448 Tenseur_ns2BB& courbure_tdt ( const Tableau<Noeud *>& tab_noeud
449                               ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const &
vplan);
450 //-----// calcul du tenseur de courbure et de sa variation
451 // plusieurs cas sont étudiés suivant l'instant considéré
452 // a l'instant  $t$ 
453 void Dcourbure_t( const Tableau<Noeud *>& tab_noeud,Tenseur_ns2BB& curb,TabOper<Tenseur_ns2BB>&
dcurb
454                  ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
455 // a l'instant  $tdt$ 
456 void Dcourbure_tdt( const Tableau<Noeud *>& tab_noeud,Tenseur_ns2BB& curb,TabOper<Tenseur_ns2BB>&
dcurb
457                    ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
458
459 // DONNEES PROTEGEES
460
461 Tableau <Coordonnee3> Ia; // base absolue
462 // calcul de la courbure et de sa variation éventuelle
463 Tenseur_ns2BB curb_0,curb_t,curb_tdt;
464 TabOper<Tenseur_ns2BB> dcurb_t,dcurb_tdt;
465 // calcul de la normale
466 Coordonnee3B N_0,N_t,N_tdt;
467 double norme_N_t, norme_N_tdt; // norme des normales
468 // variation de  $N_t$  et  $N_tdt$ 
469 Tableau < CoordonneeB > * d_N_t, * d_N_tdt;
470 //N,alpha pour les différents temps
471 CoordonneeB N_alpha_0_1,N_alpha_0_2,N_alpha_t_1,N_alpha_t_2,N_alpha_tdt_1,N_alpha_tdt_2;
472 // variation de  $N,alpha$ 
473 Tableau < CoordonneeB > * d_N_alpha_t_1, * d_N_alpha_t_2; // variation par rapport aux  $\xi$  de  $N(t),1$ 
et ,2
474 Tableau < CoordonneeB > * d_N_alpha_tdt_1, * d_N_alpha_tdt_2; // idem pour  $N(tdt)$ 
475
476 int nbNoeudCentral; // nombre de noeud de l'élément central
477 bool tCalEpais; // indique le type de prise en compte de l'épaisseur
478 // = 0 : épaisseur constante (n'est pas un ddl) passée en paramètre
479 // = 1 : épaisseur = un ddl stocké au noeud
480 // épaisseurs : soit contient les épaisseurs passées en paramètres d'entrée (tCalEpais = 0)
481 // soit contient les épaisseurs données interpolées aux noeuds (tCalEpais = 1)
482 Epai epais; // le conteneur d'épaisseur à 0, t et tdt
483 Tableau <Epai> * d_epais; // variation des épaisseurs
484 // indicateur de type de calcul de jacobien finale :
485 // =1: le jacobien = celui de la facette médiane sans courbure
486 // =2: le jacobien = celui de la facette médiane + la courbure actuelle
487 // =3: le jacobien = celui du volume = facette * épaisseur, cas d'une métrique 3D
488 int type_calcul_jacobien;
489
490 //il s'agit ici des infos concernant l'élément centrale, et sur la surface médiane
491
492 Coordonnee3 * P0; // point a t=0
493 Coordonnee3 * Pt ; // point a l'instant t
494 Coordonnee3 * Ptdt; // point a l'instant t+dt BaseB Ia; // la base absolue
495 Tableau <Coordonnee3> * d_Pt ; // dérivées au point a l'instant t, par rapport aux ddl
496 Tableau <Coordonnee3> * d_Ptdt; // dérivées au point a l'instant t+dt, par rapport aux ddl
497
498 BaseB * aiB_0 ; // base naturelle a t=0
499 BaseB * aiB_t ; // base naturelle a t
500 BaseB * aiB_tdt ; // base naturelle a t+dt
501
502 BaseH * aiH_0; // vecteur de la base duale a 0
503 BaseH * aiH_t; // vecteur de la base duale a t
504 BaseH * aiH_tdt; // vecteur de la base duale a t+dt
505
506 TenseurBB * aiBB_0; // composantes de la metrique de la base naturelle a t
507 TenseurBB * aiBB_t; // composantes de la metrique de la base naturelle a t
508 TenseurBB * aiBB_tdt; // composantes de la metrique de la base naturelle a tdt
509 TenseurHH * aiHH_0; // composantes de la metrique de la base duale a t=0

```



```

510 TenseurHH * aijHH_t; // composantes de la metrique de la base duale a t
511 TenseurHH * aijHH_tdt; // composantes de la metrique de la base duale a t+dt
512
513 double ajacobien_0; // jacobien de la facette centrale sans courbure a l'instant t=0
514 double ajacobien_t; // jacobien de la facette centrale sans courbure a l'instant t
515 double ajacobien_tdt; // jacobien de la facette centrale sans courbure a l'instant t+dt
516
517 Tableau <BaseB> * d_aiB_t; // derivees de la base naturelle par rapport
518 // aux degres de liberte a t
519 Tableau <BaseB> * d_aiB_tdt; // derivees de la base naturelle par rapport
520 // aux degres de liberte a t+dt
521
522 Tableau <BaseH> * d_aiH_t; // derivees de la base duale par rapport
523 // aux degres de liberte a t
524 Tableau <BaseH> * d_aiH_tdt; // derivees de la base duale par rapport
525 // aux degres de liberte a t+dt
526
527 Tableau <TenseurBB * > d_aijBB_t; // derivees de la metrique de la base naturelle
528 // a t par rapport aux degres de liberte
529 Tableau <TenseurBB * > d_aijBB_tdt; // derivees de la metrique de la base naturelle
530 // a t+dt par rapport aux degres de liberte
531 Tableau <TenseurHH * > d_aijHH_t; // derivees de la metrique de la base duale a t
532 Tableau <TenseurHH * > d_aijHH_tdt; // derivees de la metrique de la base duale a t+dt
533
534 Vecteur d_ajacobien_t ; // derivees a t du jacobien de la facette centrale sans courbure
535 // par rapport aux degres de liberte
536 Vecteur d_ajacobien_tdt ; // derivees a tdt du jacobien de la facette centrale sans courbure
537 // par rapport aux degres de liberte
538
539 // maintenant definition des differents conteneurs de retour des methodes globales
540 // **** ici il s'agit de contenir les infos liees a la facette !!!! *****
541 // maintenant definition des differents conteneurs de retour des methodes globales
542 // contrairement au cas de metabstraite
543 Courbure_t_tdt courbure_t_tdt; // cas du calcul de la courbure
544 Impli ex_impliFac; // cas d'un calcul implicite
545 Expli ex_expliFac; // calcul explicite 0 t
546 Expli_t_tdt ex_expli_t_tdtFac; // calcul explicite 0 t tdt
547 giJHH_0_et_giH_0 ex_aijHH_0_et_aiH_0Fac; // comme le nom l'indique !
548 Dynamiq ex_DynamiqFac; // calcul pour la dynamique (partie specifique a quelques grandeurs)
549 flambe_lin ex_flambe_linFac; // grandeurs pour le flambement lineaire
550 Umat_cont umat_contFac; // calcul relatif aux Umat
551 InfoImp ex_InfoImpFac; // sortie d'info en implicite
552 InfoExp_t ex_InfoExp_tFac; // sortie d'info en explicite a t
553 InfoExp_tdt ex_InfoExp_tdtFac; // sortie d'info en explicite a tdt
554 Info0_t_tdt ex_Info0_t_tdtFac; // sortie d'info a 0 t et tdt
555
556 // quelques variables statiques
557 static Tableau <int> indi; // pour de l'indication indirecte
558 // indicateur utilise par Verif_Dcourbure
559 static int indic_Verif_DcourbureSFE;
560
561 // pointeurs des fonctions en cours
562 Tenseur_ns2BB (Met_Sfel::*PtCourbure) ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB ,
const BaseH & aiH
563 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const &
vplan);
564 void (Met_Sfel::*PtDcourbure) ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB
565 , const Tableau <BaseB>& DaiB,const BaseH & aiH
566 , const Tableau <BaseH>& DaiH,Tenseur_ns2BB&
curb,TabOper <Tenseur_ns2BB>& dcurb
567 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const &
vplan);
568 // test si la courbure est anormalement trop grande
569 // inf_normale : indique en entree le det mini pour la courbure en locale
570 // retour 1: si tout est ok,
571 // 0: une courbure trop grande a ete detecte
572 int (Met_Sfel::*PtTest_courbure_anormale)(double inf_normale,const Tableau<Coordonnee3>& tab_coor
573 , const BaseB & aiB , const BaseH & aiH);
574
575 // fonctions privees
576 // allocation de la memoire
577 void AllocSfel ();
578 // deallocation de la memoire complete, uniquement des grandeurs specifiques
579 // a Met_Sfel
580 void DeallocSfel();
581 // uniquement des grandeurs de tib
582 void DeallocSfel(Tableau<Enum_variable_metrrique>& tib);
583 // copie en fonction de l'instanceensee
584 // concerne que les grandeurs pointees
585 void Copie_metSfel(const Met_Sfel& a);
586 // ----- routines internes pour calculer les differentes formes de courbures
587 // routines (1) generales de calcul de la courbure
588 // cas 1: calcul des 3 courbures suivant les 3 cotes
589 Tenseur_ns2BB Courbure1
590 ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB , const BaseH & aiH
591 ,Tableau <EnumTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
592 // routine generale de calcul de la courbure et de sa variation

```

```

593 // en sortie : curb , la courbure b11,b12,b22
594 //          dcurb , la variation de courbure.
595 // cas 1: calcul des 3 courbures suivant les 3 cotés
596 void Dcourbure1 ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB
597                 , const Tableau <BaseB>& DaiB,const BaseH & aiH
598                 , const Tableau <BaseH>& DaiH,Tenseur_ns2BB& curb,TabOper <Tenseur_ns2BB>& dcurb
599                 ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
600 // test si la courbure est anormalement trop grande
601 // inf_normale : indique en entrée le det mini pour la courbure en locale
602 // retour 1: si tout est ok,
603 //          0: une courbure trop grande a été détecté
604 int Test_courbure_anormale1(double inf_normale,const Tableau<Coordonnee3>& tab_coor
605                             , const BaseB & aiB , const BaseH & aiH);
606
607 // routine (2) generale de calcul de la courbure
608 // cas 2: moyenne des normales de part et autres des arrêtes
609 Tenseur_ns2BB Courbure2
610 ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB , const BaseH & aiH
611   ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
612 // routine generale de calcul de la courbure et de sa variation
613 // en sortie : curb , la courbure b11,b12,b22
614 //          dcurb , la variation de courbure.
615 // cas 2: moyenne des normales de part et autres des arrêtes
616 void Dcourbure2 ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB
617                 , const Tableau <BaseB>& DaiB,const BaseH & aiH
618                 , const Tableau <BaseH>& DaiH,Tenseur_ns2BB& curb,TabOper <Tenseur_ns2BB>& dcurb
619                 ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
620 // test si la courbure est anormalement trop grande
621 // inf_normale : indique en entrée le det mini pour la courbure en locale
622 // retour 1: si tout est ok,
623 //          0: une courbure trop grande a été détecté
624 int Test_courbure_anormale2(double inf_normale,const Tableau<Coordonnee3>& tab_coor
625                             , const BaseB & aiB , const BaseH & aiH);
626
627 // routine (3) generale de calcul de la courbure
628 // cas 3: moyenne pondérée des normales de part et autres des arrêtes, et localisation du point sur
l'arrête
629 Tenseur_ns2BB Courbure3
630 ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB , const BaseH & aiH
631   ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
632 // routine (3) generale de calcul de la courbure et de sa variation
633 // en sortie : curb , la courbure b11,b12,b22
634 //          dcurb , la variation de courbure.
635 // cas 3: moyenne pondérée des normales de part et autres des arrêtes, et localisation du point sur
l'arrête
636 void Dcourbure3 ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB
637                 , const Tableau <BaseB>& DaiB,const BaseH & aiH
638                 , const Tableau <BaseH>& DaiH,Tenseur_ns2BB& curb,TabOper <Tenseur_ns2BB>& dcurb
639                 ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
640 // test si la courbure est anormalement trop grande
641 // inf_normale : indique en entrée le det mini pour la courbure en locale
642 // retour 1: si tout est ok,
643 //          0: une courbure trop grande a été détecté
644 int Test_courbure_anormale3(double inf_normale,const Tableau<Coordonnee3>& tab_coor
645                             , const BaseB & aiB , const BaseH & aiH);
646
647 // routine (4) generale de calcul de la courbure
648 // cas 4: calcul de la courbure à partir d'un polynôme passant par les 6 points
649 Tenseur_ns2BB Courbure4
650 ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB , const BaseH & aiH
651   ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
652 // routine (4) generale de calcul de la courbure et de sa variation
653 // en sortie : curb , la courbure b11,b12,b22
654 //          dcurb , la variation de courbure.
655 // cas 4: calcul de la courbure à partir d'un polynôme passant par les 6 points
656 void Dcourbure4 ( const Tableau<Coordonnee3>& tab_coor, const BaseB & aiB
657                 , const Tableau <BaseB>& DaiB,const BaseH & aiH
658                 , const Tableau <BaseH>& DaiH,Tenseur_ns2BB& curb,TabOper <Tenseur_ns2BB>& dcurb
659                 ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
660
661 // test si la courbure est anormalement trop grande
662 // inf_normale : indique en entrée le det mini pour la courbure en locale
663 // retour 1: si tout est ok,
664 //          0: une courbure trop grande a été détecté
665 int Test_courbure_anormale4(double inf_normale,const Tableau<Coordonnee3>& tab_coor
666                             , const BaseB & aiB , const BaseH & aiH);
667
668 // programme de vérification par différences finies des dérivées
669 void Verif_Dcourbure( const Tableau<Noeud *>& tab_noeud,bool gradV_instantane
670                     ,Mat_pleine const & tabdphiS,int nombre_noeud,Vecteur const & tabphiS
671                     ,Mat_pleine const & tabdphiH,Vecteur const & tabphiH,const Epai* epais
672                     ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan);
673 // initialisation des conteneurs de retour des méthodes
674 void Init_conteneur_de_retour_pourFacette();
675
676
677 };

```

```

678 /// @} // end of group
679
680
681 #endif
682
683

```

## 7.203 Met\_Sfe1\_struc\_donnees.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *
32 *     DATE:          15/01/97
33 *
34 *     AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *     PROJET:        Herezh++
37 *
38 * *****/
39 * deuxième partie du fichier Met_Sfel.h
40 * contient en fait toutes les structures de données de passage d'informations
41 *
42 *     *****
43 *
44 *     VERIFICATION:
45 *
46 *     ! date !   auteur !           but
47 *     -----
48 *     !           !           !
49 *
50 *     *****
51 *     MODIFICATIONS:
52 *
53 *     ! date !   auteur !           but
54 *     -----
55 *
56 *
57 *
58 * *****/
59
60 #include "Tenseur.h"
61 #include "Base.h"
62
63 // deuxième partie du fichier Met_Sfel.h
64 // contient en fait toutes les structures de données de passage d'informations
65
66
67 /// @addtogroup groupe_des_metrique
68 ///   @{
69 ///
70
71 /// CLASSE CONTENEUR : cas des grandeurs relatives à la courbure, grandeurs à 0, t et tdt
72 /// les données sont publiques
73     class Courbure_t_tdt
74     { public : Courbure_t_tdt () : // constructeur par défaut
75         aiB_0 (NULL), aiH_0 (NULL), aiB_t (NULL), aiH_t (NULL), aiB_tdt (NULL), aiH_tdt (NULL)

```

```

76
,aijBB_0(NULL),aijHH_0(NULL),aijBB_t(NULL),aijHH_t(NULL),aijBB_tdt(NULL),aijHH_tdt(NULL)
77 ,curbBB_t(NULL),curbBB_tdt(NULL),curbBB_0(NULL)
78 ,ajacobien_tdt(NULL),ajacobien_t(NULL),ajacobien_0(NULL)
79 {}
80 Courbure_t_tdt // constructeur normal
81 (BaseB * aaiB_0,BaseH * aaiH_0,BaseB * aaiB_t,BaseH * aaiH_t, BaseB * aaiB_tdt
82 ,BaseH * aaiH_tdt
83 ,TenseurBB* aaijBB_0,TenseurHH* aaijHH_0,TenseurBB* aaijBB_t
84 ,TenseurHH* aaijHH_t,TenseurBB* aaijBB_tdt,TenseurHH* aaijHH_tdt
85 ,TenseurBB * gcurbBB_t,TenseurBB * gcurbBB_tdt,TenseurBB * gcurbBB_0
86 ,double* gjacobien,double* gajacobien_t,double* gajacobien_0) :
87 aiB_0(aaiB_0),aiH_0(aaiH_0),aiB_t(aaiB_t),aiH_t(aaiH_t),aiB_tdt(aaiB_tdt),
88 aiH_tdt(aaiH_tdt),
89 aijBB_0(aaijBB_0),aijHH_0(aaijHH_0)
90 ,aijBB_t(aaijBB_t),aijHH_t(aaijHH_t),aijBB_tdt(aaijBB_tdt),aijHH_tdt(aaijHH_tdt)
91 ,curbBB_t(gcurbBB_t),curbBB_tdt(gcurbBB_tdt),curbBB_0(gcurbBB_0),
92 ajacobien_tdt(gjacobien),ajacobien_t(gajacobien_t),ajacobien_0(gajacobien_0)
93 {}
94 Courbure_t_tdt (const Courbure_t_tdt& ex) : // constructeur de copie
95
aiB_0(ex.aiB_0),aiH_0(ex.aiH_0),aiB_t(ex.aiB_t),aiH_t(ex.aiH_t),aiB_tdt(ex.aiB_tdt)
96 ,aiH_tdt(ex.aiH_tdt),
97 aijBB_0(ex.aijBB_0),aijHH_0(ex.aijHH_0),aijBB_t(ex.aijBB_t),aijHH_t(ex.aijHH_t)
98 ,aijBB_tdt(ex.aijBB_tdt),aijHH_tdt(ex.aijHH_tdt)
99 ,curbBB_t(ex.curbBB_t),curbBB_tdt(ex.curbBB_tdt),curbBB_0(ex.curbBB_0),
100
ajacobien_tdt(ex.ajacobien_tdt),ajacobien_t(ex.ajacobien_t),ajacobien_0(ex.ajacobien_0)
101 {}
102
103 Courbure_t_tdt& operator= (const Courbure_t_tdt& ex) // surcharge d'affectation
104 {aiB_0=ex.aiB_0;aiH_0=ex.aiH_0;aiB_t=ex.aiB_t;aiH_t=ex.aiH_t;aiB_tdt=ex.aiB_tdt;
105 aiH_tdt=ex.aiH_tdt;
106 aijBB_0=ex.aijBB_0;aijHH_0=ex.aijHH_0; aijBB_t=ex.aijBB_t;aijHH_t=ex.aijHH_t;
107 aijBB_tdt=ex.aijBB_tdt;aijHH_tdt=ex.aijHH_tdt;
108 curbBB_t=ex.curbBB_t;curbBB_tdt=ex.curbBB_tdt;curbBB_0=ex.curbBB_0;
109
ajacobien_tdt=ex.ajacobien_tdt;ajacobien_t=ex.ajacobien_t;ajacobien_0=ex.ajacobien_0;
110 return *this;};
111 // mise à jour des grandeurs
112 void Mise_a_jour_grandeur(BaseB * aaiB_0,BaseH * aaiH_0,BaseB * aaiB_t,BaseH* aaiH_t
113 ,BaseB * aaiB_tdt,BaseH * aaiH_tdt
114 ,TenseurBB* aaijBB_0,TenseurHH* aaijHH_0,TenseurBB* aaijBB_t,TenseurHH* aaijHH_t
115 ,TenseurBB* aaijBB_tdt,TenseurHH* aaijHH_tdt
116 ,TenseurBB * gcurbBB_t,TenseurBB * gcurbBB_tdt,TenseurBB * gcurbBB_0
117 ,double* gjacobien,double* gajacobien_t,double* gajacobien_0)
118 {aiB_0=aaiB_0;aiH_0=aaiH_0;aiB_t=aaiB_t;aiH_t=aaiH_t;aiB_tdt=aaiB_tdt;
119 aiH_tdt=aaiH_tdt;
120 aijBB_0=aaijBB_0;aijHH_0=aaijHH_0;aijBB_t=aaijBB_t;aijHH_t=aaijHH_t;
121 aijBB_tdt=aaijBB_tdt;aijHH_tdt=aaijHH_tdt;
122 curbBB_t=gcurbBB_t;curbBB_tdt=gcurbBB_tdt;curbBB_0=gcurbBB_0;
123 ajacobien_tdt=gjacobien;ajacobien_t=gajacobien_t;ajacobien_0=gajacobien_0;
124 };
125 // mise à jour des grandeurs à 0 et à t sauf les variations (pas de tableau et pas à
126 tdt)
// à partir de grandeurs stockées par ailleurs (il s'aait ici donc d'affectation de
contenu et
127 // non de pointeur !)
128 void Recup_grandeur_0_t(BaseB & aaiB_0,BaseH & aaiH_0,BaseB & aaiB_t,BaseH& aaiH_t
129 ,TenseurBB& aaijBB_0,TenseurHH& aaijHH_0,TenseurBB& aaijBB_t,TenseurHH& aaijHH_t
130 ,TenseurBB & ,double& gajacobien_0,double& gajacobien_t)
131 {*aiB_0=aaiB_0;*aiH_0=aaiH_0;*aiB_t=aaiB_t;*aiH_t=aaiH_t;
132 *aijBB_0=aaijBB_0;*aijHH_0=aaijHH_0;*aijBB_t=aaijBB_t;*aijHH_t=aaijHH_t;
133 *ajacobien_t=gajacobien_t;*ajacobien_0=gajacobien_0;
134 };
135
136 // variables :
137 BaseB * aiB_0; BaseH * aiH_0; BaseB * aiB_t; BaseH * aiH_t;BaseB * aiB_tdt;
138 BaseH * aiH_tdt;
139 TenseurBB * aijBB_0; TenseurHH * aijHH_0;
140 TenseurBB * aijBB_t ; TenseurHH * aijHH_t ;TenseurBB * aijBB_tdt;TenseurHH * aijHH_tdt;
141 TenseurBB * curbBB_t;TenseurBB * curbBB_tdt;TenseurBB * curbBB_0;
142 double* ajacobien_tdt;double* ajacobien_t;double* ajacobien_0;
143 };
144 /// @} // end of group
145

```

## 7.204 PoutSfe3.h

```

1 // FICHER : PoutSfe3.h
2 // CLASSE : PoutSfe3
3 /*****
4 * UNIVERSITE DE BRETAGNE SUD (UBS) --- ENSIBS DE LORIENT *
5 *****/
6 * IRDL - Equipe DE GENIE MECANIQUE ET MATERIAUX (EG2M) *

```

```

7 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
8 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://irdl.fr *
9 *****
10 * DATE: 24/03/2018 *
11 * $ *
12 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
13 * phone 0297874573, fax : 0297874588 *
14 * $ *
15 * PROJET: Herezh++ *
16 * $ *
17 *****
18 * BUT: Element poutre SFE3. Le calcul de la normale *
19 * s'effectue à partir de l'interpolation des theta3 relativement *
20 * à la poutre centrale, ceci via un polynome quadratique complet. *
21 * Le jacobien est celui de la poutre linéaire centrale. *
22 * $ *
23 * ***** *
24 * VERIFICATION: *
25 * *
26 * ! date ! auteur ! but ! *
27 * ----- *
28 * ! ! ! ! *
29 * $ *
30 * ***** *
31 * MODIFICATIONS: *
32 * ! date ! auteur ! but ! *
33 * ----- *
34 * $ *
35 *****/
36
37 /*
38 //
39 // l'interpolation est SFE,
40 // (4)----(1)----(2)----(3)
41 */
42
43
44 #ifndef POUTSFE3_H
45 #define POUTSFE3_H
46
47 #include "ParaGlob.h"
48 #include "ElemMeca.h"
49 #include "Met_abstraite.h"
50 #include "GeomSeg.h"
51 #include "Noeud.h"
52 #include "UtilLecture.h"
53 #include "Tenseur.h"
54 #include "NevezTenseur.h"
55 #include "Deformation.h"
56 #include "SfeMembT.h"
57 #include "ElFrontiere.h"
58 #include "FrontSegLine.h"
59
60
61 class PoutSfe3 : public SfeMembT
62 {
63
64 public :
65
66 // CONSTRUCTEURS :
67 // Constructeur par default
68 PoutSfe3 ();
69
70 // Constructeur fonction d'une epaisseur et
71 // eventuellement d'un numero
72 // d'identification
73 PoutSfe3 (double epaiss,int num_mail=0,int num_id=-3);
74
75 // Constructeur fonction d'un numero de maillage et d'identification
76 PoutSfe3 (int num_mail,int num_id);
77
78 // Constructeur fonction d'une epaisseur, d'un numero d'identification,
79 // du tableau de connexite des noeuds
80 PoutSfe3 (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
81
82 // Constructeur de copie
83 PoutSfe3 (const PoutSfe3& Pout);
84
85
86 // DESTRUCTEUR :
87 ~PoutSfe3 ();
88
89 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
90 // méthode virtuelle
91 Element* Nevez_copie() const { Element * el= new PoutSfe3(*this); return el;};
92
93 // Surcharge de l'operateur = : realise l'egalite entre deux instances de PoutSfe3

```

```

94     PoutSfe3& operator= (PoutSfe3& Pout);
95
96     // METHODES :
97 // 1) derivant des virtuelles pures
98
99     // affichage dans la sortie transmise, des variables duales "nom"
100    // aux differents points d'integration
101    // dans le cas ou nom est vide, affichage de "toute" les variables
102    void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
103
104 // 2) derivant des virtuelles
105 // 3) methodes propres a l'element
106
107    // les coordonnees des points d'integration dans l'epaisseur
108    inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
109
110 protected :
111
112    // adressage des frontieres linéiques et surfacique
113    // définit dans les classes dérivées, et utilisées pour la construction des frontieres
114    ElFrontiere* new_frontiere_lin(int ,Tableau<Noeud *> & tab, DdlElement& ddelem)
115    { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
116    ElFrontiere* new_frontiere_surf(int ,Tableau<Noeud *> & tab, DdlElement& ddelem)
117    { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
118
119 // VARIABLES PRIVEES :
120 // place memoire commune a tous les elements PoutMembL1
121 static SfeMembT::DonnComSfe * doCoSfe3;
122 // idem mais pour les indicateurs qui servent pour l'initialisation
123 static SfeMembT::UneFois uneFoisSfe3;
124
125 class NombresConstruirePoutSfe3 : public NombresConstruire
126 { public: NombresConstruirePoutSfe3();
127 };
128 static NombresConstruirePoutSfe3 nombre_V; // les nombres propres à l'élément
129
130 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
131 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
132 class ConsPoutSfe3 : public ConstrucElement
133 { public : ConsPoutSfe3 ()
134     { NouvelleTypeElement nouv(POUT,SFE3,MECA_SOLIDE_DEFORMABLE,this);
135     if (ParaGlob::NiveauImpression() >= 4)
136     cout << "\n initialisation PoutSfe3" << endl;
137     Element::listTypeElement.push_back(nouv);
138     };
139     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
140     {Element * pt;
141     pt = new PoutSfe3 (num_maill,num) ;
142     return pt;};
143     // ramene true si la construction de l'element est possible en fonction
144     // des variables globales actuelles: ex en fonction de la dimension
145     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
146 };
147 static ConsPoutSfe3 consPoutSfe3;
148 };
149 #endif
150
151
152
153

```

## 7.205 Sfeg.h

```

1 /*****
2 *      UNIVERSITE DE BRETAGNE SUD    --- I.U.P/I.U.T. DE LORIENT    *
3 *****/
4 *      LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX    *
5 *      Tel 97.80.80.60    *
6 *      Centre de Genie Industriel 56520 GUIDEL-PLAGES    *
7 *****/
8 *      DATE:      23/01/97    *
9 *      $    *
10 *      AUTEUR:      G RIO    *
11 *      $    *
12 *      PROJET:      Herezh++    *
13 *      $    *
14 *****/
15 *      BUT: Definir les infos qui sont communs aux elements    *
16 *      Sfe.    *
17 *      $    *
18 *      *****/
19 *      VERIFICATION:    *
20 *      *    *
21 *      ! date ! auteur ! but    !    *

```

```

22 * ----- *
23 * ! ! ! ! *
24 * $ *
25 * ***** *
26 * MODIFICATIONS: *
27 * ! date ! auteur ! but ! *
28 * ----- *
29 * $ *
30 *****/
31 #ifndef SFEG_H
32 #define SFEG_H
33
34 #include "ElemGeom.h"
35
36 // les fonctions d'interpolation sont definis pour l'instant pour des triangles
37 //
38 // (5) ----- (4)
39 // (3) | |
40 // | |
41 // | |
42 // (1) |----- (2)
43 //
44 // | |
45 // | |
46 // | |
47 // | |
48 // | |
49 // (6)
50 //
51
52
53 class Sfeg : public ElemGeom
54 {
55 public :
56 // CONSTRUCTEURS :
57 // par default on suppose un point d'integration et
58 // un triangle lineaire a 3 noeuds
59 Sfeg( int nbi = 1, int nbne = 3);
60 // de copie
61 Sfeg(Sfeg& a);
62 // DESTRUCTEUR :
63 ~Sfeg();
64 // methodes particuliere
65
66 //----- cas de coordonnees locales quelconques -----
67 // retourne les fonctions d'interpolation au point M (en coordonnees locales)
68 Vecteur Phi(Coordonnee& M);
69 // retourne les derivees des fonctions d'interpolation au point M (en coordonnees locales)
70 Mat_pleine Dphi(Coordonnee& M);
71 // en fonction de coordonnees locales, retourne true si le point est a l'interieur
72 // de l'element, false sinon
73 bool Interieur(Coordonnee& M);
74
75
76 protected :
77 // METHODES PROTEGEES :
78 // fourni la coordonnees ksi du point d'integ i
79 inline double& KSI(int i) { return ptInteg(i)(1);};
80 inline double& ETA(int i) { return ptInteg(i)(2);};
81
82 };
83
84 #endif

```

## 7.206 SfeMembT.h

```

1 // FICHER : SfeMembT.h
2 // CLASSE : SfeMembT
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,

```

```

20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           04/07/2008                               *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)      *
35 *   PROJET:        Herezh++                                  *
36 *   $               $                                       *
37 *   $               $                                       *
38 *   $               $                                       *
39 *****/
40 *   BUT:
41 *   $
42 * La classe SfeMembT permet de declarer des elements SFE et de realiser
43 * le calcul du residu local et de la raideur locale pour une loi de
44 * comportement donnee. La dimension de l'espace pour un tel element est
45 * soit 2 ou 3 suivant que l'on travaille en contrainte plane ou 3D
46 * l'interpolation le nombre de point d'integration sont definit dans
47 * les classes derivees.
48 *
49 * l'element est virtuel.
50 *
51 * VERIFICATION:
52 *
53 * ! date ! auteur ! but
54 * -----
55 * ! ! !
56 * $
57 *
58 * MODIFICATIONS:
59 * ! date ! auteur ! but
60 * -----
61 * $
62 *****/
63
64 // -----classe pour un calcul de mecanique-----
65
66
67
68 #ifndef SFEMEMBT_H
69 #define SFEMEMBT_H
70
71 #include "ParaGlob.h"
72 #include "ElemMeca.h"
73 #include "Met_Sfel.h"
74 #include "ElemGeomCO.h"
75 #include "GeomSeg.h"
76 #include "Noeud.h"
77 #include "UtilLecture.h"
78 #include "Tenseur.h"
79 #include "NevezTenseur.h"
80 #include "DeformationSfel.h"
81 #include "EnumTypeCL.h"
82 #include "Epai.h"
83
84 /// @addtogroup groupe_des_elements_finis
85 /// @{
86 ///
87
88
89 class SfeMembT : public ElemMeca
90 {
91
92     public :
93
94         // CONSTRUCTEURS :
95         // Constructeur par default
96         SfeMembT ();
97
98         // Constructeur fonction d'un numero
99         // d'identification , d'identificateur d'interpolation et de geometrie
100        // et éventuellement un string d'information annexe
101        SfeMembT (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string info="");
102
103        // Constructeur fonction d'un numero de maillage et d'identification,
104        // du tableau de connexite des noeuds, d'identificateur d'interpolation et de geometrie
105        // et éventuellement un string d'information annexe
106        SfeMembT (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,

```



```

107         const Tableau<Noeud *>& tab,string info="" ) ;
108
109         // Constructeur de copie
110         SfeMembT (const SfeMembT& sfe);
111
112
113         // DESTRUCTEUR :
114         ~SfeMembT () ;
115
116
117         // Surcharge de l'operateur = : realise l'affectation entre deux instances de SfeMembT
118         // SfeMembT& operator=(const SfeMembT& sfe);
119
120         // METHODES :
121 // 1) derivant des virtuelles pures
122         // Lecture des donnees de la classe sur fichier
123         void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * ) ;
124
125         // affichage d'info en fonction de ordre
126         // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
127         void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> * tabMaillageNoeud)
128         { return Element::Info_com_El(nombre->nbnte,entreePrinc,ordre,tabMaillageNoeud);};
129
130         // ramene l'element geometrique
131         ElemGeomC0& ElementGeometrique() const { return *(unefois->doCoMemb->eleCentre);};
132         // ramene l'element geometrique en constant
133         const ElemGeomC0& ElementGeometrique_const() const {return *(unefois->doCoMemb->eleCentre);};
134
135         // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
136         // associé
137         // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
138         // 1) cas où l'on utilise la place passée en argument
139         // Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
140         // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
141         void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
142
143         // -- connaissances particulières sur l'élément
144         // ramène l'épaisseur de l'élément
145         // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
146         virtual double EpaisseurMoyenne(Enum_dure enu, const Coordonnee& ) {return H(enu);};
147         // ramène l'épaisseur moyenne de l'élément (indépendante du point)
148         // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
149         virtual double EpaisseurMoyenne(Enum_dure enu) {return H(enu);};
150
151         // vérification que la courbure ne soit pas anormale au temps enu
152         // inf_normale : indique en entrée le det mini pour la courbure en locale
153         // retour 1: si tout est ok,
154         // 0: une courbure trop grande a été détecté
155         // retour = -1 : il y a un pb inconnue dans le calcul
156         int Test_courbure_anormale3(Enum_dure enu,double inf_normale);
157
158         // retourne la liste des données particulières actuellement utilisés
159         // par l'élément (actif ou non), sont exclu de cette liste les données particulières des noeuds
160         // reliés à l'élément
161         // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
162         List_io <TypeQuelconque> Les_types_particuliers_internes(bool absolue) const;
163
164         // récupération de grandeurs particulières au numéro d'ordre = iteg
165         // celles-ci peuvent être quelconques
166         // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
167         // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
168         void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);
169
170         // inactive les ddl du problème primaire de mécanique
171         inline void Inactive_ddl_primaire()
172         {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};
173         // active les ddl du problème primaire de mécanique
174         inline void Active_ddl_primaire()
175         {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl);};
176         // ajout des ddl de contraintes pour les noeuds de l'élément
177         inline void Plus_ddl_Sigma()
178         {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
179         // inactive les ddl du problème de recherche d'erreur : les contraintes
180         inline void Inactive_ddl_Sigma()
181         {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
182         // active les ddl du problème de recherche d'erreur : les contraintes
183         inline void Active_ddl_Sigma()
184         {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
185         // active le premier ddl du problème de recherche d'erreur : SIGMA11
186         inline void Active_premier_ddl_Sigma()
187         {ElemMeca::Act_premier_ddl_Sigma();};
188
189         // lecture de données diverses sur le flot d'entrée
190         void LectureContraintes (UtilLecture * entreePrinc)
191         { if (unefois->CalResPrem_t == 1)
192             ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());};

```

```

192     else
193     { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
194       unefois->CalResPrem_t = 1;
195     }
196   };
197
198   // retour des contraintes en absolu retour true si elle existe sinon false
199   bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
200   { if (unefois->CalResPrem_t == 1)
201     ElemMeca::ContraintesEnAbsolues(false,lesPtMecaInt.TabSigHH_t(),tabSig);
202     else
203     { ElemMeca::ContraintesEnAbsolues(true,lesPtMecaInt.TabSigHH_t(),tabSig);
204       unefois->CalResPrem_t = 1;
205     }
206     return true; }
207
208   // methode relatives au nombre de pt d'integ
209   int Nb_pt_int_surf() { return nombre->nbis;}; // nb total de pt d'integ de surface
210   int Nb_pt_int_epai() { return nombre->nbie;}; // nb total de pt d'integ d'epaisseur
211
212   // Libere la place occupee par le residu et eventuellement la raideur
213   // par l'appel de Libere de la classe mere et libere les differents tenseurs
214   // intermediaires cree pour le calcul et les grandeurs pointee
215   // de la raideur et du residu
216   void Libere ();
217
218   // acquisition d'une loi de comportement
219   void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
220
221   // définition de conditions limites pouvant affecter l'élément
222   // on peut ainsi soit mettre une nouvelle condition, soit changer une ancienne condition
223   // en cours de calcul, la condition peut changer
224   // cas d'une seule arête, nb_ar = le numéro de l'arête
225   // si arTypeCL == RIEN_TYPE_CL, la condition est supprimée, et vpla n'est pas utilisé
226   void DefCondLim(const EnumTypeCL & arTypeCL,const Coordonnee3& vpla,const int& nb_ar);
227   // cas de plusieurs arêtes
228   // si arTypeCL(i) == RIEN_TYPE_CL, la condition est supprimée, et vpla(i) n'est pas utilisé
229   // cas de plusieurs arêtes, la dimension des tableaux = le nombre d'arêtes
230   void DefCondLim(const Tableau <EnumTypeCL> & arTypeCL,const Tableau <Coordonnee3>& vpla );
231
232   // test si l'element est complet
233   // = 1 tout est ok, =0 element incomplet
234   int TestComplet();
235   // Compléter pour la mise en place de la gestion de l'hourglass
236   Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc) {return
this;};
237
238   // procedure permettant de completer l'element apres
239   // sa creation avec les donnees du bloc transmis
240   // peut etre appeler plusieurs fois
241   // ici il s'agit de l'epaisseurs
242   Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
243
244   // ramene l'epaisseur
245   inline double H(Enum_dure enu = TEMPS_tdt )
246   { if (donnee_specif.epais != NULL){switch (enu)
247     { case TEMPS_0: return donnee_specif.epais->epaisseur0; break;
248       case TEMPS_t: return donnee_specif.epais->epaisseur_t; break;
249       case TEMPS_tdt: return donnee_specif.epais->epaisseur_tdt; break;
250     }; return 0.;}
251     else {return def->DonneeInterpoleeScalaire(EPAIS,enu);};};
252   // les coordonnees des points dans l'epaisseur
253   virtual double KSI(int i) = 0;
254
255   // ramene vrai si la surface numéro ns existe pour l'élément
256   // dans le cas des éléments sfe il ne peut y avoir qu'une
257   // seule surface
258   bool SurfExiste(int ns) const
259   { if ((ns==1)&&(ParaGlob::Dimension() >= 2)) return true; else return false;};
260
261   // ramene vrai si l'arête numéro na existe pour l'élément
262   bool AreteExiste(int na) const
263   {if ((na <= 3) || (na>= 1)) return true; else return false;};
264
265   // retourne les tableaux de ddl associés aux noeuds, gere par l'element
266   // ce tableau et specifique a l'element
267   const DdlElement & TableauDdl() const
268   { return unefois->doCoMemb->tab_ddl; };
269
270   // Calcul du residu local et de la raideur locale,
271   // pour le schema implicite
272   Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
273
274   // Calcul du residu local a t
275   // pour le schema explicite par exemple
276   Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
277   { return SfeMembT::CalculResidu(false,pa);};

```

```

278
279 // Calcul du residu local a tdt
280 // pour le schema explicit par exemple
281 Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
282 { return SfeMembT::CalculResidu(true,pa);};
283
284 // Calcul de la matrice masse pour l'élément
285 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
286
287 // ----- calcul dynamique -----
288 // calcul de la longueur d'arrête de l'élément minimal
289 // divisé par la célérité la plus rapide dans le matériau
290 double Long_arrete_mini_sur_c(Enum_dure temps)
291 { int nbn_aconsiderer = 3;
292   return ElemMeca::Interne_Long_arrete_mini_sur_c(temps,nbn_aconsiderer);};
293
294 //----- calcul d'erreur, remontée des contraintes -----
295 // 1) calcul du résidu et de la matrice de raideur pour le calcul d'erreur
296 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
297 // 2) remontée aux erreurs aux noeuds
298 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
299
300 // ----- affichage ou récupération d'informations -----
301 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
302 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
303 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
304 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
305 // temps: dit si c'est à 0 ou t ou tdt
306 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M);
307
308 // recuperation des coordonnées du point de numéro d'ordre iteg pour
309 // la grandeur enu
310 // temps: dit si c'est à 0 ou t ou tdt
311 // si erreur retourne erreur à true
312 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur);
313
314 // récupération des valeurs au numéro d'ordre = iteg pour
315 // les grandeur enu
316 Tableau <double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
enu,int iteg);
317 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
318 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
319 // de conteneurs quelconque associée
320 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure ,List_io<TypeQuelconque>& ,int );
321
322 // calcul éventuel de la normale à un noeud
323 // ce calcul existe pour les éléments 2D, 1D axi, et aussi pour les éléments 1D
324 // qui possède un repère d'orientation
325 // en retour coor = la normale si coor.Dimension() est = à la dimension de l'espace
326 // si le calcul n'existe pas --> coor.Dimension() = 0
327 // ramène un entier :
328 // == 1 : calcul normal
329 // == 0 : problème de calcul -> coor.Dimension() = 0
330 // == 2 : indique que le calcul n'est pas licite pour le noeud passé en paramètre
331 // c'est le cas par exemple des noeuds extérieurs pour les éléments SFE
332 // mais il n'y a pas d'erreur, c'est seulement que l'élément n'est pas ad hoc pour
333 // calculer la normale à ce noeud là
334 // temps: indique à quel moment on veut le calcul
335 // pour des éléments particulier (ex: SFE) la méthode est surchargée
336 virtual int CalculNormale_noeud(Enum_dure temps, const Noeud& noe,Coordonnee& coor);
337
338 //===== lecture écriture dans base info =====
339
340 // cas donne le niveau de la récupération
341 // = 1 : on récupère tout
342 // = 2 : on récupère uniquement les données variables (supposées comme telles)
343 void Lecture_base_info
344 (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
345 // cas donne le niveau de sauvegarde
346 // = 1 : on sauvegarde tout
347 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
348 void Ecriture_base_info(ofstream& sort,const int cas) ;
349
350 // 2) derivant des virtuelles
351
352 // ramène le nombre de points d'intégration de surface correspondant à un type énuméré
353 // ramène 0 si l'élément n'est pas une plaque ou coque
354 virtual int NbPtIntegSurface(Enum_ddl ) const;
355 // ramène le nombre de points d'intégration en épaisseur correspondant à un type énuméré
356 // ramène 0 si l'élément n'est pas une plaque ou coque
357 virtual int NbPtIntegEpaiss(Enum_ddl ) const ;
358 // ramene l'element geometrique de surface correspondant au ddl passé en paramètre
359 // ou null si ce n'est pas définie, dans ce cas si l'élément géométrique de surface est 2D
360 // cela signifie qu'il faut se référer à l'élément générique: ElementGeometrie(...)
361 virtual ElemGeomCO* ElementGeometrieSurface(Enum_ddl ) const ;
362 // ramene l'element geometrique d'épaisseur correspondant au ddl passé en paramètre

```

```

363         // ou null si ce n'est pas définie, dans ce cas si l'élément géométrique de surface est 2D
364         // cela signifie que tout est constant (identique) dans l'épaisseur
365         virtual ElemGeomCO* ElementGeometrieEpais(Enum_ddl ) const;
366
367
368         // retourne un tableau de ddl element, correspondant à la
369         // composante de sigma -> SIG11, pour chaque noeud qui contient
370         // des ddl de contrainte
371         // -> utilisé pour l'assemblage de la raideur d'erreur
372     inline DdlElement& Tableau_de_Sigl() const
373     { return unefois->doCoMemb->tab_Err1Sig11; } ;
374
375     // actualisation des ddl et des grandeurs actives de t+dt vers t
376     void TdtversT();
377     // actualisation des ddl et des grandeurs actives de t vers tdt
378     void TversTdt();
379
380     // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
381     // qu'une fois la remontée aux contraintes effectuées sinon aucune
382     // action. En retour la valeur de l'erreur sur l'élément
383     // type indique le type de calcul d'erreur :
384     void ErreurElement(int type,double& errElemRelative
385         ,double& numerateur, double& denominateur);
386
387     // mise à jour de la boîte d'encombrement de l'élément, suivant les axes I_a globales
388     // en retour coordonnées du point mini dans retour.Premier() et du point maxi dans .Second()
389     // la méthode est différente de la méthode générale car il faut prendre en compte l'épaisseur de
390     // l'élément
391     virtual const DeuxCoordonnees& Boite_encombre_element(Enum_dure temps);
392
393     // calcul des seconds membres suivant les chargements
394     // cas d'un chargement volumique,
395     // force indique la force volumique appliquée
396     // retourne le second membre résultant
397     // ici on l'épaisseur de l'élément pour constituer le volume
398     // -> explicite à t
399     Vecteur SM_charge_volumique_E_t
400     (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
401     { return SfeMembT::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_); } ;
402     // -> explicite à tdt
403     Vecteur SM_charge_volumique_E_tdt
404     (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
405     { return SfeMembT::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_); } ;
406     // -> implicite,
407     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
408     // retourne le second membre et la matrice de raideur correspondant
409     ResRaid SMR_charge_volumique_I
410     (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
411     ;
412
413     // cas d'un chargement surfacique, sur les frontières des éléments
414     // force indique la force surfacique appliquée
415     // numface indique le numéro de la face chargée
416     // retourne le second membre résultant
417     // -> version explicite à t
418     Vecteur SM_charge_surfacique_E_t
419     (const Coordonnee& force,Fonction_nd* pt_fonct,int numFace,const ParaAlgoControle & pa)
420     { return SfeMembT::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa); } ;
421     // -> version explicite à tdt
422     Vecteur SM_charge_surfacique_E_tdt
423     (const Coordonnee& force,Fonction_nd* pt_fonct,int numFace,const ParaAlgoControle & pa)
424     { return SfeMembT::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa); } ;
425     // -> implicite,
426     // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
427     // retourne le second membre et la matrice de raideur correspondant
428     ResRaid SMR_charge_surfacique_I
429     (const Coordonnee& force,Fonction_nd* pt_fonct,int numFace,const ParaAlgoControle & pa) ;
430
431     // cas d'un chargement lineique, sur les aretes frontières des éléments
432     // force indique la force lineique appliquée
433     // numarete indique le numéro de l'arete chargée
434     // retourne le second membre résultant
435     // NB: il y a une définition par défaut pour les éléments qui n'ont pas
436     // d'arete externe -> message d'erreur d'où le virtuel et non virtuel pur
437     // -> explicite à t
438     Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nd* pt_fonct,int numArete,const
439     ParaAlgoControle & pa)
440     { return SfeMembT::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); } ;
441     // -> explicite à tdt
442     Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nd* pt_fonct,int numArete,const
443     ParaAlgoControle & pa)
444     { return SfeMembT::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); } ;
445     // -> implicite,
446     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
447     // retourne le second membre et la matrice de raideur correspondant
448     ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nd* pt_fonct,int numArete,const
449     ParaAlgoControle & pa) ;

```

```

445
446 // cas d'un chargement lineique suiveuse, sur les aretes frontieres des éléments 2D (uniquement)
447 // force indique la force lineique appliquée
448 // numarete indique le numéro de l'arete chargée
449 // retourne le second membre résultant
450 // -> explicite à t
451 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
452 { return SfeMembT::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa); };
453 // -> explicite à tdt
454 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
455 { return SfeMembT::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa); };
456 // -> implicite,
457 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
458 // retourne le second membre et la matrice de raideur correspondant
459 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
460
461 // cas d'un chargement de type pression, sur les frontieres des éléments
462 // pression indique la pression appliquée
463 // numface indique le numéro de la face chargée
464 // retourne le second membre résultant
465 // NB: il y a une définition par défaut pour les éléments qui n'ont pas de
466 // surface externe -> message d'erreur d'où le virtuel et non virtuel pur
467 // -> explicite à t
468 Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
469 { return SfeMembT::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa); };
470 // -> explicite à tdt
471 Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
472 { return SfeMembT::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa); };
473 // -> implicite,
474 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
475 // retourne le second membre et la matrice de raideur correspondant
476 ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
477
478 // cas d'un chargement de type pression unidirectionnelle, sur les frontieres des éléments
479 // presUniDir indique le vecteur appliquée
480 // numface indique le numéro de la face chargée
481 // retourne le second membre résultant
482 // -> explicite à t
483 Vecteur SM_charge_presUniDir_E_t(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
484 { return SfeMembT::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,false,pa); };
485 // -> explicite à tdt
486 Vecteur SM_charge_presUniDir_E_tdt(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
487 { return SfeMembT::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,true,pa); };
488 // -> implicite,
489 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
490 // retourne le second membre et la matrice de raideur correspondant
491 ResRaid SMR_charge_presUniDir_I(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa);
492
493 // cas d'un chargement surfacique hydrostatique,
494 // poidvol: indique le poids volumique du liquide
495 // M_liquide : un point de la surface libre
496 // dir_normal_liquide : direction normale à la surface libre
497 // retourne le second membre résultant
498 // -> explicite à t
499 Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
500 ,int numFace,const Coordonnee& M_liquide,const ParaAlgoControle
& pa
501 ,bool sans_limitation)
502 { return
SfeMembT::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation); };
503 // -> explicite à tdt
504 Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
505 ,int numFace,const Coordonnee& M_liquide,const
ParaAlgoControle & pa
506 ,bool sans_limitation)
507 { return
SfeMembT::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation); };
508 // -> implicite,
509 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
510 // retourne le second membre et la matrice de raideur correspondant
511 ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
512 ,int numFace,const Coordonnee& M_liquide,const ParaAlgoControle
& pa
513 ,bool sans_limitation) ;
514
515 // cas d'un chargement surfacique hydro-dynamique,
516 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
517 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc unitaire)

```

```

518 // une suivant la direction normale à la vitesse de type portance
519 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
520 // une suivant la vitesse tangente de type frottement visqueux
521 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
522 // coef_mul: est un coefficient multiplicateur global (de tout)
523 // retourne le second membre résultant
524 // -> explicite à t
525 Vecteur SM_charge_hydrodynamique_E_t( CourbelD* frot_fluid,const double& poidvol
526 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
527 , CourbelD* coef_aero_t,const ParaAlgoControle & pa)
528 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
529 // -> explicite à tdt
530 Vecteur SM_charge_hydrodynamique_E_tdt( CourbelD* frot_fluid,const double& poidvol
531 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
532 , CourbelD* coef_aero_t,const ParaAlgoControle &
pa)
533 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};
534 // -> implicite,
535 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
536 // retourne le second membre et la matrice de raideur correspondant
537 ResRaid SMR_charge_hydrodynamique_I( CourbelD* frot_fluid,const double& poidvol
538 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
539 , CourbelD* coef_aero_t,const ParaAlgoControle & pa)
;
540 // ramène le nombre de points d'intégration correspondant à un type énuméré
541 virtual int NbPtInteg(Enum_ddl enu) const;
542 // ===== définition et/ou construction des frontières =====
543 // Calcul des frontieres de l'element
544 // creation des elements frontieres et retour du tableau de ces elements
545 // la création n'a lieu qu'au premier appel
546 // ou lorsque l'on force le paramètre force a true
547 // dans ce dernier cas seul les frontière effacées sont recréée
548 Tableau <ElFrontiere*> const & Frontiere(bool force = false);
549 // ramène la frontière point
550 // éventuellement création des frontieres points de l'element et stockage dans l'element
551 // si c'est la première fois sinon il y a seulement retour de l'elements
552 // a moins que le paramètre force est mis a true
553 // dans ce dernier cas la frontière effacée est recréée
554 // num indique le numéro du point à créer (numérotation EF)
555 // ElFrontiere* const Frontiere_points(int num,bool force = false);
556 // ramène la frontière linéique
557 // éventuellement création des frontieres linéique de l'element et stockage dans l'element
558 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
559 // a moins que le paramètre force est mis a true
560 // dans ce dernier cas la frontière effacée est recréée
561 // num indique le numéro de l'arête à créer (numérotation EF)
562 // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
563 // ramène la frontière surfacique
564 // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
565 // si c'est la première fois sinon il y a seulement retour de l'elements
566 // a moins que le paramètre force est mis a true
567 // dans ce dernier cas la frontière effacée est recréée
568 // num indique le numéro de la surface à créer (numérotation EF)
569 // ici normalement uniquement 1 possible
570 // ElFrontiere* const Frontiere_surfacique(int num,bool force = false);
571 // 3) methodes propres a l'element
572 // ajout du tableau specific de ddl des noeuds
573 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
574 // des noeuds constituant l'element
575 void ConstTabDdl();
576 // ----- definition de la classe conteneur de donnees communes -----
577 class DonnComSfe
578 { public :
579 DonnComSfe (ElemGeomC0* eleCentre,const GeomSeg& seg,const DdlElement& tab
580 ,DdlElement& tabErr,DdlElement& tab_Err1Sig,DdlElement& tab_ddlXi_C
581 ,const Met_Sfel& met_gene,Tableau <Vecteur *> & resEr
582 ,Mat_pleine& raidEr,ElemGeomC0* eleEr,GeomSeg& segEr,ElemGeomC0* eleS
583 ,GeomSeg& segmS,Vecteur& residu_int,Mat_pleine& raideur_int
584 ,Tableau <Vecteur*> & residus_extN,Tableau <Mat_pleine*> & raideurs_extN
585 ,Tableau <Vecteur*> & residus_extA,Tableau <Mat_pleine*> & raideurs_extA

```

```

595         ,Tableau <Vecteur* >& residus_extS,Tableau <Mat_pleine* >& raideurs_extS
596         ,Mat_pleine& mat_masse, ElemGeomC0* eleMas,const GeomSeg& segMas,int nbi
597         ,Tableau <EnuTypeCL> const & tabTypeCL,Tableau <Coordonnee3> const & vplan
598         ,Tableau <int>& nMetVTab_ddl,const Met_abstraite& met_cent );
599     DonnComSfe(DonnComSfe& a);
600     ~DonnComSfe();
601     // variables
602     ElemGeomC0* eleCentre; // contient les fonctions d'interpolation et
603         // les derivees de surface
604     GeomSeg segment; // epaisseur
605     DdlElement tab_ddl; // tableau des degres
606         //de liberte des noeuds de l'element commun a tous les elements
607     DdlElement tab_ddlXi_C; // tableau des degres de liberte Xi des noeuds de l'element
central,
608         // (sans ddl d'epaisseur s'ils existent !), et commun a tous les elements
609     Tableau <int> nMetVTab_ddl; // numerotation des ddl i de la metrique vers celle de tab_ddl:
610         // nMetVTab_ddl(i) = le numero dans Tab_ddl
611     Met_Sfel met_SfeMembT;
612     Met_abstraite met_surf_cent; // metrique de la surface centrale, et relative
uniquement aux xi centraux
613     Tableau <EnuTypeCL> tabType_rienCL; // tableau donnant les conditions limites par defaut

614     Tableau <Coordonnee3> vplan_rien; // c'est-à-dire : aucune condition
615     Mat_pleine matGeom ; // matrice geometrique
616     Mat_pleine matInit ; // matrice initiale
617     Tableau <TenseurBB * > d_epsBB; // place pour la variation des def
618     Tableau <TenseurHH * > d_sigHH; // place pour la variation des contraintes
619     Tableau < Tableau2 <TenseurBB * > > d2_epsBB; // variation seconde des deformations
620     // ---- concernant les frontieres et particulierement le calcul de second membre
621     ElemGeomC0* eleS; // contient les fonctions d'interpolation et les derivees
622     GeomSeg segS; // " " "
623     // calcul d'erreur
624     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
625     // d'erreur : contraintes
626     DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud, servant pour le
calcul
627     // d'erreur : contraintes, en fait pour l'assemblage
628     Tableau <Vecteur * > resErr; // residu pour le calcul d'erreur
629     Mat_pleine raidErr; // raideur pour le calcul d'erreur
630     ElemGeomC0* eleEr; // contient les fonctions d'interpolation et
631         // les derivees pour le calcul du hessien dans
632         //la resolution de la fonctionnelle d'erreur
633     GeomSeg segmentEr; // epaisseur
634     // ----- calcul de residus, de raideur : interne ou pour les efforts extérieurs
-----
635     // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
636     Vecteur residu_interne;
637     Mat_pleine raideur_interne;
638     Tableau <Vecteur* > residus_externeN; // pour les noeuds
639     Tableau <Mat_pleine* > raideurs_externeN; // pour les noeuds
640     Tableau <Vecteur* > residus_externeA; // pour les aretes
641     Tableau <Mat_pleine* > raideurs_externeA; // pour les aretes
642     Tableau <Vecteur* > residus_externeS; // pour la surface
643     Tableau <Mat_pleine* > raideurs_externeS; // pour la surface
644
645     // ----- données concernant la dynamique -----
646     Mat_pleine matrice_masse;
647     ElemGeomC0* eleMas; // contient les fonctions d'interpolation et ...
648         // pour les calculs relatifs à la masse
649     GeomSeg segmentMas; // epaisseur
650
651     // --- blocage éventuel de stabilisation normale à la membrane
652     // utilisé dans: ElemMeca::Cal_implicit_StabMembBiel, ElemMeca::Cal_explicit_StabMembBiel
653     Mat_pleine* sfematD; // raideur éventuelle,
654     Vecteur* sferesD; // residu éventuelle,
655     Tableau <int> * noeud_a_prendre_en_compte; // choix des noeuds à stabiliser
656
657 };
658
659 // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
660 // et un pointeur sur les données statiques communes
661 // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
662 // classe est défini. Son allocation est effectuée dans les classes dérivées
663 class UneFois
664 { public :
665     UneFois () ; // constructeur par defaut
666     ~UneFois () ; // destructeur
667
668     // VARIABLES :
669     public :
670     SfeMembT::DonnComSfe * doCoMemb;
671
672     // incicateurs permettant de dimensionner seulement au premier passage
673     // utilise dans "CalculResidu" et "Calcul_implicit"
674     int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
675     int CalimpPrem;
676     int dualSortSfe; // pour la sortie des valeurs au pt d'integ

```



```

677         int CalSMlin_t; // pour les seconds membres concernant les arretes
678         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
679         int CalSMRlin; // pour les seconds membres concernant les arretes
680         int CalSMsurf_t; // pour les seconds membres concernant les surfaces
681         int CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
682         int CalSMRsurf; // pour les seconds membres concernant les surfaces
683         int CalSMvol_t; // pour les seconds membres concernant les volumes
684         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
685         int CalSMvol; // pour les seconds membres concernant les volumes
686         int CalDynamique; // pour le calcul de la matrice de masse
687         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
688         // ----- sauvegarde du nombre d'élément en cours -----
689         int nbelem_in_Prog;
690     };
691
692     // -----
693
694     protected :
695
696     // VARIABLES PRIVEES :
697     UneFois * unefois; // pointeur défini dans la classe dérivée
698     // les données spécifiques sont groupées dans une structure pour sécuriser
699     // le passage de paramètre dans init par exemple
700     class Donnee_specif
701     { public :
702         // --- cas de l'épaisseur stockée dans l'élément ----
703         Donnee_specif() : epais(new Epai)
704         {epais->epaisseur0=epaisseur_defaut;epais->epaisseur_t=epaisseur_defaut;
705          epais->epaisseur_tdt=epaisseur_defaut;
706         };
707         Donnee_specif(double epai0,double epai_t,double epai_tdt) : epais(new Epai)
708         {epais->epaisseur0=epai0;epais->epaisseur_t=epai_t;epais->epaisseur_tdt=epai_tdt;};
709         Donnee_specif(double epai) : epais(new Epai) // cas d'une seule valeur: on initialise tout
710
711         {epais->epaisseur0=epais->epaisseur_t=epais->epaisseur_tdt=epai;};
712         Donnee_specif(const Donnee_specif& a) : epais(NULL)
713         {if (a.epais != NULL){ epais = new Epai(*a.epais);};}; // recopie via le constructeur de
714
715         // --- cas de l'épaisseur pouvant être stockée aux noeuds, donc rien à l'élément
716         // si l'on veut un élément 3D, il faut qu'ici epas == NULL
717         Donnee_specif(const Epai * epas)
718         {if (epas != NULL) {epais = new Epai(*epas);} else {epais = NULL;};};
719
720         ~Donnee_specif() {if (epais != NULL) delete epais;};
721         Donnee_specif & operator = ( const Donnee_specif& a)
722         { if (a.epais == NULL) { if(epais != NULL) {delete epais;epais=NULL;};
723           else // sinon cas où a.epais != NULL
724             {if (epais == NULL) {epais=new Epai(*a.epais);}
725              else { *epais = *a.epais;};
726             };
727         return *this;};
728         // data
729         // epaisseurs de l'element
730         Epai * epais; // pointeur qui est non null pour un élément 2D et null pour un élément 3D
731     };
732
733     Donnee_specif donnee_specif;
734
735     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
736     LesPtIntegMecaInterne lesPtMecaInt;
737
738     // information concernant des conditions limites éventuelles, qui ont des répercussions sur
739     // le calcul de la métrique par exemple. Par défaut areteTypeCL et vplan pointent sur les
740     // communes de la classe tabType_rienCL et vplan_rien, qui indiquent aucune conditions limites
741     // type de condition
742     Tableau <EnumTypeCL>* areteTypeCL; // areteTypeCL(i) : si différent de RIEN_TYPE_CL, indique le
743     // limite de l'arête i,
744     // si areteTypeCL(i) = TANGENTE_CL , alors vplan(i) contient
745     // un vecteur du plan normal à la tangente,
746     // l'arête donne un second vecteur
747     Tableau <Noeud *> t_N_centre; // tableau des noeuds du centre (sert par ex pour le calcul des
748     SM)
749
750     // type structuré et fonction virtuelle pour construire les éléments
751     // la fonction est défini dans le type dérivé
752     class NombresConstruire
753     { public:
754         int nbnce; // nb de noeud de l'element central
755         int nbnte; // nombre total de noeud
756         int nbneA ; // le nombre de noeud des aretes de l'élément central
757         int nbis; // le nombre de point d'intégration de surface pour le calcul mécanique
758         int nbie; // nombre de pt d'integ d'epaisseur pour le calcul mécanique
759         int nbisEr; // le nombre de point d'intégration de surface pour le calcul d'erreur
760         int nbieEr; // le nombre de point d'intégration d'epaisseur pour le calcul d'erreur
761         int nbisSur; // le nombre de point d'intégration pour le calcul de second membre surfacique

```



```

759     int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
760     int nbisMas; // le nombre de point d'intégration de surface pour le calcul de la matrice masse
761     int nbieMas; // le nombre de point d'intégration d'épaisseur pour le calcul de la matrice masse
762 };
763 NombresConstruire * nombre; // le pointeur défini dans la classe dérivée
764
765     // =====>> methodes appelees par les classes dérivées <<=====
766
767     // fonction d'initialisation servant dans les classes derivant
768     // au niveau du constructeur, si rien initialisation par défaut
769     SfeMembT::DonnComSfe* Init(ElemGeomC0* eleCentre, ElemGeomC0* eleEr
770         , ElemGeomC0* eleS, ElemGeomC0* eleMas, int type_calcul_jacobien = 1
771         , Donnee_specif donnee_specif = Donnee_specif(), bool sans_init_noeud =
false);
772     // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
773     void Destruction();
774
775     // =====>> methodes virtuelles dérivant d'ElemMeca =====
776     // ramene la dimension des tenseurs contraintes et déformations de l'élément
777     int Dim_sig_eps() const {return 2;};
778
779     //----- fonctions uniquement a usage interne -----
780     private :
781         // definition des données communes
782         // epaisAuNoeud indique si oui ou non l'épaisseur est défini aux noeuds
783         SfeMembT::DonnComSfe* Def_DonneeCommune(bool epaisAuNoeud, ElemGeomC0* eleCentre, ElemGeomC0* eleEr
784             , ElemGeomC0* eleS, ElemGeomC0* eleMas);
785         // Calcul du residu local a t ou tdt en fonction du booleen
786         Vecteur* CalculResidu (bool atdt, const ParaAlgoControle & pa);
787     private: // pour éviter les modifications par les classes dérivées
788         static TenseurHH * sig_bulk_pourSfe_HH; // variable de travail pour le bulk
789
790         // calcul des seconds membres suivant les chargements
791         // cas d'un chargement volumique,
792         // force indique la force volumique appliquée
793         // retourne le second membre résultant
794         // ici on l'épaisseur de l'élément pour constituer le volume
795         // -> explicite à t ou tdt en fonction de la variable booleenne atdt
796         Vecteur SM_charge_volumique_E
797             (const Coordonnee& force, Fonction_nD* pt_fonct, bool atdt, const ParaAlgoControle &
pa, bool sur_volume_finale_);
798         // cas d'un chargement surfacique, sur les frontières des éléments
799         // force indique la force surfacique appliquée
800         // numface indique le numéro de la face chargée
801         // retourne le second membre résultant
802         // -> explicite à t ou tdt en fonction de la variable booleenne atdt
803         Vecteur SM_charge_surfacique_E
804             (const Coordonnee& force, Fonction_nD* pt_fonct, int numFace, bool atdt, const
ParaAlgoControle & pa);
805         // cas d'un chargement lineique, sur les aretes frontières des éléments
806         // force indique la force lineique appliquée
807         // numarete indique le numéro de l'arete chargée
808         // retourne le second membre résultant
809         // NB: il y a une définition par défaut pour les éléments qui n'ont pas
810         // d'arete externe -> message d'erreur d'où le virtuel et non virtuel pur
811         // -> explicite à t ou tdt en fonction de la variable booleenne atdt
812         Vecteur SM_charge_lineique_E
813             (const Coordonnee& force, Fonction_nD* pt_fonct, int numArete, bool atdt, const
ParaAlgoControle & pa);
814         // cas d'un chargement lineique suiveuse, sur les aretes frontières des éléments 2D (uniquement)
815         // force indique la force lineique appliquée
816         // numarete indique le numéro de l'arete chargée
817         // retourne le second membre résultant
818         // -> explicite à t
819         Vecteur SM_charge_lineique_Suiv_E
820             (const Coordonnee& force, Fonction_nD* pt_fonct, int numArete, bool atdt, const
ParaAlgoControle & pa);
821         // cas d'un chargement de type pression, sur les frontières des éléments
822         // pression indique la pression appliquée
823         // numface indique le numéro de la face chargée
824         // retourne le second membre résultant
825         // NB: il y a une définition par défaut pour les éléments qui n'ont pas de
826         // surface externe -> message d'erreur d'où le virtuel et non virtuel pur
827         // -> explicite à t ou tdt en fonction de la variable booleenne atdt
828         Vecteur SM_charge_pression_E
829             (double pression, Fonction_nD* pt_fonct, int numFace, bool atdt, const ParaAlgoControle &
pa);
830         // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
831         // presUniDir indique le vecteur appliquée
832         // numface indique le numéro de la face chargée
833         // retourne le second membre résultant
834         // -> explicite à t ou tdt en fonction de la variable booleenne atdt
835         Vecteur SM_charge_presUniDir_E
836             (const Coordonnee& presUniDir, Fonction_nD* pt_fonct, int numFace, bool atdt, const
ParaAlgoControle & pa);
837         // cas d'un chargement surfacique hydrostatique,
838         // poidvol: indique le poids volumique du liquide

```

```

839 // M_liquide : un point de la surface libre
840 // dir_normal_liquide : direction normale à la surface libre
841 // retourne le second membre résultant
842 // -> explicite à t
843 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
844 ,int numFace,const Coordonnee& M_liquide,bool atdt
845 ,const ParaAlgoControle & pa
846 ,bool sans_limitation);
847 // cas d'un chargement surfacique hydro-dynamique,
848 // voir méthode explicite plus haut, pour les arguments
849 // retourne le second membre résultant
850 // bool atdt : permet de spécifier à t ou a t+dt
851 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
852 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
853 , CourbelD* coef_aero_t,bool atdt
854 ,const ParaAlgoControle & pa) ;
855 // calcul de la nouvelle épaisseur et de la raideur associée
856 // void CalEpaisseurEtVar(const ParaAlgoControle & pa, const double& module_compressibilite);
857 // calcul de la nouvelle épaisseur à tdt (sans raideur) avec métrique en explicite
858 // mise à jour du volume au pti
859 void CalEpaisseurAtdtExp_et_vol_pti(const double& epaisseur0, const ParaAlgoControle & pa
860 ,const double & epaisseur_t, PtIntegMecaInterne & ptIntegMeca
861 ,double & epaisseur_tdt, const Met_abstraite::Expli_t_tdt& ex);
862 // calcul de la nouvelle épaisseur (sans raideur) avec métrique en implicite
863 // mise à jour du volume au pti
864 void CalEpaisseurAtdtImp_et_vol_pti(const double& epaisseur0, const ParaAlgoControle & pa
865 ,const double & epaisseur_t, PtIntegMecaInterne & ptIntegMeca
866 ,double & epaisseur_tdt, const Met_abstraite::Implicite_t_tdt& ex);
867 // calcul de la nouvelle épaisseur moyenne finale (sans raideur)
868 // mise à jour des volumes aux pti
869 // ramène l'épaisseur moyenne calculée à atdt
870 const double& CalEpaisseurMoyenne_et_vol_pti(bool atdt);
871
872 //test si le jacobien due aux gi finaux est très différent du jacobien de la facette
873 bool Delta_Jacobien_anormal(const Deformation::SaveDefResul* don
874 ,int nisur,int niepais);
875
876
877 };
878 /// @} // end of group
879 #endif
880
881
882
883

```

## 7.207 TriaQSfe1.h

```

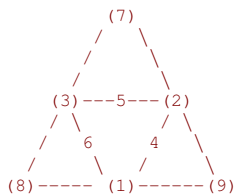
1 // FICHER : TriaQSfel.h
2 // CLASSE : TriaQSfel
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      1/03/2021      *
34 *
35 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)      *
36 *

```

```

37 *   PROJET:      Herezh++
38 *
39 *
40 *   BUT: Element triangulaire,Sfel avec une interpolation quadratique*
41 *   de la partie membrane du tenseur de déformation. Les noeuds *
42 *   concernés sont ceux du triangle central, au nombre de 6.
43 *   Le calcul de la normale est identique à celui de l'élément SFE1 *
44 *   avec membrane linéaire.
45 *   Le jacobien est celui de la facette centrale à 6 noeuds.
46 *
47 *   *****
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !   but
51 *   -----
52 *   !           !           !
53 *   $
54 *   *****
55 *   MODIFICATIONS:
56 *   ! date !   auteur !   but
57 *   -----
58 *   $
59 *
60
61 // -----classe pour un calcul de mecanique-----
62
63 /*
64 //
65 // l'interpolation est QSFE1,
66 //
67 //
68 //
69 //
70 //
71 //
72 //
73 //
74 //
75 */
76
77
78 #ifndef TRIAQSFE1_H
79 #define TRIAQSFE1_H
80
81 #include "ParaGlob.h"
82 #include "ElemMeca.h"
83 #include "Met_abstraite.h"
84 #include "GeomTriangle.h"
85 #include "Noeud.h"
86 #include "UtilLecture.h"
87 #include "Tenseur.h"
88 #include "NevezTenseur.h"
89 #include "Deformation.h"
90 #include "SfeMembT.h"
91 #include "ElFrontiere.h"
92 #include "FrontSegQuad.h"
93 #include "FrontTriaQuad.h"
94
95
96 /// @addtogroup groupe_des_elements_finis
97 /// @{
98 ///
99
100
101 class TriaQSfel : public SfeMembT
102 {
103     public :
104
105         // CONSTRUCTEURS :
106         // Constructeur par default
107         TriaQSfel ();
108
109         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
110         // d'identification
111         TriaQSfel (double epaiss,int num_mail=0,int num_id=-3);
112
113         // Constructeur fonction d'un numero de maillage et d'identification
114         TriaQSfel (int num_mail,int num_id);
115
116         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
117         // du tableau de connexite des noeuds
118         TriaQSfel (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
119
120         // Constructeur de copie
121         TriaQSfel (const TriaQSfel& tria);
122
123

```



```

124
125 // DESTRUCTEUR :
126 ~TriaQSfel ();
127
128 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
129 // méthode virtuelle
130 Element* Nevez_copie() const { Element * el= new TriaQSfel(*this); return el;};
131
132 // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaQSfel
133 TriaQSfel& operator= (TriaQSfel& tria);
134
135 // METHODES :
136 // 1) derivant des virtuelles pures
137
138 // affichage dans la sortie transmise, des variables duales "nom"
139 // aux differents points d'integration
140 // dans le cas ou nom est vide, affichage de "toute" les variables
141 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
142
143 // 2) derivant des virtuelles
144 // 3) methodes propres a l'element
145
146 // les coordonnees des points d'integration dans l'epaisseur
147 inline double KSI(int i) { return doCoSfel->segment.KSI(i);};
148
149 protected :
150
151 // adressage des frontieres linéiques et surfacique
152 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
153 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
154 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
155 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
156 { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
157
158 // VARIABLES PRIVEES :
159 // place memoire commune a tous les elements
160 static SfeMembT::DonnComSfe * doCoSfel;
161 // idem mais pour les indicateurs qui servent pour l'initialisation
162 static SfeMembT::UneFois uneFoisSfel;
163
164 class NombresConstruireTriaQSfel : public NombresConstruire
165 { public: NombresConstruireTriaQSfel();
166 };
167 static NombresConstruireTriaQSfel nombre_V; // les nombres propres à l'élément
168
169 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
170 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
171 class ConsTriaQSfel : public ConstrucElement
172 { public : ConsTriaQSfel ()
173 { NouvelleTypeElement nouv (TRIANGLE,QSFE1,MECA_SOLIDE_DEFORMABLE,this);
174 if (ParaGlob::NiveauImpression() >= 4)
175 cout << "\n initialisation TriaQSfel" << endl;
176 Element::listTypeElement.push_back(nouv);
177 };
178 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
179 {Element * pt;
180 pt = new TriaQSfel (num_maill,num) ;
181 return pt;};
182 // ramene true si la construction de l'element est possible en fonction
183 // des variables globales actuelles: ex en fonction de la dimension
184 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
185 };
186 static ConsTriaQSfel consTriaQSfel;
187 };
188 /// @} // end of group
189 #endif
190
191
192
193

```

## 7.208 TriaQSfe3.h

```

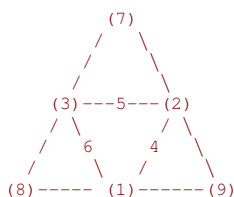
1 // FICHER : TriaQSfe3.h
2 // CLASSE : TriaQSfe3
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //

```

```

13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           28/02/2021
34 *
35 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:        Herezh++
38 *
39 *****/
40 *   BUT: Element triangulaire,Sfe3 avec une interpolation quadratique*
41 *   de la partie membrane du tenseur de déformation. Les noeuds *
42 *   concernés sont ceux du triangle central, au nombre de 6.    *
43 *   Le calcul de la normale est identique à celui de l'élément SFE3 *
44 *   avec membrane linéaire.                                       *
45 *   Il s'effectue à partir de l'interpolation des theta3 relativement*
46 *   aux noeuds sommet du triangle central, ceci via un polynome  *
47 *   quadratique complet.                                          *
48 *   Le jacobien est celui de la facette centrale à 6 noeuds.    *
49 *
50 *   *****
51 *   VERIFICATION:
52 *
53 *   ! date !   auteur !       but
54 *   -----
55 *   !       !       !
56 *
57 *   *****
58 *   MODIFICATIONS:
59 *   ! date !   auteur !       but
60 *   -----
61 *
62 *   *****/
63
64 /*
65 //
66 // l'interpolation est QSFE3,
67 //
68 //
69 //
70 //
71 //
72 //
73 //
74 //
75 //
76 */
77
78
79 #ifndef TRIAQSFE3_H
80 #define TRIAQSFE3_H
81
82 #include "ParaGlob.h"
83 #include "ElemMeca.h"
84 #include "Met_abstraite.h"
85 #include "GeomTriangle.h"
86 #include "Noeud.h"
87 #include "UtilLecture.h"
88 #include "Tenseur.h"
89 #include "NevezTenseur.h"
90 #include "Deformation.h"
91 #include "SfeMembT.h"
92 #include "ElFrontiere.h"
93 #include "FrontSegQuad.h"
94 #include "FrontTriaQuad.h"
95
96
97 /// @addtogroup groupe_des_elements_finis
98 /// @{
99 ///

```



```

100
101
102 class TriaQSfe3 : public SfeMembT
103 {
104
105     public :
106
107         // CONSTRUCTEURS :
108         // Constructeur par défaut
109         TriaQSfe3 ();
110
111         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
112         // d'identification
113         TriaQSfe3 (double epaiss,int num_mail=0,int num_id=-3);
114
115         // Constructeur fonction d'un numero de maillage et d'identification
116         TriaQSfe3 (int num_mail,int num_id);
117
118         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
119         // du tableau de connexite des noeuds
120         TriaQSfe3 (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
121
122         // Constructeur de copie
123         TriaQSfe3 (const TriaQSfe3& tria);
124
125
126         // DESTRUCTEUR :
127         ~TriaQSfe3 ();
128
129         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
130         // méthode virtuelle
131         Element* Nevez_copie() const { Element * el= new TriaQSfe3(*this); return el;};
132
133         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaQSfe3
134         TriaQSfe3& operator= (TriaQSfe3& tria);
135
136         // METHODES :
137         // 1) derivant des virtuelles pures
138
139         // affichage dans la sortie transmise, des variables duales "nom"
140         // aux differents points d'integration
141         // dans le cas ou nom est vide, affichage de "toute" les variables
142         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
143
144         // 2) derivant des virtuelles
145         // 3) methodes propres a l'element
146
147         // les coordonnees des points d'integration dans l'epaisseur
148         inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
149
150     protected :
151
152         // adressage des frontieres linéiques et surfacique
153         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
154         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
155         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
156         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
157         { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
158
159         // VARIABLES PRIVEES :
160         // place memoire commune a tous les elements TriaQSfe3
161         static SfeMembT::DonnComSfe * doCoSfe3;
162         // idem mais pour les indicateurs qui servent pour l'initialisation
163         static SfeMembT::UneFois uneFoisSfe3;
164
165         class NombresConstruireTriaQSfe3 : public NombresConstruire
166         { public: NombresConstruireTriaQSfe3();
167         };
168         static NombresConstruireTriaQSfe3 nombre_V; // les nombres propres à l'élément
169
170         // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
171         //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
172         class ConsTriaQSfe3 : public ConstrucElement
173         { public : ConsTriaQSfe3 ()
174         { NouvelleTypeElement nouv (TRIANGLE,QSFE3,MECA_SOLIDE_DEFORMABLE,this);
175         if (ParaGlob::NiveauImpression() >= 4)
176         cout << "\n initialisation TriaQSfe3" << endl;
177         Element::listTypeElement.push_back(nouv);
178         };
179         Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
180         {Element * pt;
181         pt = new TriaQSfe3 (num_maill,num) ;
182         return pt;};
183         // ramene true si la construction de l'element est possible en fonction
184         // des variables globales actuelles: ex en fonction de la dimension
185         bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
186     };

```

```

187         static ConsTriaQSfe3 consTriaQSfe3;
188     };
189     /// @} // end of group
190 #endif
191
192
193
194

```

## 7.209 TriaSfe1.h

```

1 // FICHER : TriaSfe1.h
2 // CLASSE : TriaSfe1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *
34 *   DATE:           15/01/97
35 *
36 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:        Herezh++
39 *
40 *
41 *   BUT:           Element triangulaire, Sfe 2 pt d'integ.
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   -----
49 *   !           !           !
50 *
51 *   *****
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *
58 *
59 *****/
60
61 // -----classe pour un calcul de mecanique-----
62
63 /*
64 // La classe TriaSfe1 permet de declarer des elements triangulaire Sfe et de realiser
65 // le calcul du residu local et de la raideur locale pour une loi de comportement
66 // donnee. La dimension de l'espace pour un tel element est 2
67 //
68 // l'interpolation est SFE, le nombre de point d'integration est de 1 en surface
69 // et de deux dans l'epaisseur
70 //
71 //
72 //
73 //

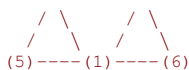
```

(4)  
 / \  
 / \  
 /-----\  
 (3)----- (2)

```

74 //
75 //
76 //
77 */
78
79
80 #ifndef TRIASFEL_H
81 #define TRIASFEL_H
82
83 #include "ParaGlob.h"
84 #include "ElemMeca.h"
85 #include "Met_abstraite.h"
86 #include "GeomTriangle.h"
87 #include "Noeud.h"
88 #include "UtilLecture.h"
89 #include "Tenseur.h"
90 #include "NevezTenseur.h"
91 #include "Deformation.h"
92 #include "SfeMembT.h"
93 #include "ElFrontiere.h"
94 #include "FrontSegLine.h"
95 #include "FrontTriaLine.h"
96
97
98 /// @addtogroup groupe_des_elements_finis
99 /// @{
100 ///
101
102
103 class TriaSfel : public SfeMembT
104 {
105
106     public :
107
108         // CONSTRUCTEURS :
109         // Constructeur par default
110         TriaSfel ();
111
112         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
113         // d'identification
114         TriaSfel (double epaiss,int num_mail=0,int num_id=-3);
115
116         // Constructeur fonction d'un numero de maillage et d'identification
117         TriaSfel (int num_mail,int num_id);
118
119         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
120         // du tableau de connexite des noeuds
121         TriaSfel (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
122
123         // Constructeur de copie
124         TriaSfel (const TriaSfel& tria);
125
126
127         // DESTRUCTEUR :
128         ~TriaSfel ();
129
130         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
131         // méthode virtuelle
132         Element* Nevez_copie() const { Element * el= new TriaSfel(*this); return el;};
133
134         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfel
135         TriaSfel& operator= (TriaSfel& tria);
136
137         // METHODES :
138         // 1) derivant des virtuelles pures
139
140         // affichage dans la sortie transmise, des variables duales "nom"
141         // aux differents points d'integration
142         // dans le cas ou nom est vide, affichage de "toute" les variables
143         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
144
145         // 2) derivant des virtuelles
146         // 3) methodes propres a l'element
147
148         // les coordonnees des points d'integration dans l'epaisseur
149         inline double KSI(int i) { return doCoSfel->segment.KSI(i);};
150
151     protected :
152
153         // adressage des frontieres linéiques et surfacique
154         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
155         ElFrontiere* new_frontiere_lin(int Tableau <Noeud *> & tab, DdlElement& ddelem)
156         { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
157         ElFrontiere* new_frontiere_surf(int Tableau <Noeud *> & tab, DdlElement& ddelem)
158         { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
159
160     // VARIABLES PRIVEES :

```





```

161         // place memoire commune a tous les elements TriaMembL1
162         static SfeMembT::DonnComSfe * doCoSfe1;
163         // idem mais pour les indicateurs qui servent pour l'initialisation
164         static SfeMembT::UneFois uneFoisSfe1;
165
166         class NombresConstruireTriaSfe1 : public NombresConstruire
167         { public: NombresConstruireTriaSfe1();
168         };
169         static NombresConstruireTriaSfe1 nombre_V; // les nombres propres à l'élément
170
171         // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
172         //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
173         class ConsTriaSfe1 : public ConstrucElement
174         { public : ConsTriaSfe1 ()
175         { NouvelleTypeElement nouv (TRIANGLE,SFE1,MECA_SOLIDE_DEFORMABLE,this);
176           if (ParaGlob::NiveauImpression() >= 4)
177             cout << "\n initialisation TriaSfe1" << endl;
178             Element::listTypeElement.push_back(nouv);
179         };
180         Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
181         {Element * pt;
182           pt = new TriaSfe1 (num_maill,num) ;
183           return pt;};
184         // ramene true si la construction de l'element est possible en fonction
185         // des variables globales actuelles: ex en fonction de la dimension
186         bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
187         };
188         static ConsTriaSfe1 consTriaSfe1;
189 };
190 /// @} // end of group
191 #endif
192
193
194
195

```

## 7.210 TriaSfe1\_cm5pti.h

```

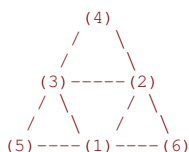
1 // FICHER : TriaSfe1_cm5pti.h
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *      DATE:      06/03/2023
34 *
35 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 *      BUT:      Element triangulaire,Sfe 5 pt d'integ.
41 *
42 *      *****
43 *
44 *      VERIFICATION:
45 *      ! date !   auteur !           but           !
46 *      -----

```

```

47 *      !      !      !      !      *
48 *      *      *      *      *
49 *      *      *      *      *
50 *      *      *      *      *
51 *      *      *      *      *
52 *      *      *      *      *
53 *      *      *      *      *
54 *      *      *      *      *
55 *      *      *      *      *
56 *      *      *      *      *
57 *      *      *      *      *
58 *      *      *      *      *
59
60 // -----classe pour un calcul de mecanique-----
61
62 /*
63 // La classe TriaSfel permet de declarer des elements triangulaire Sfe et de realiser
64 // le calcul du residu local et de la raideur locale pour une loi de comportement
65 // donnee. La dimension de l'espace pour un tel element est 2
66 //
67 // l'interpolation est SFE, le nombre de point d'integration est de 1 en surface
68 // et de 5 dans l'epaisseur
69 //
70 //
71 //
72 //
73 //
74 //
75 //
76 */
77
78
79 #ifndef TRIASFEL_5PTI_H
80 #define TRIASFEL_5PTI_H
81
82 #include "ParaGlob.h"
83 #include "ElemMeca.h"
84 #include "Met_abstraite.h"
85 #include "GeomTriangle.h"
86 #include "Noeud.h"
87 #include "UtilLecture.h"
88 #include "Tenseur.h"
89 #include "NevezTenseur.h"
90 #include "Deformation.h"
91 #include "SfeMembT.h"
92 #include "ElFrontiere.h"
93 #include "FrontSegLine.h"
94 #include "FrontTriaLine.h"
95
96
97 /// @addtogroup groupe_des_elements_finis
98 /// @{
99 ///
100
101
102 class TriaSfel_cm5pti : public SfeMembT
103 {
104
105     public :
106
107         // CONSTRUCTEURS :
108         // Constructeur par default
109         TriaSfel_cm5pti ();
110
111         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
112         // d'identification
113         TriaSfel_cm5pti (double epaiss,int num_mail=0,int num_id=-3);
114
115         // Constructeur fonction d'un numero de maillage et d'identification
116         TriaSfel_cm5pti (int num_mail,int num_id);
117
118         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
119         // du tableau de connexite des noeuds
120         TriaSfel_cm5pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
121
122         // Constructeur de copie
123         TriaSfel_cm5pti (const TriaSfel_cm5pti& tria);
124
125
126         // DESTRUCTEUR :
127         ~TriaSfel_cm5pti ();
128
129         // creation d'un element de copie: utilisation de l'operateur new et du constructeur de copie
130         // methode virtuelle
131         Element* Nevez_copie() const { Element * el= new TriaSfel_cm5pti(*this); return el;};
132
133         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfel_cm5pti

```



```

134     TriaSfel_cm5pti& operator= (TriaSfel_cm5pti& tria);
135
136     // METHODES :
137 // 1) derivant des virtuelles pures
138
139 // affichage dans la sortie transmise, des variables duales "nom"
140 // aux differents points d'integration
141 // dans le cas ou nom est vide, affichage de "toute" les variables
142 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
143
144 // 2) derivant des virtuelles
145 // 3) methodes propres a l'element
146
147 // les coordonnees des points d'integration dans l'epaisseur
148 inline double KSI(int i) { return doCoSfel->segment.KSI(i);};
149
150 protected :
151
152 // adressage des frontieres linéiques et surfacique
153 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
154 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
155 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
156 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
157 { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
158
159 // VARIABLES PRIVEES :
160 // place memoire commune a tous les elements TriaMembL1
161 static SfeMembT::DonnComSfe * doCoSfel;
162 // idem mais pour les indicateurs qui servent pour l'initialisation
163 static SfeMembT::UneFois uneFoisSfel;
164
165 class NombresConstruireTriaSfel_cm5pti : public NombresConstruire
166 { public: NombresConstruireTriaSfel_cm5pti();
167 };
168 static NombresConstruireTriaSfel_cm5pti nombre_V; // les nombres propres à l'élément
169
170 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
171 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
172 class ConsTriaSfel_cm5pti : public ConstrucElement
173 { public : ConsTriaSfel_cm5pti ()
174 { NouvelleTypeElement nouv(TRIANGLE,SFE1,MECA_SOLIDE_DEFORMABLE,this,"_cm5pti");
175 if (ParaGlob::NiveauImpression() >= 4)
176 cout << "\n initialisation TriaSfel_cm5pti" << endl;
177 Element::listTypeElement.push_back(nouv);
178 };
179 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
180 {Element * pt;
181 pt = new TriaSfel_cm5pti (num_maill,num) ;
182 return pt;};
183 // ramene true si la construction de l'element est possible en fonction
184 // des variables globales actuelles: ex en fonction de la dimension
185 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
186 };
187 static ConsTriaSfel_cm5pti consTriaSfel_cm5pti;
188 };
189 /// @} // end of group
190 #endif
191
192
193
194

```

## 7.211 TriaSfe2.h

```

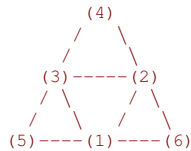
1 // FICHER : TriaSfe2.h
2 // CLASSE : TriaSfe2
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //

```

```

22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      04/07/2007
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      Element triangulaire,Sfe2. Le calcul de la normale
41 *             s'effectue à partir de l'interpolation de normale sur chaque
42 *             arrête. Celles-ci sont obtenue par moyenne pondérée (en fonc-
43 *             tion de la distance du centre de gravité ou en fait de la
44 *             longueur de la hauteur abaissée sur l'arrête) et des normales
45 *             aux triangles de part et autres de l'arrête.
46 *             La position de la normale sur l'arrête est également calculée
47 *             de manière à tenir compte de la distorsion des éléments. La
48 *             est celle exposée par Hervé dans sa thèse.
49 *
50 *   *****
51 *
52 *   VERIFICATION:
53 *   ! date !   auteur !           but
54 *   -----
55 *   !           !           !
56 *
57 *   *****
58 *   MODIFICATIONS:
59 *   ! date !   auteur !           but
60 *   -----
61 *
62 *****/
63
64 /*
65 //
66 // l'interpolation est SFE,
67 //
68 //
69 //
70 //
71 //
72 //
73 //
74 */
75
76
77 #ifndef TRIASFE2_H
78 #define TRIASFE2_H
79
80 #include "ParaGlob.h"
81 #include "ElemMeca.h"
82 #include "Met_abstraite.h"
83 #include "GeomTriangle.h"
84 #include "Noeud.h"
85 #include "UtilLecture.h"
86 #include "Tenseur.h"
87 #include "NevezTenseur.h"
88 #include "Deformation.h"
89 #include "SfeMembT.h"
90 #include "ElFrontiere.h"
91 #include "FrontSegLine.h"
92 #include "FrontTriaLine.h"
93
94
95 /// @addtogroup groupe_des_elements_finis
96 /// @{
97 ///
98
99
100 class TriaSfe2 : public SfeMembT
101 {
102
103     public :
104
105         // CONSTRUCTEURS :
106         // Constructeur par default
107         TriaSfe2 ();

```



```

108
109 // Constructeur fonction d'une epaisseur et eventuellement d'un numero
110 // d'identification
111 TriaSfe2 (double epaiss,int num_mail=0,int num_id=-3);
112
113 // Constructeur fonction d'un numero de maillage et d'identification
114 TriaSfe2 (int num_mail,int num_id);
115
116 // Constructeur fonction d'une epaisseur, d'un numero d'identification,
117 // du tableau de connexite des noeuds
118 TriaSfe2 (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
119
120 // Constructeur de copie
121 TriaSfe2 (const TriaSfe2& tria);
122
123
124 // DESTRUCTEUR :
125 ~TriaSfe2 ();
126
127 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
128 // méthode virtuelle
129 Element* Nevez_copie() const { Element * el= new TriaSfe2(*this); return el;};
130
131 // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe2
132 TriaSfe2& operator= (TriaSfe2& tria);
133
134 // METHODES :
135 // 1) derivant des virtuelles pures
136
137 // affichage dans la sortie transmise, des variables duales "nom"
138 // aux differents points d'integration
139 // dans le cas ou nom est vide, affichage de "toute" les variables
140 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
141
142 // 2) derivant des virtuelles
143 // 3) methodes propres a l'element
144
145 // les coordonnees des points d'integration dans l'epaisseur
146 inline double KSI(int i) { return doCoSfe2->segment.KSI(i);};
147
148 protected :
149
150 // adressage des frontieres linéiques et surfacique
151 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
152 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
153 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
154 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
155 { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
156
157 // VARIABLES PRIVEES :
158 // place memoire commune a tous les elements TriaMembL1
159 static SfeMembT::DonnComSfe * doCoSfe2;
160 // idem mais pour les indicateurs qui servent pour l'initialisation
161 static SfeMembT::UneFois uneFoisSfe2;
162
163 class NombresConstruireTriaSfe2 : public NombresConstruire
164 { public: NombresConstruireTriaSfe2();
165 };
166 static NombresConstruireTriaSfe2 nombre_V; // les nombres propres à l'élément
167
168 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
169 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
170 class ConsTriaSfe2 : public ConstrucElement
171 { public : ConsTriaSfe2 ()
172 { NouvelleTypeElement nouv (TRIANGLE,SFE2,MECA_SOLIDE_DEFORMABLE,this);
173 if (ParaGlob::NiveauImpression() >= 4)
174 cout << "\n initialisation TriaSfe2" << endl;
175 Element::listTypeElement.push_back(nouv);
176 };
177 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
178 {Element * pt;
179 pt = new TriaSfe2 (num_maill,num) ;
180 return pt;};
181 // ramene true si la construction de l'element est possible en fonction
182 // des variables globales actuelles: ex en fonction de la dimension
183 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
184 };
185 static ConsTriaSfe2 consTriaSfe2;
186 };
187 /// @} // end of group
188 #endif
189
190
191
192

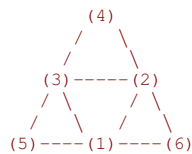
```

## 7.212 TriaSfe3.h

```

1 // FICHER : TriaSfe3.h
2 // CLASSE : TriaSfe3
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          04/07/2007
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *****/
40 *      BUT:           Element triangulaire,Sfe3C. Le calcul de la normale
41 *      s'effectue à partir de l'interpolation des theta3 relativement
42 *      au triangle central, ceci via un polynome quadratique complet.
43 *      Le jacobien est celui de la facette plane centrale.
44 *
45 *      *****
46 *      VERIFICATION:
47 *
48 *      ! date !   auteur !           but
49 *      -----
50 *      !           !           !
51 *
52 *      *****
53 *      MODIFICATIONS:
54 *      ! date !   auteur !           but
55 *      -----
56 *
57 *****/
58
59 /*
60 //
61 // l'interpolation est SFE,
62 //
63 //
64 //
65 //
66 //
67 //
68 //
69 */
70
71
72 #ifndef TRIASF3_H
73 #define TRIASF3_H
74
75 #include "ParaGlob.h"
76 #include "ElemMeca.h"
77 #include "Met_abstraite.h"
78 #include "GeomTriangle.h"
79 #include "Noeud.h"
80 #include "UtilLecture.h"
81 #include "Tenseur.h"
82 #include "NevezTenseur.h"
83 #include "Deformation.h"
84 #include "SfeMembT.h"
85 #include "ElFrontiere.h"

```



```

86 #include "FrontSegLine.h"
87 #include "FrontTriaLine.h"
88
89
90 /// @addtogroup groupe_des_elements_finis
91 /// @{
92 ///
93
94
95 class TriaSfe3 : public SfeMembT
96 {
97
98     public :
99
100         // CONSTRUCTEURS :
101         // Constructeur par défaut
102         TriaSfe3 ();
103
104         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
105         // d'identification
106         TriaSfe3 (double epaiss,int num_mail=0,int num_id=-3);
107
108         // Constructeur fonction d'un numero de maillage et d'identification
109         TriaSfe3 (int num_mail,int num_id);
110
111         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
112         // du tableau de connexite des noeuds
113         TriaSfe3 (double epaiss,int num_mail,int num_id,const Tableau<Noeud *> & tab);
114
115         // Constructeur de copie
116         TriaSfe3 (const TriaSfe3& tria);
117
118
119         // DESTRUCTEUR :
120         ~TriaSfe3 ();
121
122         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
123         // méthode virtuelle
124         Element* Nevez_copie() const { Element * el= new TriaSfe3(*this); return el;};
125
126         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe3
127         TriaSfe3& operator= (TriaSfe3& tria);
128
129         // METHODES :
130 // 1) derivant des virtuelles pures
131
132         // affichage dans la sortie transmise, des variables duales "nom"
133         // aux differents points d'integration
134         // dans le cas ou nom est vide, affichage de "toute" les variables
135         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
136
137 // 2) derivant des virtuelles
138 // 3) methodes propres a l'element
139
140         // les coordonnees des points d'integration dans l'epaisseur
141         inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
142
143     protected :
144
145         // adressage des frontieres linéiques et surfacique
146         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
147         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
148         { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
149         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
150         { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
151
152     // VARIABLES PRIVEES :
153     // place memoire commune a tous les elements TriaMembL1
154     static SfeMembT::DonnComSfe * doCoSfe3;
155     // idem mais pour les indicateurs qui servent pour l'initialisation
156     static SfeMembT::UneFois uneFoisSfe3;
157
158     class NombresConstruireTriaSfe3 : public NombresConstruire
159     { public: NombresConstruireTriaSfe3();
160     };
161     static NombresConstruireTriaSfe3 nombre_V; // les nombres propres à l'élément
162
163     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
164     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
165     class ConsTriaSfe3 : public ConstrucElement
166     { public : ConsTriaSfe3 ()
167     { NouvelleTypeElement nouv (TRIANGLE,SFE3,MECA_SOLIDE_DEFORMABLE,this);
168     if (ParaGlob::NiveauImpression() >= 4)
169     cout << "\n initialisation TriaSfe3" << endl;
170     Element::listTypeElement.push_back(nouv);
171     };
172     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien

```

```

173         {Element * pt;
174         pt = new TriaSfe3 (num_maill,num) ;
175         return pt;};
176         // ramene true si la construction de l'element est possible en fonction
177         // des variables globales actuelles: ex en fonction de la dimension
178         bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
179     };
180     static ConsTriaSfe3 consTriaSfe3;
181 };
182 /// @} // end of group
183 #endif
184
185
186
187

```

## 7.213 TriaSfe3\_3D.h

```

1 // FICHER : TriaSfe3_3D.h
2 // CLASSE : TriaSfe3_3D
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      07/07/2008
34 *
35 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      BUT:      Element triangulaire,3DSfe3C. Le calcul de la normale
41 *               s'effectue à partir de l'interpolation des theta3 relativement
42 *               au triangle central, ceci via un polynome quadratique complet.
43 *               Le jacobien est celui de la facette plane centrale.
44 *               Par rapport à l'élément TriaSfe3, cette élément utilise une
45 *               épaisseur définie aux noeuds, la métrique associée est 3D
46 *               Les lois de comportement sont donc en 3D
47 *
48 *               *****
49 *               VERIFICATION:
50 *
51 *               ! date ! auteur ! but
52 *               -----
53 *               ! ! !
54 *               $
55 *               *****
56 *               MODIFICATIONS:
57 *               ! date ! auteur ! but
58 *               -----
59 *               $
60 *****/
61
62 /*
63 //
64 // l'interpolation est SFE,
65 //
66 //
67 //

```

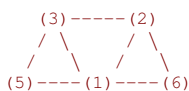
(4)



```

68 //
69 //
70 //
71 //
72 */
73
74
75 #ifndef TRIASF3_3D_H
76 #define TRIASF3_3D_H
77
78 #include "ParaGlob.h"
79 #include "ElemMeca.h"
80 #include "Met_abstraite.h"
81 #include "GeomTriangle.h"
82 #include "Noeud.h"
83 #include "UtilLecture.h"
84 #include "Tenseur.h"
85 #include "NevezTenseur.h"
86 #include "Deformation.h"
87 #include "SfeMembT.h"
88 #include "ElFrontiere.h"
89 #include "FrontSegLine.h"
90 #include "FrontTriaLine.h"
91
92
93 /// @addtogroup groupe_des_elements_finis
94 /// @{
95 ///
96
97
98 class TriaSfe3_3D : public SfeMembT
99 {
100
101     public :
102
103         // CONSTRUCTEURS :
104         // Constructeur par défaut
105         TriaSfe3_3D ();
106
107         // Constructeur fonction d'un numero de maillage et d'identification
108         TriaSfe3_3D (int num_mail,int num_id);
109
110         // Constructeur fonction d'un numero d'identification,
111         // du tableau de connexite des noeuds
112         TriaSfe3_3D (int num_mail,int num_id,const Tableau<Noeud *>& tab);
113
114         // Constructeur de copie
115         TriaSfe3_3D (const TriaSfe3_3D& tria);
116
117
118         // DESTRUCTEUR :
119         ~TriaSfe3_3D ();
120
121         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
122         // méthode virtuelle
123         Element* Nevez_copie() const { Element * el= new TriaSfe3_3D(*this); return el;};
124
125         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe3_3D
126         TriaSfe3_3D& operator= (TriaSfe3_3D& tria);
127
128         // METHODES :
129 // 1) derivant des virtuelles pures
130
131         // affichage dans la sortie transmise, des variables duales "nom"
132         // aux differents points d'integration
133         // dans le cas ou nom est vide, affichage de "toute" les variables
134         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
135
136 // 2) derivant des virtuelles
137 // 3) methodes propres a l'element
138
139         // les coordonnees des points d'integration dans l'epaisseur
140         inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
141
142     protected :
143
144         // adressage des frontieres linéiques et surfacique
145         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
146         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
147         { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
148         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
149         { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
150
151         // VARIABLES PRIVEES :
152         // place memoire commune a tous les elements TriaSfe3_3D
153         static SfeMembT::DonnComSfe * doCoSfe3;
154         // idem mais pour les indicateurs qui servent pour l'initialisation

```



```

155     static SfeMembT::UneFois  uneFoisSfe3;
156
157     class NombresConstruireTriaSfe3_3D : public NombresConstruire
158     { public: NombresConstruireTriaSfe3_3D();
159     };
160     static NombresConstruireTriaSfe3_3D nombre_V; // les nombres propres à l'élément
161
162     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
163     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
164     class ConsTriaSfe3_3D : public ConstrucElement
165     { public : ConsTriaSfe3_3D ()
166     { NouvelleTypeElement nouv (TRIANGLE,SFE3_3D,MECA_SOLIDE_DEFORMABLE,this);
167       if (ParaGlob::NiveauImpression() >= 4)
168         cout << "\n initialisation TriaSfe3_3D" << endl;
169       Element::listTypeElement.push_back(nouv);
170     };
171     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
172     {Element * pt;
173       pt = new TriaSfe3_3D (num_maill,num) ;
174       return pt;};
175     // ramene true si la construction de l'element est possible en fonction
176     // des variables globales actuelles: ex en fonction de la dimension
177     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
178     };
179     static ConsTriaSfe3_3D consTriaSfe3_3D;
180 };
181 /// @} // end of group
182 #endif
183
184
185
186

```

## 7.214 TriaSfe3\_cm12pti.h

```

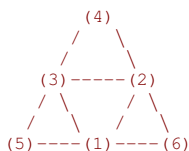
1 // FICHER : TriaSfe3_cm6pti.h
2 // CLASSE : TriaSfe3_cm6pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          04/07/2007
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *
40 *      BUT:           Element triangulaire,Sfe3C. Le calcul de la normale
41 *                    s'effectue à partir de l'interpolation des theta3 relativement
42 *                    au triangle central, ceci via un polynome quadratique complet.
43 *                    Le jacobien est celui de la facette plane centrale.
44 *                    Ici le nombre de point d'intégration est de 2 en épaisseur
45 *                    et 3 en surface.
46 *
47 *                    *****
48 *
49 *      VERIFICATION:
50 *

```

```

50 *      ! date !   auteur !           but           ! *
51 *      -----
52 *      !           !           !           !           *
53 *      $ *
54 *      ***** *
55 *      MODIFICATIONS: *
56 *      ! date !   auteur !           but           ! *
57 *      -----
58 *      $ *
59 *****/
60
61 //
62 // l'interpolation est SFE,
63 //
64 //
65 //
66 //
67 //
68 //
69 //
70
71
72
73 #ifndef TRIASF3_CM12PTI_H
74 #define TRIASF3_CM12PTI_H
75
76 #include "ParaGlob.h"
77 #include "ElemMeca.h"
78 #include "Met_abstraite.h"
79 #include "GeomTriangle.h"
80 #include "Noeud.h"
81 #include "UtilLecture.h"
82 #include "Tenseur.h"
83 #include "NevezTenseur.h"
84 #include "Deformation.h"
85 #include "SfeMembT.h"
86 #include "ElFrontiere.h"
87 #include "FrontSegLine.h"
88 #include "FrontTriaLine.h"
89
90
91 /// @addtogroup groupe_des_elements_finis
92 /// @{
93 ///
94
95
96 class TriaSfe3_cm12pti : public SfeMembT
97 {
98
99     public :
100
101         // CONSTRUCTEURS :
102         // Constructeur par défaut
103         TriaSfe3_cm12pti ();
104
105         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
106         // d'identification
107         TriaSfe3_cm12pti (double epaiss,int num_mail=0,int num_id=-3);
108
109         // Constructeur fonction d'un numero de maillage et d'identification
110         TriaSfe3_cm12pti (int num_mail,int num_id);
111
112         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
113         // du tableau de connexite des noeuds
114         TriaSfe3_cm12pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
115
116         // Constructeur de copie
117         TriaSfe3_cm12pti (const TriaSfe3_cm12pti& tria);
118
119
120         // DESTRUCTEUR :
121         ~TriaSfe3_cm12pti ();
122
123         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
124         // méthode virtuelle
125         Element* Nevez_copie() const { Element * el= new TriaSfe3_cm12pti(*this); return el;};
126
127         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe3_cm12pti
128         TriaSfe3_cm12pti& operator= (TriaSfe3_cm12pti& tria);
129
130         // METHODES :
131         // 1) derivant des virtuelles pures
132
133         // affichage dans la sortie transmise, des variables duales "nom"
134         // aux differents points d'integration
135         // dans le cas ou nom est vide, affichage de "toute" les variables
136         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);

```



```

137
138 // 2) derivant des virtuelles
139 // 3) methodes propres a l'element
140
141 // les coordonnees des points d'integration dans l'epaisseur
142 inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
143
144 protected :
145
146 // adressage des frontieres linéiques et surfacique
147 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
148 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
149 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
150 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
151 { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
152
153 // VARIABLES PRIVEES :
154 // place memoire commune a tous les elements TriaMembl1
155 static SfeMemblT::DonnComSfe * doCoSfe3;
156 // idem mais pour les indicateurs qui servent pour l'initialisation
157 static SfeMemblT::UneFois uneFoisSfe3;
158
159 class NombresConstruireTriaSfe3_cm12pti : public NombresConstruire
160 { public: NombresConstruireTriaSfe3_cm12pti();
161 };
162 static NombresConstruireTriaSfe3_cm12pti nombre_V; // les nombres propres à l'élément
163
164 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
165 //ajout de l'element dans la liste : listTypeElement, geree par la class Element
166 class ConsTriaSfe3_cm12pti : public ConstrucElement
167 { public : ConsTriaSfe3_cm12pti ()
168 { NouvelleTypeElement nouv(TRIANGLE,SFE3,MECA_SOLIDE_DEFORMABLE,this,"_cm12pti");
169 if (ParaGlob::NiveauImpression() >= 4)
170 cout << "\n initialisation TriaSfe3_cm12pti" << endl;
171 Element::listTypeElement.push_back(nouv);
172 };
173 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
174 {Element * pt;
175 pt = new TriaSfe3_cm12pti (num_maill,num) ;
176 return pt;};
177 // ramene true si la construction de l'element est possible en fonction
178 // des variables globales actuelles: ex en fonction de la dimension
179 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
180 };
181 static ConsTriaSfe3_cm12pti consTriaSfe3_cm12pti;
182 };
183 /// @} // end of group
184 #endif
185
186
187
188

```

## 7.215 TriaSfe3\_cm13pti.h

```

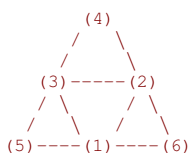
1 // FICHER : TriaSfe3_cm13pti.h
2 // CLASSE : TriaSfe3_cm13pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.

```

```

31
32 /*****
33 *   DATE:      13/11/2012
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *   ****
40 *   BUT:      Element triangulaire,Sfe3. Le calcul de la normale
41 *             s'effectue à partir de l'interpolation des theta3 relativement
42 *             au triangle central, ceci via un polynome quadratique complet.
43 *             Le jacobien est celui de la facette plane centrale.
44 *             Ici le nombre de point d'intégration est de 13 en épaisseur
45 *             et un en surface. Le type est Gauss Lobatto
46 *
47 *             *****
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !       !           !
53 *
54 *             *****
55 *   MODIFICATIONS:
56 *   ! date !   auteur !           but
57 *   -----
58 *
59 *             *****/
60
61 //
62 // l'interpolation est SFE,
63 //
64 //
65 //
66 //
67 //
68 //
69 //
70 //
71 //
72 //
73 #ifndef TRIASF3_CM13PTI_H
74 #define TRIASF3_CM13PTI_H
75
76 #include "ParaGlob.h"
77 #include "ElemMeca.h"
78 #include "Met_abstraite.h"
79 #include "GeomTriangle.h"
80 #include "Noeud.h"
81 #include "UtilLecture.h"
82 #include "Tenseur.h"
83 #include "NevezTenseur.h"
84 #include "Deformation.h"
85 #include "SfeMembT.h"
86 #include "ElFrontiere.h"
87 #include "FrontSegLine.h"
88 #include "FrontTriaLine.h"
89
90
91 /// @addtogroup groupe_des_elements_finis
92 /// @{
93 ///
94
95
96 class TriaSfe3_cm13pti : public SfeMembT
97 {
98
99     public :
100
101         // CONSTRUCTEURS :
102         // Constructeur par défaut
103         TriaSfe3_cm13pti ();
104
105         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
106         // d'identification
107         TriaSfe3_cm13pti (double epaiss,int num_mail=0,int num_id=-3);
108
109         // Constructeur fonction d'un numero de maillage et d'identification
110         TriaSfe3_cm13pti (int num_mail,int num_id);
111
112         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
113         // du tableau de connexite des noeuds
114         TriaSfe3_cm13pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
115
116         // Constructeur de copie
117         TriaSfe3_cm13pti (const TriaSfe3_cm13pti& tria);

```



```

118
119
120 // DESTRUCTEUR :
121 ~TriaSfe3_cm13pti ();
122
123 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
124 // méthode virtuelle
125 Element* Nevez_copie() const { Element * el= new TriaSfe3_cm13pti(*this); return el;};
126
127 // Surcharge de l'opérateur = : realise l'egalite entre deux instances de TriaSfe3_cm13pti
128 TriaSfe3_cm13pti& operator= (TriaSfe3_cm13pti& tria);
129
130 // METHODES :
131 // 1) derivant des virtuelles pures
132
133 // affichage dans la sortie transmise, des variables duales "nom"
134 // aux differents points d'integration
135 // dans le cas ou nom est vide, affichage de "toute" les variables
136 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
137
138 // 2) derivant des virtuelles
139 // 3) methodes propres a l'element
140
141 // les coordonnees des points d'integration dans l'epaisseur
142 inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
143
144 protected :
145
146 // adressage des frontieres linéiques et surfacique
147 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
148 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
149 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
150 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
151 { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
152
153 // VARIABLES PRIVEES :
154 // place memoire commune a tous les elements TriaMembl1
155 static SfeMemblT::DonnComSfe * doCoSfe3;
156 // idem mais pour les indicateurs qui servent pour l'initialisation
157 static SfeMemblT::UneFois uneFoisSfe3;
158
159 class NombresConstruireTriaSfe3_cm13pti : public NombresConstruire
160 { public: NombresConstruireTriaSfe3_cm13pti();
161 };
162 static NombresConstruireTriaSfe3_cm13pti nombre_V; // les nombres propres à l'élément
163
164 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
165 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
166 class ConsTriaSfe3_cm13pti : public ConstrucElement
167 { public : ConsTriaSfe3_cm13pti ()
168 { NouvelleTypeElement nouv (TRIANGLE,SFE3,MECA_SOLIDE_DEFORMABLE,this,"_cm13pti");
169 if (ParaGlob::NiveauImpression() >= 4)
170 cout << "\n initialisation TriaSfe3_cm13pti" << endl;
171 Element::listTypeElement.push_back(nouv);
172 };
173 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
174 {Element * pt;
175 pt = new TriaSfe3_cm13pti (num_maill,num) ;
176 return pt;};
177 // ramene true si la construction de l'element est possible en fonction
178 // des variables globales actuelles: ex en fonction de la dimension
179 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
180 };
181 static ConsTriaSfe3_cm13pti consTriaSfe3_cm13pti;
182 };
183 /// @} // end of group
184 #endif
185
186
187
188

```

## 7.216 TriaSfe3\_cm3pti.h

```

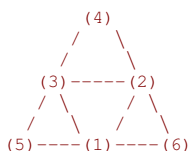
1 // FICHER : TriaSfe3_cm3pti.h
2 // CLASSE : TriaSfe3_cm3pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.

```

```

12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31 //
32 /*****
33 *   DATE:      13/11/2012
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      Element triangulaire,Sfe3. Le calcul de la normale *
41 *             s'effectue à partir de l'interpolation des theta3 relativement *
42 *             au triangle central, ceci via un polynome quadratique complet. *
43 *             Le jacobien est celui de la facette plane centrale. *
44 *             Ici le nombre de point d'intégration est de 3 en épaisseur *
45 *             et un en surface. Le type est Gauss Lobatto *
46 *
47 *             ***** *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !           !           !
53 *
54 *             ***** *
55 *   MODIFICATIONS:
56 *   ! date !   auteur !           but
57 *   -----
58 *
59 *****/
60
61 /*
62 //
63 // l'interpolation est SFE,
64 //
65 //
66 //
67 //
68 //
69 //
70 //
71 */
72
73
74 #ifndef TRIASFE3_CM3PTI_H
75 #define TRIASFE3_CM3PTI_H
76
77 #include "ParaGlob.h"
78 #include "ElemMeca.h"
79 #include "Met_abstraite.h"
80 #include "GeomTriangle.h"
81 #include "Noeud.h"
82 #include "UtilLecture.h"
83 #include "Tenseur.h"
84 #include "NevezTenseur.h"
85 #include "Deformation.h"
86 #include "SfeMembT.h"
87 #include "ElFrontiere.h"
88 #include "FrontSegLine.h"
89 #include "FrontTriaLine.h"
90
91
92 /// @addtogroup groupe_des_elements_finis
93 /// @{
94 ///
95
96
97 class TriaSfe3_cm3pti : public SfeMembT
98 {

```



```

99
100 public :
101
102     // CONSTRUCTEURS :
103     // Constructeur par default
104     TriaSfe3_cm3pti ();
105
106     // Constructeur fonction d'une epaisseur et eventuellement d'un numero
107     // d'identification
108     TriaSfe3_cm3pti (double epaiss,int num_mail=0,int num_id=-3);
109
110     // Constructeur fonction d'un numero de maillage et d'identification
111     TriaSfe3_cm3pti (int num_mail,int num_id);
112
113     // Constructeur fonction d'une epaisseur, d'un numero d'identification,
114     // du tableau de connexite des noeuds
115     TriaSfe3_cm3pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
116
117     // Constructeur de copie
118     TriaSfe3_cm3pti (const TriaSfe3_cm3pti& tria);
119
120
121     // DESTRUCTEUR :
122     ~TriaSfe3_cm3pti ();
123
124     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
125     // méthode virtuelle
126     Element* Nevez_copie() const { Element * el= new TriaSfe3_cm3pti(*this); return el;};
127
128     // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe3_cm3pti
129     TriaSfe3_cm3pti& operator= (TriaSfe3_cm3pti& tria);
130
131     // METHODES :
132 // 1) derivant des virtuelles pures
133
134     // affichage dans la sortie transmise, des variables duales "nom"
135     // aux differents points d'integration
136     // dans le cas ou nom est vide, affichage de "toute" les variables
137     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
138
139 // 2) derivant des virtuelles
140 // 3) methodes propres a l'element
141
142     // les coordonnees des points d'integration dans l'epaisseur
143     inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
144
145 protected :
146
147     // adressage des frontieres linéiques et surfacique
148     // définit dans les classes dérivées, et utilisées pour la construction des frontieres
149     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
150     { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
151     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
152     { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
153
154 // VARIABLES PRIVEES :
155 // place memoire commune a tous les elements TriaMembL1
156 static SfeMembT::DonnComSfe * doCoSfe3;
157 // idem mais pour les indicateurs qui servent pour l'initialisation
158 static SfeMembT::UneFois uneFoisSfe3;
159
160 class NombresConstruireTriaSfe3_cm3pti : public NombresConstruire
161 { public: NombresConstruireTriaSfe3_cm3pti();
162 };
163 static NombresConstruireTriaSfe3_cm3pti nombre_V; // les nombres propres à l'élément
164
165 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
166 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
167 class ConsTriaSfe3_cm3pti : public ConstrucElement
168 { public : ConsTriaSfe3_cm3pti ()
169     { NouvelleTypeElement nouv (TRIANGLE,SFE3,MECA_SOLIDE_DEFORMABLE,this,"_cm3pti");
170     if (ParaGlob::NiveauImpression() >= 4)
171     cout << "\n initialisation TriaSfe3_cm3pti" << endl;
172     Element::listTypeElement.push_back(nouv);
173     };
174     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
175     {Element * pt;
176     pt = new TriaSfe3_cm3pti (num_maill,num) ;
177     return pt;};
178     // ramene true si la construction de l'element est possible en fonction
179     // des variables globales actuelles: ex en fonction de la dimension
180     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
181 };
182 static ConsTriaSfe3_cm3pti consTriaSfe3_cm3pti;
183 };
184 /// @} // end of group
185 #endif

```



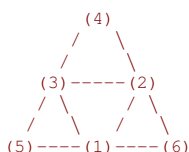
186  
187  
188  
189

## 7.217 TriaSfe3\_cm4pti.h

```

1 // FICHER : TriaSfe3_cm4pti.h
2 // CLASSE : TriaSfe3_cm4pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           04/07/2007
34 *
35 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:        Herezh++
38 *
39 *   ****
40 *   BUT:           Element triangulaire,Sfe3C. Le calcul de la normale
41 *   s'effectue à partir de l'interpolation des theta3 relativement
42 *   au triangle central, ceci via un polynome quadratique complet.
43 *   Le jacobien est celui de la facette plane centrale.
44 *   Ici le nombre de point d'intégration est de 4 en épaisseur
45 *   et un en surface.
46 *
47 *   *****
48 *
49 *   VERIFICATION:
50 *
51 *   ! date !   auteur !           but
52 *   -----
53 *   !           !           !           !
54 *   *****
55 *   MODIFICATIONS:
56 *   ! date !   auteur !           but
57 *   -----
58 *
59 *   *****/
60
61 /*
62 //
63 // l'interpolation est SFE,
64 //
65 //
66 //
67 //
68 //
69 //
70 //
71 */
72
73
74 #ifndef TRIASF3_CM4PTI_H
75 #define TRIASF3_CM4PTI_H
76
77 #include "ParaGlob.h"

```



```

78 #include "ElemMeca.h"
79 #include "Met_abstraite.h"
80 #include "GeomTriangle.h"
81 #include "Noeud.h"
82 #include "UtilLecture.h"
83 #include "Tenseur.h"
84 #include "NevezTenseur.h"
85 #include "Deformation.h"
86 #include "SfeMembT.h"
87 #include "ElFrontiere.h"
88 #include "FrontSegLine.h"
89 #include "FrontTriaLine.h"
90
91
92 /// @addtogroup groupe_des_elements_finis
93 /// @{
94 ///
95
96
97 class TriaSfe3_cm4pti : public SfeMembT
98 {
99
100 public :
101
102     // CONSTRUCTEURS :
103     // Constructeur par default
104     TriaSfe3_cm4pti ();
105
106     // Constructeur fonction d'une epaisseur et eventuellement d'un numero
107     // d'identification
108     TriaSfe3_cm4pti (double epaiss,int num_mail=0,int num_id=-3);
109
110     // Constructeur fonction d'un numero de maillage et d'identification
111     TriaSfe3_cm4pti (int num_mail,int num_id);
112
113     // Constructeur fonction d'une epaisseur, d'un numero d'identification,
114     // du tableau de connexite des noeuds
115     TriaSfe3_cm4pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
116
117     // Constructeur de copie
118     TriaSfe3_cm4pti (const TriaSfe3_cm4pti& tria);
119
120
121     // DESTRUCTEUR :
122     ~TriaSfe3_cm4pti ();
123
124     // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
125     // méthode virtuelle
126     Element* Nevez_copie() const { Element * el= new TriaSfe3_cm4pti(*this); return el;};
127
128     // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe3_cm4pti
129     TriaSfe3_cm4pti& operator= (TriaSfe3_cm4pti& tria);
130
131     // METHODES :
132     // 1) derivant des virtuelles pures
133
134     // affichage dans la sortie transmise, des variables duales "nom"
135     // aux differents points d'integration
136     // dans le cas ou nom est vide, affichage de "toute" les variables
137     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
138
139     // 2) derivant des virtuelles
140     // 3) methodes propres a l'element
141
142     // les coordonnees des points d'integration dans l'epaisseur
143     inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
144
145 protected :
146
147     // adressage des frontieres linéiques et surfacique
148     // définit dans les classes dérivées, et utilisées pour la construction des frontieres
149     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
150     { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
151     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
152     { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
153
154     // VARIABLES PRIVEES :
155     // place memoire commune a tous les elements TriaMembL1
156     static SfeMembT::DonnComSfe * doCoSfe3;
157     // idem mais pour les indicateurs qui servent pour l'initialisation
158     static SfeMembT::UneFois uneFoisSfe3;
159
160     class NombresConstruireTriaSfe3_cm4pti : public NombresConstruire
161     { public: NombresConstruireTriaSfe3_cm4pti();
162     };
163     static NombresConstruireTriaSfe3_cm4pti nombre_V; // les nombres propres à l'élément
164

```

```

165 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
166 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
167 class ConsTriaSfe3_cm4pti : public ConstrucElement
168 { public : ConsTriaSfe3_cm4pti ()
169 { NouvelleTypeElement nouv (TRIANGLE,SFE3,MECA_SOLIDE_DEFORMABLE,this,"_cm4pti");
170   if (ParaGlob::NiveauImpression() >= 4)
171     cout << "\n initialisation TriaSfe3_cm4pti" << endl;
172   Element::listTypeElement.push_back(nouv);
173 };
174   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
175   {Element * pt;
176    pt = new TriaSfe3_cm4pti (num_maill,num) ;
177    return pt;};
178 // ramene true si la construction de l'element est possible en fonction
179 // des variables globales actuelles: ex en fonction de la dimension
180 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
181 };
182 static ConsTriaSfe3_cm4pti consTriaSfe3_cm4pti;
183 };
184 /// @} // end of group
185 #endif
186
187
188
189

```

## 7.218 TriaSfe3\_cm5pti.h

```

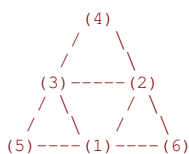
1 // FICHER : TriaSfe3_cm5pti.h
2 // CLASSE : TriaSfe3_cm5pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      13/11/2012
34 *
35 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *
40 *      BUT:      Element triangulaire,Sfe3. Le calcul de la normale
41 *      s'effectue à partir de l'interpolation des theta3 relativement
42 *      au triangle central, ceci via un polynome quadratique complet.
43 *      Le jacobien est celui de la facette plane centrale.
44 *      Ici le nombre de point d'intégration est de 5 en épaisseur
45 *      et un en surface. Le type est Gauss Lobatto
46 *
47 *      *****
48 *      VERIFICATION:
49 *
50 *      ! date ! auteur ! but
51 *      -----
52 *      ! ! ! ! !
53 *      $
54 *      *****
55 *      MODIFICATIONS:
56 *      ! date ! auteur ! but
57 *      -----

```

```

58 *
59 *****$*
60
61 /*
62 //
63 // l'interpolation est SFE,
64 //
65 //
66 //
67 //
68 //
69 //
70 //
71 //
72 */
73
74
75
76 #ifndef TRIASF3_CM5PTI_H
77 #define TRIASF3_CM5PTI_H
78
79 #include "ParaGlob.h"
80 #include "ElemMeca.h"
81 #include "Met_abstraite.h"
82 #include "GeomTriangle.h"
83 #include "Noeud.h"
84 #include "UtilLecture.h"
85 #include "Tenseur.h"
86 #include "NevezTenseur.h"
87 #include "Deformation.h"
88 #include "SfeMembT.h"
89 #include "ElFrontiere.h"
90 #include "FrontSegLine.h"
91 #include "FrontTriaLine.h"
92
93
94 /// @addtogroup groupe_des_elements_finis
95 /// @{
96 ///
97
98
99 class TriaSfe3_cm5pti : public SfeMembT
100 {
101
102     public :
103
104         // CONSTRUCTEURS :
105         // Constructeur par défaut
106         TriaSfe3_cm5pti ();
107
108         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
109         // d'identification
110         TriaSfe3_cm5pti (double epaiss,int num_mail=0,int num_id=-3);
111
112         // Constructeur fonction d'un numero de maillage et d'identification
113         TriaSfe3_cm5pti (int num_mail,int num_id);
114
115         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
116         // du tableau de connexion des noeuds
117         TriaSfe3_cm5pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
118
119         // Constructeur de copie
120         TriaSfe3_cm5pti (const TriaSfe3_cm5pti& tria);
121
122
123         // DESTRUCTEUR :
124         ~TriaSfe3_cm5pti ();
125
126         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
127         // méthode virtuelle
128         Element* Nevez_copie() const { Element * el= new TriaSfe3_cm5pti(*this); return el;};
129
130         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe3_cm5pti
131         TriaSfe3_cm5pti& operator= (TriaSfe3_cm5pti& tria);
132
133         // METHODES :
134         // 1) derivant des virtuelles pures
135
136         // affichage dans la sortie transmise, des variables duales "nom"
137         // aux differents points d'integration
138         // dans le cas ou nom est vide, affichage de "toute" les variables
139         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
140
141         // 2) derivant des virtuelles
142         // 3) methodes propres a l'element
143
144         // les coordonnees des points d'integration dans l'epaisseur

```



```

145     inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
146
147     protected :
148
149     // adressage des frontières linéiques et surfacique
150     // définit dans les classes dérivées, et utilisées pour la construction des frontières
151     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
152     { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
153     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
154     { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
155
156     // VARIABLES PRIVEES :
157     // place memoire commune a tous les elements TriaMembL1
158     static SfeMembT::DonnComSfe * doCoSfe3;
159     // idem mais pour les indicateurs qui servent pour l'initialisation
160     static SfeMembT::UneFois  uneFoisSfe3;
161
162     class NombresConstruireTriaSfe3_cm5pti : public NombresConstruire
163     { public: NombresConstruireTriaSfe3_cm5pti();
164       };
165     static NombresConstruireTriaSfe3_cm5pti nombre_V; // les nombres propres à l'élément
166
167     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
168     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
169     class ConsTriaSfe3_cm5pti : public ConstrucElement
170     { public : ConsTriaSfe3_cm5pti ()
171       { NouvelleTypeElement nouv(TRIANGLE,SFE3,MECA_SOLIDE_DEFORMABLE,this,"_cm5pti");
172         if (ParaGlob::NiveauImpression() >= 4)
173           cout << "\n initialisation TriaSfe3_cm5pti" << endl;
174         Element::listTypeElement.push_back(nouv);
175       };
176       Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
177       {Element * pt;
178         pt = new TriaSfe3_cm5pti (num_maill,num) ;
179         return pt;};
180       // ramene true si la construction de l'element est possible en fonction
181       // des variables globales actuelles: ex en fonction de la dimension
182       bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
183     };
184     static ConsTriaSfe3_cm5pti consTriaSfe3_cm5pti;
185 };
186 /// @} // end of group
187 #endif
188
189
190
191

```

## 7.219 TriaSfe3\_cm6pti.h

```

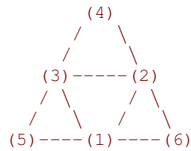
1 // FICHER : TriaSfe3_cm6pti.h
2 // CLASSE : TriaSfe3_cm6pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *    DATE:      04/07/2007
34 *
35 *    AUTEUR:    G RIO    (mailto:gerardrio56@free.fr)
36 *
37 *****/

```

```

36 *                                     $ *
37 *   PROJET:      Herezh++                                     *
38 *                                     $ *
39 *****
40 *   BUT:         Element triangulaire,Sfe3C. Le calcul de la normale *
41 *               s'effectue à partir de l'interpolation des theta3 relativement *
42 *               au triangle central, ceci via un polynome quadratique complet. *
43 *               Le jacobien est celui de la facette plane centrale. *
44 *               Ici le nombre de point d'intégration est de 2 en épaisseur *
45 *               et 3 en surface. *
46 *                                     $ *
47 *   ***** *
48 *   *
49 *   VERIFICATION: *
50 *   ! date !   auteur !   but *
51 *   ----- *
52 *   !       !       !       ! *
53 *   *                                     $ *
54 *   ***** *
55 *   MODIFICATIONS: *
56 *   ! date !   auteur !   but *
57 *   ----- *
58 *   *                                     $ *
59 *****/
60
61 //
62 // l'interpolation est SFE,
63 /*
64 //
65 //
66 //
67 //
68 //
69 //
70 */
71
72
73 #ifndef TRIASFE3_CM6PTI_H
74 #define TRIASFE3_CM6PTI_H
75
76 #include "ParaGlob.h"
77 #include "ElemMeca.h"
78 #include "Met_abstraite.h"
79 #include "GeomTriangle.h"
80 #include "Noeud.h"
81 #include "UtilLecture.h"
82 #include "Tenseur.h"
83 #include "NevezTenseur.h"
84 #include "Deformation.h"
85 #include "SfeMembT.h"
86 #include "ElFrontiere.h"
87 #include "FrontSegLine.h"
88 #include "FrontTriaLine.h"
89
90
91 /// @addtogroup groupe_des_elements_finis
92 /// @{
93 ///
94
95
96 class TriaSfe3_cm6pti : public SfeMembT
97 {
98     public :
99
100     // CONSTRUCTEURS :
101     // Constructeur par default
102     TriaSfe3_cm6pti ();
103
104     // Constructeur fonction d'une epaisseur et eventuellement d'un numero
105     // d'identification
106     TriaSfe3_cm6pti (double epaiss,int num_mail=0,int num_id=-3);
107
108     // Constructeur fonction d'un numero de maillage et d'identification
109     TriaSfe3_cm6pti (int num_mail,int num_id);
110
111     // Constructeur fonction d'une epaisseur, d'un numero d'identification,
112     // du tableau de connexite des noeuds
113     TriaSfe3_cm6pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
114
115     // Constructeur de copie
116     TriaSfe3_cm6pti (const TriaSfe3_cm6pti& tria);
117
118
119     // DESTRUCTEUR :
120     ~TriaSfe3_cm6pti ();

```



```

122
123 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
124 // méthode virtuelle
125 Element* Nevez_copie() const { Element * el= new TriaSfe3_cm6pti(*this); return el;};
126
127 // Surcharge de l'opérateur = : realise l'egalite entre deux instances de TriaSfe3_cm6pti
128 TriaSfe3_cm6pti& operator= (TriaSfe3_cm6pti& tria);
129
130 // METHODES :
131 // 1) derivant des virtuelles pures
132
133 // affichage dans la sortie transmise, des variables duales "nom"
134 // aux differents points d'integration
135 // dans le cas ou nom est vide, affichage de "toute" les variables
136 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
137
138 // 2) derivant des virtuelles
139 // 3) methodes propres a l'element
140
141 // les coordonnees des points d'integration dans l'epaisseur
142 inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
143
144 protected :
145
146 // adressage des frontieres linéiques et surfacique
147 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
148 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
149 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
150 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
151 { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
152
153 // VARIABLES PRIVEES :
154 // place memoire commune a tous les elements TriaMembL1
155 static SfeMembT::DonnComSfe * doCoSfe3;
156 // idem mais pour les indicateurs qui servent pour l'initialisation
157 static SfeMembT::UneFois uneFoisSfe3;
158
159 class NombresConstruireTriaSfe3_cm6pti : public NombresConstruire
160 { public: NombresConstruireTriaSfe3_cm6pti();
161 };
162 static NombresConstruireTriaSfe3_cm6pti nombre_V; // les nombres propres à l'élément
163
164 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
165 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
166 class ConsTriaSfe3_cm6pti : public ConstrucElement
167 { public : ConsTriaSfe3_cm6pti ()
168 { NouvelleTypeElement nouv (TRIANGLE,SFE3,MECA_SOLIDE_DEFORMABLE,this,"_cm6pti");
169 if (ParaGlob::NiveauImpression() >= 4)
170 cout << "\n initialisation TriaSfe3_cm6pti" << endl;
171 Element::listTypeElement.push_back(nouv);
172 };
173 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
174 {Element * pt;
175 pt = new TriaSfe3_cm6pti (num_maill,num) ;
176 return pt;};
177 // ramene true si la construction de l'element est possible en fonction
178 // des variables globales actuelles: ex en fonction de la dimension
179 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
180 };
181 static ConsTriaSfe3_cm6pti consTriaSfe3_cm6pti;
182 };
183 /// @} // end of group
184 #endif
185
186
187
188

```

## 7.220 TriaSfe3\_cm7pti.h

```

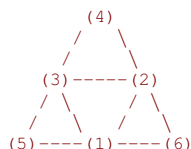
1 // FICHER : TriaSfe3_cm7pti.h
2 // CLASSE : TriaSfe3_cm7pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr

```

```

16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      13/11/2012
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      Element triangulaire,Sfe3. Le calcul de la normale
41 *             s'effectue à partir de l'interpolation des theta3 relativement
42 *             au triangle central, ceci via un polynome quadratique complet.
43 *             Le jacobien est celui de la facette plane centrale.
44 *             Ici le nombre de point d'intégration est de 7 en épaisseur
45 *             et un en surface. Le type est Gauss Lobatto
46 *
47 *             *****
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !           !           !
53 *
54 *             *****
55 *   MODIFICATIONS:
56 *   ! date !   auteur !           but
57 *   -----
58 *
59 *****/
60 //
61 //
62 // l'interpolation est SFE,
63 //
64 //
65 //
66 //
67 //
68 //
69 //
70 //
71 //
72 #ifndef TRIASFE3_CM7PTI_H
73 #define TRIASFE3_CM7PTI_H
74
75 #include "ParaGlob.h"
76 #include "ElemMeca.h"
77 #include "Met_abstraite.h"
78 #include "GeomTriangle.h"
79 #include "Noeud.h"
80 #include "UtilLecture.h"
81 #include "Tenseur.h"
82 #include "NevezTenseur.h"
83 #include "Deformation.h"
84 #include "SfeMembT.h"
85 #include "ElFrontiere.h"
86 #include "FrontSegLine.h"
87 #include "FrontTriaLine.h"
88
89
90 /// @addtogroup groupe_des_elements_finis
91 /// @{
92 ///
93
94
95 class TriaSfe3_cm7pti : public SfeMembT
96 {
97
98     public :
99
100         // CONSTRUCTEURS :
101         // Constructeur par default
102         TriaSfe3_cm7pti ();

```





```

103
104 // Constructeur fonction d'une epaisseur et eventuellement d'un numero
105 // d'identification
106 TriaSfe3_cm7pti (double epaiss,int num_mail=0,int num_id=-3);
107
108 // Constructeur fonction d'un numero de maillage et d'identification
109 TriaSfe3_cm7pti (int num_mail,int num_id);
110
111 // Constructeur fonction d'une epaisseur, d'un numero d'identification,
112 // du tableau de connexite des noeuds
113 TriaSfe3_cm7pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
114
115 // Constructeur de copie
116 TriaSfe3_cm7pti (const TriaSfe3_cm7pti& tria);
117
118
119 // DESTRUCTEUR :
120 ~TriaSfe3_cm7pti ();
121
122 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
123 // méthode virtuelle
124 Element* Nevez_copie() const { Element * el= new TriaSfe3_cm7pti(*this); return el;};
125
126 // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe3_cm7pti
127 TriaSfe3_cm7pti& operator= (TriaSfe3_cm7pti& tria);
128
129 // METHODES :
130 // 1) derivant des virtuelles pures
131
132 // affichage dans la sortie transmise, des variables duales "nom"
133 // aux differents points d'integration
134 // dans le cas ou nom est vide, affichage de "toute" les variables
135 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
136
137 // 2) derivant des virtuelles
138 // 3) methodes propres a l'element
139
140 // les coordonnees des points d'integration dans l'epaisseur
141 inline double KSI(int i) { return doCoSfe3->segment.KSI(i);};
142
143 protected :
144
145 // adressage des frontieres linéiques et surfacique
146 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
147 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
148 { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
149 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
150 { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
151
152 // VARIABLES PRIVEES :
153 // place memoire commune a tous les elements TriaMembL1
154 static SfeMembT::DonnComSfe * doCoSfe3;
155 // idem mais pour les indicateurs qui servent pour l'initialisation
156 static SfeMembT::UneFois uneFoisSfe3;
157
158 class NombresConstruireTriaSfe3_cm7pti : public NombresConstruire
159 { public: NombresConstruireTriaSfe3_cm7pti();
160 };
161 static NombresConstruireTriaSfe3_cm7pti nombre_V; // les nombres propres à l'élément
162
163 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
164 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
165 class ConsTriaSfe3_cm7pti : public ConstrucElement
166 { public : ConsTriaSfe3_cm7pti ()
167 { NouvelleTypeElement nouv (TRIANGLE,SFE3,MECA_SOLIDE_DEFORMABLE,this,"_cm7pti");
168 if (ParaGlob::NiveauImpression() >= 4)
169 cout << "\n initialisation TriaSfe3_cm7pti" << endl;
170 Element::listTypeElement.push_back(nouv);
171 };
172 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
173 {Element * pt;
174 pt = new TriaSfe3_cm7pti (num_maill,num) ;
175 return pt;};
176 // ramene true si la construction de l'element est possible en fonction
177 // des variables globales actuelles: ex en fonction de la dimension
178 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
179 };
180 static ConsTriaSfe3_cm7pti consTriaSfe3_cm7pti;
181 };
182 /// @} // end of group
183 #endif
184
185
186
187

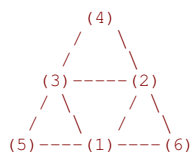
```

## 7.221 TriaSfe3C.h

```

1 // FICHER : TriaSfe3C.h
2 // CLASSE : TriaSfe3C
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          04/07/2007
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *****/
40 *      BUT:           Element triangulaire, Sfe3C. Le calcul de la normale
41 *      s'effectue à partir de l'interpolation des theta3 relativement
42 *      au triangle central, ceci via un polynome quadratique complet.
43 *      Contrairement au cas du Sfe3, le calcul du jacobien est relatif
44 *      à celui du triangle central courbe.
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *      ! date ! auteur ! but
50 *      -----
51 *      ! ! !
52 *      $
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *      $
58 *****/
59
60 /*
61 // l'interpolation est SFE,
62 //
63 //
64 //
65 //
66 //
67 //
68 //
69 */
70
71
72
73 #ifndef TRIASF3C_H
74 #define TRIASF3C_H
75
76 #include "ParaGlob.h"
77 #include "ElemMeca.h"
78 #include "Met_abstraite.h"
79 #include "GeomTriangle.h"
80 #include "Noeud.h"
81 #include "UtilLecture.h"
82 #include "Tenseur.h"
83 #include "NevezTenseur.h"
84 #include "Deformation.h"

```



```

85 #include "SfeMembT.h"
86 #include "ElFrontiere.h"
87 #include "FrontSegLine.h"
88 #include "FrontTriaLine.h"
89
90
91 /// @addtogroup groupe_des_elements_finis
92 /// @{
93 ///
94
95
96 class TriaSfe3C : public SfeMembT
97 {
98
99     public :
100
101         // CONSTRUCTEURS :
102         // Constructeur par défaut
103         TriaSfe3C ();
104
105         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
106         // d'identification
107         TriaSfe3C (double epaiss,int num_mail=0,int num_id=-3);
108
109         // Constructeur fonction d'un numero de maillage et d'identification
110         TriaSfe3C (int num_mail,int num_id);
111
112         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
113         // du tableau de connexite des noeuds
114         TriaSfe3C (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
115
116         // Constructeur de copie
117         TriaSfe3C (const TriaSfe3C& tria);
118
119
120         // DESTRUCTEUR :
121         ~TriaSfe3C ();
122
123         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
124         // méthode virtuelle
125         Element* Nevez_copie() const { Element * el= new TriaSfe3C(*this); return el;};
126
127         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaSfe3C
128         TriaSfe3C& operator= (TriaSfe3C& tria);
129
130         // METHODES :
131 // 1) derivant des virtuelles pures
132
133         // affichage dans la sortie transmise, des variables duales "nom"
134         // aux differents points d'integration
135         // dans le cas ou nom est vide, affichage de "toute" les variables
136         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
137
138 // 2) derivant des virtuelles
139 // 3) methodes propres a l'element
140
141         // les coordonnees des points d'integration dans l'epaisseur
142         inline double KSI(int i) { return doCoSfe3C->segment.KSI(i);};
143
144     protected :
145
146         // adressage des frontieres linéiques et surfacique
147         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
148         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
149         { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
150         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
151         { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
152
153     // VARIABLES PRIVEES :
154     // place memoire commune a tous les elements TriaSfe3C
155     static SfeMembT::DonnComSfe * doCoSfe3C;
156     // idem mais pour les indicateurs qui servent pour l'initialisation
157     static SfeMembT::UneFois uneFoisSfe3C;
158
159     class NombresConstruireTriaSfe3C : public NombresConstruire
160     { public: NombresConstruireTriaSfe3C();
161       };
162     static NombresConstruireTriaSfe3C nombre_V; // les nombres propres à l'élément
163
164     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
165     //ajout de l'element dans la liste : listTypeElement, geree par la class Element
166     class ConsTriaSfe3C : public ConstrucElement
167     { public : ConsTriaSfe3C ()
168       { NouvelleTypeElement nouv (TRIANGLE,SFE3C,MECA_SOLIDE_DEFORMABLE,this);
169         if (ParaGlob::NiveauImpression() >= 4)
170           cout << "\n initialisation TriaSfe3C" << endl;
171         Element::listTypeElement.push_back(nouv);

```

```

172         };
173         Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
174         {Element * pt;
175         pt = new TriaSfe3C (num_maill,num) ;
176         return pt;};
177         // ramene true si la construction de l'element est possible en fonction
178         // des variables globales actuelles: ex en fonction de la dimension
179         bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
180     };
181     static ConsTriaSfe3C consTriaSfe3C;
182 };
183 /// @} // end of group
184 #endif
185
186
187
188

```

## 7.222 Tetra.h

```

1 // FICHER : Tetra.h
2 // CLASSE : Tetra
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      15/11/99
34 *
35 *      AUTEUR:      G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:      Herezh++
38 *
39 *****/
40 *      BUT:      La classe Tetra permet de declarer des elements
41 *      Tetraedriques Trilineaires et de realiser
42 *      le calcul du residu local et de la raideur locale pour une loi de
43 *      comportement donnee. La dimension de l'espace pour un tel element
44 *      est 3.
45 *
46 *      l'interpolation est trilineaire a 4 noeuds, le nombre de
47 *      point d'integration est de 1.
48 *
49 *
50 *      VERIFICATION:
51 *
52 *      ! date ! auteur ! but
53 *      -----
54 *      ! ! !
55 *      $
56 *
57 *      MODIFICATIONS:
58 *
59 *      ! date ! auteur ! but
60 *      -----
61 *      $
62 // -----classe pour un calcul de mecanique-----
63
64

```

```

65
66 #ifndef TETRA_H
67 #define TETRA_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomTriangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "TetraMemb.h"
79 #include "FrontTriaLine.h"
80 #include "FrontSegLine.h"
81
82
83 /// @addtogroup groupe_des_elements_finis
84 /// @
85 ///
86
87
88 class Tetra : public TetraMemb
89 {
90
91     public :
92
93         // CONSTRUCTEURS :
94
95         // Constructeur par default
96         Tetra ();
97
98         // Constructeur fonction d'un numero
99         // d'identification
100        Tetra (int num_mail,int num_id);
101
102        // Constructeur fonction d'un numero de maillage et d'identification et
103        // du tableau de connexite des noeuds
104        Tetra (int num_mail,int num_id,const Tableau<Noeud *>& tab);
105
106        // Constructeur de copie
107        Tetra (const Tetra& tetra);
108
109
110        // DESTRUCTEUR :
111        ~Tetra ();
112
113        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
114        // méthode virtuelle
115        Element* Nevez_copie() const { Element * el= new Tetra(*this); return el;};
116
117        // Surcharge de l'operateur = : realise l'egalite entre deux instances de Tetra
118        Tetra& operator= (Tetra& tetra);
119
120        // METHODES :
121        // 1) derivant des virtuelles pures
122
123        // affichage dans la sortie transmise, des variables duales "nom"
124        // aux differents points d'integration
125        // dans le cas ou nom est vide, affichage de "toute" les variables
126        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
127
128        // 2) derivant des virtuelles
129        // 3) methodes propres a l'element
130
131        protected :
132
133        // adressage des frontières linéiques et surfacique
134        // définit dans les classes dérivées, et utilisées pour la construction des frontières
135        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
136        { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
137        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
138        { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
139
140        // VARIABLES PRIVEES :
141        // place memoire commune a tous les elements Tetra
142        static TetraMemb::DonnComTetra * doCoTetra;
143        // idem mais pour les indicateurs qui servent pour l'initialisation
144        static TetraMemb::UneFois uneFois;
145
146        class NombresConstruireTetra : public NombresConstruire
147        { public: NombresConstruireTetra();
148          };
149        static NombresConstruireTetra nombre_V; // les nombres propres à l'élément
150
151        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME

```

```

152 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
153 class ConsTetra : public ConstrucElement
154 { public : ConsTetra ()
155   { NouvelleTypeElement nouv (TETRAEDRE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this);
156     if (ParaGlob::NiveauImpression() >= 4)
157       cout << "\n initialisation Tetra" << endl;
158     nouv.el = this;
159     Element::listTypeElement.push_back(nouv);
160   };
161   Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
162   {Element * pt;
163     pt = new Tetra (num_maill,num) ;
164     return pt;};
165   // ramene true si la construction de l'element est possible en fonction
166   // des variables globales actuelles: ex en fonction de la dimension
167   bool Element_possible() { if (ParaGlob::Dimension() ==3) return true; else return false;};
168 };
169 static ConsTetra consTetra;
170 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
171 };
172 /// @} // end of group
173 #endif
174
175
176
177

```

## 7.223 TetraMemb.h

```

1 // FICHER : TetraMemb.h
2 // CLASSE : TetraMemb
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *
35 *   DATE:      15/01/97
36 *
37 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:    Herezh++
40 *
41 *
42 * *****
43 * La classe TetraMemb permet de declarer des elements tetrahedriques et de realiser
44 * le calcul du residu local et de la raideur locale pour une loi de comportement
45 * donnee. La dimension de l'espace pour un tel element est 3
46 * l'interpolation le nombre de point d'integration sont definit dans les classes derivees
47 *
48 * l'element est virtuel
49 *
50 *   $
51 *
52 * VERIFICATION:
53 *
54 *   ! date ! auteur ! but
55 *   -----
56 *   ! ! ! !

```

```

56 *
57 *
58 *      *****
59 *      MODIFICATIONS:
60 *
61 *      ! date !   auteur !   but
62 *      -----
63 *
64 *
65 *
66 *      *****/
67
68 // -----classe pour un calcul de mecanique-----
69
70 // La classe TetraMemb permet de declarer des elements tetrahedriques et de realiser
71 // le calcul du residu local et de la raideur locale pour une loi de comportement
72 // donnee. La dimension de l'espace pour un tel element est 3
73 //
74 // l'interpolation le nombre de point d'integration sont definit dans les classes derivees
75 //
76 // l'element est virtuel
77
78
79 #ifndef TETRAMEMB_H
80 #define TETRAMEMB_H
81
82 #include "ParaGlob.h"
83 #include "ElemMeca.h"
84 #include "Met_abstraite.h"
85 #include "ElemGeomC0.h"
86 #include "Noeud.h"
87 #include "UtilLecture.h"
88 #include "Tenseur.h"
89 #include "NevezTenseur.h"
90 #include "Deformation.h"
91 #include "GeomSeg.h"
92 #include "GeomTriangle.h"
93
94 /// @addtogroup groupe_des_elements_finis
95 /// @{
96 ///
97
98
99 class TetraMemb : public ElemMeca
100 {
101
102     public :
103
104         // CONSTRUCTEURS :
105         // Constructeur par default
106         TetraMemb ();
107
108         // Constructeur fonction d'un numero
109         // d'identification , d'identificateur d'interpolation et de geometrie
110         // et éventuellement un string d'information annexe
111         TetraMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string
info="");
112
113         // Constructeur fonction d'un numero de maillage et d'identification,
114         // du tableau de connexite des noeuds, d'identificateur d'interpolation et de geometrie
115         // et éventuellement un string d'information annexe
116         TetraMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,
const Tableau<Noeud *>& tab,string info="") ;
117
118
119         // Constructeur de copie
120         TetraMemb (const TetraMemb& tetraMem);
121
122
123         // DESTRUCTEUR :
124         ~TetraMemb ();
125
126
127         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TetraMemb
128         TetraMemb& operator= (TetraMemb& tetraMem);
129
130         // METHODES :
131         // 1) derivant des virtuelles pures
132
133         // Lecture des donnees de la classe sur fichier
134         void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
135
136         // affichage d'info en fonction de ordre
137         // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
138         void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> *
tabMaillageNoeud)
139         { return Element::Info_com_El (nombre->nbne,entreePrinc,ordre,tabMaillageNoeud);};

```

```

140
141 // ramene l'element geometrique
142 ElemGeomC0& ElementGeometrique() const { return *(unefois->doCoMemb->tetraed);};
143 // ramene l'element geometrique en constant
144 const ElemGeomC0& ElementGeometrique_const() const {return *(unefois->doCoMemb->tetraed);};
145
146 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
    associé
147 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
148 // 1) cas où l'on utilise la place passée en argument
149 Coordonnee & Point_physique(const Coordonnee& c_int, Coordonnee & co, Enum_dure temps);
150 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
151 void Point_physique(const Coordonnee& c_int, Tableau <Coordonnee> & t_co);
152
153 // inactive les ddl du problème primaire de mécanique
154 inline void Inactive_ddl_primaire()
155 {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};
156 // active les ddl du problème primaire de mécanique
157 inline void Active_ddl_primaire()
158 {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl);};
159 // ajout des ddl de contraintes pour les noeuds de l'élément
160 inline void Plus_ddl_Sigma()
161 {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
162 // inactive les ddl du problème de recherche d'erreur : les contraintes
163 inline void Inactive_ddl_Sigma()
164 {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
165 // active les ddl du problème de recherche d'erreur : les contraintes
166 inline void Active_ddl_Sigma()
167 {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
168 // active le premier ddl du problème de recherche d'erreur : SIGMA11
169 inline void Active_premier_ddl_Sigma()
170 {ElemMeca::Act_premier_ddl_Sigma();};
171
172 // lecture de données diverses sur le flot d'entrée
173 void LectureContraintes(UtilLecture * entreePrinc)
174 { if (unefois->CalResPrem_t == 1)
175     ElemMeca::LectureDesContraintes (false, entreePrinc, lesPtMecaInt.TabSigHH_t());
176     else
177     { ElemMeca::LectureDesContraintes (true, entreePrinc, lesPtMecaInt.TabSigHH_t());
178       unefois->CalResPrem_t = 1;
179     }
180 };
181
182 // retour des contraintes en absolu retour true si elle existe sinon false
183 bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
184 { if (unefois->CalResPrem_t == 1)
185     ElemMeca::ContraintesEnAbsolues (false, lesPtMecaInt.TabSigHH_t(), tabSig);
186     else
187     { ElemMeca::ContraintesEnAbsolues (true, lesPtMecaInt.TabSigHH_t(), tabSig);
188       unefois->CalResPrem_t = 1;
189     }
190     return true; }
191
192 // Libere la place occupee par le residu et eventuellement la raideur
193 // par l'appel de Libere de la classe mere et libere les differents tenseurs
194 // intermediaires cree pour le calcul et les grandeurs pointee
195 // de la raideur et du residu
196 void Libere ();
197
198 // acquisition d'une loi de comportement
199 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
200
201 // test si l'element est complet
202 // = 1 tout est ok, =0 element incomplet
203 int TestComplet();
204
205 // procedure permettant de completer l'element apres
206 // sa creation avec les donnees du bloc transmis
207 // peut etre appeler plusieurs fois
208 // ici pour l'instant ne fait rien
209 Element* Complete(BlocGen & bloc, LesFonctions_nD* lesFonctionsnD);
210 // Compléter pour la mise en place de la gestion de l'hourglass
211 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc);
212
213 // ramene vrai si la surface numéro ns existe pour l'élément
214 // dans le cas des éléments Tetraedrique il peut y avoir de 1 à 4 surfaces
215 bool SurfExiste(int ns) const
216 { if ((ns>=1)&&(ns<=4)) return true; else return false;};
217
218 // ramene vrai si l'arête numéro na existe pour l'élément
219 bool AreteExiste(int na) const
220 {if ((na <= 6) || (na>= 1)) return true; else return false;};
221
222 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
223 // ce tableau et specifique a l'element
224 const DdlElement & TableauDdl() const
225 { return unefois->doCoMemb->tab_ddl; };

```



```

226
227 // Calcul du residu local et de la raideur locale,
228 // pour le schema implicite
229 Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
230
231 // Calcul du residu local a t
232 // pour le schema explicite par exemple
233 Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
234 { return TetraMemb::CalculResidu(false,pa);};
235
236 // Calcul du residu local a tdt
237 // pour le schema explicite par exemple
238 Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
239 { return TetraMemb::CalculResidu(true,pa);};
240
241 // Calcul de la matrice masse pour l'élément
242 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
243
244 // ----- calcul dynamique -----
245 // calcul de la longueur d'arrête de l'élément minimal
246 // divisé par la célérité la plus rapide dans le matériau
247 double Long_arrete_mini_sur_c(Enum_dure temps)
248 { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
249
250 //----- calcul d'erreur, remontée des contraintes -----
251 // 1) calcul du résidu et de la matrice de raideur pour le calcul d'erreur
252 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
253 // 2) remontée aux erreurs aux noeuds
254 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
255
256 // ----- affichage ou récupération d'informations -----
257 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
258 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
259 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
260 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
261 // temps: dit si c'est à 0 ou t ou tdt
262 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
263 { return PtLePlusPres(temps,enu,M);};
264
265 // recuperation des coordonnées du point de numéro d'ordre iteg pour
266 // la grandeur enu
267 // temps: dit si c'est à 0 ou t ou tdt
268 // si erreur retourne erreur à true
269 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
270 { return CoordPtInt(temps,enu,iteg,erreur);};
271
272 // récupération des valeurs au numéro d'ordre = iteg pour
273 // les grandeur enu
274 Tableau<double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const
List_io<Ddl_enum_etendu>& enu,int iteg);
275 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
276 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
277 // de conteneurs quelconque associée
278 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int
iteg);
279
280 //===== lecture écriture dans base info =====
281
282 // cas donne le niveau de la récupération
283 // = 1 : on récupère tout
284 // = 2 : on récupère uniquement les données variables (supposées comme telles)
285 void Lecture_base_info
286 (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
287 // cas donne le niveau de sauvegarde
288 // = 1 : on sauvegarde tout
289 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
290 void Ecriture_base_info(ofstream& sort,const int cas) ;
291
292 // 2) derivant des virtuelles
293
294 // retourne un tableau de ddl element, correspondant à la
295 // composante de sigma -> SIG11, pour chaque noeud qui contient
296 // des ddl de contrainte
297 // -> utilisé pour l'assemblage de la raideur d'erreur
298 DdlElement& Tableau_de_Sig1() const
299 {return unefois->doCoMemb->tab_Err1Sig11;};
300
301 // actualisation des ddl et des grandeurs actives de t+dt vers t
302 void TdtversT();
303 // actualisation des ddl et des grandeurs actives de t vers tdt
304 void TversTdt();
305
306 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
307 // qu'une fois la remontée aux contraintes effectuées sinon aucune
308 // action. En retour la valeur de l'erreur sur l'élément
309 // type indique le type de calcul d'erreur :
310 void ErreurElement(int type,double& errElemRelative

```

```

311         ,double& numerateur, double& denominateur);
312
313     // calcul des seconds membres suivant les chargements
314     // cas d'un chargement volumique,
315     // force indique la force volumique appliquée
316     // retourne le second membre résultant
317     // ici on l'épaisseur de l'élément pour constituer le volume
318     // -> explicite à t
319     Vecteur SM_charge_volumique_E_t
320     (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_)
321     { return TetraMemb::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_); };
322     // -> explicite à tdt
323     Vecteur SM_charge_volumique_E_tdt
324     (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_)
325     { return TetraMemb::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_); };
326     // -> implicite,
327     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
328     // retourne le second membre et la matrice de raideur correspondant
329     ResRaid SMR_charge_volumique_I
330     (const Coordonnee& force,Fonction_nD* pt_fonct,const ParaAlgoControle & pa,bool
sur_volume_finale_) ;
331     // cas d'un chargement surfacique, sur les frontières des éléments
332     // force indique la force surfacique appliquée
333     // numface indique le numéro de la face chargée
334     // retourne le second membre résultant
335     // -> version explicite à t
336     Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
337     { return TetraMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa); };
338     // -> version explicite à tdt
339     Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
340     { return TetraMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa); };
341     // -> implicite,
342     // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
343     // retourne le second membre et la matrice de raideur correspondant
344     ResRaid SMR_charge_surfacique_I
345     (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const ParaAlgoControle & pa)
;
346
347     // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
348     // presUniDir indique le vecteur appliquée
349     // numface indique le numéro de la face chargée
350     // retourne le second membre résultant
351     // -> explicite à t
352     Vecteur SM_charge_presUniDir_E_t(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
353     { return TetraMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,false,pa); };
354     // -> explicite à tdt
355     Vecteur SM_charge_presUniDir_E_tdt(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
356     { return TetraMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,true,pa); };
357     // -> implicite,
358     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
359     // retourne le second membre et la matrice de raideur correspondant
360     ResRaid SMR_charge_presUniDir_I(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa);
361
362     // cas d'un chargement lineique, sur les aretes frontières des éléments
363     // force indique la force lineique appliquée
364     // numarete indique le numéro de l'arete chargée
365     // retourne le second membre résultant
366     // NB: il y a une définition par défaut pour les éléments qui n'ont pas
367     // d'arete externe -> message d'erreur d'où le virtuel et non virtuel pur
368     // -> explicite à t
369     Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
370     { return TetraMemb::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); };
371     // -> explicite à tdt
372     Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
373     { return TetraMemb::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); };
374     // -> implicite,
375     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
376     // retourne le second membre et la matrice de raideur correspondant
377     ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
378
379     // cas d'un chargement de type pression, sur les frontières des éléments
380     // pression indique la pression appliquée
381     // numface indique le numéro de la face chargée
382     // retourne le second membre résultant
383     // NB: il y a une définition par défaut pour les éléments qui n'ont pas de
384     // surface externe -> message d'erreur d'où le virtuel et non virtuel pur
385     // -> explicite à t

```

```

386     Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
387     { return TetraMemb::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa);};
388     // -> explicite à tdt
389     Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
390     { return TetraMemb::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa);};
391     // -> implicite,
392     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
393     // retourne le second membre et la matrice de raideur correspondant
394     ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
395
396     // cas d'un chargement surfacique hydrostatique,
397     // poidvol: indique le poids volumique du liquide
398     // M_liquide : un point de la surface libre
399     // dir_normal_liquide : direction normale à la surface libre
400     // retourne le second membre résultant
401     // -> explicite à t
402     Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
403     ,int numFace,const Coordonnee& M_liquide,Fonction_nD*
pt_fonct
404     ,const ParaAlgoControle & pa
405     ,bool sans_limitation)
406     { return
TetraMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation);};
407     // -> explicite à tdt
408     Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
409     ,int numFace,const Coordonnee& M_liquide,Fonction_nD*
pt_fonct
410     ,const ParaAlgoControle & pa
411     ,bool sans_limitation)
412     { return
TetraMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation);};
413     // -> implicite,
414     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
415     // retourne le second membre et la matrice de raideur correspondant
416     ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
417     ,int numFace,const Coordonnee& M_liquide
418     ,const ParaAlgoControle & pa
419     ,bool sans_limitation) ;
420
421     // cas d'un chargement surfacique hydro-dynamique,
422     // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
423     // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc
unitaire)
424     // une suivant la direction normale à la vitesse de type portance
425     // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
426     // une suivant la vitesse tangente de type frottement visqueux
427     // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
428     // coef_mul: est un coefficient multiplicateur global (de tout)
429     // retourne le second membre résultant
430     // -> explicite à t
431     Vecteur SM_charge_hydrodynamique_E_t( CourbeID* frot_fluid,const double& poidvol
432     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
433     , CourbeID* coef_aero_t
434     ,const ParaAlgoControle & pa)
435     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
436     // -> explicite à tdt
437     Vecteur SM_charge_hydrodynamique_E_tdt( CourbeID* frot_fluid,const double& poidvol
438     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
439     , CourbeID* coef_aero_t
440     ,const ParaAlgoControle & pa)
441     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};
442     // -> implicite,
443     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
444     // retourne le second membre et la matrice de raideur correspondant
445     ResRaid SMR_charge_hydrodynamique_I( CourbeID* frot_fluid,const double& poidvol
446     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
447     , CourbeID* coef_aero_t
448     ,const ParaAlgoControle & pa) ;
449
450     // ===== définition et/ou construction des frontières =====
451
452     // Calcul des frontieres de l'element
453     // creation des elements frontieres et retour du tableau de ces elements
454     // la création n'a lieu qu'au premier appel
455     // ou lorsque l'on force le paramètre force a true
456     // dans ce dernier cas seul les frontière effacées sont recréée
457     Tableau <ElFrontiere*> const & Frontiere(bool force = false);
458
459     // ramène la frontière point

```

```

460 // éventuellement création des frontieres points de l'element et stockage dans l'element
461 // si c'est la première fois sinon il y a seulement retour de l'elements
462 // a moins que le paramètre force est mis a true
463 // dans ce dernier cas la frontière effacée est recréée
464 // num indique le numéro du point à créer (numérotation EF)
465 // ElFrontiere* const Frontiere_points(int num,bool force = false);
466
467 // ramène la frontière linéique
468 // éventuellement création des frontieres linéique de l'element et stockage dans l'element
469 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
470 // a moins que le paramètre force est mis a true
471 // dans ce dernier cas la frontière effacée est recréée
472 // num indique le numéro de l'arête à créer (numérotation EF)
473 // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
474
475 // ramène la frontière surfacique
476 // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
477 // si c'est la première fois sinon il y a seulement retour de l'elements
478 // a moins que le paramètre force est mis a true
479 // dans ce dernier cas la frontière effacée est recréée
480 // num indique le numéro de la surface à créer (numérotation EF)
481 // ElFrontiere* const Frontiere_surfacique(int num,bool force = false);
482
483 // 3) methodes propres a l'element
484
485 // ajout du tableau specific de ddl des noeuds
486 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
487 // des noeuds constituants l'element
488 void ConstTabDdl();
489
490 public :
491 // ----- definition de la classe conteneur de donnees communes -----
492 class DonnComTetra
493 { public :
494 // hexal doit pointer sur un element deja existant via un new
495 DonnComTetra (ElemGeomC0* tetra1,DdlElement& tab,DdlElement& tabErr,DdlElement&
tab_Err1Sig,
496 Met_abstraite& met_gene,
497 Tableau <Vecteur * > & resEr,Mat_pleine& raidEr,ElemGeomC0* tetraeEr,
498 GeomTriangle triS,GeomSeg& seggS,
499 Vecteur& residu_int,Mat_pleine& raideur_int,
500 Tableau <Vecteur * > & residus_extN,Tableau <Mat_pleine * > & raideurs_extN,
501 Tableau <Vecteur * > & residus_extA,Tableau <Mat_pleine * > & raideurs_extA,
502 Tableau <Vecteur * > & residus_extS,Tableau <Mat_pleine * > & raideurs_extS,
503 Mat_pleine& mat_masse, ElemGeomC0* tetraeMas,int nbi,
504 ElemGeomC0* tetraeHourg
505 );
506 DonnComTetra(DonnComTetra& a) ;
507 ~DonnComTetra();
508 // variables
509 ElemGeomC0* tetraed; // contient les fonctions d'interpolation et
510 // les derivees
511 DdlElement tab_ddl; // tableau des degres
512 //de liberte des noeuds de l'element commun a tous les
513 // elements
514 Met_abstraite metrique;
515 Mat_pleine matGeom ; // matrice géométrique
516 Mat_pleine matInit ; // matrice initiale
517 Tableau <TenseurBB * > d_epsBB; // place pour la variation des def
518 Tableau <TenseurHH * > d_sigHH; // place pour la variation des contraintes
519 Tableau < Tableau2 <TenseurBB * > > d2_epsBB; // variation seconde des déformations
520 // ---- concernant les frontières et particulièrement le calcul de second membre
521 GeomTriangle triaS; // contient les fonctions d'interpolation
522 // et les derivees pour les surfaces
523 GeomSeg segS; // idem pour les arêtes
524
525 // calcul d'erreur
526 DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
527 // d'erreur : contraintes
528 DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud, servant pour le
calcul
529 // d'erreur : contraintes, en fait pour l'assemblage
530
531 Tableau <Vecteur * > resErr; // residu pour le calcul d'erreur
532 Mat_pleine raidErr; // raideur pour le calcul d'erreur
533 ElemGeomC0* tetraeEr; // contient les fonctions d'interpolation et
534 // les derivees pour le calcul du hessien dans
535 //la résolution de la fonctionnelle d'erreur
536 // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
537 -----
538 // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
539 Vecteur residu_interne;
540 Mat_pleine raideur_interne;
541 Tableau <Vecteur * > residus_externeN; // pour les noeuds
542 Tableau <Mat_pleine * > raideurs_externeN; // pour les noeuds
543 Tableau <Vecteur * > residus_externeA; // pour les aretes
544 Tableau <Mat_pleine * > raideurs_externeA; // pour les aretes

```

```

544     Tableau <Vecteur* > residus_externes; // pour les surfaces
545     Tableau <Mat_pleine* > raideurs_externes; // pour les surfaces
546     // ----- données concernant la dynamique -----
547     Mat_pleine matrice_masse;
548     ElemGeomC0* tetraedMas; // contient les fonctions d'interpolation et ...
549     // pour les calculs relatifs à la masse
550     // ----- blocage éventuel d'hourglass
551     // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
Cal_explici_hourglass
552     ElemGeomC0* tetraedHourg; // contient les fonctions d'interpolation
553     };
554
555     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
556     // et un pointeur sur les données statiques communes
557     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
558     // classe est défini. Son allocation est effectuée dans les classes dérivées
559     class UneFois
560     { public :
561         UneFois () ; // constructeur par défaut
562         ~UneFois () ; // destructeur
563
564     // VARIABLES :
565     public :
566         TetraMemb::DonnComTetra * doCoMemb;
567
568         // incicateurs permettant de dimensionner seulement au premier passage
569         // utilise dans "CalculResidu" et "Calcul_implicit"
570         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
571         int CalimpPrem;
572         int dualSortTetra; // pour la sortie des valeurs au pt d'integ
573         int CalSMlin_t; // pour les seconds membres concernant les arretes
574         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
575         int CalSMRlin; // pour les seconds membres concernant les arretes
576         int CalSMSurf_t; // pour les seconds membres concernant les surfaces
577         int CalSMSurf_tdt; // pour les seconds membres concernant les surfaces
578         int CalSMRsurf; // pour les seconds membres concernant les surfaces
579         int CalSMvol_t; // pour les seconds membres concernant les volumes
580         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
581         int CalSMvol; // pour les seconds membres concernant les volumes
582         int CalDynamique; // pour le calcul de la matrice de masse
583         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
584         // ----- sauvegarde du nombre d'élément en cours -----
585         int nbelem_in_Prog;
586     };
587
588     // -----
589
590 protected :
591
592     // VARIABLES PRIVEES :
593     UneFois * unefois; // pointeur défini dans la classe dérivée
594
595     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
596     LesPtIntegMecaInterne lesPtMecaInt;
597
598     // type structuré et pointeur pour construire les éléments
599     // le pointeur est défini dans le type dérivé
600     class NombresConstruire
601     { public:
602         int nbne; // le nombre de noeud de l'élément
603         int nbneS ; // le nombre de noeud des facettes
604         int nbneA ; // le nombre de noeud des aretes
605         int nbi; // le nombre de point d'intégration pour le calcul mécanique
606         int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
607         int nbiV; // le nombre de point d'intégration pour le calcul de second membre volumique
608         int nbiS; // le nombre de point d'intégration pour le calcul de second membre surfacique
609         int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
610         int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse
611         int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
612     };
613     NombresConstruire * nombre; // le pointeur défini dans la classe dérivée d'hexamemb
614
615     // =====>>> methodes appelees par les classes derivees <<=====
616
617     // fonction d'initialisation servant dans les classes derivant
618     // au niveau du constructeur
619     // les pointeurs d'éléments géométriques sont non nul uniquement lorsque doCoMemb est null
620     // c'est-à-dire pour l'initialisation
621     TetraMemb::DonnComTetra* Init (ElemGeomC0* tetra,ElemGeomC0* tetraEr,ElemGeomC0* tetraMas
622     ,ElemGeomC0* tetraeHourg,bool sans_init_noeud = false);
623     // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
624     void Destruction();
625
626     // =====>>> methodes virtuelles derivant d'ElemMeca =====
627     // ramene la dimension des tenseurs contraintes et déformations de l'élément
628     int Dim_sig_eps() const {return 3;};
629

```

```

630 //----- fonctions uniquement a usage interne -----
631 private :
632 // definition des données communes : doCoTetra
633 // nbiS est pour le nombre de point d'intégration de surface triangle pour le second membre
634 TetraMemb::DonnComTetra* Def_DonneeCommune(ElemGeomCO* tetra,ElemGeomCO* tetraEr
635 ,ElemGeomCO* tetraMas,ElemGeomCO* tetraeHourg);
636 // Calcul du residu local a t ou tdt en fonction du booleen
637 Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
638 // cas d'un chargement de type pression, sur les frontières des éléments
639 // pression indique la pression appliquée
640 // numface indique le numéro de la face chargée
641 // retourne le second membre résultant
642 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
643 Vecteur SM_charge_pression_E(double pression,Fonction_nD* pt_fonct,int numFace,bool atdt,const
ParaAlgoControle & pa);
644 // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
645 // presUniDir indique le vecteur appliquée
646 // numface indique le numéro de la face chargée
647 // retourne le second membre résultant
648 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
649 Vecteur SM_charge_presUniDir_E
650 (const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int numFace,bool atdt,const
ParaAlgoControle & pa) ;
651 // cas d'un chargement lineique, sur les aretes frontières des éléments
652 // force indique la force lineique appliquée
653 // numarete indique le numéro de l'arete chargée
654 // retourne le second membre résultant
655 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
656 Vecteur SM_charge_lineique_E
657 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
658 // cas d'un chargement surfacique, sur les frontières des éléments
659 // force indique la force surfacique appliquée
660 // numface indique le numéro de la face chargée
661 // retourne le second membre résultant
662 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
663 Vecteur SM_charge_surfacique_E
664 (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,bool atdt,const
ParaAlgoControle & pa);
665 // calcul des seconds membres suivant les chargements
666 // cas d'un chargement volumique,
667 // force indique la force volumique appliquée
668 // retourne le second membre résultant
669 // ici on l'épaisseur de l'élément pour constituer le volume
670 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
671 Vecteur SM_charge_volumique_E
672 (const Coordonnee& force,Fonction_nD* pt_fonct,bool atdt,const ParaAlgoControle &
pa,bool sur_volume_finale_);
673 // cas d'un chargement surfacique hydrostatique,
674 // poidvol: indique le poids volumique du liquide
675 // M_liquide : un point de la surface libre
676 // dir_normal_liquide : direction normale à la surface libre
677 // retourne le second membre résultant
678 // -> explicite à t
679 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
680 ,int numFace,const Coordonnee& M_liquide,bool atdt
681 ,const ParaAlgoControle & pa
682 ,bool sans_limitation);
683 // cas d'un chargement surfacique hydro-dynamique,
684 // voir méthode explicite plus haut, pour les arguments
685 // retourne le second membre résultant
686 // bool atdt : permet de spécifier à t ou a t+dt
687 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
688 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
689 , CourbelD* coef_aero_t,bool atdt
690 ,const ParaAlgoControle & pa) ;
691 };
692 /// @} // end of group
693
694
695 #endif
696
697
698
699

```

## 7.224 TetraQ.h

```

1 // FICHER : TetraQ.h
2 // CLASSE : TetraQ
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field

```

```

7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      15/11/99
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:   La classe Tetra permet de declarer des elements
41 * Tetraedriques Triquadratiques complets et de realiser
42 * le calcul du residu local et de la raideur locale pour une loi de
43 * comportement donnee. La dimension de l'espace pour un tel element
44 * est 3.
45 *
46 * l'interpolation est Triquadratiques incomplet a 10 noeuds,le nombre
47 * de point d'integration est de 4.
48 *   *****
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !           !
55 *   *****
56 *   MODIFICATIONS:
57 *
58 *   ! date !   auteur !           but
59 *   -----
60 *   *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef TETRAQ_H
67 #define TETRAQ_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "TetraMemb.h"
79 #include "FrontTriaQuad.h"
80 #include "FrontSegQuad.h"
81
82 /// @addtogroup groupe_des_elements_finis
83 /// @{
84 ///
85
86
87 class TetraQ : public TetraMemb
88 {
89
90     public :
91
92     // CONSTRUCTEURS :

```

```

93
94 // Constructeur par default
95 TetraQ ();
96
97 // Constructeur fonction d'un numero
98 // d'identification
99 TetraQ (int num_maill,int num_id);
100
101 // Constructeur fonction d'un numero de maillage et d'identification et
102 // du tableau de connexite des noeuds
103 TetraQ (int num_maill,int num_id,const Tableau<Noeud *>& tab);
104
105 // Constructeur de copie
106 TetraQ (const TetraQ& tetra);
107
108
109 // DESTRUCTEUR :
110 ~TetraQ ();
111
112 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113 // méthode virtuelle
114 Element* Nevez_copie() const { Element * el= new TetraQ(*this); return el;};
115
116 // Surcharge de l'operateur = : realise l'egalite entre deux instances de TetraQ
117 TetraQ& operator= (TetraQ& tetra);
118
119 // METHODES :
120 // 1) derivant des virtuelles pures
121
122 // affichage dans la sortie transmise, des variables duales "nom"
123 // aux differents points d'integration
124 // dans le cas ou nom est vide, affichage de "toute" les variables
125 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127 protected :
128
129 // adressage des frontieres linéiques et surfacique
130 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
131 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
133 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
134 { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
135
136 // VARIABLES PRIVEES :
137 // place memoire commune a tous les elements TetraQ
138 static TetraMemb::DonnComTetra * doCoTetraQ;
139 // idem mais pour les indicateurs qui servent pour l'initialisation
140 static TetraMemb::UneFois uneFois;
141
142 class NombresConstruireTetraQ : public NombresConstruire
143 { public: NombresConstruireTetraQ();
144 };
145 static NombresConstruireTetraQ nombre_V; // les nombres propres à l'élément
146
147 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
148 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
149 class ConsTetraQ : public ConstrucElement
150 { public : ConsTetraQ ()
151 { NouvelleTypeElement nouv (TETRAEDRE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this);
152 if (ParaGlob::NiveauImpression() >= 4)
153 cout << "\n initialisation TetraQ" << endl;
154 Element::listTypeElement.push_back(nouv);
155 };
156 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
157 {Element * pt;
158 pt = new TetraQ (num_maill,num) ;
159 return pt;};
160 // ramene true si la construction de l'element est possible en fonction
161 // des variables globales actuelles: ex en fonction de la dimension
162 bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
163 };
164 static ConsTetraQ consTetraQ;
165 static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
166 };
167 /// @} // end of group
168 #endif
169
170
171
172

```

## 7.225 TetraQ\_cm15pti.h

```

1 // FICHER : TetraQ_cml5pti.h
2

```



```

3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *   DATE:           06/03/2023
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 * *****/
39 *   BUT:   La classe TetraQ_15pti permet de declarer des elements
40 * Tetraedriques Triquadratiques complets avec 15 pti et de realiser
41 * le calcul du residu local et de la raideur locale pour une loi de
42 * comportement donnee. La dimension de l'espace pour un tel element
43 * est 3.
44 *
45 * l'interpolation est Triquadratiques complet a 10 noeuds, le nombre
46 * de point d'integration est de 15.
47 *   *****
48 *
49 *   VERIFICATION:
50 *
51 *   ! date !   auteur !           but
52 *   -----
53 *   !           !           !
54 *   $
55 *   *****
56 *   MODIFICATIONS:
57 *   ! date !   auteur !           but
58 *   -----
59 *   $
60 * *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef TETRAQ_CM15PTI_H
67 #define TETRAQ_CM15PTI_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomQuadrangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "TetraMemb.h"
79 #include "FrontTriaQuad.h"
80 #include "FrontSegQuad.h"
81
82
83 /// @addtogroup groupe_des_elements_finis
84 /// @{
85 ///
86
87 class TetraQ_15pti : public TetraMemb
88 {
89

```

```

90     public :
91
92         // CONSTRUCTEURS :
93
94         // Constructeur par défaut
95         TetraQ_15pti ();
96
97         // Constructeur fonction d'un numero
98         // d'identification
99         TetraQ_15pti (int num_mail,int num_id);
100
101         // Constructeur fonction d'un numero de maillage et d'identification et
102         // du tableau de connexite des noeuds
103         TetraQ_15pti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
104
105         // Constructeur de copie
106         TetraQ_15pti (const TetraQ_15pti& tetra);
107
108
109         // DESTRUCTEUR :
110         ~TetraQ_15pti ();
111
112         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113         // méthode virtuelle
114         Element* Nevez_copie() const { Element * el= new TetraQ_15pti(*this); return el;};
115
116         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TetraQ_15pti
117         TetraQ_15pti& operator= (TetraQ_15pti& tetra);
118
119         // METHODES :
120 // 1) derivant des virtuelles pures
121
122         // affichage dans la sortie transmise, des variables duales "nom"
123         // aux differents points d'integration
124         // dans le cas ou nom est vide, affichage de "toute" les variables
125         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127     protected :
128
129         // adressage des frontières linéiques et surfacique
130         // définit dans les classes dérivées, et utilisées pour la construction des frontières
131         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
133         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
134         { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
135
136     // VARIABLES PRIVEES :
137     // place memoire commune a tous les elements TetraQ_15pti
138     static TetraMemb::DonnComTetra * doCoTetraQ_15pti;
139     // idem mais pour les indicateurs qui servent pour l'initialisation
140     static TetraMemb::UneFois uneFois;
141
142     class NombresConstruireTetraQ_15pti : public NombresConstruire
143     { public: NombresConstruireTetraQ_15pti();
144     };
145     static NombresConstruireTetraQ_15pti nombre_V; // les nombres propres à l'élément
146
147     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
148     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
149     class ConsTetraQ_15pti : public ConstrucElement
150     { public : ConsTetraQ_15pti ()
151     { NouvelleTypeElement nouv (TETRAEDRE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cm15pti");
152     if (ParaGlob::NiveauImpression() >= 4)
153     cout << "\n initialisation TetraQ_15pti" << endl;
154     Element::listTypeElement.push_back(nouv);
155     };
156     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
157     {Element * pt;
158     pt = new TetraQ_15pti (num_maill,num) ;
159     return pt;};
160     // ramene true si la construction de l'element est possible en fonction
161     // des variables globales actuelles: ex en fonction de la dimension
162     bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
163     };
164     static ConsTetraQ_15pti consTetraQ_15pti;
165     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
166 };
167 /// @} // end of group
168 #endif
169
170
171
172

```

## 7.226 TetraQ\_cm1pti.h

```

1 // FICHER : TetraQ_cm1pti.h
2 // CLASSE : TetraQ_cm1pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      15/11/99
34 *
35 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *      $
40 *
41 *      *****
42 *      BUT:      La classe Tetra permet de declarer des elements
43 *      Tetraedriques Triquadratiques complets et de realiser
44 *      le calcul du residu local et de la raideur locale pour une loi de
45 *      comportement donnee. La dimension de l'espace pour un tel element
46 *      est 3.
47 *
48 *      l'interpolation est Triquadratiques incomplet a 10 noeuds, le nombre
49 *      de point d'integration est de 1.
50 *
51 *      *****
52 *
53 *      VERIFICATION:
54 *
55 *      ! date !      auteur !      but
56 *      -----
57 *      !      !      !
58 *
59 *      $
60 *
61 *      *****
62 *
63 *      MODIFICATIONS:
64 *
65 *      ! date !      auteur !      but
66 *      -----
67 *
68 *      $
69 *
70 *      *****
71 */***/
72
73 // -----classe pour un calcul de mecanique-----
74
75 #ifndef TETRAQ_CM1PTI_H
76 #define TETRAQ_CM1PTI_H
77
78 #include "ParaGlob.h"
79 #include "ElemMeca.h"
80 #include "Met_abstraite.h"
81 #include "GeomQuadrangle.h"
82 #include "Noeud.h"
83 #include "UtilLecture.h"
84 #include "Tenseur.h"
85 #include "NevezTenseur.h"
86 #include "Deformation.h"
87 #include "TetraMemb.h"
88 #include "FrontTriaQuad.h"
89 #include "FrontSegQuad.h"
90
91
92 /// @addtogroup groupe_des_elements_finis
93 /// @{
94 ///

```

```

85
86
87 class TetraQ_cmlpti : public TetraMemb
88 {
89
90     public :
91
92         // CONSTRUCTEURS :
93
94         // Constructeur par défaut
95         TetraQ_cmlpti ();
96
97         // Constructeur fonction d'un numero
98         // d'identification
99         TetraQ_cmlpti (int num_mail,int num_id);
100
101         // Constructeur fonction d'un numero de maillage et d'identification et
102         // du tableau de connexite des noeuds
103         TetraQ_cmlpti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
104
105         // Constructeur de copie
106         TetraQ_cmlpti (const TetraQ_cmlpti& tetra);
107
108
109         // DESTRUCTEUR :
110         ~TetraQ_cmlpti ();
111
112         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113         // méthode virtuelle
114         Element* Nevez_copie() const { Element * el= new TetraQ_cmlpti(*this); return el;};
115
116         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TetraQ_cmlpti
117         TetraQ_cmlpti& operator= (TetraQ_cmlpti& tetra);
118
119         // METHODES :
120 // 1) derivant des virtuelles pures
121
122         // affichage dans la sortie transmise, des variables duales "nom"
123         // aux differents points d'integration
124         // dans le cas ou nom est vide, affichage de "toute" les variables
125         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
126
127     protected :
128
129         // adressage des frontières linéiques et surfacique
130         // définit dans les classes dérivées, et utilisées pour la construction des frontières
131         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
133         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
134         { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
135
136     // VARIABLES PRIVEES :
137     // place memoire commune a tous les elements TetraQ_cmlpti
138     static TetraMemb::DonnComTetra * doCoTetraQ_cmlpti;
139     // idem mais pour les indicateurs qui servent pour l'initialisation
140     static TetraMemb::UneFois uneFois;
141
142     class NombresConstruireTetraQ_cmlpti : public NombresConstruire
143     { public: NombresConstruireTetraQ_cmlpti();
144       };
145     static NombresConstruireTetraQ_cmlpti nombre_V; // les nombres propres à l'élément
146
147     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
148     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
149     class ConsTetraQ_cmlpti : public ConstrucElement
150     { public : ConsTetraQ_cmlpti ()
151       { NouvelleTypeElement nouv (TETRAEDRE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cmlpti");
152         if (ParaGlob::NiveauImpression() >= 4)
153           cout << "\n initialisation TetraQ_cmlpti" << endl;
154           Element::listTypeElement.push_back (nouv);
155       };
156       Element * NouvelElement (int num_maill,int num) // un nouvel élément sans rien
157       {Element * pt;
158         pt = new TetraQ_cmlpti (num_maill,num) ;
159         return pt;};
160       // ramene true si la construction de l'element est possible en fonction
161       // des variables globales actuelles: ex en fonction de la dimension
162       bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
163     };
164     static ConsTetraQ_cmlpti consTetraQ_cmlpti;
165     static int bidon; // a virer lorsque l'on n'aura plus a declarer une instance dans herezh
166 };
167 /// @} // end of group
168 #endif
169
170
171

```

172

## 7.227 TriaAxiL1.h

```

1 // FICHER : TriaAxiL1.h
2 // CLASSE : TriaAxiL1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *
35 *   DATE:           15/01/97
36 *
37 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:         Herezh++
40 *
41 *
42 *   But: définition d'un élément triangulaire axisymétrique linéaire
43 *
44 *   *****
45 *
46 *   VERIFICATION:
47 *
48 *   ! date ! auteur ! but
49 *   -----
50 *   !     !     !
51 *
52 *   *****
53 *
54 *   MODIFICATIONS:
55 *
56 *   ! date ! auteur ! but
57 *   -----
58 *
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64
65
66 #ifndef TRIA_AXI_L1_H
67 #define TRIA_AXI_L1_H
68
69 #include "ParaGlob.h"
70 #include "ElemMeca.h"
71 #include "Met_abstraite.h"
72 #include "GeomTriangle.h"
73 #include "Noeud.h"
74 #include "UtilLecture.h"
75 #include "Tenseur.h"
76 #include "NevezTenseur.h"
77 #include "Deformation.h"
78 #include "TriaAxiMemb.h"
79 #include "ElFrontiere.h"
80 #include "FrontSegLine.h"

```

```

81 #include "FrontTriaLine.h"
82
83 /// @addtogroup groupe_des_elements_finis
84 /// @{
85 ///
86
87
88 class TriaAxil1 : public TriaAximemb
89 {
90
91     public :
92
93         // CONSTRUCTEURS :
94         // Constructeur par défaut
95         TriaAxil1 ();
96
97         // Constructeur fonction d'un numero de maillage et d'identification
98         TriaAxil1 (int num_mail,int num_id);
99
100        // Constructeur fonction d'un numero de maillage et d'identification,
101        // du tableau de connexite des noeuds
102        TriaAxil1 (int num_mail,int num_id,const Tableau<Noeud *>& tab);
103
104        // Constructeur de copie
105        TriaAxil1 (const TriaAxil1& tria);
106
107
108        // DESTRUCTEUR :
109        ~TriaAxil1 ();
110
111        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
112        // méthode virtuelle
113        Element* Nevez_copie() const { Element * el= new TriaAxil1(*this); return el;};
114
115        // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaAxil1
116        TriaAxil1& operator= (TriaAxil1& tria);
117
118        // METHODES :
119        // 1) derivant des virtuelles pures
120
121        // affichage dans la sortie transmise, des variables duales "nom"
122        // aux differents points d'integration
123        // dans le cas ou nom est vide, affichage de "toute" les variables
124        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
125
126        // 2) derivant des virtuelles
127        // 3) methodes propres a l'element
128
129        protected :
130
131        // adressage des frontieres linéiques et surfacique
132        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
133        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
134        { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
135        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
136        { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
137
138        // VARIABLES PRIVEES :
139        // place memoire commune a tous les elements TriaAxil1
140        static TriaAximemb::DonnComTria * doCoMembL1;
141        // idem mais pour les indicateurs qui servent pour l'initialisation
142        static TriaAximemb::UneFois uneFoisL1;
143
144        class NombresConstruireTriaAxil1 : public NombresConstruire
145        { public: NombresConstruireTriaAxil1();
146          };
147        static NombresConstruireTriaAxil1 nombre_V; // les nombres propres à l'élément
148
149        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
150        //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
151        class ConsTriaAxil1 : public ConstrucElement
152        { public : ConsTriaAxil1 ()
153          { NouvelleTypeElement nouv (TRIA_AXI,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this);
154            if (ParaGlob::NiveauImpression() >= 4)
155              cout << "\n initialisation TriaAxil1" << endl;
156            Element::listTypeElement.push_back (nouv);
157          };
158          Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
159          {Element * pt;
160            pt = new TriaAxil1 (num_maill,num) ;
161            return pt;};
162          // ramene true si la construction de l'element est possible en fonction
163          // des variables globales actuelles: ex en fonction de la dimension
164          bool Element_possible() { if (ParaGlob::Dimension() == 3) return true; else return false;};
165          };
166        static ConsTriaAxil1 consTriaAxil1;
167 };

```

```

168 /// @} // end of group
169 #endif
170
171
172
173

```

## 7.228 TriaAxiMemb.h

```

1 // FICHER : TriaAxiMemb.h
2 // CLASSE : TriaAxiMemb
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          29/12/2004
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *
40 *      BUT:           La classe TriaAxiMemb permet de declarer des éléments
41 *                   triangulaires axisymétriques. La dimension de l'espace
42 *                   est 3 avec z qui est hors service.
43 *                   Les éléments finis sont ainsi de dimensions 3.
44 *                   L'interpolation le nombre de point d'integration sont
45 *                   definit dans les classes derivees.
46 *                   La classe est virtuelle.
47 *
48 *      *****
49 *
50 *      VERIFICATION:
51 *
52 *      ! date ! auteur ! but
53 *      -----
54 *      ! ! ! !
55 *      *****
56 *
57 *      MODIFICATIONS:
58 *
59 *      ! date ! auteur ! but
60 *      -----
61 *
62 *      *****/
63
64 #ifndef TRIA_AXI_MEMB_H
65 #define TRIA_AXI_MEMB_H
66
67 #include "ParaGlob.h"
68 #include "ElemMeca.h"
69 #include "MetAxisymetrique3D.h"
70 #include "GeomTriangle.h"
71 #include "GeomSeg.h"
72 #include "Noeud.h"
73 #include "UtilLecture.h"
74 #include "Tenseur.h"
75 #include "NevezTenseur.h"
76 #include "Deformation.h"
77

```

```

76 /// @addtogroup groupe_des_elements_finis
77 /// @{
78 ///
79
80
81 class TriaAxiMemb : public ElemMeca
82 {
83
84     public :
85
86         // CONSTRUCTEURS :
87         // Constructeur par default
88         TriaAxiMemb ();
89
90         // Constructeur fonction d'un numero
91         // d'identification , d'identificateur d'interpolation et de geometrie
92         // et éventuellement un string d'information annexe
93         TriaAxiMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string info="");
94
95         // Constructeur fonction d'un numero de maillage et d'identification,
96         // du tableau de connexite des noeuds, d'identificateur d'interpolation et de geometrie
97         // et éventuellement un string d'information annexe
98         TriaAxiMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,
99                     const Tableau<Noeud *>& tab,string info="") ;
100
101         // Constructeur de copie
102         TriaAxiMemb (const TriaAxiMemb& tria);
103
104
105         // DESTRUCTEUR :
106         ~TriaAxiMemb ();
107
108
109         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaAxiMemb
110         TriaAxiMemb& operator= (const TriaAxiMemb& tria);
111
112         // METHODES :
113         // 1) derivant des virtuelles pures
114
115         // Lecture des donnees de la classe sur fichier
116         void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
117
118         // affichage d'info en fonction de ordre
119         // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
120         void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> * tabMaillageNoeud)
121         { return Element::Info_com_El (nombre->nbne,entreePrinc,ordre,tabMaillageNoeud);};
122
123         // ramene l'element geometrique
124         ElemGeomC0& ElementGeometrique() const { return unefois->doCoMemb->tria};
125         // ramene l'element geometrique en constant
126         const ElemGeomC0& ElementGeometrique_const() const { return unefois->doCoMemb->tria};
127
128         // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
129         // associé
130         // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
131         // 1) cas où l'on utilise la place passée en argument
132         Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
133         // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
134         void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
135
136         // inactive les ddl du problème primaire de mécanique
137         inline void Inactive_ddl_primaire()
138         {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};
139         // active les ddl du problème primaire de mécanique
140         inline void Active_ddl_primaire()
141         {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl);};
142         // ajout des ddl de contraintes pour les noeuds de l'élément
143         inline void Plus_ddl_Sigma()
144         {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
145         // inactive les ddl du problème de recherche d'erreur : les contraintes
146         inline void Inactive_ddl_Sigma()
147         {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
148         // active les ddl du problème de recherche d'erreur : les contraintes
149         inline void Active_ddl_Sigma()
150         {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
151         // active le premier ddl du problème de recherche d'erreur : SIGMA11
152         inline void Active_premier_ddl_Sigma()
153         {ElemMeca::Act_premier_ddl_Sigma();};
154
155         // lecture de données diverses sur le flot d'entrée
156         void LectureContraintes (UtilLecture * entreePrinc)
157         { if (unefois->CalResPrem_t == 1)
158             ElemMeca::LectureDesContraintes (false,entreePrinc,lesPtMecaInt.TabSigHH_t());
159             else
160             { ElemMeca::LectureDesContraintes (true,entreePrinc,lesPtMecaInt.TabSigHH_t());
161                 unefois->CalResPrem_t = 1;
162             };
163

```



```

162     };
163
164 // retour des contraintes en absolu retour true si elle existe sinon false
165 bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
166 { if (unefois->CalResPrem_t == 1)
167     ElemMeca::ContraintesEnAbsolues(false,lesPtMecaInt.TabSigHH_t(),tabSig);
168     else
169     { ElemMeca::ContraintesEnAbsolues(true,lesPtMecaInt.TabSigHH_t(),tabSig);
170     unefois->CalResPrem_t = 1;
171     }
172     return true;
173 };
174
175
176 // Libere la place occupee par le residu et eventuellement la raideur
177 // par l'appel de Libere de la classe mere et libere les differents tenseurs
178 // intermediaires cree pour le calcul et les grandeurs pointee
179 // de la raideur et du residu
180 void Libere ();
181
182 // acquisition d'une loi de comportement
183 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
184
185 // test si l'element est complet
186 // = 1 tout est ok, =0 element incomplet
187 int TestComplet();
188
189 // procedure permettant de completer l'element apres
190 // sa creation avec les donnees du bloc transmis
191 // peut etre appeler plusieurs fois
192 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
193 // Compléter pour la mise en place de la gestion de l'hourglass
194 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc) ;
195
196 // ramene vrai si la surface numéro ns existe pour l'élément
197 // dans le cas des éléments triangulaires il ne peut y avoir qu'une
198 // seule surface
199 bool SurfExiste(int ns) const
200     { if ((ns==1)&&(ParaGlob::Dimension() >= 2)) return true; else return false;};
201
202 // ramene vrai si l'arête numéro na existe pour l'élément
203 bool AreteExiste(int na) const
204     {if ((na <= 3) || (na>= 1)) return true; else return false;};
205
206 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
207 // ce tableau et specifique a l'element
208 const DdlElement & TableauDdl() const
209     { return unefois->doCoMemb->tab_ddl; };
210
211 // Calcul du residu local et de la raideur locale,
212 // pour le schema implicite
213 Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
214
215 // Calcul du residu local a t
216 // pour le schema explicite par exemple
217 Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
218     { return TriaAxiMemb::CalculResidu(false,pa);};
219
220 // Calcul du residu local a tdt
221 // pour le schema explicite par exemple
222 Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
223     { return TriaAxiMemb::CalculResidu(true,pa);};
224
225 // Calcul de la matrice masse pour l'élément
226 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
227
228 // ----- calcul dynamique -----
229 // calcul de la longueur d'arrête de l'élément minimal
230 // divisé par la célérité la plus rapide dans le matériau
231 double Long_arrete_mini_sur_c(Enum_dure temps)
232     { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
233
234 //----- calcul d'erreur, remontée des contraintes -----
235 // 1) calcul du résidu et de la matrice de raideur pour le calcul d'erreur
236 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
237 // 2) remontée aux erreurs aux noeuds
238 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
239
240 // ----- affichage ou récupération d'informations -----
241 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
242 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
243 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
244 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
245 // temps: dit si c'est à 0 ou t ou tdt
246 int PointLePlusPres (Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
247     { return PtLePlusPres (temps,enu,M);};
248

```

```

249 // recuperation des coordonnées du point de numéro d'ordre iteg pour
250 // la grandeur enu
251 // temps: dit si c'est à 0 ou t ou tdt
252 // si erreur retourne erreur à true
253 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
254 { return CoordPtInt(temps,enu,iteg,erreur);};
255
256 // récupération des valeurs au numéro d'ordre = iteg pour
257 // les grandeur enu
258 Tableau<double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
enu,int iteg);
259 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
260 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
261 // de conteneurs quelconque associée
262 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int iteg);
263
264 //===== lecture écriture dans base info =====
265
266 // cas donne le niveau de la récupération
267 // = 1 : on récupère tout
268 // = 2 : on récupère uniquement les données variables (supposées comme telles)
269 void Lecture_base_info
270 (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
271 // cas donne le niveau de sauvegarde
272 // = 1 : on sauvegarde tout
273 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
274 void Ecriture_base_info(ofstream& sort,const int cas) ;
275
276 // 2) derivant des virtuelles
277
278 // retourne un tableau de ddl element, correspondant à la
279 // composante de sigma -> SIG11, pour chaque noeud qui contient
280 // des ddl de contrainte
281 // -> utilisé pour l'assemblage de la raideur d'erreur
282 DdlElement& Tableau_de_Sig1() const
283 {return uneFois->doCoMemb->tab_Err1Sig1;};
284
285 // actualisation des ddl et des grandeurs actives de t+dt vers t
286 void TdtversT();
287 // actualisation des ddl et des grandeurs actives de t vers tdt
288 void TversTdt();
289
290 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
291 // qu'une fois la remontée aux contraintes effectuées sinon aucune
292 // action. En retour la valeur de l'erreur sur l'élément
293 // type indique le type de calcul d'erreur :
294 void ErreurElement(int type,double& errElemRelative
295 ,double& numerateur, double& denominateur);
296
297 // calcul des seconds membres suivant les chargements
298 // cas d'un chargement volumique,
299 // force indique la force volumique appliquée
300 // retourne le second membre résultant
301 // ici on l'épaisseur de l'élément pour constituer le volume
302 // -> explicite à t
303 Vecteur SM_charge_volumique_E_t
304 (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
305 { return TriaAxiMemb::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_);};
306 // -> explicite à tdt
307 Vecteur SM_charge_volumique_E_tdt
308 (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
309 { return TriaAxiMemb::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_);};
310 // -> implicite,
311 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
312 // retourne le second membre et la matrice de raideur correspondant
313 ResRaid SMR_charge_volumique_I
314 (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
315 ;
316
317 // cas d'un chargement surfacique, sur les frontières des éléments
318 // force indique la force surfacique appliquée
319 // numface indique le numéro de la face chargée
320 // retourne le second membre résultant
321 // -> version explicite à t
322 Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nd* pt_fonct,int numFace,const
ParaAlgoControle & pa)
323 { return TriaAxiMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa);};
324 // -> version explicite à tdt
325 Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nd* pt_fonct,int numFace,const
ParaAlgoControle & pa)
326 { return TriaAxiMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa);};
327 // -> implicite,
328 // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
329 // retourne le second membre et la matrice de raideur correspondant
330 ResRaid SMR_charge_surfacique_I
331 (const Coordonnee& force,Fonction_nd* pt_fonct,int numFace,const ParaAlgoControle & pa) ;

```

```

332 // cas d'un chargement lineique, sur les aretes frontieres des éléments
333 // force indique la force lineique appliquée
334 // numarete indique le numéro de l'arete chargée
335 // retourne le second membre résultant
336 // -> explicite à t
337 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
338 { return TriaAxiMemb::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); };
339 // -> explicite à tdt
340 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
341 { return TriaAxiMemb::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); };
342 // -> implicite,
343 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
344 // retourne le second membre et la matrice de raideur correspondant
345 ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
346
347 // cas d'un chargement lineique suiveuse, sur les aretes frontieres des éléments 2D (uniquement)
348 // force indique la force lineique appliquée
349 // numarete indique le numéro de l'arete chargée
350 // retourne le second membre résultant
351 // -> explicite à t
352 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
353 { return TriaAxiMemb::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa); };
354 // -> explicite à tdt
355 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
356 { return TriaAxiMemb::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa); };
357 // -> implicite,
358 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
359 // retourne le second membre et la matrice de raideur correspondant
360 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
361
362 // cas d'un chargement de type pression, sur les frontieres des éléments
363 // pression indique la pression appliquée
364 // numface indique le numéro de la face chargée
365 // retourne le second membre résultant
366 // -> explicite à t
367 Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
368 { return TriaAxiMemb::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa); };
369 // -> explicite à tdt
370 Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
371 { return TriaAxiMemb::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa); };
372 // -> implicite,
373 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
374 // retourne le second membre et la matrice de raideur correspondant
375 ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
376
377 // cas d'un chargement surfacique hydrostatique,
378 // poidvol: indique le poids volumique du liquide
379 // M_liquide : un point de la surface libre
380 // dir_normal_liquide : direction normale à la surface libre
381 // retourne le second membre résultant
382 // -> explicite à t
383 Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
384 ,int numFace,const Coordonnee& M_liquide
385 ,const ParaAlgoControle & pa
386 ,bool sans_limitation)
387 { return
TriaAxiMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation); };
388 // -> explicite à tdt
389 Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
390 ,int numFace,const Coordonnee& M_liquide
391 ,const ParaAlgoControle & pa
392 ,bool sans_limitation)
393 { return
TriaAxiMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation); };
394 // -> implicite,
395 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
396 // retourne le second membre et la matrice de raideur correspondant
397 ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
398 ,int numFace,const Coordonnee& M_liquide
399 ,const ParaAlgoControle & pa
400 ,bool sans_limitation) ;
401
402 // cas d'un chargement surfacique hydro-dynamique,
403 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
404 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc unitaire)
405 // une suivant la direction normale à la vitesse de type portance
406 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
407 // une suivant la vitesse tangente de type frottement visqueux

```

```

408 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
409 // coef_mul: est un coefficient multiplicateur global (de tout)
410 // retourne le second membre résultant
411 // -> explicite à t
412 Vecteur SM_charge_hydrodynamique_E_t( CourbeID* frot_fluid,const double& poidvol
413                                     , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
414                                     , CourbeID* coef_aero_t
415                                     ,const ParaAlgoControle & pa)
416 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa)};
417 // -> explicite à tdt
418 Vecteur SM_charge_hydrodynamique_E_tdt( CourbeID* frot_fluid,const double& poidvol
419                                         , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
420                                         , CourbeID* coef_aero_t
421                                         ,const ParaAlgoControle & pa)
422 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa)};
423 // -> implicite,
424 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
425 // retourne le second membre et la matrice de raideur correspondant
426 ResRaid SMR_charge_hydrodynamique_I( CourbeID* frot_fluid,const double& poidvol
427                                       , CourbeID* coef_aero_n,int numFace,const double&
coef_mul
428                                       , CourbeID* coef_aero_t
429                                       ,const ParaAlgoControle & pa) ;
430
431 // ===== définition et/ou construction des frontières =====
432
433 // Calcul des frontieres de l'element
434 // creation des elements frontieres et retour du tableau de ces elements
435 // la création n'a lieu qu'au premier appel
436 // ou lorsque l'on force le paramètre force a true
437 // dans ce dernier cas seul les frontière effacées sont recréée
438 Tableau <ElFrontiere*> const & Frontiere(bool force = false);
439
440 // ramène la frontière point
441 // éventuellement création des frontieres points de l'element et stockage dans l'element
442 // si c'est la première fois sinon il y a seulement retour de l'elements
443 // a moins que le paramètre force est mis a true
444 // dans ce dernier cas la frontière effacée est recréée
445 // num indique le numéro du point à créer (numérotation EF)
446 // ElFrontiere* const Frontiere_points(int num,bool force = false);
447
448
449 // ramène la frontière surfacique
450 // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
451 // si c'est la première fois sinon il y a seulement retour de l'elements
452 // a moins que le paramètre force est mis a true
453 // dans ce dernier cas la frontière effacée est recréée
454 // num indique le numéro de la surface à créer (numérotation EF)
455 // ici normalement uniquement 1 possible
456 // ElFrontiere* const Frontiere_surfacique(int num,bool force = false);
457
458
459 // 3) methodes propres a l'element
460
461 // ajout du tableau specific de ddl des noeuds
462 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
463 // des noeuds constitutants l'element
464 void ConstTabDdl();
465
466 // ----- definition de la classe conteneur de donnees communes -----
467 class DonnComTria
468 { public :
469     DonnComTria (GeomTriangle& tri,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
470                 MetAxisymetrique3D& met_gene, Tableau <Vecteur * > & resEr,Mat_pleine&
raidEr,
471                 GeomTriangle& triEr,GeomTriangle& triS,
472                 GeomSeg& segmS,Vecteur& residu_int,Mat_pleine& raideur_int,
473                 Tableau <Vecteur* > & residus_extN,Tableau <Mat_pleine* >& raideurs_extN,
474                 Tableau <Vecteur* > & residus_extA,Tableau <Mat_pleine* >& raideurs_extA,
475                 Tableau <Vecteur* >& residus_extS,Tableau <Mat_pleine* >& raideurs_extS,
476                 Mat_pleine& mat_masse,GeomTriangle& triMas,int nbi,GeomTriangle* triHourg
477                 );
478     DonnComTria(DonnComTria& a) ;
479     ~DonnComTria();
480     // variables
481     GeomTriangle tria; // contient les fonctions d'interpolation et
482                       // les derivees pour le calcul d'équilibre en déplacement
483     DdlElement tab_ddl; // tableau des degres
484                       //de liberte des noeuds de l'element commun a tous les
485                       // elements. Il s'agit des ddl primaires pour le problème de mécanique
486     MetAxisymetrique3D met_TriaAxiMemb;
487     Mat_pleine matGeom ; // matrice géométrique
488     Mat_pleine matInit ; // matrice initiale

```

```

489     Tableau <TenseurBB *> d_epsBB; // place pour la variation des def
490     Tableau <TenseurHH *> d_sigHH; // place pour la variation des contraintes
491     Tableau <Tableau2 <TenseurBB *> > d2_epsBB; // variation seconde des déformations
492     // ---- concernant les frontières et particulièrement le calcul de second membre
493     GeomTriangle triaS; // contient les fonctions d'interpolation et les derivees
494     GeomSeg      segS; // " " " "
495
496     //----- calcul d'erreur -----
497     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
498     // d'erreur : contraintes
499     DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud, servant pour le
calcul
500     // d'erreur : contraintes, en fait pour l'assemblage
501     Tableau <Vecteur* > resErr; // residu pour le calcul d'erreur
502     Mat_pleine raidErr; // raideur pour le calcul d'erreur
503     GeomTriangle triaEr; // contient les fonctions d'interpolation et
504     // les derivees pour le calcul du hessien dans
505     // la résolution de la fonctionnelle d'erreur
506     // ----- calcul de résidus, de raideur : interne ou pour les efforts extérieurs
-----
507     // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
508     Vecteur residu_interne;
509     Mat_pleine raideur_interne;
510     Tableau <Vecteur* > residus_externesN; // pour les noeuds
511     Tableau <Mat_pleine* > raideurs_externesN; // pour les noeuds
512     Tableau <Vecteur* > residus_externesA; // pour les aretes
513     Tableau <Mat_pleine* > raideurs_externesA; // pour les aretes
514     Tableau <Vecteur* > residus_externesS; // pour les surfaces
515     Tableau <Mat_pleine* > raideurs_externesS; // pour les surfaces
516     // ----- données concernant la dynamique -----
517     Mat_pleine matrice_masse;
518     GeomTriangle triaMas; // contient les fonctions d'interpolation et ...
519     // pour les calculs relatifs à la masse
520     // ----- blocage éventuel d'hourglass
521     // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
Cal_explici_hourglass
522     GeomTriangle* triaHourg; // contient les fonctions d'interpolation
523     };
524
525     // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
526     // et un pointeur sur les données statiques communes
527     // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
528     // classe est défini. Son allocation est effectuée dans les classes dérivées
529     class UneFois
530     { public :
531         UneFois () ; // constructeur par défaut
532         ~UneFois () ; // destructeur
533
534         // VARIABLES :
535     public :
536         TriaAxiMemb::DonnComTria * doCoMemb;
537
538         // incicateurs permettant de dimensionner seulement au premier passage
539         // utilise dans "CalculResidu" et "Calcul_implicit"
540         int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
541         int CalimpPrem;
542         int dualSortTria; // pour la sortie des valeurs au pt d'integ
543         int CalSMlin_t; // pour les seconds membres concernant les arretes
544         int CalSMlin_tdt; // pour les seconds membres concernant les arretes
545         int CalSMRlin; // pour les seconds membres concernant les arretes
546         int CalSMsurf_t; // pour les seconds membres concernant les surfaces
547         int CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
548         int CalSMRsurf; // pour les seconds membres concernant les surfaces
549         int CalSMvol_t; // pour les seconds membres concernant les volumes
550         int CalSMvol_tdt; // pour les seconds membres concernant les volumes
551         int CalSMvol; // pour les seconds membres concernant les volumes
552         int CalDynamique; // pour le calcul de la matrice de masse
553         int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
554         // ----- sauvegarde du nombre d'élément en cours -----
555         int nbelem_in_Prog;
556     };
557
558     // -----
559
560     protected :
561     // VARIABLES PROTÉGÉES :
562     UneFois * unefois; // pointeur défini dans la classe dérivée
563     // les données spécifiques sont groupées dans une structure pour sécuriser
564     // le passage de paramètre dans init par exemple
565     class Donnee_specif
566     { public :
567         Donnee_specif() : // défaut
568             cas_pti_nbi(0),cas_pti_nbiEr(0)
569             ,cas_pti_nbiS(0),cas_pti_nbiMas(0)
570             {};
571         Donnee_specif(int casptnbi,int casptnbiEr,int casptnbiS,int casptnbiMas) : // les indicateurs
572             cas_pti_nbi(casptnbi),cas_pti_nbiEr(casptnbiEr)

```

```

573     ,cas_pti_nbiS(casptnbiS),cas_pti_nbiMas(casptnbiMas)
574     {});
575     Donnee_specif(const Donnee_specif& a) :
576     cas_pti_nbi(a.cas_pti_nbi),cas_pti_nbiEr(a.cas_pti_nbiEr)
577     ,cas_pti_nbiS(a.cas_pti_nbiS),cas_pti_nbiMas(a.cas_pti_nbiMas)
578     {}; // recopie via le constructeur de copie
579     ~Donnee_specif() {});
580     Donnee_specif & operator = ( const Donnee_specif& a)
581     { cas_pti_nbi=a.cas_pti_nbi;cas_pti_nbiEr=a.cas_pti_nbiEr;
582     cas_pti_nbiS=a.cas_pti_nbiS;cas_pti_nbiMas=a.cas_pti_nbiMas;
583     return *this;};
584     // data
585     int cas_pti_nbi; // permet de différencier les différents cas de pt d'integ identique
586     // =0: valeur par défaut, ensuite si diff de 1 donne les différents
cas
587     // est par exemple utilisé pour différencier le cas de 3 pt sur les
arrêtes
588     // ou 3 pt en interne, dans ce cas vaut 1, il peut ainsi y avoir plus
de 2 cas
589     int cas_pti_nbiEr; // idem pour l'erreur
590     int cas_pti_nbiS; // idem pour le calcul de second membre surfacique
591     int cas_pti_nbiMas; // idem pour le calcul de la matrice masse
592     };
593     Donnee_specif donnee_specif;
594
595     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
596     LesPtIntegMecaInterne lesPtMecaInt;
597
598     // type structuré et fonction virtuelle pour construire les éléments
599     // la fonction est défini dans le type dérivé
600     class NombresConstruire
601     { public:
602     NombresConstruire():nbne(0),nbneA(0),nbi(0)
603     ,nbiEr(0),nbiS(0),nbiA(0),nbiMas(0),nbiHour(0) {});
604     int nbne; // le nombre de noeud de l'élément
605     int nbneA; // le nombre de noeud des aretes
606     int nbi; // le nombre de point d'intégration pour le calcul mécanique
607     int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
608     int nbiS; // le nombre de point d'intégration pour le calcul de second membre surfacique
609     int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
610     int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse
611     int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
612     };
613     NombresConstruire * nombre; // le pointeur défini dans la classe dérivée d'hexamemb
614
615     // METHODES
616     // =====» methodes appelees par les classes derivees «=====
617
618     // fonction d'initialisation servant dans les classes derivant
619     // au niveau du constructeur, si rien initialisation par default
620     TriaAxiMemb::DonnComTria* Init (bool sans_init_noeud = false);
621     TriaAxiMemb::DonnComTria* Init (Donnee_specif donnee_specif,bool sans_init_noeud = false);
622     // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
623     void Destruction();
624
625
626     // ==== » methodes virtuelles derivant d'ElemMeca =====
627     // ramene la dimension des tenseurs contraintes et déformations de l'élément
628     int Dim_sig_eps() const {return 3;};
629
630     //----- fonctions uniquement a usage interne -----
631     private :
632     // definition des données communes : CoTria
633     TriaAxiMemb::DonnComTria* Def_DonneeCommune();
634     // Calcul du residu local a t ou tdt en fonction du booleen
635     Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
636     // calcul des seconds membres suivant les chargements
637     // cas d'un chargement volumique,
638     // force indique la force volumique appliquée
639     // retourne le second membre résultant
640     // ici on l'épaisseur de l'élément pour constituer le volume
641     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
642     Vecteur SM_charge_volumique_E
643     (const Coordonnee& force,Fonction_nD* pt_fonct,bool atdt,const ParaAlgoControle & pa,bool
sur_volume_finale_);
644     // cas d'un chargement surfacique, sur les frontières des éléments
645     // force indique la force surfacique appliquée
646     // numface indique le numéro de la face chargée
647     // retourne le second membre résultant
648     // -> explicite à t ou tdt en fonction de la variable booleenne atdt
649     Vecteur SM_charge_surfacique_E
650     (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,bool atdt,const
ParaAlgoControle & pa);
651     // cas d'un chargement lineique, sur les aretes frontières des éléments
652     // force indique la force lineique appliquée
653     // numarete indique le numéro de l'arete chargée
654     // retourne le second membre résultant

```

```

655 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
656 Vecteur SM_charge_lineique_E
657 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
658 // cas d'un chargement lineique suivieuse, sur les aretes frontieres des éléments 2D (uniquement)
659 // force indique la force lineique appliquee
660 // numarete indique le numéro de l'arete chargée
661 // retourne le second membre résultant
662 // -> explicite à t
663 Vecteur SM_charge_lineique_Suiv_E
664 (const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,bool atdt,const
ParaAlgoControle & pa);
665 // cas d'un chargement de type pression, sur les frontieres des éléments
666 // pression indique la pression appliquee
667 // numface indique le numéro de la face chargée
668 // retourne le second membre résultant
669 // -> explicite à t ou tdt en fonction de la variable booléenne atdt
670 Vecteur SM_charge_pression_E
671 (double pression,Fonction_nD* pt_fonct,int numFace,bool atdt,const ParaAlgoControle &
pa);
672 // cas d'un chargement surfacique hydrostatique,
673 // poidvol: indique le poids volumique du liquide
674 // M_liquide : un point de la surface libre
675 // dir_normal_liquide : direction normale à la surface libre
676 // retourne le second membre résultant
677 // -> explicite à t
678 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
679 ,int numFace,const Coordonnee& M_liquide,bool atdt
680 ,const ParaAlgoControle & pa
681 ,bool sans_limitation);
682 // cas d'un chargement surfacique hydro-dynamique,
683 // voir methode explicite plus haut, pour les arguments
684 // retourne le second membre résultant
685 // bool atdt : permet de spécifier à t ou a t+dt
686 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
687 , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
688 , CourbelD* coef_aero_t,bool atdt
,const ParaAlgoControle & pa) ;
689 };
690 /// @} // end of group
691 #endif
692
693
694
695

```

## 7.229 TriaAxiQ3.h

```

1 // FICHER : TriaAxiQ3.h
2 // CLASSE : TriaAxiQ3
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *
34 * DATE: 15/01/97 *
35 * * *
36 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *

```

```

37 *                                     $ *
38 *   PROJET:      Herezh++                                     *
39 *                                     $ *
40 *****
41 *   BUT:      Element triangulaire, quadratique 3 pt d'integ. *
42 *             ici les 3 pti sont sur les arêtes                 *
43 *                                     $ *
44 *   ***** *
45 *
46 *   VERIFICATION: *
47 *
48 *   ! date !   auteur !           but           ! *
49 *   ----- *
50 *   !           !           !           ! *
51 *
52 *                                     $ *
53 *   ***** *
54 *   MODIFICATIONS: *
55 *
56 *   ! date !   auteur !           but           ! *
57 *   ----- *
58 *
59 *                                     $ *
60 *
61 *****/
62
63 // -----classe pour un calcul de mecanique-----
64
65 // La classe TriaAxiMemb permet de declarer des elements triangulaire membrane et de realiser
66 // le calcul du residu local et de la raideur locale pour une loi de comportement
67 // donnee. La dimension de l'espace pour un tel element est 2
68 //
69 // l'interpolation est quadratique, le nombre de point d'integration est de 3
70
71
72 #ifndef TRIA_AXI_Q3_H
73 #define TRIA_AXI_Q3_H
74
75 #include "ParaGlob.h"
76 #include "ElemMeca.h"
77 #include "Met_abstraite.h"
78 #include "GeomTriangle.h"
79 #include "Noeud.h"
80 #include "UtilLecture.h"
81 #include "Tenseur.h"
82 #include "NevezTenseur.h"
83 #include "Deformation.h"
84 #include "TriaAxiMemb.h"
85 #include "FrontSegQuad.h"
86 #include "FrontTriaQuad.h"
87
88
89 /// @addtogroup groupe_des_elements_finis
90 /// @{
91 ///
92
93 class TriaAxiQ3 : public TriaAxiMemb
94 {
95
96     public :
97
98         // CONSTRUCTEURS :
99         // Constructeur par default
100         TriaAxiQ3 ();
101
102         // Constructeur fonction d'un numero de maillage et d'identification
103         TriaAxiQ3 (int num_mail,int num_id);
104
105         // Constructeur fonction d'un numero d'identification,
106         // du tableau de connexite des noeuds
107         TriaAxiQ3 (int num_mail,int num_id,const Tableau<Noeud *>& tab);
108
109         // Constructeur de copie
110         TriaAxiQ3 (const TriaAxiQ3& tria);
111
112
113         // DESTRUCTEUR :
114         ~TriaAxiQ3 ();
115
116         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
117         // méthode virtuelle
118         Element* Nevez_copie() const { Element * el= new TriaAxiQ3(*this); return el;};
119
120         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaAxiQ3
121         TriaAxiQ3& operator= (TriaAxiQ3& tria);
122
123         // METHODES :

```



```

124 // 1) derivant des virtuelles pures
125
126     // affichage dans la sortie transmise, des variables duales "nom"
127     // aux differents points d'integration
128     // dans le cas ou nom est vide, affichage de "toute" les variables
129     void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
130
131 protected :
132
133     // adressage des frontieres linéiques et surfacique
134     // définit dans les classes dérivées, et utilisées pour la construction des frontieres
135     ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
136     { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
137     ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
138     { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
139
140 // VARIABLES PRIVEES :
141 // place memoire commune a tous les elements TriaAxiQ3
142 static TriaAxiMemb::DonnComTria * doCoMembQ3;
143 // idem mais pour les indicateurs qui servent pour l'initialisation
144 static TriaAxiMemb::UneFois uneFoisQ3;
145
146 class NombresConstruireTriaAxiQ3 : public NombresConstruire
147 { public: NombresConstruireTriaAxiQ3();
148 };
149 static NombresConstruireTriaAxiQ3 nombre_V; // les nombres propres à l'élément
150
151 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
152 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
153 class ConsTriaAxiQ3 : public ConstrucElement
154 { public : ConsTriaAxiQ3 ()
155     { NouvelleTypeElement nouv(TRIA_AXI,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this);
156     if (ParaGlob::NiveauImpression() >= 4)
157     cout << "\n initialisation TriaAxiQ3" << endl;
158     Element::listTypeElement.push_back(nouv);
159     };
160     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
161     {Element * pt;
162     pt = new TriaAxiQ3 (num_maill,num) ;
163     return pt;};
164     // ramene true si la construction de l'element est possible en fonction
165     // des variables globales actuelles: ex en fonction de la dimension
166     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
167 };
168 static ConsTriaAxiQ3 consTriaAxiQ3;
169 };
170 /// @} // end of group
171 #endif
172
173
174
175

```

## 7.230 TriaAxiQ3\_cm1pti.h

```

1 // FICHER : TriaAxiQ3_cmlpti.h
2 // CLASSE : TriaAxiQ3_cmlpti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.

```

```

31
32 /*****
33 *
34 *   DATE:      15/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *****/
41 *   BUT:      Element triangulaire, quadratique 1 pt d'integ.
42 *            et gestion des modes d'hourglass.
43 *
44 *   *****
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !       but
48 *   -----
49 *   !       !       !
50 *
51 *   *****
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !       but
55 *   -----
56 *
57 *
58 *
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64 // La classe TriaAxiMemb permet de declarer des elements triangulaire membrane et de realiser
65 // le calcul du residu local et de la raideur locale pour une loi de comportement
66 // donnee. La dimension de l'espace pour un tel element est 2
67 //
68 // l'interpolation est quadratique, le nombre de point d'integration est de 3
69
70
71 #ifndef TRIA_AXI_Q3_CM1PTI_H
72 #define TRIA_AXI_Q3_CM1PTI_H
73
74 #include "ParaGlob.h"
75 #include "ElemMeca.h"
76 #include "Met_abstraite.h"
77 #include "GeomTriangle.h"
78 #include "Noeud.h"
79 #include "UtilLecture.h"
80 #include "Tenseur.h"
81 #include "NevezTenseur.h"
82 #include "Deformation.h"
83 #include "TriaAxiMemb.h"
84 #include "FrontSegQuad.h"
85 #include "FrontTriaQuad.h"
86
87 /// @addtogroup groupe_des_elements_finis
88 /// @{
89 ///
90
91
92 class TriaAxiQ3_cmlpti : public TriaAxiMemb
93 {
94     public :
95
96         // CONSTRUCTEURS :
97         // Constructeur par defaut
98         TriaAxiQ3_cmlpti ();
99
100        // Constructeur fonction d'un numero de maillage et d'identification
101        TriaAxiQ3_cmlpti (int num_mail,int num_id);
102
103        // Constructeur fonction d'un numero d'identification,
104        // du tableau de connexite des noeuds
105        TriaAxiQ3_cmlpti (int num_mail,int num_id,const Tableau<Noeud *>& tab);
106
107        // Constructeur de copie
108        TriaAxiQ3_cmlpti (const TriaAxiQ3_cmlpti& tria);
109
110
111        // DESTRUCTEUR :
112        ~TriaAxiQ3_cmlpti ();
113
114        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
115        // méthode virtuelle
116        Element* Nevez_copie() const { Element * el= new TriaAxiQ3_cmlpti(*this); return el;};
117

```

```

118
119 // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaAxiQ3_cmplpti
120 TriaAxiQ3_cmplpti& operator= (TriaAxiQ3_cmplpti& tria);
121
122 // METHODES :
123 // 1) derivant des virtuelles pures
124
125 // affichage dans la sortie transmise, des variables duales "nom"
126 // aux differents points d'integration
127 // dans le cas ou nom est vide, affichage de "toute" les variables
128 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
129
130 protected :
131
132 // adressage des frontieres linéiques et surfacique
133 // définit dans les classes dérivées, et utilisées pour la construction des frontieres
134 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
136 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
137 { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
138
139 // VARIABLES PRIVEES :
140 // place memoire commune a tous les elements TriaAxiQ3_cmplpti
141 static TriaAxiMemb::DonnComTria * doCoMembQ3;
142 // idem mais pour les indicateurs qui servent pour l'initialisation
143 static TriaAxiMemb::UneFois uneFoisQ3;
144
145 class NombresConstruireTriaAxiQ3_cmplpti : public NombresConstruire
146 { public: NombresConstruireTriaAxiQ3_cmplpti();
147 };
148 static NombresConstruireTriaAxiQ3_cmplpti nombre_V; // les nombres propres à l'élément
149
150 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
151 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
152 class ConsTriaAxiQ3_cmplpti : public ConstrucElement
153 { public : ConsTriaAxiQ3_cmplpti ()
154 { NouvelleTypeElement nouv (TRIA_AXI,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cmplpti");
155 if (ParaGlob::NiveauImpression() >= 4)
156 cout << "\n initialisation TriaAxiQ3_cmplpti" << endl;
157 Element::listTypeElement.push_back(nouv);
158 };
159 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
160 {Element * pt;
161 pt = new TriaAxiQ3_cmplpti (num_maill,num) ;
162 return pt;};
163 // ramene true si la construction de l'element est possible en fonction
164 // des variables globales actuelles: ex en fonction de la dimension
165 bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
166 };
167 static ConsTriaAxiQ3_cmplpti consTriaAxiQ3_cmplpti;
168 };
169 /// @} // end of group
170 #endif
171
172
173
174

```

## 7.231 TriaAxiQ3\_cmpti1003.h

```

1 // FICHER : TriaAxiQ3_cmpti1003.h
2 // CLASSE : TriaAxiQ3_cmpti1003
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.

```

```

26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           09/11/2016                               $   *
34 *                                                         $   *
35 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)      $   *
36 *                                                         $   *
37 *   PROJET:        Herezh++                                $   *
38 *                                                         $   *
39 *****/
40 *   BUT:           Element triangulaire, quadratique 3 pt d'integ
41 *                 qui ne sont pas sur les arêtes.          $   *
42 *                                                         $   *
43 *   *****
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !           but                       !   *
47 *   -----
48 *   !       !           !           !                       !   *
49 *
50 *                                                         $   *
51 *   *****
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but                       !   *
55 *   -----
56 *
57 *                                                         $   *
58 *                                                         *
59 *****/
60
61 // -----classe pour un calcul de mecanique-----
62
63
64
65 #ifndef TRIA_AXI_Q3_CMPTI1003H
66 #define TRIA_AXI_Q3_CMPTI1003H
67
68 #include "ParaGlob.h"
69 #include "ElemMeca.h"
70 #include "Met_abstraite.h"
71 #include "GeomTriangle.h"
72 #include "Noeud.h"
73 #include "UtilLecture.h"
74 #include "Tenseur.h"
75 #include "NevezTenseur.h"
76 #include "Deformation.h"
77 #include "TriaAxiMemb.h"
78 #include "FrontSegQuad.h"
79 #include "FrontTriaQuad.h"
80
81 /// @addtogroup groupe_des_elements_finis
82 /// @{
83 ///
84
85
86 class TriaAxiQ3_cmpti1003 : public TriaAxiMemb
87 {
88
89     public :
90
91         // CONSTRUCTEURS :
92         // Constructeur par default
93         TriaAxiQ3_cmpti1003 ();
94
95         // Constructeur fonction d'un numero de maillage et d'identification
96         TriaAxiQ3_cmpti1003 (int num_mail,int num_id);
97
98         // Constructeur fonction d'un numero d'identification,
99         // du tableau de connexite des noeuds
100        TriaAxiQ3_cmpti1003 (int num_mail,int num_id,const Tableau<Noeud *>& tab);
101
102        // Constructeur de copie
103        TriaAxiQ3_cmpti1003 (const TriaAxiQ3_cmpti1003& tria);
104
105
106        // DESTRUCTEUR :
107        ~TriaAxiQ3_cmpti1003 ();
108
109        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
110        // méthode virtuelle
111        Element* Nevez_copie() const { Element * el= new TriaAxiQ3_cmpti1003(*this); return el;};
112

```

```

113         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaAxiQ3_cmptil1003
114         TriaAxiQ3_cmptil1003& operator= (TriaAxiQ3_cmptil1003& tria);
115
116         // METHODES :
117 // 1) derivant des virtuelles pures
118
119         // affichage dans la sortie transmise, des variables duales "nom"
120         // aux differents points d'integration
121         // dans le cas ou nom est vide, affichage de "toute" les variables
122         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
123
124     protected :
125
126         // adressage des frontieres lineiques et surfacique
127         // definit dans les classes derivees, et utilisees pour la construction des frontieres
128         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
129         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
130         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
131         { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
132
133     // VARIABLES PRIVEES :
134     // place memoire commune a tous les elements TriaAxiQ3_cmptil1003
135     static TriaAxiMemb::DonnComTria * doCoMembQ3;
136     // idem mais pour les indicateurs qui servent pour l'initialisation
137     static TriaAxiMemb::UneFois  uneFoisQ3;
138
139     class NombresConstruireTriaAxiQ3_cmptil1003 : public NombresConstruire
140     { public: NombresConstruireTriaAxiQ3_cmptil1003();
141       };
142     static NombresConstruireTriaAxiQ3_cmptil1003 nombre_V; // les nombres propres à l'élément
143
144     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
145     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
146     class ConsTriaAxiQ3_cmptil1003 : public ConstrucElement
147     { public : ConsTriaAxiQ3_cmptil1003 ()
148       { NouvelleTypeElement nouv(TRIA_AXI,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cm3pti");
149         if (ParaGlob::NiveauImpression() >= 4)
150           cout << "\n initialisation TriaAxiQ3_cmptil1003" << endl;
151         Element::listTypeElement.push_back(nouv);
152       };
153       Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
154       {Element * pt;
155         pt = new TriaAxiQ3_cmptil1003 (num_maill,num) ;
156         return pt;};
157       // ramene true si la construction de l'element est possible en fonction
158       // des variables globales actuelles: ex en fonction de la dimension
159       bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
160     };
161     static ConsTriaAxiQ3_cmptil1003 consTriaAxiQ3_cmptil1003;
162 };
163 /// @} // end of group
164 #endif
165
166
167
168

```

## 7.232 Met\_triMemb.h

```

1 // FICHER : Met_triMemb.h
2 // CLASSE : Met_triMemb
3
4 // La classe Met_triMemb est une classe derivee de la classe Met_abstraite
5 // et permet de realiser tous les calculs lies a la metrique d'une facette triangulaire
6 // avec interpolation classique
7 // N.B. : La dimension de l'espace est 2
8
9
10 #ifndef MET_TRIAMEMB_H
11 #define MET_TRIAMEMB_H
12
13 #include "Met_abstraite.h"
14
15
16 class Met_triMemb : public Met_abstraite
17 {
18
19     public :
20
21
22
23         // CONSTRUCTEUR :
24
25         // Constructeur par default
26         Met_triMemb ();

```

```

27     // constructeur permettant de dimensionner unique ment certaine variables
28     // dim_base = dimension de l'espace, nb de vecteur des bases ici = 2 (cf Met_abstraite),
29     // tab = liste des variables a initialiser
30     Met_triaMemb (int dim_base,DdlElement& tabddl,Tableau<Enum_variable_metrique> & tab);
31     // constructeur de copie
32     Met_triaMemb (Met_triaMemb&);
33     // DESTRUCTEUR :
34
35     ~Met_triaMemb ();
36
37
38 };
39
40
41 #endif
42
43

```

## 7.233 TriaCub.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *
32 *   DATE:           21/04/2005
33 *
34 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:         Herezh++
37 *
38 *
39 *   BUT:            Element triangulaire, cubique 3 pt d'integ.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   !-----!-----!-----!-----!
47 *
48 *
49 *   *****
50 *
51 *   MODIFICATIONS:
52 *
53 *   ! date !   auteur !           but
54 *   !-----!-----!-----!-----!
55 *
56 *
57 *   *****/
58
59 // -----classe pour un calcul de mecanique-----
60
61
62 #ifndef TRIACUB_H
63 #define TRIACUB_H
64

```

```

65 #include "ParaGlob.h"
66 #include "ElemMeca.h"
67 #include "Met_abstraite.h"
68 #include "GeomTriangle.h"
69 #include "Noeud.h"
70 #include "UtilLecture.h"
71 #include "Tenseur.h"
72 #include "NevezTenseur.h"
73 #include "Deformation.h"
74 #include "TriaMemb.h"
75 #include "FrontSegQuad.h"
76 #include "FrontTriaQuad.h"
77
78 /// @addtogroup groupe_des_elements_finis
79 /// @{
80 ///
81
82
83 class TriaCub : public TriaMemb
84 {
85
86     public :
87
88         // CONSTRUCTEURS :
89         // Constructeur par défaut
90         TriaCub ();
91
92         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
93         // d'identification
94         TriaCub (double epaiss,int num_mail=0,int num_id=-3);
95
96         // Constructeur fonction d'un numero de maillage et d'identification
97         TriaCub (int num_mail,int num_id);
98
99         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
100        // du tableau de connexite des noeuds
101        TriaCub (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
102
103        // Constructeur de copie
104        TriaCub (const TriaCub& tria);
105
106
107        // DESTRUCTEUR :
108        ~TriaCub ();
109
110        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
111        // méthode virtuelle
112        Element* Nevez_copie() const { Element * el= new TriaCub(*this); return el;};
113
114        // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaCub
115        TriaCub& operator= (TriaCub& tria);
116
117        // METHODES :
118        // 1) derivant des virtuelles pures
119
120        // affichage dans la sortie transmise, des variables duales "nom"
121        // aux differents points d'integration
122        // dans le cas ou nom est vide, affichage de "toute" les variables
123        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
124
125    protected :
126
127        // adressage des frontières linéiques et surfacique
128        // définit dans les classes dérivées, et utilisées pour la construction des frontières
129        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
130        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
131        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132        { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
133
134    // VARIABLES PRIVEES :
135    // place memoire commune a tous les elements TriaCub
136    static TriaMemb::DonnComTria * doCoMembQ3;
137    // idem mais pour les indicateurs qui servent pour l'initialisation
138    static TriaMemb::UneFois uneFoisQ3;
139
140    class NombresConstruireTriaCub : public NombresConstruire
141    { public: NombresConstruireTriaCub();
142      };
143    static NombresConstruireTriaCub nombre_V; // les nombres propres à l'élément
144
145    // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
146    //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
147    class ConsTriaCub : public ConstrucElement
148    { public : ConsTriaCub ()
149      { NouvelleTypeElement nouv (TRIANGLE,CUBIQUE,MECA_SOLIDE_DEFORMABLE,this);
150        if (ParaGlob::NiveauImpression() >= 4)
151          cout << "\n initialisation TriaCub" << endl;

```

```

152         Element::listTypeElement.push_back(nouv);
153     };
154     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
155     {Element * pt;
156       pt = new TriaCub (num_maill,num) ;
157       return pt;};
158     // ramene true si la construction de l'element est possible en fonction
159     // des variables globales actuelles: ex en fonction de la dimension
160     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
161 };
162     static ConsTriaCub consTriaCub;
163 };
164 /// @} // end of group
165 #endif
166
167
168
169

```

## 7.234 TriaCub\_cm4pti.h

```

1 // FICHER : TriaCub_cm4pti.h
2 // CLASSE : TriaCub_cm4pti
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *
34 *   DATE:           3/05/2011
35 *
36 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:        Herezh++
39 *
40 *
41 *   BUT:           Element triangulaire, cubique 4 pt d'integ.
42 *                 et gestion des modes d'hourglass.
43 *
44 *   *****
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   -----
49 *   !           !           !           !
50 *
51 *
52 *   *****
53 *   MODIFICATIONS:
54 *
55 *   ! date !   auteur !           but
56 *   -----
57 *
58 *
59 *
60 *****/
61
62 // -----classe pour un calcul de mecanique-----
63
64

```



```

65 #ifndef TRIACUB_CM4PTI_H
66 #define TRIACUB_CM4PTI_H
67
68 #include "ParaGlob.h"
69 #include "ElemMeca.h"
70 #include "Met_abstraite.h"
71 #include "GeomTriangle.h"
72 #include "Noeud.h"
73 #include "UtilLecture.h"
74 #include "Tenseur.h"
75 #include "NevezTenseur.h"
76 #include "Deformation.h"
77 #include "TriaMemb.h"
78 #include "FrontSegQuad.h"
79 #include "FrontTriaQuad.h"
80
81 /// @addtogroup groupe_des_elements_finis
82 /// @{
83 ///
84
85
86 class TriaCub_cm4pti : public TriaMemb
87 {
88
89     public :
90
91         // CONSTRUCTEURS :
92         // Constructeur par default
93         TriaCub_cm4pti ();
94
95         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
96         // d'identification
97         TriaCub_cm4pti (double epaiss,int num_mail=0,int num_id=-3);
98
99         // Constructeur fonction d'un numero de maillage et d'identification
100        TriaCub_cm4pti (int num_mail,int num_id);
101
102        // Constructeur fonction d'une epaisseur, d'un numero d'identification,
103        // du tableau de connexite des noeuds
104        TriaCub_cm4pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
105
106        // Constructeur de copie
107        TriaCub_cm4pti (const TriaCub_cm4pti & tria);
108
109
110        // DESTRUCTEUR :
111        ~TriaCub_cm4pti ();
112
113        // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
114        // méthode virtuelle
115        Element* Nevez_copie() const { Element * el= new TriaCub_cm4pti(*this); return el;};
116
117        // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaCub_cm4pti
118        TriaCub_cm4pti& operator= (TriaCub_cm4pti& tria);
119
120        // METHODES :
121        // 1) derivant des virtuelles pures
122
123        // affichage dans la sortie transmise, des variables duales "nom"
124        // aux differents points d'integration
125        // dans le cas ou nom est vide, affichage de "toute" les variables
126        void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
127
128        protected :
129
130        // adressage des frontieres linéiques et surfacique
131        // définit dans les classes dérivées, et utilisées pour la construction des frontieres
132        ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
133        { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
134        ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
135        { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
136
137        // VARIABLES PRIVEES :
138        // place memoire commune a tous les elements TriaCub_cm4pti
139        static TriaMemb::DonnComTria * doCoMembQ3;
140        // idem mais pour les indicateurs qui servent pour l'initialisation
141        static TriaMemb::UneFois uneFoisQ3;
142
143        class NombresConstruireTriaCub_cm4pti : public NombresConstruire
144        { public: NombresConstruireTriaCub_cm4pti();
145          };
146        static NombresConstruireTriaCub_cm4pti nombre_V; // les nombres propres à l'élément
147
148        // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
149        //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
150        class ConsTriaCub_cm4pti : public ConstrucElement
151        { public : ConsTriaCub_cm4pti ()

```

```

152         { NouvelleTypeElement nouv(TRIANGLE,CUBIQUE,MECA_SOLIDE_DEFORMABLE,this,"_cm4pti");
153         if (ParaGlob::NiveauImpression() >= 4)
154             cout << "\n initialisation TriaCub_cm4pti" << endl;
155         Element::listTypeElement.push_back(nouv);
156     };
157     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
158     {Element * pt;
159     pt = new TriaCub_cm4pti (num_maill,num) ;
160     return pt;};
161     // ramene true si la construction de l'element est possible en fonction
162     // des variables globales actuelles: ex en fonction de la dimension
163     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
164 };
165     static ConsTriaCub_cm4pti consTriaCub_cm4pti;
166 };
167 /// @} // end of group
168 #endif
169
170
171
172

```

## 7.235 TriaMemb.h

```

1 // FICHER : TriaMemb.h
2 // CLASSE : TriaMemb
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *
35 *     DATE:          15/01/97
36 *
37 *     AUTEUR:       G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *     PROJET:       Herezh++
40 *
41 *
42 * La classe TriaMemb permet de declarer des elements triangulaire membrane et de realiser
43 * le calcul du residu local et de la raideur locale pour une loi de comportement
44 * donnee. La dimension de l'espace pour un tel element est 2
45 *
46 * l'interpolation le nombre de point d'integration sont definit dans les classes derivees
47 *
48 * l'element est virtuel
49 *
50 *     *****
51 *
52 * VERIFICATION:
53 *
54 * ! date ! auteur ! but
55 * -----
56 * ! ! !
57 *
58 *     *****
59 *
60 * MODIFICATIONS:

```

```

61 *      ! date !   auteur !           but           !      *
62 *      -----
63 *
64 *
65 *
66 * *****/
67
68
69 // -----classe pour un calcul de mecanique-----
70
71 // La classe TriaMemb permet de declarer des elements triangulaire membrane et de realiser
72 // le calcul du residu local et de la raideur locale pour une loi de comportement
73 // donnee. La dimension de l'espace pour un tel element est 2
74 //
75 // l'interpolation le nombre de point d'integration sont definit dans les classes derivees
76 //
77 // l'element est virtuel
78
79
80 #ifndef TRIAMEMB_H
81 #define TRIAMEMB_H
82
83 #include "ParaGlob.h"
84 #include "ElemMeca.h"
85 #include "Met_abstraite.h"
86 #include "GeomTriangle.h"
87 #include "GeomSeg.h"
88 #include "Noeud.h"
89 #include "UtilLecture.h"
90 #include "Tenseur.h"
91 #include "NevezTenseur.h"
92 #include "Deformation.h"
93 #include "Epai.h"
94
95 /// @addtogroup groupe_des_elements_finis
96 /// @{
97 ///
98
99
100 class TriaMemb : public ElemMeca
101 {
102
103     public :
104
105         // CONSTRUCTEURS :
106         // Constructeur par default
107         TriaMemb ();
108
109         // Constructeur fonction d'un numero
110         // d'identification , d'identificateur d'interpolation et de geometrie
111         // et éventuellement un string d'information annexe
112         TriaMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string info="");
113
114         // Constructeur fonction d'un numero de maillage et d'identification,
115         // du tableau de connexite des noeuds, d'identificateur d'interpolation et de geometrie
116         // et éventuellement un string d'information annexe
117         TriaMemb (int num_mail,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,
118                 const Tableau<Noeud *>& tab,string info="") ;
119
120         // Constructeur de copie
121         TriaMemb (const TriaMemb& tria);
122
123
124         // DESTRUCTEUR :
125         ~TriaMemb ();
126
127
128         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaMemb
129         TriaMemb& operator= (const TriaMemb& tria);
130
131         // METHODES :
132         // 1) derivant des virtuelles pures
133
134         // Lecture des donnees de la classe sur fichier
135         void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
136
137         // affichage d'info en fonction de ordre
138         // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
139         void Info_com_Element (UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> * tabMaillageNoeud)
140         { return Element::Info_com_El (nombre->nbne,entreePrinc,ordre,tabMaillageNoeud);};
141
142         // ramene l'element geometrique
143         ElemGeomC0& ElementGeometrique() const { return unefois->doCoMemb->tria;};
144         // ramene l'element geometrique en constant
145         const ElemGeomC0& ElementGeometrique_const() const { return unefois->doCoMemb->tria;};
146
147         // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence

```

```

    associé
148     // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
149     // 1) cas où l'on utilise la place passée en argument
150     Coordonnee & Point_physique(const Coordonnee& c_int, Coordonnee & co, Enum_dure temps);
151     // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co
152     void Point_physique(const Coordonnee& c_int, Tableau <Coordonnee> & t_co);
153
154     // -- connaissances particulières sur l'élément
155     // ramène l'épaisseur de l'élément
156     // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
157     virtual double Epaisseurs(Enum_dure enu , const Coordonnee& );
158     // ramène l'épaisseur moyenne de l'élément (indépendante du point)
159     // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
160     virtual double EpaisseurMoyenne(Enum_dure enu ) {return H_moy(enu);};
161
162     // retourne la liste des données particulières actuellement utilisés
163     // par l'élément (actif ou non), sont exclu de cette liste les données particulières des noeuds
164     // reliés à l'élément
165     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
166     List_io <TypeQuelconque> Les_types_particuliers_internes(bool absolue) const;
167
168     // récupération de grandeurs particulières au numéro d'ordre = iteg
169     // celles-ci peuvent être quelconques
170     // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
171     // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
172     void Grandeur_particuliere (bool absolue, List_io<TypeQuelconque>& liTQ, int iteg);
173
174     // inactive les ddl du problème primaire de mécanique
175     inline void Inactive_ddl_primaire()
176     {ElemMeca::Inact_ddl_primaire(unefois->doCoMemb->tab_ddl);};
177     // active les ddl du problème primaire de mécanique
178     inline void Active_ddl_primaire()
179     {ElemMeca::Act_ddl_primaire(unefois->doCoMemb->tab_ddl);};
180     // ajout des ddl de contraintes pour les noeuds de l'élément
181     inline void Plus_ddl_Sigma()
182     {ElemMeca::Ad_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
183     // inactive les ddl du problème de recherche d'erreur : les contraintes
184     inline void Inactive_ddl_Sigma()
185     {ElemMeca::Inact_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
186     // active les ddl du problème de recherche d'erreur : les contraintes
187     inline void Active_ddl_Sigma()
188     {ElemMeca::Act_ddl_Sigma(unefois->doCoMemb->tab_ddlErr);};
189     // active le premier ddl du problème de recherche d'erreur : SIGMA11
190     inline void Active_premier_ddl_Sigma()
191     {ElemMeca::Act_premier_ddl_Sigma();};
192
193     // lecture de données diverses sur le flot d'entrée
194     void LectureContraintes(UtilLecture * entreePrinc)
195     { if (unefois->CalResPrem_t == 1)
196       ElemMeca::LectureDesContraintes (false, entreePrinc, lesPtMecaInt.TabSigHH_t());
197     else
198       { ElemMeca::LectureDesContraintes (true, entreePrinc, lesPtMecaInt.TabSigHH_t());
199         unefois->CalResPrem_t = 1;
200       }
201     };
202
203     // retour des contraintes en absolu retour true si elle existe sinon false
204     bool ContraintesAbsolues(Tableau <Vecteur>& tabSig)
205     { if (unefois->CalResPrem_t == 1)
206       ElemMeca::ContraintesEnAbsolues (false, lesPtMecaInt.TabSigHH_t(), tabSig);
207     else
208       { ElemMeca::ContraintesEnAbsolues (true, lesPtMecaInt.TabSigHH_t(), tabSig);
209         unefois->CalResPrem_t = 1;
210       }
211     return true;
212     };
213
214
215     // Libere la place occupee par le residu et eventuellement la raideur
216     // par l'appel de Libere de la classe mere et libere les differents tenseurs
217     // intermediaires cree pour le calcul et les grandeurs pointee
218     // de la raideur et du residu
219     void Libere ();
220
221     // acquisition d'une loi de comportement
222     void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
223
224     // test si l'element est complet
225     // = 1 tout est ok, =0 element incomplet
226     int TestComplet();
227
228     // procedure permettant de completer l'element apres
229     // sa creation avec les donnees du bloc transmis
230     // peut etre appeler plusieurs fois
231     // ici il s'agit de l'epaisseurs
232     Element* Complete(BlocGen & bloc, LesFonctions_nD* lesFonctionsnD);
233     // Compléter pour la mise en place de la gestion de l'hourglass

```

```

234     Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc) ;
235
236     // ramene l'epaisseur au point d'intégration ni
237 // dans le cas où l'élément n'est pas totalement construit -> retour 0.
238     inline double H(int ni, Enum_dure enu = TEMPS_tdt )
239     {if (donnee_specif.epais.Taille())
240         {switch (enu)
241             { case TEMPS_0: return donnee_specif.epais(ni).epaisseur0; break;
242               case TEMPS_t: return donnee_specif.epais(ni).epaisseur_t; break;
243               case TEMPS_tdt: return donnee_specif.epais(ni).epaisseur_tdt; break;
244             };
245         }
246     else
247         return 0.; // cas épaisseur pas définit
248     };
249
250 // ramene l'epaisseur moyenne de tous les pti
251 // dans le cas où l'élément n'est pas totalement construit -> retour 0.
252     inline double H_moy(Enum_dure enu)
253     {double retour=0.;
254     int nb_epais_actuel = donnee_specif.epais.Taille();
255     for (int i=1; i<= nb_epais_actuel;i++)
256         {switch (enu)
257             { case TEMPS_0: retour += donnee_specif.epais(i).epaisseur0; break;
258               case TEMPS_t: retour += donnee_specif.epais(i).epaisseur_t; break;
259               case TEMPS_tdt: retour += donnee_specif.epais(i).epaisseur_tdt; break;
260               default: cout << "\n erreur dans le calcul de l'epaisseur moyenne "
261                       << "\n H_moy(...)";
262                       Sortie(1);
263             };
264         };
265     return retour/nombre->nbi; // retour de la moyenne
266     };
267
268 // ramene vrai si la surface numéro ns existe pour l'élément
269 // dans le cas des éléments triangulaires il ne peut y avoir qu'une
270 // seule surface
271     bool SurfExiste(int ns) const
272     { if ((ns==1)&&(ParaGlob::Dimension() >= 2)) return true; else return false;};
273
274 // ramene vrai si l'arête numéro na existe pour l'élément
275     bool AreteExiste(int na) const
276     {if ((na <= 3) || (na>= 1)) return true; else return false;};
277
278 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
279 // ce tableau et spécifique a l'element
280     const DdlElement & TableauDdl() const
281     { return unefois->doCoMemb->tab_ddl; };
282
283 // Calcul du residu local et de la raideur locale,
284 // pour le schema implicite
285     Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
286
287 // Calcul du residu local a t
288 // pour le schema explicit par exemple
289     Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
290     { return TriaMemb::CalculResidu(false,pa);};
291
292 // Calcul du residu local a tdt
293 // pour le schema explicit par exemple
294     Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
295     { return TriaMemb::CalculResidu(true,pa);};
296
297 // Calcul de la matrice masse pour l'élément
298     Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
299
300 // ----- calcul dynamique -----
301 // calcul de la longueur d'arrête de l'élément minimal
302 // divisé par la célérité la plus rapide dans le matériau
303     double Long_arrete_mini_sur_c(Enum_dure temps)
304     { return ElemMeca::Interne_Long_arrete_mini_sur_c(temps);};
305
306 //----- calcul d'erreur, remontée des contraintes -----
307 // 1) calcul du résidu et de la matrice de raideur pour le calcul d'erreur
308     Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
309 // 2) remontée aux erreurs aux noeuds
310     Element::Er_ResRaid ErreurAuNoeud_ResRaid();
311
312 // ----- affichage ou récupération d'informations -----
313 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
314 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
315 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
316 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
317 // temps: dit si c'est à 0 ou t ou tdt
318     int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
319     { return PtLePlusPres(temps,enu,M);};
320

```

```

321 // recuperation des coordonnées du point de numéro d'ordre iteg pour
322 // la grandeur enu
323 // temps: dit si c'est à 0 ou t ou tdt
324 // si erreur retourne erreur à true
325 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
326 { return CoordPtInt(temps,enu,iteg,erreur);};
327
328 // récupération des valeurs au numéro d'ordre = iteg pour
329 // les grandeur enu
330 Tableau <double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
331 enu,int iteg);
332 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
333 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
334 // de conteneurs quelconque associée
335 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int iteg);
336
337 //===== lecture écriture dans base info =====
338 // cas donne le niveau de la récupération
339 // = 1 : on récupère tout
340 // = 2 : on récupère uniquement les données variables (supposées comme telles)
341 void Lecture_base_info
342 (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
343 // cas donne le niveau de sauvegarde
344 // = 1 : on sauvegarde tout
345 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
346 void Ecriture_base_info(ofstream& sort,const int cas) ;
347
348 // 2) derivant des virtuelles
349
350 // retourne un tableau de ddl element, correspondant à la
351 // composante de sigma -> SIG11, pour chaque noeud qui contient
352 // des ddl de contrainte
353 // -> utilisé pour l'assemblage de la raideur d'erreur
354 DdlElement& Tableau_de_Sig1() const
355 {return uneFois->doCoMemb->tab_Err1Sig11;};
356
357 // actualisation des ddl et des grandeurs actives de t+dt vers t
358 void TdtversT();
359 // actualisation des ddl et des grandeurs actives de t vers tdt
360 void TversTdt();
361
362 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
363 // qu'une fois la remontée aux contraintes effectuées sinon aucune
364 // action. En retour la valeur de l'erreur sur l'élément
365 // type indique le type de calcul d'erreur :
366 void ErreurElement(int type,double& errElemRelative
367 ,double& numerateur, double& denominateur);
368
369 // mise à jour de la boîte d'encombrement de l'élément, suivant les axes I_a globales
370 // en retour coordonnées du point mini dans retour.Premier() et du point maxi dans .Second()
371 // la méthode est différente de la méthode générale car il faut prendre en compte l'épaisseur de
372 // l'élément
373 virtual const DeuxCoordonnees& Boite_encombre_element(Enum_dure temps);
374
375 // calcul des seconds membres suivant les chargements
376 // cas d'un chargement volumique,
377 // force indique la force volumique appliquée
378 // retourne le second membre résultant
379 // ici on l'épaisseur de l'élément pour constituer le volume
380 // -> explicite à t
381 Vecteur SM_charge_volumique_E_t
382 (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
383 { return TriaMemb::SM_charge_volumique_E(force,pt_fonct,false,pa,sur_volume_finale_);};
384 // -> explicite à tdt
385 Vecteur SM_charge_volumique_E_tdt
386 (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
387 { return TriaMemb::SM_charge_volumique_E(force,pt_fonct,true,pa,sur_volume_finale_);};
388 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
389 // retourne le second membre et la matrice de raideur correspondant
390 ResRaid SMR_charge_volumique_I
391 (const Coordonnee& force,Fonction_nd* pt_fonct,const ParaAlgoControle & pa,bool sur_volume_finale_)
392 ;
393
394 // cas d'un chargement surfacique, sur les frontières des éléments
395 // force indique la force surfacique appliquée
396 // numface indique le numéro de la face chargée
397 // retourne le second membre résultant
398 // -> version explicite à t
399 Vecteur SM_charge_surfacique_E_t(const Coordonnee& force,Fonction_nd* pt_fonct,int numFace,const
400 ParaAlgoControle & pa)
401 { return TriaMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,false,pa);};
402 // -> version explicite à tdt
403 Vecteur SM_charge_surfacique_E_tdt(const Coordonnee& force,Fonction_nd* pt_fonct,int numFace,const
404 ParaAlgoControle & pa)
405 { return TriaMemb::SM_charge_surfacique_E(force,pt_fonct,numFace,true,pa);};

```

```

403     // -> implicite,
404     // pa : permet de déterminer si oui ou non on calcul la contribution à la raideur
405     // retourne le second membre et la matrice de raideur correspondant
406     ResRaid SMR_charge_surfacique_I
407         (const Coordonnee& force,Fonction_nD* pt_fonct,int numFace,const ParaAlgoControle & pa) ;
408
409     // cas d'un chargement lineique, sur les aretes frontieres des éléments
410     // force indique la force lineique appliquée
411     // numarete indique le numéro de l'arete chargée
412     // retourne le second membre résultant
413     // -> explicite à t
414     Vecteur SM_charge_lineique_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
415     { return TriaMemb::SM_charge_lineique_E(force,pt_fonct,numArete,false,pa); } ;
416     // -> explicite à tdt
417     Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
418     { return TriaMemb::SM_charge_lineique_E(force,pt_fonct,numArete,true,pa); } ;
419     // -> implicite,
420     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
421     // retourne le second membre et la matrice de raideur correspondant
422     ResRaid SMR_charge_lineique_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
423
424     // cas d'un chargement lineique suiveuse, sur les aretes frontieres des éléments 2D (uniquement)
425     // force indique la force lineique appliquée
426     // numarete indique le numéro de l'arete chargée
427     // retourne le second membre résultant
428     // -> explicite à t
429     Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
430     { return TriaMemb::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,false,pa); } ;
431     // -> explicite à tdt
432     Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa)
433     { return TriaMemb::SM_charge_lineique_Suiv_E(force,pt_fonct,numArete,true,pa); } ;
434     // -> implicite,
435     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
436     // retourne le second membre et la matrice de raideur correspondant
437     ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,Fonction_nD* pt_fonct,int numArete,const
ParaAlgoControle & pa) ;
438
439     // cas d'un chargement de type pression, sur les frontieres des éléments
440     // pression indique la pression appliquée
441     // numface indique le numéro de la face chargée
442     // retourne le second membre résultant
443     // -> explicite à t
444     Vecteur SM_charge_pression_E_t(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
445     { return TriaMemb::SM_charge_pression_E(pression,pt_fonct,numFace,false,pa); } ;
446     // -> explicite à tdt
447     Vecteur SM_charge_pression_E_tdt(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa)
448     { return TriaMemb::SM_charge_pression_E(pression,pt_fonct,numFace,true,pa); } ;
449     // -> implicite,
450     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
451     // retourne le second membre et la matrice de raideur correspondant
452     ResRaid SMR_charge_pression_I(double pression,Fonction_nD* pt_fonct,int numFace,const
ParaAlgoControle & pa) ;
453
454     // cas d'un chargement de type pression unidirectionnelle, sur les frontieres des éléments
455     // presUniDir indique le vecteur appliquée
456     // numface indique le numéro de la face chargée
457     // retourne le second membre résultant
458     // -> explicite à t
459     Vecteur SM_charge_presUniDir_E_t(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
460     { return TriaMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,false,pa); } ;
461     // -> explicite à tdt
462     Vecteur SM_charge_presUniDir_E_tdt(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa)
463     { return TriaMemb::SM_charge_presUniDir_E(presUniDir,pt_fonct,numFace,true,pa); } ;
464     // -> implicite,
465     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
466     // retourne le second membre et la matrice de raideur correspondant
467     ResRaid SMR_charge_presUniDir_I(const Coordonnee& presUniDir,Fonction_nD* pt_fonct,int
numFace,const ParaAlgoControle & pa);
468
469     // cas d'un chargement surfacique hydrostatique,
470     // poidvol: indique le poids volumique du liquide
471     // M_liquide : un point de la surface libre
472     // dir_normal_liquide : direction normale à la surface libre
473     // retourne le second membre résultant
474     // -> explicite à t
475     Vecteur SM_charge_hydrostatique_E_t(const Coordonnee& dir_normal_liquide,const double& poidvol
,int numFace,const Coordonnee& M_liquide
476     ,const ParaAlgoControle & pa
477     ,const ParaAlgoControle & pa) ;

```

```

478                                     ,bool sans_limitation)
479     { return
TriaMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,false,pa,sans_limitation));};
480 // -> explicite à tdt
481 Vecteur SM_charge_hydrostatique_E_tdt(const Coordonnee& dir_normal_liquide,const double& poidvol
482                                     ,int numFace,const Coordonnee& M_liquide
483                                     ,const ParaAlgoControle & pa
484                                     ,bool sans_limitation)
485     { return
TriaMemb::SM_charge_hydrostatique_E(dir_normal_liquide,poidvol,numFace,M_liquide,true,pa,sans_limitation));};
486 // -> implicite,
487 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
488 // retourne le second membre et la matrice de raideur correspondant
489 ResRaid SMR_charge_hydrostatique_I(const Coordonnee& dir_normal_liquide,const double& poidvol
490                                   ,int numFace,const Coordonnee& M_liquide
491                                   ,const ParaAlgoControle & pa
492                                   ,bool sans_limitation) ;
493
494 // cas d'un chargement surfacique hydro-dynamique,
495 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
496 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc unitaire)
497 // une suivant la direction normale à la vitesse de type portance
498 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
499 // une suivant la vitesse tangente de type frottement visqueux
500 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
501 // coef_mul: est un coefficient multiplicateur global (de tout)
502 // retourne le second membre résultant
503 // -> explicite à t
504 Vecteur SM_charge_hydrodynamique_E_t( CourbelD* frot_fluid,const double& poidvol
505                                     , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
506                                     , CourbelD* coef_aero_t
507                                     ,const ParaAlgoControle & pa)
508     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa));};
509 // -> explicite à tdt
510 Vecteur SM_charge_hydrodynamique_E_tdt( CourbelD* frot_fluid,const double& poidvol
511                                       , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
512                                       , CourbelD* coef_aero_t
513                                       ,const ParaAlgoControle & pa)
514     {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa));};
515 // -> implicite,
516 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
517 // retourne le second membre et la matrice de raideur correspondant
518 ResRaid SMR_charge_hydrodynamique_I( CourbelD* frot_fluid,const double& poidvol
519                                   , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
520                                   , CourbelD* coef_aero_t
521                                   ,const ParaAlgoControle & pa) ;
522
523 // ===== définition et/ou construction des frontières =====
524
525 // Calcul des frontieres de l'element
526 // creation des elements frontieres et retour du tableau de ces elements
527 // la création n'a lieu qu'au premier appel
528 // ou lorsque l'on force le paramètre force a true
529 // dans ce dernier cas seul les frontière effacées sont recréée
530 Tableau <ElFrontiere*> const & Frontiere(bool force = false);
531
532 // ramène la frontière point
533 // éventuellement création des frontieres points de l'element et stockage dans l'element
534 // si c'est la première fois sinon il y a seulement retour de l'elements
535 // a moins que le paramètre force est mis a true
536 // dans ce dernier cas la frontière effacée est recréée
537 // num indique le numéro du point à créer (numérotation EF)
538 // ElFrontiere* const Frontiere_points(int num,bool force = false);
539
540 // ramène la frontière linéique
541 // éventuellement création des frontieres linéique de l'element et stockage dans l'element
542 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
543 // a moins que le paramètre force est mis a true
544 // dans ce dernier cas la frontière effacée est recréée
545 // num indique le numéro de l'arête à créer (numérotation EF)
546 // ElFrontiere* const Frontiere_lineique(int num,bool force = false);
547
548 // ramène la frontière surfacique
549 // éventuellement création des frontieres surfacique de l'element et stockage dans l'element
550 // si c'est la première fois sinon il y a seulement retour de l'elements
551 // a moins que le paramètre force est mis a true
552 // dans ce dernier cas la frontière effacée est recréée
553 // num indique le numéro de la surface à créer (numérotation EF)
554 // ici normalement uniquement 1 possible
555 // ElFrontiere* const Frontiere_surfacique(int num,bool force = false);
556
557

```



```

558 // 3) methodes propres a l'element
559
560 // ajout du tableau specific de ddl des noeuds
561 // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
562 // des noeuds constituant l'element
563 void ConstTabDdl();
564
565 // ----- definition de la classe conteneur de donnees communes -----
566 class DonnComTria
567 { public :
568     DonnComTria (GeomTriangle& tri,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1Sig,
569                 Met_abstraite& met_gene, Tableau <Vecteur * > & resEr,Mat_pleine& raidEr,
570                 GeomTriangle& triEr,GeomTriangle& triS,
571                 GeomSeg& segmS,Vecteur& residu_int,Mat_pleine& raideur_int,
572                 Tableau <Vecteur* > & residus_extN,Tableau <Mat_pleine* >& raideurs_extN,
573                 Tableau <Vecteur* > & residus_extA,Tableau <Mat_pleine* >& raideurs_extA,
574                 Tableau <Vecteur* >& residus_extS,Tableau <Mat_pleine* >& raideurs_extS,
575                 Mat_pleine& mat_masse,GeomTriangle& triMas,int nbi,GeomTriangle* triHourg,
576                 Mat_pleine* trimatD,Vecteur* triresD
577     ) ;
578     DonnComTria(DonnComTria& a) ;
579     ~DonnComTria();
580     // variables
581     GeomTriangle tria; // contient les fonctions d'interpolation et
582                       // les derivees pour le calcul d'equilibre en deplacement
583     DdlElement tab_ddl; // tableau des degres
584                       //de liberte des noeuds de l'element commun a tous les
585                       // elements. Il s'agit des ddl primaires pour le probleme de mecanique
586     Met_abstraite met_triaMemb;
587     Mat_pleine matGeom ; // matrice geometrique
588     Mat_pleine matInit ; // matrice initiale
589     Tableau <TenseurBB * > d_epsBB; // place pour la variation des def
590     Tableau <TenseurHH * > d_sigHH; // place pour la variation des contraintes
591     Tableau < Tableau2 <TenseurBB * > > d2_epsBB; // variation seconde des deformations
592     // ---- concernant les frontieres et particulierement le calcul de second membre
593     GeomTriangle triaS; // contient les fonctions d'interpolation et les derivees
594     GeomSeg segS; // " " "
595
596     //----- calcul d'erreur -----
597     DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
598     // d'erreur : contraintes
599     DdlElement tab_Err1Sig11; // tableau du ddl SIG11 pour chaque noeud, servant pour le
calcul
600     // d'erreur : contraintes, en fait pour l'assemblage
601     Tableau <Vecteur* > resErr; // residu pour le calcul d'erreur
602     Mat_pleine raidErr; // raideur pour le calcul d'erreur
603     GeomTriangle triaEr; // contient les fonctions d'interpolation et
604                       // les derivees pour le calcul du hessien dans
605                       //la resolution de la fonctionnelle d'erreur
606     // ----- calcul de residus, de raideur : interne ou pour les efforts exterieurs
-----
607     // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
608     Vecteur residu_interne;
609     Mat_pleine raideur_interne;
610     Tableau <Vecteur* > residus_externeN; // pour les noeuds
611     Tableau <Mat_pleine* > raideurs_externeN; // pour les noeuds
612     Tableau <Vecteur* > residus_externeA; // pour les aretes
613     Tableau <Mat_pleine* > raideurs_externeA; // pour les aretes
614     Tableau <Vecteur* > residus_externeS; // pour les surfaces
615     Tableau <Mat_pleine* > raideurs_externeS; // pour les surfaces
616     // ----- données concernant la dynamique -----
617     Mat_pleine matrice_masse;
618     GeomTriangle triaMas; // contient les fonctions d'interpolation et ...
619                       // pour les calculs relatifs à la masse
620     // ----- blocage éventuel d'hourglass
621     // utiliser dans ElemMeca::Cal_mat_hourglass_comp, Cal_implicit_hourglass,
Cal_explici_hourglass
622     GeomTriangle* triaHourg; // contient les fonctions d'interpolation
623
624     // --- blocage éventuel de stabilisation normale à la membrane
625     // utilisé dans: ElemMeca::Cal_implicit_StabMembBiel, ElemMeca::Cal_explicit_StabMembBiel
626     Mat_pleine* triamatD; // raideur éventuelle,
627     Vecteur* triaresD; // résidu éventuelle,
628 };
629
630 // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
631 // et un pointeur sur les données statiques communes
632 // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
633 // classe est défini. Son allocation est effectuée dans les classes dérivées
634 class UneFois
635 { public :
636     UneFois () ; // constructeur par défaut
637     ~UneFois () ; // destructeur
638
639     // VARIABLES :
640     public :
641     TriaMemb::DonnComTria * doCoMemb;

```

```

642
643 // incicateurs permettant de dimensionner seulement au premier passage
644 // utilise dans "CalculResidu" et "Calcul_implicit"
645 int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
646 int CalimpPrem;
647 int dualSortTria; // pour la sortie des valeurs au pt d'integ
648 int CalSMLin_t; // pour les seconds membres concernant les arretes
649 int CalSMLin_tdt; // pour les seconds membres concernant les arretes
650 int CalSMRlin; // pour les seconds membres concernant les arretes
651 int CalSMsurf_t; // pour les seconds membres concernant les surfaces
652 int CalSMsurf_tdt; // pour les seconds membres concernant les surfaces
653 int CalSMRsurf; // pour les seconds membres concernant les surfaces
654 int CalSMvol_t; // pour les seconds membres concernant les volumes
655 int CalSMvol_tdt; // pour les seconds membres concernant les volumes
656 int CalSMvol; // pour les seconds membres concernant les volumes
657 int CalDynamique; // pour le calcul de la matrice de masse
658 int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
659 // ----- sauvegarde du nombre d'élément en cours -----
660 int nbelem_in_Prog;
661 };
662
663 // -----
664
665 protected :
666 // VARIABLES PROTÉGÉES :
667 UnePois * unefois; // pointeur défini dans la classe dérivée
668
669 // type structuré et fonction virtuelle pour construire les éléments
670 // la fonction est défini dans le type dérivé
671 class NombresConstruire
672 { public:
673 NombresConstruire():nbne(0),nbneA(0),nbi(0)
674 ,nbiEr(0),nbiS(0),nbiA(0),nbiMas(0),nbiHour(0) {};
675 int nbne; // le nombre de noeud de l'élément
676 int nbneA ; // le nombre de noeud des aretes
677 int nbi; // le nombre de point d'intégration pour le calcul mécanique
678 int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
679 int nbiS; // le nombre de point d'intégration pour le calcul de second membre surfacique
680 int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
681 int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse
682 int nbiHour; // éventuellement, le nombre de point d'intégration un blocage d'hourglass
683 };
684 NombresConstruire * nombre; // le pointeur défini dans la classe dérivée d'hexamemb
685
686 // les données spécifiques sont groupées dans une structure pour sécuriser
687 // le passage de paramètre dans init par exemple
688 class Donnee_specif
689 { public :
690 Donnee_specif() : // défaut
691 epais(),use_epais_moyenne(1)
692 ,cas_pti_nbi(0),cas_pti_nbiEr(0)
693 ,cas_pti_nbiS(0),cas_pti_nbiMas(0)
694 {};
695 Donnee_specif(double epai,int nbi) : // uniquement l'épaisseur et le nombre de pti
696 epais(nbi,Epai(epai,epai,epai)),use_epais_moyenne(1)
697 ,cas_pti_nbi(0),cas_pti_nbiEr(0)
698 ,cas_pti_nbiS(0),cas_pti_nbiMas(0)
699 {};
700 Donnee_specif(int casptnbi,int casptnbiEr,int casptnbiS,int casptnbiMas,int nbi) : // les
701 indicateurs
702 epais(nbi,Epai(Element::epaisseur_defaut,Element::epaisseur_defaut,Element::epaisseur_defaut))
703 ,use_epais_moyenne(1),cas_pti_nbi(casptnbi),cas_pti_nbiEr(casptnbiEr)
704 ,cas_pti_nbiS(casptnbiS),cas_pti_nbiMas(casptnbiMas)
705 {};
706 Donnee_specif(double epai0,double epai_t,double epai_tdt
707 ,int casptnbi,int casptnbiEr,int casptnbiS,int casptnbiMas,int nbi) : //
708 tous
709 epais(nbi,Epai(epai0,epai_t,epai_tdt)),use_epais_moyenne(1)
710 ,cas_pti_nbi(casptnbi),cas_pti_nbiEr(casptnbiEr),cas_pti_nbiS(casptnbiS),cas_pti_nbiMas(casptnbiMas)
711 {};
712 Donnee_specif(const Donnee_specif& a) :
713 epais(a.epais ),use_epais_moyenne(a.use_epais_moyenne)
714 ,cas_pti_nbi(a.cas_pti_nbi),cas_pti_nbiEr(a.cas_pti_nbiEr)
715 ,cas_pti_nbiS(a.cas_pti_nbiS),cas_pti_nbiMas(a.cas_pti_nbiMas)
716 {}; // recopie via le constructeur de copie
717 ~Donnee_specif() {};
718 Donnee_specif & operator = ( const Donnee_specif& a)
719 { epais = a.epais;use_epais_moyenne = a.use_epais_moyenne;
720 cas_pti_nbi=a.cas_pti_nbi;cas_pti_nbiEr=a.cas_pti_nbiEr;
721 cas_pti_nbiS=a.cas_pti_nbiS;cas_pti_nbiMas=a.cas_pti_nbiMas;
722 return *this;};
723 // data
724 // epaisseurs de l'element
725 Tableau < Epai > epais; // épaisseur à chaque point d'intégration
726
727 int cas_pti_nbi; // permet de différencier les différents cas de pt d'integ identique

```

```

726                                     // =0: valeur par défaut, ensuite si diff de 1 donne les différents
cas
727                                     // est par exemple utilisé pour différencier le cas de 3 pt sur les
arrêtes
728                                     // ou 3 pt en interne, dans ce cas vaut 1, il peut ainsi y avoir plus
de 2 cas
729     int cas_pti_nbiEr; // idem pour l'erreur
730     int cas_pti_nbiS; // idem pour le calcul de second membre surfacique
731     int cas_pti_nbiMas; // idem pour le calcul de la matrice masse
732     int use_epais_moyenne; // = 1 : (val par défaut) c'est l'épaisseur moyenne qui est mise à jour
733                                     // = 0 : c'est l'épaisseur à chaque pti qui est mise à jour
734 };
735     Donnee_specif donnee_specif;
736
737     // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
738     LesPtIntegMecaInterne lesPtMecaInt;
739
740 // METHODES
741 // =====» methodes appelees par les classes dérivées «=====
742
743 // fonction d'initialisation servant dans les classes derivant
744 // au niveau du constructeur, si rien initialisation par défaut
745 TriaMemb::DonnComTria* Init (bool sans_init_noeud = false);
746 TriaMemb::DonnComTria* Init (Donnee_specif donnee_specif, bool sans_init_noeud = false);
747 // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
748 void Destruction();
749
750 // ===== » methodes virtuelles dérivant d'ElemMeca =====
751 // ramene la dimension des tenseurs contraintes et déformations de l'élément
752 int Dim_sig_eps() const {return 2;};
753
754 //----- fonctions uniquement a usage interne -----
755 private :
756 // definition des données communes : CoTria
757 TriaMemb::DonnComTria* Def_DonneeCommune();
758 // Calcul du residu local a t ou tdt en fonction du booleen
759 Vecteur* CalculResidu (bool atdt, const ParaAlgoControle & pa);
760 // calcul des seconds membres suivant les chargements
761 // cas d'un chargement volumique,
762 // force indique la force volumique appliquée
763 // retourne le second membre résultant
764 // ici on l'épaisseur de l'élément pour constituer le volume
765 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
766 Vecteur SM_charge_volumique_E
767     (const Coordonnee& force, Fonction_nd* pt_fonct, bool atdt, const ParaAlgoControle &
pa, bool sur_volume_finale_);
768 // cas d'un chargement surfacique, sur les frontières des éléments
769 // force indique la force surfacique appliquée
770 // numface indique le numéro de la face chargée
771 // retourne le second membre résultant
772 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
773 Vecteur SM_charge_surfacique_E
774     (const Coordonnee& force, Fonction_nd* pt_fonct, int numFace, bool atdt, const
ParaAlgoControle & pa);
775 // cas d'un chargement lineique, sur les aretes frontières des éléments
776 // force indique la force lineique appliquée
777 // numarete indique le numéro de l'arete chargée
778 // retourne le second membre résultant
779 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
780 Vecteur SM_charge_lineique_E
781     (const Coordonnee& force, Fonction_nd* pt_fonct, int numArete, bool atdt, const
ParaAlgoControle & pa);
782 // cas d'un chargement lineique suiveuse, sur les aretes frontières des éléments 2D (uniquement)
783 // force indique la force lineique appliquée
784 // numarete indique le numéro de l'arete chargée
785 // retourne le second membre résultant
786 // -> explicite à t
787 Vecteur SM_charge_lineique_Suiv_E
788     (const Coordonnee& force, Fonction_nd* pt_fonct, int numArete, bool atdt, const
ParaAlgoControle & pa);
789 // cas d'un chargement de type pression, sur les frontières des éléments
790 // pression indique la pression appliquée
791 // numface indique le numéro de la face chargée
792 // retourne le second membre résultant
793 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
794 Vecteur SM_charge_pression_E
795     (double pression, Fonction_nd* pt_fonct, int numFace, bool atdt, const ParaAlgoControle &
pa);
796 // cas d'un chargement de type pression unidirectionnelle, sur les frontières des éléments
797 // presUniDir indique le vecteur appliquée
798 // numface indique le numéro de la face chargée
799 // retourne le second membre résultant
800 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
801 Vecteur SM_charge_presUniDir_E
802     (const Coordonnee& presUniDir, Fonction_nd* pt_fonct, int numFace, bool atdt, const
ParaAlgoControle & pa);
803 // cas d'un chargement surfacique hydrostatique,

```

```

804 // poidvol: indique le poids volumique du liquide
805 // M_liquide : un point de la surface libre
806 // dir_normal_liquide : direction normale à la surface libre
807 // retourne le second membre résultant
808 // -> explicite à t
809 Vecteur SM_charge_hydrostatique_E(const Coordonnee& dir_normal_liquide,const double& poidvol
810                                   ,int numFace,const Coordonnee& M_liquide,bool atdt
811                                   ,const ParaAlgoControle & pa
812                                   ,bool sans_limitation);
813 // cas d'un chargement surfacique hydro-dynamique,
814 // voir méthode explicite plus haut, pour les arguments
815 // retourne le second membre résultant
816 // bool atdt : permet de spécifier à t ou à t+dt
817 Vecteur SM_charge_hydrodynamique_E( CourbelD* frot_fluid,const double& poidvol
818                                   , CourbelD* coef_aero_n,int numFace,const double&
coef_mul
819                                   , CourbelD* coef_aero_t,bool atdt
820                                   ,const ParaAlgoControle & pa) ;
821 // calcul de la nouvelle épaisseur moyenne finale (sans raideur)
822 // ramène l'épaisseur moyenne calculée à atdt ou t
823 // met à jour les épaisseurs aux pti en fonction de l'épaisseur moyenne calculée
824 const double CalEpaisseurMoyenne_et_transfert_pti(const bool atdt);
825 };
826 /// @} // end of group
827 #endif
828
829
830
831

```

## 7.236 TriaMembL1.h

```

1 // FICHER : TriaMembL1.h
2 // CLASSE : TriaMembL1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *
34 *   DATE:      15/01/97
35 *
36 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *
41 * *****
42 * La classe TriaMemb permet de declarer des elements triangulaire membrane et de realiser
43 * le calcul du residu local et de la raideur locale pour une loi de comportement
44 * donnee. La dimension de l'espace pour un tel element est 2 ou 3
45 * l'interpolation est lineaire, le nombre de point d'integration est de 1
46 *
47 *
48 * VERIFICATION:
49 *
50 * ! date ! auteur ! but
51 *
52 * ! ! !

```

```

53 *
54 *
55 *
56 *   MODIFICATIONS:
57 *
58 *   ! date !   auteur   !           but           !
59 *   -----
60 *
61 *
62 *
63 *****/
64
65
66 // -----classe pour un calcul de mecanique-----
67
68 // La classe TriaMemb permet de declarer des elements triangulaire membrane et de realiser
69 // le calcul du residu local et de la raideur locale pour une loi de comportement
70 // donnee. La dimension de l'espace pour un tel element est 2 ou 3
71 //
72 // l'interpolation est lineaire, le nombre de point d'integration est de 1
73
74
75 #ifndef TRIAMEMBL1_H
76 #define TRIAMEMBL1_H
77
78 #include "ParaGlob.h"
79 #include "ElemMeca.h"
80 #include "Met_abstraite.h"
81 #include "GeomTriangle.h"
82 #include "Noeud.h"
83 #include "UtilLecture.h"
84 #include "Tenseur.h"
85 #include "NevezTenseur.h"
86 #include "Deformation.h"
87 #include "TriaMemb.h"
88 #include "ElFrontiere.h"
89 #include "FrontSegLine.h"
90 #include "FrontTriaLine.h"
91
92 /// @addtogroup groupe_des_elements_finis
93 /// @{
94 ///
95
96
97 class TriaMembL1 : public TriaMemb
98 {
99
100     public :
101
102         // CONSTRUCTEURS :
103         // Constructeur par default
104         TriaMembL1 ();
105
106         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
107         // d'identification
108         TriaMembL1 (double epaiss,int num_maill=0,int num_id=-3);
109
110         // Constructeur fonction d'un numero de maillage et d'identification
111         TriaMembL1 (int num_mail,int num_id);
112
113         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
114         // du tableau de connexite des noeuds
115         TriaMembL1 (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
116
117         // Constructeur de copie
118         TriaMembL1 (const TriaMembL1& tria);
119
120
121         // DESTRUCTEUR :
122         ~TriaMembL1 ();
123
124         // creation d'un element de copie: utilisation de l'operateur new et du constructeur de copie
125         // methode virtuelle
126         Element* Nevez_copie() const { Element * el= new TriaMembL1(*this); return el;};
127
128         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaMembL1
129         TriaMembL1& operator= (TriaMembL1& tria);
130
131         // METHODES :
132         // 1) derivant des virtuelles pures
133
134         // affichage dans la sortie transmise, des variables duales "nom"
135         // aux differents points d'integration
136         // dans le cas ou nom est vide, affichage de "toute" les variables
137         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
138
139         // 2) derivant des virtuelles

```

```

140 // 3) methodes propres a l'element
141
142     protected :
143
144         // adressage des frontieres lineiques et surfacique
145         // definit dans les classes derivees, et utilisees pour la construction des frontieres
146         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
147         { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
148         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
149         { return ((ElFrontiere*) (new FrontTriaLine(tab,ddelem)));};
150
151     // VARIABLES PRIVEES :
152     // place memoire commune a tous les elements TriaMembl1
153     static TriaMemb::DonnComTria * doCoMembl1;
154     // idem mais pour les indicateurs qui servent pour l'initialisation
155     static TriaMemb::UneFois uneFoisL1;
156
157     class NombresConstruireTriaMembl1 : public NombresConstruire
158     { public: NombresConstruireTriaMembl1();
159     };
160     static NombresConstruireTriaMembl1 nombre_V; // les nombres propres a l'element
161
162     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
163     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
164     class ConsTriaMembl1 : public ConstrucElement
165     { public : ConsTriaMembl1 ()
166     { NouvelleTypeElement nouv (TRIANGLE,LINEAIRE,MECA_SOLIDE_DEFORMABLE,this);
167     if (ParaGlob::NiveauImpression() >= 4)
168     cout << "\n initialisation TriaMembl1" << endl;
169     Element::listTypeElement.push_back(nouv);
170     };
171     Element * NouvelElement(int num_maill,int num) // un nouvel element sans rien
172     {Element * pt;
173     pt = new TriaMembl1 (num_maill,num) ;
174     return pt;};
175     // ramene true si la construction de l'element est possible en fonction
176     // des variables globales actuelles: ex en fonction de la dimension
177     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
178     };
179     static ConsTriaMembl1 consTriaMembl1;
180 };
181 /// @} // end of group
182 #endif
183
184
185
186

```

## 7.237 TriaMembQ3.h

```

1 // FICHER : TriaMembQ3.h
2 // CLASSE : TriaMembQ3
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *
34 *     DATE:          15/01/97
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```

```

36 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)           *
37 *                                                     $   *
38 *   PROJET:      Herezh++                                     *
39 *                                                     $   *
40 * *****
41 *   BUT:         Element triangulaire, quadratique 3 pt d'integ. *
42 *                                                     $   *
43 *   *****
44 *   VERIFICATION:                                           *
45 *                                                     *
46 *   ! date !   auteur !           but                       !   *
47 *   -----
48 *   !           !           !                               !   *
49 *                                                     *
50 *                                                     $   *
51 *   *****
52 *   MODIFICATIONS:                                           *
53 *                                                     *
54 *   ! date !   auteur !           but                       !   *
55 *   -----
56 *                                                     *
57 *                                                     $   *
58 *                                                     *
59 * *****/
60
61 // -----classe pour un calcul de mecanique-----
62
63 // La classe TriaMemb permet de declarer des elements triangulaire membrane et de realiser
64 // le calcul du residu local et de la raideur locale pour une loi de comportement
65 // donnee. La dimension de l'espace pour un tel element est 2
66 //
67 // l'interpolation est quadratique, le nombre de point d'integration est de 3
68
69
70 #ifndef TRIAMEMBQ3_H
71 #define TRIAMEMBQ3_H
72
73 #include "ParaGlob.h"
74 #include "ElemMeca.h"
75 #include "Met_abstraite.h"
76 #include "GeomTriangle.h"
77 #include "Noeud.h"
78 #include "UtilLecture.h"
79 #include "Tenseur.h"
80 #include "NevezTenseur.h"
81 #include "Deformation.h"
82 #include "TriaMemb.h"
83 #include "FrontSegQuad.h"
84 #include "FrontTriaQuad.h"
85
86 /// @addtogroup groupe_des_elements_finis
87 /// @{
88 ///
89
90
91 class TriaMembQ3 : public TriaMemb
92 {
93
94     public :
95
96         // CONSTRUCTEURS :
97         // Constructeur par default
98         TriaMembQ3 ();
99
100        // Constructeur fonction d'une epaisseur et eventuellement d'un numero
101        // d'identification
102        TriaMembQ3 (double epaiss,int num_maill=0,int num_id=-3);
103
104        // Constructeur fonction d'un numero de maillage et d'identification
105        TriaMembQ3 (int num_mail,int num_id);
106
107        // Constructeur fonction d'une epaisseur, d'un numero d'identification,
108        // du tableau de connexite des noeuds
109        TriaMembQ3 (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
110
111        // Constructeur de copie
112        TriaMembQ3 (const TriaMembQ3& tria);
113
114
115        // DESTRUCTEUR :
116        ~TriaMembQ3 ();
117
118        // creation d'un element de copie: utilisation de l'operateur new et du constructeur de copie
119        // methode virtuelle
120        Element* Nevez_copie() const { Element * el= new TriaMembQ3(*this); return el;};
121

```

```

122         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaMembQ3
123         TriaMembQ3& operator= (TriaMembQ3& tria);
124
125         // METHODES :
126 // 1) derivant des virtuelles pures
127
128         // affichage dans la sortie transmise, des variables duales "nom"
129         // aux differents points d'integration
130         // dans le cas ou nom est vide, affichage de "toute" les variables
131         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
132
133     protected :
134
135         // adressage des frontieres lineiques et surfacique
136         // definit dans les classes derivees, et utilisees pour la construction des frontieres
137         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
138         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
139         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
140         { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
141
142     // VARIABLES PRIVEES :
143     // place memoire commune a tous les elements TriaMembQ3
144     static TriaMemb::DonnComTria * doCoMembQ3;
145     // idem mais pour les indicateurs qui servent pour l'initialisation
146     static TriaMemb::UneFois uneFoisQ3;
147
148     class NombresConstruireTriaMembQ3 : public NombresConstruire
149     { public: NombresConstruireTriaMembQ3();
150     };
151     static NombresConstruireTriaMembQ3 nombre_V; // les nombres propres à l'élément
152
153     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
154     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
155     class ConsTriaMembQ3 : public ConstrucElement
156     { public : ConsTriaMembQ3 ()
157     { NouvelleTypeElement nouv (TRIANGLE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this);
158     if (ParaGlob::NiveauImpression() >= 4)
159     cout << "\n initialisation TriaMembQ3" << endl;
160     Element::listTypeElement.push_back(nouv);
161     };
162     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
163     {Element * pt;
164     pt = new TriaMembQ3 (num_maill,num) ;
165     return pt;};
166     // ramene true si la construction de l'element est possible en fonction
167     // des variables globales actuelles: ex en fonction de la dimension
168     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
169     };
170     static ConsTriaMembQ3 consTriaMembQ3;
171 };
172 /// @} // end of group
173 #endif
174
175
176
177

```

## 7.238 TriaMembQ3\_cm1pti.h

```

1 // FICHER : TriaMembQ3_cmlpti.h
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License

```



```

27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *
33 *   DATE:      03/05/2011
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 *   BUT:      Element triangulaire, quadratique 1 pt d'integ.
41 * + gestion des modes d'hourglass.
42 *
43 *   *****
44 *   VERIFICATION:
45 *
46 *   ! date !   auteur !       but
47 *   -----
48 *   !       !       !
49 *
50 *   *****
51 *   MODIFICATIONS:
52 *
53 *   ! date !   auteur !       but
54 *   -----
55 *
56 *
57 *
58 *
59 *****/
60
61
62 #ifndef TRIAMEMBQ3_CM1PTI_H
63 #define TRIAMEMBQ3_CM1PTI_H
64
65 #include "ParaGlob.h"
66 #include "ElemMeca.h"
67 #include "Met_abstraite.h"
68 #include "GeomTriangle.h"
69 #include "Noeud.h"
70 #include "UtilLecture.h"
71 #include "Tenseur.h"
72 #include "NevezTenseur.h"
73 #include "Deformation.h"
74 #include "TriaMemb.h"
75 #include "FrontSegQuad.h"
76 #include "FrontTriaQuad.h"
77
78 /// @addtogroup groupe_des_elements_finis
79 /// @{
80 ///
81
82
83 class TriaMembQ3_cm1pti : public TriaMemb
84 {
85
86     public :
87
88         // CONSTRUCTEURS :
89         // Constructeur par default
90         TriaMembQ3_cm1pti ();
91
92         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
93         // d'identification
94         TriaMembQ3_cm1pti (double epaiss,int num_maill=0,int num_id=-3);
95
96         // Constructeur fonction d'un numero de maillage et d'identification
97         TriaMembQ3_cm1pti (int num_mail,int num_id);
98
99         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
100         // du tableau de connexion des noeuds
101         TriaMembQ3_cm1pti (double epaiss,int num_mail,int num_id,const Tableau<Noeud *>& tab);
102
103         // Constructeur de copie
104         TriaMembQ3_cm1pti (const TriaMembQ3_cm1pti& tria);
105
106
107         // DESTRUCTEUR :
108         ~TriaMembQ3_cm1pti ();
109
110         // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
111         // méthode virtuelle
112         Element* Nevez_copie() const { Element * el= new TriaMembQ3_cm1pti(*this); return el;};
113

```

```

114         // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaMembQ3_cmlpti
115         TriaMembQ3_cmlpti& operator= (TriaMembQ3_cmlpti& tria);
116
117         // METHODES :
118 // 1) derivant des virtuelles pures
119
120         // affichage dans la sortie transmise, des variables duales "nom"
121         // aux differents points d'integration
122         // dans le cas ou nom est vide, affichage de "toute" les variables
123         void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
124
125     protected :
126
127         // adressage des frontieres linéiques et surfacique
128         // définit dans les classes dérivées, et utilisées pour la construction des frontieres
129         ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
130         { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
131         ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
132         { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
133
134     // VARIABLES PRIVEES :
135     // place memoire commune a tous les elements TriaMembQ3_cmlpti
136     static TriaMemb::DonnComTria * doCoMembQ3;
137     // idem mais pour les indicateurs qui servent pour l'initialisation
138     static TriaMemb::UneFois uneFoisQ3;
139
140     class NombresConstruireTriaMembQ3_cmlpti : public NombresConstruire
141     { public: NombresConstruireTriaMembQ3_cmlpti();
142     };
143     static NombresConstruireTriaMembQ3_cmlpti nombre_V; // les nombres propres à l'élément
144
145     // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
146     //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
147     class ConsTriaMembQ3_cmlpti : public ConstrucElement
148     { public : ConsTriaMembQ3_cmlpti ()
149     { NouvelleTypeElement nouv(TRIANGLE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cmlpti");
150     if (ParaGlob::NiveauImpression() >= 4)
151     cout << "\n initialisation TriaMembQ3_cmlpti" << endl;
152     Element::listTypeElement.push_back(nouv);
153     };
154     Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
155     {Element * pt;
156     pt = new TriaMembQ3_cmlpti (num_maill,num) ;
157     return pt;};
158     // ramene true si la construction de l'element est possible en fonction
159     // des variables globales actuelles: ex en fonction de la dimension
160     bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
161     };
162     static ConsTriaMembQ3_cmlpti consTriaMembQ3_cmlpti;
163 };
164 /// @} // end of group
165 #endif
166
167
168
169

```

## 7.239 TriaQ3\_cmpti1003.h

```

1 // FICHER : TriaQ3_cmpti1003.h
2 // CLASSE : TriaQ3_cmpti1003
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //

```

```

27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *
34 *   DATE:      15/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *
41 *   BUT:  Element triangulaire, quadratique 3 pt d'integ
42 *         avec la particularité d'être interne à l'élément et non
43 *         sur les arrêtes.
44 *
45 *   *****/
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !       but
50 *   -----
51 *   !       !       !
52 *
53 *   *****/
54 *
55 *   MODIFICATIONS:
56 *
57 *   ! date !   auteur !       but
58 *   -----
59 *
60 *
61 *****/
62
63 // -----classe pour un calcul de mecanique-----
64
65 // La classe TriaMemb permet de declarer des elements triangulaire membrane et de realiser
66 // le calcul du residu local et de la raideur locale pour une loi de comportement
67 // donnee. La dimension de l'espace pour un tel element est 2
68 //
69 // l'interpolation est quadratique, le nombre de point d'integration est de 3
70
71
72 #ifndef TRIAQ3_CMPTI1003H
73 #define TRIAQ3_CMPTI1003H
74
75 #include "ParaGlob.h"
76 #include "ElemMeca.h"
77 #include "Met_abstraite.h"
78 #include "GeomTriangle.h"
79 #include "Noeud.h"
80 #include "UtilLecture.h"
81 #include "Tenseur.h"
82 #include "NevezTenseur.h"
83 #include "Deformation.h"
84 #include "TriaMemb.h"
85 #include "FrontSegQuad.h"
86 #include "FrontTriaQuad.h"
87
88 /// @addtogroup groupe_des_elements_finis
89 /// @{
90 ///
91
92
93 class TriaQ3_cmpti1003 : public TriaMemb
94 {
95
96     public :
97
98         // CONSTRUCTEURS :
99         // Constructeur par defaut
100         TriaQ3_cmpti1003 ();
101
102         // Constructeur fonction d'une epaisseur et eventuellement d'un numero
103         // d'identification
104         TriaQ3_cmpti1003 (double epaiss,int num_maill=0,int num_id=-3);
105
106         // Constructeur fonction d'un numero de maillage et d'identification
107         TriaQ3_cmpti1003 (int num_maill,int num_id);
108
109         // Constructeur fonction d'une epaisseur, d'un numero d'identification,
110         // du tableau de connexite des noeuds
111         TriaQ3_cmpti1003 (double epaiss,int num_maill,int num_id,const Tableau<Noeud *>& tab);
112

```

```

113 // Constructeur de copie
114 TriaQ3_cmptil003 (const TriaQ3_cmptil003& tria);
115
116
117 // DESTRUCTEUR :
118 ~TriaQ3_cmptil003 ();
119
120 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
121 // méthode virtuelle
122 Element* Nevez_copie() const { Element * el= new TriaQ3_cmptil003(*this); return el;};
123
124 // Surcharge de l'operateur = : realise l'egalite entre deux instances de TriaQ3_cmptil003
125 TriaQ3_cmptil003& operator= (TriaQ3_cmptil003& tria);
126
127 // METHODES :
128 // 1) derivant des virtuelles pures
129
130 // affichage dans la sortie transmise, des variables duales "nom"
131 // aux differents points d'integration
132 // dans le cas ou nom est vide, affichage de "toute" les variables
133 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
134
135 protected :
136
137 // adressage des frontières linéiques et surfacique
138 // définit dans les classes dérivées, et utilisées pour la construction des frontières
139 ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
140 { return ((ElFrontiere*) (new FrontSegQuad(tab,ddelem)));};
141 ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
142 { return ((ElFrontiere*) (new FrontTriaQuad(tab,ddelem)));};
143
144 // VARIABLES PRIVEES :
145 // place memoire commune a tous les elements TriaQ3_cmptil003
146 static TriaMemb::DonnComTria * doCoMembQ3;
147 // idem mais pour les indicateurs qui servent pour l'initialisation
148 static TriaMemb::UneFois uneFoisQ3;
149
150 class NombresConstruireTriaQ3_cmptil003 : public NombresConstruire
151 { public: NombresConstruireTriaQ3_cmptil003();
152 };
153 static NombresConstruireTriaQ3_cmptil003 nombre_V; // les nombres propres à l'élément
154
155 // GESTION AUTOMATIQUE D'AJOUT D'ELEMENT DANS LE PROGRAMME
156 //ajout de l'element dans la liste : listTypeElemen, geree par la class Element
157 class ConsTriaQ3_cmptil003 : public ConstrucElement
158 { public : ConsTriaQ3_cmptil003 ()
159 { NouvelleTypeElement nouv (TRIANGLE,QUADRACOMPL,MECA_SOLIDE_DEFORMABLE,this,"_cm3pti");
160 if (ParaGlob::NiveauImpression() >= 4)
161 cout << "\n initialisation TriaQ3_cmptil003" << endl;
162 Element::listTypeElement.push_back(nouv);
163 };
164 Element * NouvelElement(int num_maill,int num) // un nouvel élément sans rien
165 {Element * pt;
166 pt = new TriaQ3_cmptil003 (num_maill,num) ;
167 return pt;};
168 // ramene true si la construction de l'element est possible en fonction
169 // des variables globales actuelles: ex en fonction de la dimension
170 bool Element_possible() { if (ParaGlob::Dimension() >= 2) return true; else return false;};
171 };
172 static ConsTriaQ3_cmptil003 consTriaQ3_cmptil003;
173 };
174 /// @} // end of group
175 #endif
176
177
178
179

```

## 7.240 BielleThermi.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by

```

```

16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           06/03/2023
31 *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:        Herezh++
35 *
36 * *****/
37 *   BUT:   La classe BielletteThermi permet de declarer des elements *
38 *   biellettes et de realiser le calcul du residu local et de la raideur *
39 *   locale pour une loi de comportement donnee. La dimension de l'espace *
40 *   pour un tel element est 1.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *   ! date !   auteur !           but
46 *   -----
47 *   !           !           !
48 *
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date !   auteur !           but
52 *   -----
53 *
54 * *****/
55 // -----classe pour un calcul de mecanique-----
56
57
58
59 #ifndef BIELLETTETHERMI_H
60 #define BIELLETTETHERMI_H
61
62 #include "ParaGlob.h"
63 #include "ElemThermi.h"
64 // #include "Loi_comp_abstraite.h"
65 #include "Met_abstraite.h"
66 #include "Met_biellette.h"
67 #include "Noeud.h"
68 #include "UtilLecture.h"
69 #include "Tenseur.h"
70 #include "NevezTenseur.h"
71 #include "Deformation.h"
72 #include "ElFrontiere.h"
73 #include "GeomSeg.h"
74 #include "ParaAlgoControle.h"
75 #include "FrontSegLine.h"
76 #include "Section.h"
77
78 class ConstrucElementbiel;
79
80 /// @addtogroup groupe_des_elements_finis
81 /// @{
82 ///
83
84
85 class BielletteThermi : public ElemThermi
86 {
87
88     public :
89
90         // CONSTRUCTEURS :
91         // Constructeur par defaut
92         BielletteThermi ();
93
94         // Constructeur fonction d'une section et eventuellement d'un numero
95         // d'identification et de maillage
96         BielletteThermi (double sect,int num_maill=0,int num_id=-3);
97
98         // Constructeur fonction d'un numero de maillage et d'identification
99         BielletteThermi (int num_maill,int num_id);
100
101         // Constructeur fonction d'une section, d'un numero de maillage et d'identification,

```

```

102 // du tableau de connexite des noeuds
103 BielleTThermi (double sect,int num_maill,int num_id,const Tableau<Noeud *>& tab);
104
105 // Constructeur de copie
106 BielleTThermi (const BielleTThermi& biel);
107
108
109 // DESTRUCTEUR :
110 ~BielleTThermi ();
111
112 // création d'un élément de copie: utilisation de l'opérateur new et du constructeur de copie
113 // méthode virtuelle
114 Element* Nevez_copie() const { Element * el= new BielleTThermi(*this); return el;};
115
116 // Surcharge de l'operateur = : realise l'egalite entre deux instances de BielleTThermi
117 BielleTThermi& operator= (BielleTThermi& biel);
118
119 // METHODES :
120 // 1) derivant des virtuelles pures
121 // Lecture des donnees de la classe sur fichier
122 void LectureDonneesParticulieres (UtilLecture *,Tableau<Noeud *> * );
123
124 // Calcul du residu local et de la raideur locale,
125 // pour le schema implicite
126 Element::ResRaid Calcul_implicit (const ParaAlgoControle & pa);
127
128 // Calcul du residu local a t
129 // pour le schema explicit par exemple
130 Vecteur* CalculResidu_t (const ParaAlgoControle & pa)
131 { return BielleTThermi::CalculResidu(false,pa);};
132
133 // Calcul du residu local a tdt
134 // pour le schema explicit par exemple
135 Vecteur* CalculResidu_tdt (const ParaAlgoControle & pa)
136 { return BielleTThermi::CalculResidu(true,pa);};
137
138 // Calcul de la matrice masse pour l'élément
139 Mat_pleine * CalculMatriceMasse (Enum_calcul_masse id_calcul_masse) ;
140
141 // ----- calcul dynamique -----
142 // calcul de la longueur d'arrête de l'élément minimal
143 // divisé par la célérité la plus rapide dans le matériau
144 double Long_arrete_mini_sur_c(Enum_dure temps)
145 { return ElemThermi::Interne_Long_arrete_mini_sur_c(temps);};
146
147 //----- calcul d'erreur, remontée des contraintes -----
148 // 1)calcul du résidu et de la matrice de raideur pour le calcul d'erreur
149 Element::Er_ResRaid ContrainteAuNoeud_ResRaid();
150 // 2) remontée aux erreurs aux noeuds
151 Element::Er_ResRaid ErreurAuNoeud_ResRaid();
152
153 // retourne les tableaux de ddl associés aux noeuds, gere par l'element
154 // ce tableau et specifique a l'element
155 const DdlElement & TableauDdl() const ;
156
157
158 // Libere la place occupee par le residu et eventuellement la raideur
159 // par l'appel de Libere de la classe mere et libere les differents tenseurs
160 // intermediaires cree pour le calcul et les grandeurs pointee
161 // de la raideur et du residu
162 void Libere ();
163
164 // acquisition d'une loi de comportement
165 void DefLoi (LoiAbstraiteGeneral * NouvelleLoi);
166
167 // test si l'element est complet
168 // = 1 tout est ok, =0 element incomplet
169 int TestComplet();
170
171 // procedure permettant de completer l'element apres
172 // sa creation avec les donnees du bloc transmis
173 // peut etre appeler plusieurs fois
174 Element* Complete(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
175 // Compléter pour la mise en place de la gestion de l'hourglass
176 Element* Complet_Hourglass(LoiAbstraiteGeneral * NouvelleLoi, const BlocGen & bloc) {return this;};
177
178 // ramene l'element geometrique
179 ElemGeomC0& ElementGeometrique() const { return doCo->segment;};
180 // ramene l'element geometrique en constant
181 const ElemGeomC0& ElementGeometrique_const() const { return doCo->segment;};
182
183 // calcul d'un point dans l'élément réel en fonction des coordonnées dans l'élément de référence
184 // associé
185 // temps: indique si l'on veut les coordonnées à t = 0, ou t ou tdt
186 // 1) cas où l'on utilise la place passée en argument
187 Coordonnee & Point_physique(const Coordonnee& c_int,Coordonnee & co,Enum_dure temps);
188 // 3) cas où l'on veut les coordonnées aux 1, 2 ou trois temps selon la taille du tableau t_co

```

```

188 void Point_physique(const Coordonnee& c_int,Tableau <Coordonnee> & t_co);
189
190 // -- connaissances particulières sur l'élément
191 // ramène l'épaisseur de l'élément
192 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
193 virtual double Section(Enum_dure enu , const Coordonnee& ) {return S(enu);};
194 // ramène l'épaisseur moyenne de l'élément (indépendante du point)
195 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
196 virtual double SectionMoyenne(Enum_dure enu ) {return S(enu);};
197
198 // affichage dans la sortie transmise, des variables duales "nom"
199 // dans le cas ou nom est vide, affichage de "toute" les variables
200 void AfficheVarDual(ofstream& sort, Tableau<string>& nom);
201
202 // affichage d'info en fonction de ordre
203 // ordre = "commande" : affichage d'un exemple d'entree pour l'élément
204 void Info_com_Element(UtilLecture * entreePrinc,string& ordre,Tableau<Noeud *> * tabMaillageNoeud)
205
206     { return Element::Info_com_El(2,entreePrinc,ordre,tabMaillageNoeud);};
207
208 // retourne un numero d'ordre d'un point le plus près ou est exprimé la grandeur enum
209 // par exemple un point d'intégration, mais n'est utilisable qu'avec des méthodes particulières
210 // par exemple CoordPtInteg, ou Valeur_a_diff_temps
211 // car le numéro d'ordre peut-être différent du numéro d'intégration au sens classique
212 // temps: dit si c'est à 0 ou t ou tdt
213 int PointLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M)
214     { return PtLePlusPres(temps,enu,M);};
215
216 // recuperation des coordonnées du point de numéro d'ordre iteg pour
217 // la grandeur enu
218 // temps: dit si c'est à 0 ou t ou tdt
219 // si erreur retourne erreur à true
220 Coordonnee CoordPtInteg(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur)
221     { return CoordPtInt(temps,enu,iteg,erreur);};
222
223 // récupération des valeurs au numéro d'ordre = iteg pour
224 // les grandeurs enu
225 Tableau <double> Valeur_a_diff_temps(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>&
226 enu,int iteg) ;
227
228 // récupération des valeurs au numéro d'ordre = iteg pour les grandeurs enu
229 // ici il s'agit de grandeurs tensorielles, le retour s'effectue dans la liste
230 // de conteneurs quelconque associée
231 void ValTensorielle_a_diff_temps(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu,int
232 iteg);
233
234 // ramene vrai si la surface numéro ns existe pour l'élément
235 // dans le cas de la biellette il n'y a pas de surface
236 bool SurfExiste(int ) const
237     { return false;};
238
239 // ramene vrai si l'arête numéro na existe pour l'élément
240 bool AreteExiste(int na) const {if (na==1) return true; else return false;};
241
242 //===== lecture écriture dans base info =====
243
244 // cas donne le niveau de la récupération
245 // = 1 : on récupère tout
246 // = 2 : on récupère uniquement les données variables (supposées comme telles)
247 void Lecture_base_info
248     (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
249 // cas donne le niveau de sauvegarde
250 // = 1 : on sauvegarde tout
251 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
252 void Ecriture_base_info(ofstream& sort,const int cas) ;
253
254 // METHODES VIRTUELLES:
255 // ----- calculs utils dans le cadre de la recherche du flambement linéaire
256 // Calcul de la matrice géométrique et initiale
257 ElemThermi::MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
258 // retourne la liste des données particulières actuellement utilisés
259 // par l'élément (actif ou non), sont exclu de cette liste les données particulières des noeuds
260 // reliés à l'élément
261 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
262 // particulière
263 List_io <TypeQuelconque> Les_types_particuliers_internes(bool absolue) const;
264
265 // récupération de grandeurs particulières au numéro d'ordre = iteg
266 // celles-ci peuvent être quelconques
267 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
268 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
269 // particulière
270 void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);
271
272 // inactive les ddl du problème primaire de mécanique
273 inline void Inactive_ddl_primaire()

```

```

270             {ElemThermi::Inact_ddl_primaire(doCo->tab_ddl)};};
271 // active les ddl du problème primaire de mécanique
272 inline void Active_ddl_primaire()
273             {ElemThermi::Act_ddl_primaire(doCo->tab_ddl)};};
274 // ----- calcul d'erreur, calculs du champs de Flux continu -----
275
276 // ajout des ddl de flux pour les noeuds de l'élément
277 inline void Plus_ddl_Flux()
278             {ElemThermi::Ad_ddl_Flux(doCo->tab_ddlErr)};};
279 // inactive les ddl du problème de recherche d'erreur : les flux
280 inline void Inactive_ddl_Flux()
281             {ElemThermi::Inact_ddl_Flux(doCo->tab_ddlErr)};};
282 // active les ddl du problème de recherche d'erreur : les flux
283 inline void Active_ddl_Flux()
284             {ElemThermi::Act_ddl_Flux(doCo->tab_ddlErr)};};
285 // active le premier ddl du problème de recherche d'erreur : SIGMA11
286 inline void Active_premier_ddl_Flux()
287             {ElemThermi::Act_premier_ddl_Flux()};};
288
289 // lecture de données diverses sur le flot d'entrée
290 void LectureFlux(UtilLecture * entreePrinc)
291     {if (unefois.CalResPrem_t == 1)
292         ElemThermi::LectureDesFlux (false,entreePrinc,lesPtThermiInt.TabfluxH_t());
293     else
294         { ElemThermi::LectureDesFlux (true,entreePrinc,lesPtThermiInt.TabfluxH_t());
295           unefois.CalResPrem_t = 1;
296         };
297     };
298
299 // retour des flux en absolu retour true si elle existe sinon false
300 bool FluxAbsolues(Tableau <Vecteur>& tabFlux)
301     { if (unefois.CalResPrem_t == 1)
302         ElemThermi::FluxEnAbsolues (false,lesPtThermiInt.TabfluxH_t(),tabFlux);
303     else
304         { unefois.CalResPrem_t = 1;
305           ElemThermi::FluxEnAbsolues (true,lesPtThermiInt.TabfluxH_t(),tabFlux);
306         };
307     return true;
308     };
309
310 // 2) derivant des virtuelles
311 // retourne un tableau de ddl element, correspondant à la
312 // composante de sigma -> SIG11, pour chaque noeud qui contient
313 // des ddl de contrainte
314 // -> utilisé pour l'assemblage de la raideur d'erreur
315 inline DdlElement& Tableau_de_Flux1() const
316     {return doCo->tab_Err1FLUX;} ;
317
318 // actualisation des ddl et des grandeurs actives de t+dt vers t
319 void TdtversT();
320 // actualisation des ddl et des grandeurs actives de t vers tdt
321 void TversTdt();
322
323 // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
324 // qu'une fois la remontée aux contraintes effectuées sinon aucune
325 // action. En retour la valeur de l'erreur sur l'élément
326 // type indique le type de calcul d'erreur :
327 void ErreurElement(int type,double& errElemRelative
328                   ,double& numerateur, double& denominateur);
329
330 // mise à jour de la boîte d'encombrement de l'élément, suivant les axes I_a globales
331 // en retour coordonnées du point mini dans retour.Premier() et du point maxi dans .Second()
332 // la méthode est différente de la méthode générale car il faut prendre en compte l'épaisseur de
333 // l'élément
334 virtual const DeuxCoordonnees& Boite_encombre_element(Enum_dure temps);
335
336 // calcul des seconds membres suivant les chargements
337 // cas d'un chargement volumique,
338 // force indique la force volumique appliquée
339 // retourne le second membre résultant
340 // ici on l'épaisseur de l'élément pour constituer le volume
341 // -> explicite à t
342 Vecteur SM_charge_volumique_E_t(const Coordonnee& force,const ParaAlgoControle & pa)
343     { return BiellletteThermi::SM_charge_volumique_E(force,false,pa);} ;
344 // -> explicite à tdt
345 Vecteur SM_charge_volumique_E_tdt(const Coordonnee& force,const ParaAlgoControle & pa)
346     { return BiellletteThermi::SM_charge_volumique_E(force,true,pa);} ;
347 // -> implicite,
348 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
349 // retourne le second membre et la matrice de raideur correspondant
350 ResRaid SMR_charge_volumique_I(const Coordonnee& force,const ParaAlgoControle & pa) ;
351
352 // cas d'un chargement lineique, sur les aretes frontieres des éléments
353 // force indique la force lineique appliquée
354 // numarete indique le numéro de l'arete chargée
355 // retourne le second membre résultant

```



```

356 // -> explicite à t
357 Vecteur SM_charge_lineique_E_t(const Coordonnee& force,int numArete,const ParaAlgoControle & pa)
358 { return BiелletteThermi::SM_charge_lineique_E(force,numArete,false,pa); } ;
359 // -> explicite à tdt
360 Vecteur SM_charge_lineique_E_tdt(const Coordonnee& force,int numArete,const ParaAlgoControle & pa)
361 { return BiелletteThermi::SM_charge_lineique_E(force,numArete,true,pa); } ;
362 // -> implicite,
363 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
364 // retourne le second membre et la matrice de raideur correspondant
365 ResRaid SMR_charge_lineique_I(const Coordonnee& force,int numArete,const ParaAlgoControle & pa) ;
366
367 // cas d'un chargement lineique suiveuse, sur l'arrête frontière de
368 // la biелlette (2D uniquement)
369 // force indique la force lineique appliquée
370 // numarete indique le numéro de l'arete chargée
371 // retourne le second membre résultant
372 // -> explicite à t
373 Vecteur SM_charge_lineique_Suiv_E_t(const Coordonnee& force,int numArete,const ParaAlgoControle &
pa)
374 { return BiелletteThermi::SM_charge_lineique_Suiv_E(force,numArete,false,pa); } ;
375 // -> explicite à tdt
376 Vecteur SM_charge_lineique_Suiv_E_tdt(const Coordonnee& force,int numArete,const ParaAlgoControle
& pa)
377 { return BiелletteThermi::SM_charge_lineique_Suiv_E(force,numArete,true,pa); } ;
378 // -> implicite,
379 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
380 // retourne le second membre et la matrice de raideur correspondant
381 ResRaid SMR_charge_lineique_Suiv_I(const Coordonnee& force,int numArete,const ParaAlgoControle &
pa) ;
382
383 // cas d'un chargement surfacique hydro-dynamique,
384 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
385 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc
unitaire)
386 // une suivant la direction normale à la vitesse de type portance
387 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
388 // une suivant la vitesse tangente de type frottement visqueux
389 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
390 // coef_mul: est un coefficient multiplicateur global (de tout)
391 // retourne le second membre résultant
392 // -> explicite à t
393 Vecteur SM_charge_hydrodynamique_E_t( CourbeID* frot_fluid,const double& poidvol
, CourbeID* coef_aero_n,int numFace,const double&
coef_mul
, CourbeID* coef_aero_t,const ParaAlgoControle & pa)
395 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,false,pa);};
396
397 // -> explicite à tdt
398 Vecteur SM_charge_hydrodynamique_E_tdt( CourbeID* frot_fluid,const double& poidvol
, CourbeID* coef_aero_n,int numFace,const double&
coef_mul
, CourbeID* coef_aero_t,const ParaAlgoControle & pa)
400 {return
SM_charge_hydrodynamique_E(frot_fluid,poidvol,coef_aero_n,numFace,coef_mul,coef_aero_t,true,pa);};
401
402 // -> implicite,
403 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
404 // retourne le second membre et la matrice de raideur correspondant
405 ResRaid SMR_charge_hydrodynamique_I( CourbeID* frot_fluid,const double& poidvol
, CourbeID* coef_aero_n,int numFace,const double&
coef_mul
, CourbeID* coef_aero_t,const ParaAlgoControle & pa)
407 ;
408
409
410 // ===== définition et/ou construction des frontières =====
411
412 // Calcul des frontières de l'element
413 // creation des elements frontieres et retour du tableau de ces elements
414 // la création n'a lieu qu'au premier appel
415 // ou lorsque l'on force le paramètre force a true
416 // dans ce dernier cas seul les frontière effacées sont recréée
417 Tableau <ElFrontiere*> const & Frontiere(bool force = false);
418
419
420
421 // Retourne la section de l'element
422 inline double S(Enum_dure enu = TEMPS_tdt)
423 { switch (enu)
424 { case TEMPS_0: return donnee_specif.secti.section0; break;
425 case TEMPS_t: return donnee_specif.secti.section_t; break;
426 case TEMPS_tdt: return donnee_specif.secti.section_tdt; break;
427 };
428 return 0.; // cas n'arrivant normalement jamais
429 };

```

```

430
431
432
433     // ajout du tableau specific de ddl des noeuds de la biellette
434     // la procedure met a jour les ddl(relatif a l'element, c-a-d Xi)
435     // des noeuds constituant l'element
436     void ConstTabDdl();
437 protected:
438
439     // ===> methodes virtuelles derivant d'ElemThermi =====
440     // ramene la dimension des vecteurs flux et gradient de temperature de l'element
441     int Dim_flux_gradT() const {return 1;};
442
443 // ----- calcul de frontieres en protected -----
444
445     // --- fonction necessaire pour la construction des Frontieres lineiques ou surfaciques
particuliere a l'element
446     // adressage des frontieres lineiques et surfacique
447     // definit dans les classes derivees, et utilisees pour la construction des frontieres
448     virtual ElFrontiere* new_frontiere_lin(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
449     { return ((ElFrontiere*) (new FrontSegLine(tab,ddelem)));};
450     virtual ElFrontiere* new_frontiere_surf(int ,Tableau <Noeud *> & tab, DdlElement& ddelem)
451     {return NULL;}; // il n'y a pas de surface possible
452
453
454 private :
455
456     // VARIABLES PRIVEES :
457
458
459     class DonneeCommune
460     { public :
461         DonneeCommune (GeomSeg& seg,DdlElement& tab,DdlElement& tabErr,DdlElement& tab_Err1FLUX,
462             Met_biellette& met_bie,
463             Tableau <Vecteur *> & resEr,Mat_pleine& raidEr,
464             GeomSeg& seEr,Vecteur& residu_int,Mat_pleine& raideur_int,
465             Tableau <Vecteur* > & residus_extN,Tableau <Mat_pleine* >& raideurs_extN,
466             Tableau <Vecteur* > & residus_extA,Tableau <Mat_pleine* >& raideurs_extA,
467             Mat_pleine& mat_masse ,GeomSeg& seMa,int nbi
468             ) ;
469         DonneeCommune(DonneeCommune& a);
470         ~DonneeCommune();
471         // variables
472         GeomSeg segment ; // element geometrique correspondant
473         DdlElement tab_ddl; // tableau des degres
474             //de liberte des noeuds de l'element commun a tous les
475             // elements: ici de thermique
476         Met_biellette met_biellette;
477         Mat_pleine matGeom ; // matrice geometrique
478         Mat_pleine matInit ; // matrice initiale
479         Tableau <CoordonneeB> d_gradTB; // place pour la variation du gradient
480         Tableau <CoordonneeH> d_fluxH; // place pour la variation du flux
481         Tableau <Tableau2 <CoordonneeB> * > d2_gradTB; // variation seconde des deformations
482             // calcul d'erreur
483         DdlElement tab_ddlErr; // tableau des degres servant pour le calcul
484             // d'erreur : contraintes
485         DdlElement tab_Err1FLUX; // tableau du ddl FLUX pour chaque noeud,
486             //servant pour le calcul d'erreur : contraintes, en fait pour l'assemblage
487         Tableau <Vecteur *> resErr; // residu pour le calcul d'erreur
488         Mat_pleine raidErr; // raideur pour le calcul d'erreur
489         GeomSeg segmentEr; // contient les fonctions d'interpolation et
490             // les derivees pour le calcul du hessien dans
491             //la resolution de la fonctionnelle d'erreur
492     // ----- calcul de residus, de raideur : interne ou pour les efforts exterieurs
-----
493     // on utilise des pointeurs pour optimiser la place (même place pointé éventuellement)
494     Vecteur residu_interne;
495     Mat_pleine raideur_interne;
496     Tableau <Vecteur* > residus_externeN; // pour les noeuds
497     Tableau <Mat_pleine* > raideurs_externeN; // pour les noeuds
498     Tableau <Vecteur* > residus_externeA; // pour l' aretes
499     Tableau <Mat_pleine* > raideurs_externeA; // pour l' aretes
500     // ----- données concernant la dynamique -----
501     Mat_pleine matrice_masse;
502     GeomSeg segmentMas; // contient les fonctions d'interpolation et les derivees
503     // pour les calculs relatifs au calcul de la masse
504 };
505
506
507 // classe contenant tous les indicateurs statique qui sont modifiés une seule fois
508 // et un pointeur sur les données statiques communes
509 // la classe est interne, toutes les variables sont publique. Un pointeur sur une instance de la
510 // classe est défini. Son allocation est effectuée dans les classes derivees
511 class UneFois
512 { public :
513     UneFois () ; // constructeur par défaut
514     ~UneFois () ; // destructeur

```

```

515
516 // VARIABLES :
517 public :
518     DonneCommune * doCoMemb;
519
520     // indicateurs permettant de dimensionner seulement au premier passage
521     // utilise dans "CalculResidu" et "Calcul_implicit"
522     int CalResPrem_t; int CalResPrem_tdt; // à t ou à tdt
523     int CalimpPrem;
524     int dualSortbiel; // pour la sortie des valeurs au pt d'integ
525     int CalSMlin_t; // pour les seconds membres concernant les arretes
526     int CalSMlin_tdt; // pour les seconds membres concernant les arretes
527     int CalSMRlin; // pour les seconds membres concernant les arretes
528     int CalSMvol_t; // pour les seconds membres concernant les volumes
529     int CalSMvol_tdt; // pour les seconds membres concernant les volumes
530     int CalSMvol; // pour les seconds membres concernant les volumes
531     int CalDynamique; // pour le calcul de la matrice de masse
532     int CalPt_0_t_tdt; // pour le calcul de point à 0 t et tdt
533     // ----- sauvegarde du nombre d'élément en cours -----
534     int nbelem_in_Prog;
535 };
536
537
538 // -----
539
540 protected :
541 // VARIABLES PROTÉGÉES :
542 // les données spécifiques sont groupées dans une structure pour sécuriser
543 // le passage de paramètre dans init par exemple
544 class Donnee_specif
545 { public :
546     Donnee_specif() : // défaut
547         secti(Element::section_defaut,Element::section_defaut,Element::section_defaut)
548     ,variation_section(true)
549     {};
550     Donnee_specif(double section) : // uniquement la section
551     secti(section,section,section),variation_section(true)
552     {};
553     Donnee_specif(double epai0,double epai_t,double epai_tdt,bool variation) : // tous
554     secti(epai0,epai_t,epai_tdt),variation_section(variation)
555     {};
556     Donnee_specif(const Donnee_specif& a) :
557     secti(a.secti ),variation_section(a.variation_section)
558     {}; // copie via le constructeur de copie
559     ~Donnee_specif() {};
560     Donnee_specif & operator = ( const Donnee_specif& a)
561     { secti = a.secti;variation_section=a.variation_section;
562       return *this;};
563     // data
564     // sections de l'element
565     Sect secti; // épaisseur
566     bool variation_section; // permet éventuellement de ne pas prendre en compte la variation
567 };
568 Donnee_specif donnee_specif;
569
570 // grandeurs aux points d'intégration: contraintes, déformations, vitesses de def etc.
571 LesPtIntegThermiInterne lesPtThermiInt;
572
573 // place memoire commune a tous les elements biellettes
574 static DonneCommune * doCo;
575 // idem mais pour les indicateurs qui servent pour l'initialisation
576 static UneFois unefois;
577
578 // type structuré pour construire les éléments
579 class NombresConstruire
580 { public:
581     NombresConstruire();
582     int nbne; // le nombre de noeud de l'élément
583     int nbneA ; // le nombre de noeud des arretes
584     int nbi; // le nombre de point d'intégration pour le calcul mécanique
585     int nbiEr; // le nombre de point d'intégration pour le calcul d'erreur
586     int nbiA; // le nombre de point d'intégration pour le calcul de second membre linéique
587     int nbiMas; // le nombre de point d'intégration pour le calcul de la matrice masse consistante
588 };
589 static NombresConstruire nombre_V; // les nombres propres à l'élément
590
591 // fonction privée
592 // fonction d'initialisation servant au niveau du constructeur
593 BielletteThermi::DonneeCommune * Init(Donnee_specif donnee_specif = Donnee_specif()
594     ,bool sans_init_noeud = false);
595 void Def_DonneCommune();
596 // destructions de certaines grandeurs pointées, créées au niveau de l'initialisation
597 void Destruction();
598
599 // pour l'ajout d'element dans la liste : listTypeElemen, geree par la class Element
600 class ConstrucElementbiel : public ConstrucElement
601 { public : ConstrucElementbiel ()

```

```

602         { NouvelleTypeElement nouv(POUT,BIE1,THERMIQUE,this);
603           if (ParaGlob::NiveauImpression() >= 4)
604             cout << "\n initialisation BielleTThermi" << endl;
605           Element::listTypeElement.push_back(nouv);
606         };
607     Element * NouvelElement(int nb_mail,int num) // un nouvel élément sans rien
608     {Element * pt;
609       pt = new BielleTThermi (nb_mail,num) ;
610       return pt;};
611     // ramene true si la construction de l'element est possible en fonction
612     // des variables globales actuelles: ex en fonction de la dimension
613     bool Element_possible() {return true;};
614 };
615 static ConstrucElementbiel construcElementbiel;
616
617 // Calcul du residu local a t ou tdt en fonction du booleen
618 Vecteur* CalculResidu (bool atdt,const ParaAlgoControle & pa);
619 // calcul des seconds membres suivant les chargements
620 // cas d'un chargement volumique,
621 // force indique la force volumique appliquée
622 // retourne le second membre résultant
623 // ici on l'épaisseur de l'élément pour constituer le volume
624 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
625 Vecteur SM_charge_volumique_E
626     (const Coordonnee& force,bool atdt,const ParaAlgoControle & pa);
627 // cas d'un chargement lineique, sur les aretes frontieres des éléments
628 // force indique la force lineique appliquée
629 // numarete indique le numéro de l'arete chargée
630 // retourne le second membre résultant
631 // -> explicite à t ou tdt en fonction de la variable booleenne atdt
632 Vecteur SM_charge_lineique_E
633     (const Coordonnee& force,int numArete,bool atdt,const ParaAlgoControle & pa);
634 // cas d'un chargement lineique suiveuse, sur l'arete frontiere
635 //de la bielleTte (2D uniquement)
636 // force indique la force lineique appliquée
637 // numarete indique le numéro de l'arete chargée
638 // retourne le second membre résultant
639 // -> explicite à t
640 Vecteur SM_charge_lineique_Suiv_E
641     (const Coordonnee& force,int numArete,bool atdt,const ParaAlgoControle & pa);
642 // cas d'un chargement surfacique hydro-dynamique,
643 // voir methode explicite plus haut, pour les arguments
644 // retourne le second membre résultant
645 // bool atdt : permet de spécifier à t ou a t+dt
646 Vecteur SM_charge_hydrodynamique_E( CourbeID* frot_fluid,const double& poidvol
647     , CourbeID* coef_aero_n,int numFace,const double&
648     , CourbeID* coef_aero_t,bool atdt,const
649     ParaAlgoControle & pa) ;
650 // calcul de la nouvelle section moyenne finale (sans raideur)
651 // mise à jour des volumes aux pti
652 // ramène la section moyenne calculée à atdt
653 const double& CalSectionMoyenne_et_vol_pti(const bool atdt);
654 };
655 /// @} // end of group
656
657 #endif
658
659
660
661

```

## 7.241 ElemThermi.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty

```

```

21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      06/03/2023
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *****/
37 *   BUT:      Defini l'element generique de thermique.
38 *
39 *   *****
40 *
41 *   VERIFICATION:
42 *
43 *   ! date !   auteur !       but
44 *   -----
45 *   !       !       !
46 *   *****
47 *
48 *   MODIFICATIONS:
49 *
50 *   ! date !   auteur !       but
51 *   -----
52 *   $
53 *
54 *****/
52 #ifndef ELEM_THERMI_H
53 #define ELEM_THERMI_H
54
55 #include "Element.h"
56 #include "Tenseur.h"
57 #include "NevezTenseur.h"
58 #include "Deformation.h"
59 #include "Loi_comp_abstraite.h"
60 #include "Enum_calcul_masse.h"
61 #include "Basiques.h"
62 #include "Enum_dure.h"
63 #include "CompThermoPhysiqueAbstraite.h"
64 #include "CompProtAbstraite.h"
65 #include "LesPtIntegThermiInterne.h"
66 #include "Enum_StabHourglass.h"
67 #include "EnergieThermi.h"
68
69
70 /// @addtogroup groupe_des_elements_finis
71 /// @{
72 ///
73
74 class ElemThermi : public Element
75 {
76 public :
77     // VARIABLES PUBLIQUES :
78
79     // CONSTRUCTEURS :
80     ElemThermi ();
81     // Constructeur utile quand le numero de maillage et d'identification de l'element est connu
82     ElemThermi (int num_maill,int num_id );
83     // Constructeur utile quand le numero de maillage et d'identification et le tableau des noeuds
84     // de l'element sont connus
85     ElemThermi (int num_maill,int num_id,const Tableau<Noeud *>& tab);
86     // Constructeur utile quand le numero de maillage et d'identification est connu,
87     // ainsi que la geometrie et le type d'interpolation de l'element
88     ElemThermi (int num_maill,int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt,string
89 info="");
89     // Constructeur utile quand le numero de maillage et d'identification est connu,
90     // ainsi que la geometrie et le type d'interpolation de l'element
91     ElemThermi (int num_maill,int num_id,char* nom_interp,char* nom_geom,string info="");
92     // Constructeur utile quand toutes les donnees de la classe Element sont connues
93     ElemThermi (int num_maill,int num_id,const Tableau<Noeud *>& tab,Enum_interpol id_interp_elt,
94 Enum_geom id_geom_elt,string info="");
95     // Constructeur utile quand toutes les donnees de la classe Element sont connues
96     ElemThermi (int num_maill,int num_id,const Tableau<Noeud *>& tab,char* nom_interp,
97 char* nom_geom,string info="");
98     // Constructeur de copie
99     ElemThermi (const ElemThermi& elt);
100
101 // DESTRUCTEUR :
102 ~ElemThermi ();
103
104 // METHODES PUBLIQUES :
105 // test si l'element est complet

```

```

106 // = 1 tout est ok, =0 element incomplet
107 int TestComplet();
108
109 // calcul si un point est a l'interieur de l'element ou non
110 // il faut que M est la dimension globale
111 // les trois fonctions sont pour l'etude a t=0, t et tdt
112 // retour : =0 le point est externe, =1 le point est interne ,
113 // = 2 le point est sur la frontiere a la precision pres
114 // coor_locales : s'il est different de NULL, est affecte des coordonnees locales calculees,
115 // uniquement precises si le point est interne
116 int Interne_0(const Coordonnee& M, Coordonnee* coor_locales=NULL);
117 int Interne_t(const Coordonnee& M, Coordonnee* coor_locales=NULL);
118 int Interne_tdt(const Coordonnee& M, Coordonnee* coor_locales=NULL);
119
120 // recuperation des energies integrees sur l'elements, resultants d'un precedent calcul
121 // explicite, ou implicite
122 const EnergieThermi& EnergieTotaleElement() const {return energie_totale;};
123
124 // test pour savoir si le calcul de Flux en absolu est possible
125 bool FluxAbsoluePossible();
126
127 // METHODES VIRTUELLES:
128
129 // retourne la liste de tous les types de ddl interne actuellement utilises
130 // par l'element (actif ou non), sont exclu de cette liste les ddl des noeuds
131 // relies a l'element (ddl implique grandeur uniquement scalaire !)
132 virtual List_io <Ddl_enum_etendu> Les_type_de_ddl_internes(bool absolue) const ;
133 // idem pour les grandeurs evoluees c'est-a-dire directement sous forme de vecteur, tenseurs
134
135 virtual List_io <TypeQuelconque> Les_type_evolues_internes(bool absolue) const ;
136 // idem pour les donnees particulieres
137 virtual List_io <TypeQuelconque> Les_types_particuliers_internes(bool absolue) const;
138
139 // retourne la liste de toutes les grandeurs quelconques relatives aux faces de
140 // l'element (actif ou non),
141 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particuliere
142 virtual List_io <TypeQuelconque> Les_type_quelconque_de_face(bool absolue) const ;
143
144 // retourne la liste de toutes les grandeurs quelconques relatives aux arêtes de
145 // l'element (actif ou non),
146 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particuliere
147 virtual List_io <TypeQuelconque> Les_type_quelconque_de_arete(bool absolue) const;
148
149 // ----- calculs utils dans le cadre de la recherche du flambement lineaire
150 // dans un premier temps uniquement virtuelles, ensuite se sera virtuelle pure pour eviter
151 // les oublis de definition ----> AMODIFIER !!!!!!!
152 // Calcul de la matrice geometrique et initiale
153 class MatGeomInit // pour le retour des pointeurs sur des matrices stockees
154 // une paire par classe d'elements
155 { public : MatGeomInit(Mat_pleine * matG, Mat_pleine * matI) :
156 matGeom(matG), matInit(matI) {};
157 Mat_pleine * matGeom ; Mat_pleine * matInit ;
158 };
159 virtual MatGeomInit MatricesGeometrique_Et_Initiale (const ParaAlgoControle & pa) ;
160
161 // ----- calcul d'erreur, calculs du champs de Flux continu -----
162 // ajout des ddl de Flux pour les noeuds de l'element
163 virtual void Plus_ddl_Flux() = 0;
164 // inactive les ddl du probleme de recherche d'erreur : les Flux
165 virtual void Inactive_ddl_Flux() = 0;
166 // active les ddl du probleme de recherche d'erreur : les Flux
167 virtual void Active_ddl_Flux() = 0 ;
168 // active le premier ddl du probleme de recherche d'erreur : FLUX
169 virtual void Active_premier_ddl_Flux() = 0 ;
170 // retourne un tableau de ddl element, correspondant a la
171 // composante de flux -> FLUX, pour chaque noeud qui contient
172 // des ddl de flux
173 // -> utilise pour l'assemblage de la raideur d'erreur
174 //!!!!!!!!!!!!!!!!!!!! pour l'instant en virtuelle il faudra apres en
175 // virtuelle pure !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
176 virtual DdlElement& Tableau_de_Flux1() const ;
177 // retourne un tableau de ddl element, correspondant a la
178 // composante d'erreur -> ERREUR, pour chaque noeud
179 // -> utilise pour l'assemblage de la raideur d'erreur
180 // dans cette version tous les noeuds sont supposes avoir un ddl erreur
181 // dans le cas contraire il faut redéfinir la fonction dans l'element terminal
182 virtual DdlElement Tableau_de_ERREUR() const ;
183 // calcul de l'erreur sur l'element. Ce calcul n'est disponible
184 // qu'une fois la remontée aux Flux effectuées sinon aucune
185 // action. En retour la valeur de l'erreur sur l'element
186 // type indique le type de calcul d'erreur :
187 // = 1 : erreur = (int (delta sigma):(delta sigma) dv)/(int sigma:sigma dv)
188 // le numerateur et le denominateur sont tel que :
189 // errElemRelative = numerateur / denominateur , si denominateur different de 0
190 // sinon denominateur = numerateur si numerateur est different de 0, sinon

```

```

190 // tous sont nuls mais on n'effectue pas la division
191 //!!!!!!!!!!!!!! pour l'instant en virtuelle il faudra après en
192 // virtuelle pure !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
193 virtual void ErreurElement(int type,double& errElemRelative
194                          ,double& numerateur, double& denominateur);
195 // les 3 routines qui suivent sont virtuelles, car la définition
196 //qui est faite dans ElemThermi.cp considère qu'il y a un ddl erreur
197 // par noeud de l'élément de manière systématique,
198 // avec le status virtuel on peut définir dans la classe dérivée
199 // un cas particulier
200 // ajout des ddl d'erreur pour les noeuds de l'élément
201 virtual void Plus_ddl_Erreur() ;
202 // inactive les ddl d'erreur
203 virtual void Inactive_ddl_Erreur() ;
204 // active les ddl d'erreur
205 virtual void Active_ddl_Erreur() ;
206 // test pour savoir si l'erreur a été calculée
207 bool ErreurDejaCalculee()
208     { if (fluxErreur == NULL)
209         return false;
210         else return true;};
211 // sortie de l'erreur à l'élément
212 double Erreur( )
213     { return (*fluxErreur);};
214
215 // lecture de données diverses sur le flot d'entrée
216 // l'implantation est faite dans les classe dérivées
217 virtual void LectureFlux(UtilLecture * entreePrinc) =0 ;
218
219 // retour des Flux en absolu retour true si elle existe sinon false
220 virtual bool FluxAbsolues(Tableau <Vecteur>& tabFlux) = 0;
221
222 // ----- calcul dynamique -----
223
224 // calcul de la longueur d'arrête de l'élément minimal
225 // divisé par la célérité dans le matériau
226 virtual double Long_arrete_mini_sur_c(Enum_dure temps) = 0;
227 // cas du bulk viscosity
228 double E_elem_bulk_t,E_elem_bulk_tdt,P_elem_bulk;
229 static void ActiveBulkViscosity(int choix) {bulk_viscosity=choix;};
230 static void InactiveBulkViscosity() {bulk_viscosity=false;};
231 static void ChangeCoefsBulkViscosity(const DeuxDoubles & coef)
232     { c_traceBulk=coef.un;c_carreBulk=coef.deux;};
233
234 // initialisation pour le calcul de la matrice masse dans le cas de l'algorithme
235 // de relaxation dynamique avec optimisation en continu de la matrice masse
236 // casMass_relax: permet de choisir entre différentes méthodes de calcul de la masse
237 void InitCalculMatriceMassePourRelaxationDynamique(int casMass_relax);
238 // phase de calcul de la matrice masse dans le cas de l'algo de relaxation dynamique
239 // mi=fonction de (alpha*K+beta*mu+gamma*Isig/3+theta/2*Sig_mises
240 // ep: epaisseur, K module de compressibilité, mu: module de cisaillement, Isig trace de sigma,
241
242 // Sig_mises la contrainte de mises
243 // casMass_relax: permet de choisir entre différentes méthodes de calcul de la masse
244 void CalculMatriceMassePourRelaxationDynamique
245     (const double& alph, const double& beta, const double & lambda
246      ,const double & gamma,const double & theta, int casMass_relax);
247 // ----- informations annexes -----
248
249 // récupération de la base locales au noeud noe, pour le temps: temps
250 const BaseB & Gib_elemeca(Enum_dure temps, const Noeud * noe);
251
252 // récupération de grandeurs particulières au numéro d'ordre = iteg
253 // celles-ci peuvent être quelconques
254 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
255 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
256 void Grandeur_particuliere (bool absolue,List_io<TypeQuelconque>& liTQ,int iteg);
257
258 // récupération de grandeurs particulières pour une face au numéro d'ordre = iteg
259 // celles-ci peuvent être quelconques
260 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
261 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
262 void Grandeur_particuliere_face (bool absolue,List_io<TypeQuelconque>& liTQ,int face, int iteg);
263
264 // récupération de grandeurs particulières pour une arête au numéro d'ordre = iteg
265 // celles-ci peuvent être quelconques
266 // en retour liTQ est modifié et contient les infos sur les grandeurs particulières
267 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
268 void Grandeur_particuliere_arete (bool absolue,List_io<TypeQuelconque>& liTQ,int arete, int
iteg);
269
270
271 // récupération de la loi de frottement, dans le cas où elle n'existe pas

```

```

272 // retour d'un pointeur nul
273 CompFrotAbstraite* LoiDeFrottement() const {return loiFrot;};
274
275 // --- transfert des grandeurs des points d'intégration aux noeuds
276 // transfert de ddl des points d'intégrations (de tous) d'un éléments (on ajoute aux
noeuds, on ne remplace pas)
277 // les ddl doivent déjà exister aux noeuds sinon erreur
278 // il doit s'agir du même type de répartition de pt d'integ pour toutes les grandeurs
279 // tab_val(i)(j) : valeur associée au i ième pt d'integ et au j ième ddl_enum_etendu
280 void TransfertAjoutAuNoeuds(const List_io < Ddl_enum_etendu >& lietendu
281 ,const Tableau <Tableau <double> > & tab_val,int cas);
282 // transfert de type quelconque des points d'intégrations (de tous) aux noeuds d'un éléments (on
ajoute aux noeuds,
283 // on ne remplace pas). Les types quelconques doivent déjà exister
284 // un tableau dans tab_liQ correspondent aux grandeurs quelconque pour tous les pt integ,
285 // tab_liQ(i) pour le pt d'integ i
286 // liQ_travail: est une liste de travail qui sera utilisée dans le transfert

287 void TransfertAjoutAuNoeuds(const Tableau <List_io < TypeQuelconque > >& tab_liQ
288 ,List_io < TypeQuelconque > & liQ_travail,int cas);
289
290 // accumulation aux noeuds de grandeurs venant de l'éléments vers ses noeuds (exemple la
pression appliquée)
291 // autres que celles aux pti classiques, mais directements disponibles
292 // le contenu du conteneur stockées dans liQ est utilisé en variable intermédiaire
293 void Accumul_aux_noeuds(const List_io < Ddl_enum_etendu >& lietendu
294 ,List_io < TypeQuelconque > & liQ,int cas);
295
296
297 // modification de l'orientation de l'élément en fonction de cas_orientation
298 // =0: inversion simple (sans condition) de l'orientation
299 // si cas_orientation est diff de 0: on calcul le jacobien aux différents points d'intégration
300 // 1. si tous les jacobiens sont négatifs on change d'orientation
301 // 2. si tous les jacobiens sont positifs on ne fait rien
302 // 3. si certains jacobiens sont positifs et d'autres négatifs message
303 // d'erreur et on ne fait rien
304 // ramène true: s'il y a eu changement effectif, sinon false
305 bool Modif_orient_elem(int cas_orientation);
306
307 // calcul éventuel de la normale à un noeud
308 // ce calcul existe pour les éléments 2D, 1D axi, et aussi pour les éléments 1D
309 // qui possède un repère d'orientation
310 // en retour coor = la normale si coor.Dimension() est = à la dimension de l'espace
311 // si le calcul n'existe pas --> coor.Dimension() = 0
312 // ramène un entier :
313 // == 1 : calcul normal
314 // == 0 : problème de calcul -> coor.Dimension() = 0
315 // == 2 : indique que le calcul n'est pas licite pour le noeud passé en paramètre
316 // c'est le cas par exemple des noeuds extérieurs pour les éléments SFE
317 // mais il n'y a pas d'erreur, c'est seulement que l'élément n'est pas ad hoc pour
318 // calculer la normale à ce noeud là
319 // temps: indique à quel moment on veut le calcul
320 virtual int CalculNormale_noeud(Enum_dure temps, const Noeud& noe,Coordonnee& coor);
321
322 // calcul si un point est a l'interieur de l'element ou non
323 // il faut que M est la dimension globale
324 // retour : =0 le point est externe, =1 le point est interne ,
325 // = 2 le point est sur la frontière à la précision près
326 // coor_locales : s'il est différent de NULL, est affecté des coordonnées locales calculées,
327 // uniquement précises si le point est interne
328 int Interne(Enum_dure temps,const Coordonnee& M,Coordonnee* coor_locales=NULL);
329 // -- connaissances particulières sur l'élément
330 // ramène l'épaisseur de l'élément
331 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
332 virtual double Epaisseurs(Enum_dure , const Coordonnee& ) {return 0.;;};
333 // ramène l'épaisseur moyenne de l'élément (indépendante du point)
334 // =0. si la notion d'épaisseurs ne veut rien dire pour l'élément
335 virtual double EpaisseurMoyenne(Enum_dure ) {return 0.;;};
336 // ramène la section de l'élément
337 // =0. si la notion de section ne veut rien dire pour l'élément
338 virtual double Section(Enum_dure , const Coordonnee& ) {return 0.;;};
339 // ramène la section moyenne de l'élément (indépendante du point)
340 // =0. si la notion de section ne veut rien dire pour l'élément
341 virtual double SectionMoyenne(Enum_dure ) {return 0.;;};
342
343 // fonction a renseigner par les classes dérivées, concernant les répercussions
344 // éventuelles due à la suppression de tous les frontières
345 // nums_i : donnent les listes de frontières supprimées
346 virtual void Prise_en_compte_des_consequences_suppression_tous_frontieres();
347 // idem pour une frontière (avant qu'elle soit supprimée)
348 virtual void Prise_en_compte_des_consequences_suppression_une_frontiere(ElFrontiere* elemFront);
349
350 // ----- calcul de frontières -----
351
352
353 // ramène la frontière point
354 // éventuellement création des frontières points de l'element et stockage dans l'element

```



```

355 // si c'est la première fois sinon il y a seulement retour de l'elements
356 // a moins que le paramètre force est mis a true
357 // dans ce dernier cas la frontière effacée est recrée
358 // num indique le numéro du point à créer (numérotation EF)
359 virtual ElFrontiere* const Frontiere_points(int num,bool force);
360
361 // ramène la frontière linéique
362 // éventuellement création des frontières linéique de l'element et stockage dans l'element
363 // si c'est la première fois et en 3D sinon il y a seulement retour de l'elements
364 // a moins que le paramètre force est mis a true
365 // dans ce dernier cas la frontière effacée est recrée
366 // num indique le numéro de l'arête à créer (numérotation EF)
367 virtual ElFrontiere* const Frontiere_lineique(int num,bool force);
368
369 // ramène la frontière surfacique
370 // éventuellement création des frontières surfacique de l'element et stockage dans l'element
371 // si c'est la première fois sinon il y a seulement retour de l'elements
372 // a moins que le paramètre force est mis a true
373 // dans ce dernier cas la frontière effacée est recrée
374 // num indique le numéro de la surface à créer (numérotation EF)
375 virtual ElFrontiere* const Frontiere_surfacique(int num,bool force);
376
377 // ----- init éventuelle avant le chargement -----
378 // initialisation éventuelle, nécessaire avant d'appliquer l'ensemble des charges
379 // par exemple des stockages intermédiaires
380 virtual void Initialisation_avant_chargement() {};
381
382 //=====
383 protected :
384 //=====
385 // METHODES PROTEGEES utilisables par les classes derivees :
386
387 // Calcul des frontières de l'element
388 // creation des elements frontieres et retour du tableau de ces elements
389 // la création n'a lieu qu'au premier appel
390 // ou lorsque l'on force le paramètre force a true
391 // dans ce dernier cas seul les frontière effacées sont recrée
392 // cas :
393 // = 0 -> on veut toutes les frontières
394 // = 1 -> on veut uniquement les surfaces
395 // = 2 -> on veut uniquement les lignes
396 // = 3 -> on veut uniquement les points
397 // = 4 -> on veut les surfaces + les lignes
398 // = 5 -> on veut les surfaces + les points
399 // = 6 -> on veut les lignes + les points
400 Tableau <ElFrontiere*> const & Frontiere_elethermi(int cas, bool force = false);
401
402 // -----
403 // cas où l'on intègre que selon une liste (un axe, un plan, un volume)
404 // -----
405 // Calcul du residu local et de la raideur locale,
406 // pour le schema implicite d'ou a l'instant t + dt
407 // ddl represente les degres de liberte specifiques a l'element
408 // tabDepsBB = vitesse de déformation, tabDeltaEpsBB = incrément de def entre t et t+dt
409 // cald_Dvirtuelle = indique si l'on doit calculer la dérivée de la vitesse de déformation
virtuelle
410 void Cal_implicit (DdlElement & tab_ddl,Tableau <CoordonneeB >& d_gradTB
411 ,Tableau < Tableau2 <CoordonneeB> * > d2_gradTB,Tableau <CoordonneeH >& d_fluxH,int
nbint
412 ,const Vecteur& poids,const ParaAlgoControle & pa,bool cald_DGradTvirtuelle);
413
414 // Calcul du residu local a l'instant t ou tdt
415 // atdt = true : calcul à tdt, valeur par défaut
416 // = false: calcul à t
417 // ddl represente les degres de liberte specifiques a l'element
418 // nbint = nb de pt d'integration , poids = poids d'integration
419 void Cal_explicit (DdlElement & ddl,Tableau <CoordonneeB >& d_gradTB,int nbint
420 ,const Vecteur& poids,const ParaAlgoControle & pa,bool atdt=true);
421
422 // Calcul de la matrice géométrique et de la matrice initiale
423 // cette fonction est éventuellement appelée par les classes dérivées
424 // ddl represente les degres de liberte specifiques a l'element
425 // nbint = nb de pt d'integration , poids = poids d'integration
426 // cald_Dvirtuelle = indique si l'on doit calculer la dérivée de la vitesse de déformation
virtuelle
427 void Cal_matGeom_Init (Mat_pleine & matGeom, Mat_pleine & matInit
428 ,DdlElement & ddl,Tableau <CoordonneeB >& d_gradTB,Tableau < Tableau2 <CoordonneeB> * >
d2_gradTB
429 ,Tableau <CoordonneeH>& d_fluxH,int nbint,const Vecteur& poids
430 ,const ParaAlgoControle & pa,bool cald_Dvirtuelle);
431
432 // Calcul de la matrice masse selon différent choix donné par type_matrice_masse,
433 // a l'instant initial.
434 void Cal_Mat_masse (DdlElement & tab_ddl,Enum_calcul_masse type_matrice_masse,
435 int nbint,const Tableau <Vecteur>& taphi,int nbne
436 ,const Vecteur& poids);
437

```

```

438 // -----
439 // cas où l'on intègre selon deux listes (ex un axe et un plan, etc.)
440 // -----
441 // Calcul du residu local et de la raideur locale,
442 // pour le schema implicite d'ou a l'instant t + dt
443 // ddl represente les degres de liberte specifiques a l'element
444 void Cal_implicitap (DdlElement & tab_ddl,Tableau <TenseurBB *> & d_epsBB
445 ,Tableau < Tableau2 <CoordonneeB> * > d2_gradTB,Tableau <CoordonneeH>& d_fluxH
446 ,int nbint1,Vecteur& poids1,int nbint2,const Vecteur& poids2
447 ,const ParaAlgoControle & pa);
448
449 // Calcul du residu local a l'instant t ou tdt
450 // atdt = true : calcul à tdt, valeur par défaut
451 // = false: calcul à t
452 // ddl represente les degres de liberte specifiques a l'element
453 // d_epsbb = variation des def
454 // nbint = nb de pt d'integration , poids = poids d'integration
455 void Cal_explicitap (DdlElement & ddl,Tableau <TenseurBB *>& d_epsBB
456 ,int nbint,const Vecteur& poids,bool atdt=true);
457
458 // Calcul de la matrice géométrique et de la matrice initiale
459 // cette fonction est éventuellement appelée par les classes dérivées
460 // ddl represente les degres de liberte specifiques a l'element
461 // d_epsbb = variation des def
462 // nbint = nb de pt d'integration , poids = poids d'integration
463 void Cal_matGeom_Initap (Mat_pleine & matGeom, Mat_pleine & matInit
464 ,DdlElement & ddl,Tableau <TenseurBB *>& d_epsBB,Tableau < Tableau2 <CoordonneeB> * >
d2_gradTB
465 ,Tableau <CoordonneeH>& d_fluxH,int nbint,const Vecteur& poids);
466
467 // ----- calcul de second membre -----
468 // calcul des seconds membres suivant les chargements
469 // cas d'un chargement surfacique, sur les frontieres des éléments
470 // force indique la force surfacique appliquée
471 // retourne le second membre résultant
472 // nSurf : le numéro de la surface externe
473 // calcul à l'instant tdt ou t en fonction de la variable atdt
474 Vecteur& SM_charge_surf_E (DdlElement & ddls,int nSurf
475 ,const Tableau <Vecteur>& taphi,int nbne
476 ,const Vecteur& poids,const Coordonnee& force
477 ,const ParaAlgoControle & pa,bool atdt=true);
478 // idem SM_charge_surf_E mais -> implicite,
479 // pa : permet de déterminer si l'on fait ou non le calcul de la contribution à la raideur
480 // retourne le second membre et la matrice de raideur correspondant
481 Element::ResRaid SMR_charge_surf_I (DdlElement & ddls,int nSurf
482 ,const Tableau <Vecteur>& taphi,int nbne
483 ,const Vecteur& poids,const Coordonnee& force
484 ,const ParaAlgoControle & pa);
485 // calcul des seconds membres suivant les chargements
486 // cas d'un chargement pression, sur les frontieres des éléments
487 // pression indique la pression appliquée
488 // retourne le second membre résultant
489 // nSurf : le numéro de la surface externe
490 // calcul à l'instant tdt ou t en fonction de la variable atdt
491 Vecteur& SM_charge_pres_E (DdlElement & ddls,int nSurf
492 ,const Tableau <Vecteur>& taphi,int nbne
493 ,const Vecteur& poids,double pression
494 ,const ParaAlgoControle & pa,bool atdt=true);
495 // idem SM_charge_pres_E mais -> implicite,
496 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
497 // retourne le second membre et la matrice de raideur correspondant
498 Element::ResRaid SMR_charge_pres_I (DdlElement & ddls,int nSurf
499 ,const Tableau <Vecteur>& taphi,int nbne
500 ,const Vecteur& poids,double pression
501 ,const ParaAlgoControle & pa);
502 // cas d'un chargement lineique, sur les arêtes frontieres des éléments
503 // force indique la force lineique appliquée
504 // retourne le second membre résultant
505 // nArete : le numéro de l'arête externe
506 // calcul à l'instant tdt ou t en fonction de la variable atdt
507 Vecteur& SM_charge_line_E (DdlElement & ddls,int nArete
508 ,const Tableau <Vecteur>& taphi,int nbne
509 ,const Vecteur& poids,const Coordonnee& force
510 ,const ParaAlgoControle & pa,bool atdt=true);
511 // idem SM_charge_line_E mais -> implicite,
512 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
513 // retourne le second membre et la matrice de raideur correspondant
514 Element::ResRaid SMR_charge_line_I (DdlElement & ddls,int nArete
515 ,const Tableau <Vecteur>& taphi,int nbne
516 ,const Vecteur& poids,const Coordonnee& force
517 ,const ParaAlgoControle & pa);
518
519 // cas d'un chargement lineique suiveur, sur les arêtes frontieres des éléments
520 // pas valable pour des éléments 3D !
521 // force indique la force lineique appliquée
522 // retourne le second membre résultant
523 // nArete : le numéro de l'arête externe

```

```

524 // calcul à l'instant tdt ou t en fonction de la variable atdt
525 Vecteur& SM_charge_line_Suiv_E (DdlElement & ddls,int nArete
526     ,const Tableau <Vecteur>& taphi,int nbne
527     ,const Vecteur& poids,const Coordonnee& force
528     ,const ParaAlgoControle & pa,bool atdt=true);
529 // idem SM_charge_line_E mais -> implicite,
530 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
531 // retourne le second membre et la matrice de raideur correspondant
532 Element::ResRaid SMR_charge_line_Suiv_I (DdlElement & ddlA,int nArete
533     ,const Tableau <Vecteur>& taphi,int nbne
534     ,const Vecteur& poids,const Coordonnee& force
535     ,const ParaAlgoControle & pa);
536
537
538 // cas d'un chargement surfacique suiveur, sur les surfaces de l'élément
539 // la direction varie selon le système suivant: on définit les coordonnées matérielles
540 // de la direction, ce qui sert ensuite à calculer les nouvelles directions. L'intensité
541 // elle est constante.
542 // force indique la force surfacique appliquée
543 // retourne le second membre résultant
544 // nSurf : le numéro de la surface externe
545 // calcul à l'instant tdt ou t en fonction de la variable atdt
546 Vecteur& SM_charge_surf_Suiv_E (DdlElement & ddls,int nSurf
547     ,const Tableau <Vecteur>& taphi,int nbne
548     ,const Vecteur& poids,const Coordonnee& force
549     ,const ParaAlgoControle & pa,bool atdt=true);
550 // idem SM_charge_surf_Suiv_E mais -> implicite,
551 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
552 // retourne le second membre et la matrice de raideur correspondant
553 Element::ResRaid SMR_charge_surf_Suiv_I (DdlElement & ddlA,int nSurf
554     ,const Tableau <Vecteur>& taphi,int nbne
555     ,const Vecteur& poids,const Coordonnee& force
556     ,const ParaAlgoControle & pa);
557
558 // cas d'un chargement volumique, sur l'élément
559 // force indique la force volumique appliquée
560 // retourne le second membre résultant
561 // calcul à l'instant tdt ou t en fonction de la variable atdt
562 Vecteur& SM_charge_vol_E (DdlElement & ddls
563     ,const Tableau <Vecteur>& taphi,int nbne
564     ,const Vecteur& poids,const Coordonnee& force
565     ,const ParaAlgoControle & pa,bool atdt=true);
566
567 // idem SM_charge_vol_E mais -> implicite,
568 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
569 // retourne le second membre et la matrice de raideur correspondant
570 void SMR_charge_vol_I (DdlElement & ddls
571     ,const Tableau <Vecteur>& taphi,int nbne
572     ,const Vecteur& poids,const Coordonnee& force
573     ,const ParaAlgoControle & pa);
574
575 // cas d'un chargement hydrostatique, sur les surfaces de l'élément
576 // la charge dépend de la hauteur à la surface libre du liquide déterminée par un point
577 // et une direction normale à la surface libre:
578 // nSurf : le numéro de la surface externe
579 // poidvol: indique le poids volumique du liquide
580 // M_liquide : un point de la surface libre
581 // dir_normal_liquide : direction normale à la surface libre
582 // retourne le second membre résultant
583 // calcul à l'instant tdt ou t en fonction de la variable atdt
584 Vecteur& SM_charge_hydro_E (DdlElement & ddls,int nSurf
585     ,const Tableau <Vecteur>& taphi,int nbne
586     ,const Vecteur& poids
587     ,const Coordonnee& dir_normal_liquide,const double& poidvol
588     ,const Coordonnee& M_liquide
589     ,const ParaAlgoControle & pa,bool atdt=true);
590 // idem SM_charge_hydro_E mais -> implicite,
591 // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
592 // retourne le second membre et la matrice de raideur correspondant
593 Element::ResRaid SMR_charge_hydro_I (DdlElement & ddlA,int nSurf
594     ,const Tableau <Vecteur>& taphi,int nbne
595     ,const Vecteur& poids
596     ,const Coordonnee& dir_normal_liquide,const double& poidvol
597     ,const Coordonnee& M_liquide
598     ,const ParaAlgoControle & pa);
599
600 // cas d'un chargement aero-hydrodynamique, sur les frontières de l'élément
601 // Il y a trois forces: une suivant la direction de la vitesse: de type traînée aerodynamique
602 // Fn = poids_volu * fn(V) * S * (normale*u) * u, u étant le vecteur directeur de V (donc
unitaire)
603 // une suivant la direction normale à la vitesse de type portance
604 // Ft = poids_volu * ft(V) * S * (normale*u) * w, w unitaire, normal à V, et dans le plan n et V
605 // une suivant la vitesse tangente de type frottement visqueux
606 // T = to(Vt) * S * ut, Vt étant la vitesse tangentielle et ut étant le vecteur directeur de Vt
607 // retourne le second membre résultant
608 // calcul à l'instant tdt ou t en fonction de la variable atdt
609 // coef_mul: est un coefficient multiplicateur global (de tout)

```

```

610     Vecteur& SM_charge_hydrodyn_E (const double& poidvol,const Tableau <Vecteur>& taphi,int nbne
611     ,CourbelD* frot_fluid,const Vecteur& poids
612     ,CourbelD* coef_aero_n,int numfront,const double& coef_mul
613     ,CourbelD* coef_aero_t,const ParaAlgoControle & ,bool
        atdt=true);
614     // idem SM_charge_hydrodyn_E mais -> implicite,
615     // pa: permet de déterminer si oui ou non on calcul la contribution à la raideur
616     // retourne le second membre et la matrice de raideur correspondant
617     Element::ResRaid SM_charge_hydrodyn_I (const double& poidvol,const Tableau <Vecteur>& taphi,int
        nbne
618     ,CourbelD* frot_fluid,const Vecteur& poids,DdlElement & ddls
619     ,CourbelD* coef_aero_n,int numfront,const double& coef_mul
620     ,CourbelD* coef_aero_t,const ParaAlgoControle & pa);
621
622 // ----- calcul de frontières en protected -----
623
624     // --- fonction nécessaire pour la construction des Frontières linéiques ou surfaciques
        particulière à l'élément
625     // adressage des frontières linéiques et surfacique
626     // définit dans les classes dérivées, et utilisées pour la construction des frontières
627     virtual ElFrontiere* new_frontiere_lin(int num,Tableau <Noeud *> & tab, DdlElement& ddelem) = 0;
628     virtual ElFrontiere* new_frontiere_surf(int num,Tableau <Noeud *> & tab, DdlElement& ddelem) =
        0;
629
630 // ----- stabilisation d'hourglass -----
631     // calcul d'élément de contrôle d'hourglass associée à un comportement
632     void Init_hourglass_comp(const ElemGeomC0& elgeHour, const string & str_precision
633     ,LoiAbstraiteGeneral * loiHourglass,const BlocGen & bloc);
634
635     // stabilisation pour un calcul implicite
636     void Cal_implicit_hourglass();
637
638     // stabilisation pour un calcul explicite
639     void Cal_explicit_hourglass(bool atdt);
640
641     public :
642     // récupération de l'énergie d'hourglass éventuelle
643     double Energie_Hourglass();
644     // idem pour l'énergie et la puissance de bulk viscosity
645     double Energie_Bulk() const {return E_elem_bulk_tdt;};
646     double Puissance_Bulk() const {return P_elem_bulk;};
647
        protected :
648 // ----- calcul d'erreur, remontée des Flux -----
649
650     // calcul du résidu et de la matrice de raideur pour le calcul d'erreur
651     // cas d'une intégration suivant une seule liste
652     void FluxAuNoeud_ResRaid(const int nbne,const Tableau <Vecteur>& taphi
653     ,const Vecteur& poids,Tableau <Vecteur *>& resErr,Mat_pleine& raidErr
654     ,const Tableau <Vecteur>& taphiEr,const Vecteur& poidsEr );
655
656     // calcul de l'erreur sur l'élément. Ce calcul n'est disponible
657     // qu'une fois la remontée aux Flux effectuées sinon aucune
658     // action. pour les autres arguments de retour, idem ErreurElement
659     // qui est la fonction generique, les autres variables sont spécifiques
660     // a l'element.
661     void Cal_ErrElem(int type,double& errElemRelative
662     ,double& numerateur, double& denominateur,const int nbne,const Tableau <Vecteur>& taphi
663     ,const Vecteur& poids,const Tableau <Vecteur>& taphiEr,const Vecteur& poidsEr);
664
665     // calcul de l'erreur aux noeuds. Contrairement au cas des Flux
666     // seul le résidu est calculé. Cas d'une intégration suivant une liste
667     void Cal_ErrAuxNoeuds(const int nbne, const Tableau <Vecteur>& taphi,
668     const Vecteur& poids,Tableau <Vecteur *>& resErr );
669
670
671 // -----
672     // affichage dans la sortie transmise, des variables duales "nom"
673     // aux differents points d'integration
674     // dans le cas ou nom est vide, affichage de "toute" les variables
675     // cas = 1 -> premier passage pour de l'implicit
676     // cas = 2 -> premier passage pour de l'explicit
677     // cas = 11 -> passage autre que le premier pour de l'implicit
678     // cas = 12 -> passage autre que le premier pour de l'explicit
679     void VarDualSort(ofstream& sort, Tableau<string>& nom,int nbint,int cas);
680     // utilitaires de VarDualSort
681     // affiche en fonction d'indic les differentes variables et appel
682     // AffDefContiD en fonction de la dimension i
683     //
684     void AffDefCont( ofstream& sort,CompThermoPhysiqueAbstraite::SaveResul * saveDon,
685     CoordonneeB& gradTB,Coordonnee& gradT,double& norme_gradT,
686     CoordonneeB& DgradTB,Coordonnee& DgradT,double& norme_dGradT,
687     CoordonneeH& fluxDH,Coordonnee& fluxD,double& norme_flux,
688     double& temperature, int indic);
689
        // cas 1D
690     void AffDefCont1D( ofstream& sort,CompThermoPhysiqueAbstraite::SaveResul * saveDon,
691     CoordonneeB& gradTB,Coordonnee& gradT,double& norme_gradT,
692     CoordonneeB& DgradTB,Coordonnee& DgradT,double& norme_dGradT,

```

```

693         CoordonneeH& fluxDH,Coordonnee & fluxD,double& norme_flux,
694         double& temperature, int indic);
695     // cas 2D
696 void AffDefCont2D( ofstream& sort,CompThermoPhysiqueAbstraite::SaveResul * saveDon,
697         CoordonneeB& gradTB,Coordonnee& gradT,double& norme_gradT,
698         CoordonneeB& DgradTB,Coordonnee& DgradT,double& norme_dGradT,
699         CoordonneeH& fluxDH,Coordonnee & fluxD,double& norme_flux,
700         double& temperature, int indic);
701     // cas 3D
702 void AffDefCont3D( ofstream& sort,CompThermoPhysiqueAbstraite::SaveResul * saveDon,
703         CoordonneeB& gradTB,Coordonnee& gradT,double& norme_gradT,
704         CoordonneeB& DgradTB,Coordonnee& DgradT,double& norme_dGradT,
705         CoordonneeH& fluxDH,Coordonnee & fluxD,double& norme_flux,
706         double& temperature, int indic);
707
708
709 // -----
710
711 // --- méthodes relatives aux calculs d'erreurs -----
712 // ---- utilisées par les classes dérivées -----
713 // ajout des ddl relatif aux Flux pour les noeuds de l'élément
714 void Ad_ddl_Flux(const DdlElement& tab_ddlErr);
715 // inactive les ddl du problème primaire de mécanique
716 void Inact_ddl_primaire(DdlElement& tab_ddl);
717 // active les ddl du problème primaire de mécanique
718 void Act_ddl_primaire(DdlElement& tab_ddl);
719 // inactive les ddl du problème de recherche d'erreur : les Flux
720 void Inact_ddl_Flux(DdlElement& tab_ddlErr);
721 // active les ddl du problème de recherche d'erreur : les Flux
722 void Act_ddl_Flux(DdlElement& tab_ddlErr);
723 // active le premier ddl du problème de recherche d'erreur : FLUX
724 void Act_premier_ddl_Flux();
725
726 // -----
727 // lecture des Flux sur le flot d'entrée
728 void LectureDesFlux
729     (bool cas,UtilLecture * entreePrinc,Tableau <CoordonneeH *>& tabfluxH);
730
731 // retour des Flux en absolu retour true si ils existent sinon false
732 void FluxEnAbsolues
733     (bool cas,Tableau <CoordonneeH *>& tabfluxH,Tableau <Vecteur>& tabflux);
734
735
736 // ----- lecture écriture dans base info -----
737 // programmes utilisés par les classes dérivées
738
739 // cas donne le niveau de la récupération
740 // = 1 : on récupère tout
741 // = 2 : on récupère uniquement les données variables (supposées comme telles)
742 void Lecture_bas_inf
743     (ifstream& ent,const Tableau<Noeud *> * tabMaillageNoeud,const int cas) ;
744 // cas donne le niveau de sauvegarde
745 // = 1 : on sauvegarde tout
746 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
747 void Ecriture_bas_inf(ofstream& sort,const int cas) ;
748
749 // ----- utilitaires pour la dynamique
750 // calcul de la longueur d'arrête de l'élément minimal
751 // divisé par la célérité la plus rapide dans le matériau
752 // appelé par les classes dérivées
753 // nb_noeud : =0 indique que l'on utilise tous les noeuds du tableau de noeuds
754 // = un nombre > 0, indique le nombre de noeuds à utiliser au début du tableau
755 double Interne_Long_arrete_mini_sur_c(Enum_dure temps,int nb_noeud=0);
756
757 // recuperation des coordonnées du point d'intégration numéro = iteg pour
758 // la grandeur enu
759 // temps: dit si c'est à 0 ou t ou tdt
760 // si erreur retourne erreur à true
761 Coordonnee CoordPtInt(Enum_dure temps,Enum_ddl enu,int iteg,bool& erreur);
762
763 // recuperation des coordonnées du point d'intégration numéro = iteg pour
764 // la face : face
765 // temps: dit si c'est à 0 ou t ou tdt
766 // si erreur retourne erreur à true
767 Coordonnee CoordPtIntFace(int face, Enum_dure temps,int iteg,bool& erreur);
768 // recuperation des coordonnées du point d'intégration numéro = iteg pour
769 // la face : face
770 // temps: dit si c'est à 0 ou t ou tdt
771 // si erreur retourne erreur à true
772 Coordonnee CoordPtIntArête(int arête, Enum_dure temps,int iteg,bool& erreur);
773
774
775 // procedure permettant de completer l'element, en ce qui concerne les variables gérés
776 // par ElemThermi, apres sa creation avec les donnees du bloc transmis
777 // peut etre appeler plusieurs fois
778 Element* Complete_ElemThermi(BlocGen & bloc,LesFonctions_nD* lesFonctionsnD);
779

```

```

780 // retourne le numero du pt d'ing le plus près ou est exprimé la grandeur enum
781 // temps: dit si c'est à 0 ou t ou tdt
782 int PtLePlusPres(Enum_dure temps,Enum_ddl enu, const Coordonnee& M);
783
784 // récupération des valeurs au numéro d'ordre = iteg pour
785 // les grandeur enu
786 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
787 Tableau <double> Valeur_multi(bool absolue,Enum_dure enu_t,const List_io<Ddl_enum_etendu>& enu
788 ,int iteg,int cas ) ;
789
790 // récupération des valeurs Tensorielles (et non scalaire comme avec Valeur_multi)
791 // au numéro d'ordre = iteg pour les grandeur enu
792 // enu contient les grandeurs de retour
793 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base
particulière
794 void Valeurs_Tensorielles(bool absolue,Enum_dure enu_t,List_io<TypeQuelconque>& enu
795 ,int iteg,int cas ) ;
796
797 // ==== »» methodes virtuelles definis dans les classes mères =====
798 // ramene l'element geometrique correspondant au ddl passé en paramètre
799 virtual ElemGeomCO& ElementGeometrie(Enum_ddl ddl) const ;
800 // ramène le nombre de grandeurs génératrices pour un pt d'integ, correspondant à un type
enuméré
801 // peut-être surchargé pour des éléments particuliers
802 virtual int NbGrandeurGene(Enum_ddl enu) const;
803
804 // ==== »» methodes virtuelles definis dans les classes dérivées =====
805 // ramene la dimension des vecteurs flux et gradient de température de l'élément
806 virtual int Dim_flux_gradT() const = 0;
807
808 // VARIABLES PROTEGEES :
809
810 //---- variables: Flux, déformations etc.. aux points d'intégrations -----
811 // lesPtIntegThermiInterne pointe vers une entité entièrement gérés par les classes dérivées
812 // contenant les grandeurs mécaniques (Flux, déformations etc.) stockées aux points
d'intégration
813 LesPtIntegThermiInterne * lesPtIntegThermiInterne;
814
815 Loi_comp_abstraite * loiComp; // loi de comportement mécanique defini dans les classes derivees
816 CompThermoPhysiqueAbstraite * loiTP; // éventuellement une loi de comportement thermo physique
817 // defini dans les classes derivees
818 CompFrotAbstraite * loiFrot; // éventuellement une loi de comportement au frottement pour ses
frontières
819
820 bool dilatation; // indique si oui ou non on tiend compte de la dilatation thermique
821
822 Met_abstraite * met; // definition spécifique dans les classes derivees
823 Deformation * def; // definition spécifique dans les classes derivees
// mais relative à la déformation mécanique
824 Deformation * defEr; // idem que def mais pour la remonte aux Flux
825 Tableau <Deformation *> defSurf; // idem mais pour les déformations des surfaces
826 // externes (frontières) si cela est nécessaire
827 Tableau <Deformation *> defArete; // idem mais pour les déformations des arretes
828 // externes (frontières) si cela est nécessaire
829 Deformation * defMas; // idem que def mais pour le calcul de la matrice masse
830 double* fluxErreur; // erreur sur l'élément dans le calcul des densités de flux;
831 Tableau <Loi_comp_abstraite::SaveResul *> tabSaveDon; // donnée particulière à la loi mécanique
832 Tableau <CompThermoPhysiqueAbstraite::SaveResul *> tabSaveTP; // donnée parti à la loi thermo
physique
833 Tableau <Deformation::SaveDefResul *> tabSaveDefDon; // donnée particulière pour la déf
834 Tableau <EnergieThermi > tab_energ,tab_energ_t; //les différentes énergies mécaniques mises en
jeux
835 // dimensionné dans les classes dérivées, a t+tdt, et a t
836 EnergieThermi energie_totale; // le bilan sur l'élément de l'énergie
837 bool premier_calcul_thermi_impli_expli; // au premier calcul soit en implicite, explicite, mat
geom
838 // toutes les grandeurs à t=0 de la métrique sont calculées et stockées dans la déformation
839 // ensuite elles ne sont plus calculées -> premier_calcul_thermi_impli_expli sert pour
l'indiquer
840 double masse_volumique; // masse volumique de l'élément
841 //--- stabilisation d'hourglass éventuelle
842 Enum_StabHourglass type_stabHourglass; // méthode de stabilisation
843 double E_Hourglass;
844
845 private:
846 Tableau <ElemThermi*> tab_elHourglass; // tableau de pointeurs éventuels sur des éléments
servant à la stabilisation
847 // les éléments sont définies au travers de l'appel à la méthode
Cal_hourglass_comp, par une classe dérivée,
848 double coefStabHourglass ; // coef de stabilisation
849 Mat_pleine* raid_hourglass_transitoire; // raideur transitoire d'hourglass, éventuellement
définie
850
851 // ----- pour faciliter les routines Interne_t _0 et _tdt + paramètres de contrôle ----
852 static const Coordonnee & (Met_abstraite::*PointM)
853 (const Tableau<Noeud *>& tab_noeud,const Vecteur& phi);

```





```

40 pour l'instant en essai pour une classe générale au dessus du solide et des fluides
41 {
42     public :
43         // VARIABLES PUBLIQUES :
44
45         // CONSTRUCTEURS :
46         ElemMeca ();
47         // Constructeur utile quand le numero d'identification de l'element est connu
48         ElemMeca (int num_id) ;
49         // Constructeur utile quand le numero et le tableau des noeuds
50         // de l'element sont connus
51         ElemMeca (int num_id,const Tableau<Noeud *>& tab);
52         // Constructeur utile quand le numero d'identification est connu,
53         // ainsi que la geometrie et le type d'interpolation de l'element
54         ElemMeca (int num_id,Enum_interpol id_interp_elt,Enum_geom id_geom_elt);
55         // Constructeur utile quand le numero d'identification est connu,
56         // ainsi que la geometrie et le type d'interpolation de l'element
57         ElemMeca (int num_id,char* nom_interpol,char* nom_geom);
58         // Constructeur utile quand toutes les donnees de la classe Element sont connues
59         ElemMeca (int num_id,const Tableau<Noeud *>& tab,Enum_interpol id_interp_elt,
60                 Enum_geom id_geom_elt);
61         // Constructeur utile quand toutes les donnees de la classe Element sont connues
62         ElemMeca (int num_id,const Tableau<Noeud *>& tab,char* nom_interpol,
63                 char* nom_geom);
64         // Constructeur de copie
65         ElemMeca ( ElemMeca& elt);
66
67         // DESTRUCTEUR :
68         ~ElemMeca ();
69
70         // METHODES PUBLIQUES :
71         // test si l'element est complet
72         // = 1 tout est ok, =0 element incomplet
73         int TestCompleet();
74
75         // calcul si un point est a l'interieur de l'element ou non
76         // condition d'utilisation : il faut que la dimension de M soit
77         // celle de l'element, c'a-dire si dim(M) = 3 il faut que l'element soit 3D
78         // si dim(M) = 2, l'element doit etre surfacique et si dim(M) = 1 il faut un
79         // element geometrique de type FrontPointF
80         // il faut egalement que M est la dimension globale
81         // les trois fonctions sont pour l'etude a t=0, t et tdt
82         bool Interne_0(Coordonnee& M);
83         bool Interne_t(Coordonnee& M);
84         bool Interne_tdt(Coordonnee& M);
85
86     protected :
87         // METHODES PROTEGEES utilisables par les classes derivees :
88
89         // Calcul du residu local et de la raideur locale,
90         // pour le schema implicite d'ou a l'instant t + dt
91         // ddl represente les degres de liberte specifiques a l'element
92         void Cal_implicit (DdlElement & tab_ddl,Tableau <TenseurBB *>& tabEpsBB,
93                 Tableau <TenseurBB *> & d_epsBB,Tableau <TenseurHH *>& tabSigHH,
94                 int nbint,Vecteur& poids);
95
96         // Calcul du residu local a l'instant t
97         // pour le schema explicite
98         // ddl represente les degres de liberte specifiques a l'element
99         // epsBB = deformation, sigHH = contrainte, d_epsbb = variation des def
100        // nbint = nb de pt d'integration , poids = poids d'integration
101        void Cal_explicit (DdlElement & ddl,Tableau <TenseurBB *>& tabEpsBB,
102                Tableau <TenseurBB *>& d_epsBB,Tableau <TenseurHH *>& tabSigHH,
103                int nbint,Vecteur& poids);
104
105        // affichage dans la sortie transmise, des variables duales "nom"
106        // aux differents points d'integration
107        // dans le cas ou nom est vide, affichage de "toute" les variables
108        // cas =1 -> premier passage pour de l'implicit
109        // cas = 2 -> premier passage pour de l'explicit
110        // cas = 11 -> passage autre que le premier pour de l'implicit
111        // cas = 12 -> passage autre que le premier pour de l'explicit
112        void VarDualSort(ofstream& sort, Tableau<string>& nom,
113                Tableau <TenseurHH *>& tabSigHH,Tableau <TenseurBB *> & tabEpsBB,
114                int nbint,int cas);
115        // utilitaires de VarDualSort
116        // affiche en fonction d'indic les differentes variables et appel
117        // AffDefContiD en fonction de la dimension i
118        //
119        void AffDefCont( ofstream& sort,Loi_comp_abstraite::SaveResul * saveDon,
120                TenseurBB& eps0BB,TenseurBB& epsBB,
121                TenseurHH& sigHH,
122                TenseurHB& epsHB,TenseurHB& sigHB,
123                Vecteur& valPropreEps,Vecteur& valPropreSig,
124                int indic);
125        // cas 1D
126        void AffDefCont1D( ofstream& sort,Loi_comp_abstraite::SaveResul * saveDon,

```



```

127         TenseurBB& eps0BB, TenseurBB& epsBB,
128         TenseurHH& sigHH,
129         TenseurHB& epsHB, TenseurHB& sigHB,
130         Vecteur& valPropreEps, Vecteur& valPropreSig,
131         int indic);
132     // cas 2D
133     void AffDefCont2D( ofstream& sort, Loi_comp_abstraite::SaveResul * saveDon,
134         TenseurBB& eps0BB, TenseurBB& epsBB,
135         TenseurHH& sigHH,
136         TenseurHB& epsHB, TenseurHB& sigHB,
137         Vecteur& valPropreEps, Vecteur& valPropreSig,
138         int indic);
139     // cas 3D
140     void AffDefCont3D( ofstream& sort, Loi_comp_abstraite::SaveResul * saveDon,
141         TenseurBB& eps0BB, TenseurBB& epsBB,
142         TenseurHH& sigHH,
143         TenseurHB& epsHB, TenseurHB& sigHB,
144         Vecteur& valPropreEps, Vecteur& valPropreSig,
145         int indic);
146
147
148     // recherche si un point est interne a un element
149     bool Interne(Coordonnee& M);
150
151     // VARIABLES PROTEGEES :
152     Loi_comp_abstraite * loiComp; // loi de comportement defini dans les classes derivees
153     Met_abstraite * met; // definition specifique dans les classes derivees
154     Deformation * def; // definition specifique dans les classes derivees
155     Tableau <Loi_comp_abstraite::SaveResul *> tabSaveDon; //
156     // pour faciliter les routines Interne_t _0 et _tdt
157     static Coordonnee & (Met_abstraite::*PointM) (Tableau<Noeud *>& tab_noeud, Vecteur& Phi);
158     static void (Met_abstraite::*BaseND)
159         (Tableau<Noeud *>& tab_noeud, Mat_pleine& dphi, BaseB& bB, BaseH& bH);
160
161 };
162
163
164
165 #endif

```

## 7.243 ExceptionsElemThermi.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           06/03/2023
31 *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:         Herezh++
35 *
36 *   *****
37 *   BUT:   Définir des classes d'exception pour la gestion d'erreur
38 *           concernant les éléments thermiques.
39 *
40 *   *****
41 *   VERIFICATION:
42 *

```

```

43 *      ! date !   auteur !           but           !      *
44 *      -----
45 *      !           !           !           !           !      *
46 *      $           $           $           $           $      *
47 *      ***** *
48 *      MODIFICATIONS:
49 *      ! date !   auteur !           but           !      *
50 *      -----
51 *      $           $           $           $           $      *
52 *****/
53 #ifndef EXCEPTIONSELEMTHERMI_H
54 #define EXCEPTIONSELEMTHERMI_H
55
56
57 /// @addtogroup groupe_des_elements_finis
58 /// @{
59 ///
60
61 // cas d'une erreur survenue à cause d'un jacobien négatif
62
63 class ErrJacobienNegatif_ElemThermi {};
64 /// @} // end of group
65
66
67 /// @addtogroup groupe_des_elements_finis
68 /// @{
69 ///
70
71 // cas d'une erreur survenue à cause d'une variation de jacobien trop grande
72
73 class ErrVarJacobienMini_ElemThermi {};
74 /// @} // end of group
75
76 #endif

```

## 7.244 LesPtlIntegThermilInterne.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *      DATE:           06/03/2023
31 *
32 *      AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *      PROJET:         Herezh++
35 *
36 *      $           $           $           $           $
37 *      *****
38 *      BUT: Classe pour stocker l'ensemble des informations aux points
39 *      d'intégration thermique
40 *
41 *      $           $           $           $           $
42 *      *****
43 *
44 *      VERIFICATION:
45 *      ! date !   auteur !           but           !      *
46 *      -----
47 *      !           !           !           !           !      *
48 *      $           $           $           $           $      *
49 *      ***** *

```

```

48 *      MODIFICATIONS:
49 *      ! date ! auteur ! but
50 *      -----
51 *
52 *
53 #ifndef LESPTINTEGHERMIINTERNE_H
54 #define LESPTINTEGHERMIINTERNE_H
55
56 #include "PtIntegThermiInterne.h"
57
58 /// @addtogroup Groupe_concernant_les_points_integration
59 /// @{
60 ///
61
62
63 class LesPtIntegThermiInterne
64 {
65     // surcharge de l'operator de lecture
66     friend istream & operator » (istream &, LesPtIntegThermiInterne &);
67     // surcharge de l'operator d'écriture
68     friend ostream & operator « (ostream &, const LesPtIntegThermiInterne &);
69
70 public :
71     // CONSTRUCTEURS :
72     // constructeur par défaut
73     LesPtIntegThermiInterne();
74     // constructeur fonction du nombre de points d'intégration et de la dimension des coordonnees
75     LesPtIntegThermiInterne(int nbpti, int dimcoor);
76     // constructeur de copie
77     LesPtIntegThermiInterne(const LesPtIntegThermiInterne & lespti);
78
79     // DESTRUCTEUR :
80     ~LesPtIntegThermiInterne();
81
82     // METHODES PUBLIQUES :
83     // Surcharge de l'operateur =
84     LesPtIntegThermiInterne& operator= ( const LesPtIntegThermiInterne& lespti);
85
86     // le tableau des grandeurs aux points d'intégration
87     // en lecture écriture
88     Tableau <PtIntegThermiInterne>& TabPtIntThermi() {return tabPtInt;};
89     // l'élément PtIntegThermiInterne de numéro i
90     PtIntegThermiInterne& operator () (int i) {return tabPtInt(i);};
91
92
93     // les tableaux des densité de flux
94     Tableau <CoordonneeH *>& TabfluxH() {return tabfluxH;}; // densité flux finale
95     Tableau <CoordonneeH *>& TabfluxH_t() {return tabfluxH_t;}; // densité flux au début de l'incrément
96     // nombre de points d'intégration
97     int NbPti() const {return tabPtInt.Taille();};
98     // changement de taille donc de nombre de points d'intégration
99     // fonction du nombre de points d'intégration et de la dimension des coordonnees
100    void Change_taille_PtIntegThermi(int nbpti, int dimtens);
101    // idem, mais les instances ajoutées ou retirées ont la même dimension de coordonnees que celles
102    // qui existent déjà
103    void Change_taille_PtIntegThermi(int nbpti);
104    // retour la dimension des vecteurs gérés
105    int DimCoord() const;
106    // actualisation des grandeurs actives de t+dt vers t, pour celles qui existent
107    // sous ces deux formes
108    void TdtversT();
109    // actualisation des grandeurs actives de t vers tdt, pour celles qui existent
110    // sous ces deux formes
111    void TversTdt();
112
113    //===== lecture écriture dans base info =====
114    // cas donne le niveau de la récupération
115    // = 1 : on récupère tout
116    // = 2 : on récupère uniquement les données variables (supposées comme telles)
117    void Lecture_base_info (ifstream& ent,const int cas);
118    // cas donne le niveau de sauvegarde
119    // = 1 : on sauvegarde tout
120    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
121    void Ecriture_base_info(ofstream& sort,const int cas);
122
123 protected:
124     // données protégées
125     // grandeurs aux points d'intégration
126     Tableau <PtIntegThermiInterne> tabPtInt;
127     // densité de flux groupées sous forme de tableau, qui pointent sur celles de tabPtThermiInt
128     Tableau <CoordonneeH *> tabfluxH; // densité flux finale
129     Tableau <CoordonneeH *> tabfluxH_t; // densité flux au début de l'incrément
130
131 };
132 /// @} // end of group
133
134 #endif

```

## 7.245 PtIntegThermiInterne.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           06/03/2023                               $   *
31 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)     $   *
32 *   PROJET:         Herezh++                                $   *
33 *   BUT: Classe pour stocker les informations aux points    $   *
34 *   d'intégration thermique                                $   *
35 *   *****
36 *   VERIFICATION:
37 *   ! date ! auteur ! but
38 *   ! ! !
39 *   *****
40 *   MODIFICATIONS:
41 *   ! date ! auteur ! but
42 *   ! ! !
43 *   *****
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *****/
53 #ifndef PTINTEGHERMIINTERNE_H
54 #define PTINTEGHERMIINTERNE_H
55
56 #include "Tenseur.h"
57 #include "Vecteur.h"
58 #include "Temps_CPU_HZpp.h"
59
60 /// @addtogroup Groupe_concernant_les_points_integration
61 /// @{
62 ///
63
64
65 class PtIntegThermiInterne
66
67 { // surcharge de l'operator de lecture
68   friend istream & operator » (istream &, PtIntegThermiInterne &);
69   // surcharge de l'operator d'écriture
70   friend ostream & operator « (ostream &, const PtIntegThermiInterne &);
71
72 public :
73   // CONSTRUCTEURS :
74   // constructeur par défaut
75   PtIntegThermiInterne();
76   // constructeur fonction de la dimension de tenseurs
77   PtIntegThermiInterne(int dimtens);
78   // constructeur de copie
79   PtIntegThermiInterne(const PtIntegThermiInterne& pti);
80
81   // DESTRUCTEUR :
82   ~PtIntegThermiInterne();
83
84   // METHODES PUBLIQUES :

```

```

85 // Surcharge de l'opérateur =
86 PtIntegThermiInterne& operator= ( const PtIntegThermiInterne& pti);
87
88 // la température
89 double& Temperature() {return temperature;};
90 // la température à t
91 double& Temperature_t() {return temperature_t;};
92 // gradient thermique finale
93 CoordonneeB& GradTB() {return gradTB;};
94 // vitesse finale du gradient thermique
95 CoordonneeB& DgradTB() {return dgradTB;};
96 // variation du gradient thermique entre t et t + delta t
97 CoordonneeB& DeltaGradTB() {return deltaGradTB;};
98 // vecteur densité du flux thermique finale
99 CoordonneeH& FluxH() {return fluxH;};
100 // vecteur densité du flux thermique en début d'incrément
101 CoordonneeH& FluxH_t() {return fluxH_t;};
102 // --- temps cpu
103 // tps cpu relatif à la métrique uniquement
104 Temps_CPU_HZpp& TpsMetrrique() {return tpsMetrrique;};
105 // temps cumulé relatif à la loi de comportement
106 Temps_CPU_HZpp& Tps_cpu_loi_comp() {return tps_cpu_loi_comp;};
107
108
109 // ---- acces idem en constants
110 // la température
111 const double& Temperature_const() const {return temperature;};
112 // la température à t
113 const double& Temperature_t_const() const {return temperature_t;};
114 // gradient thermique finale
115 const CoordonneeB & GradTB_const() const {return gradTB;};
116 // vitesse finale du gradient thermique
117 const CoordonneeB & DgradTB_const() const {return dgradTB;};
118 // variation du gradient thermique entre t et t + delta t
119 const CoordonneeB & DeltaGradTB_const() const {return deltaGradTB;};
120 // densité de flux finale
121 const CoordonneeH & FluxH_const() const {return fluxH;};
122 // densité de flux en début d'incrément
123 const CoordonneeH & FluxH_t_const() const {return fluxH_t;};
124 // --- temps cpu
125 // tps cpu relatif à la métrique uniquement
126 const Temps_CPU_HZpp& TpsMetrrique_const() const {return tpsMetrrique;};
127 // temps cumulé relatif à la loi de comportement
128 const Temps_CPU_HZpp& Tps_cpu_loi_comp_const() const {return tps_cpu_loi_comp;};
129
130
131
132 // invariant du gradient thermique
133 double& Norme_gradT() {return norme_gradT;};
134 const double& Norme_gradT_const() const {return norme_gradT;};
135 // invariant de la vitesse du gradient thermique
136 double& Norme_DGradT() {return norme_dGradT;};
137 const double& Norme_DGradT_const() const {return norme_dGradT;};
138 // invariant de la densité de flux
139 double& Norme_flux() {return norme_flux;};
140 const double& Norme_flux_const() const {return norme_flux;};
141
142 // actualisation des grandeurs actives de t+dt vers t, pour celles qui existent
143 // sous ces deux formes
144 void TdtversT();
145 // actualisation des grandeurs actives de t vers tdt, pour celles qui existent
146 // sous ces deux formes
147 void TversTdt();
148
149 //==== méthode particulière pour un passage de l'ordre 2D à 3D des tenseurs et l'inverse
150 //====
151 // plusZero: = true: indique qu'il faut compléter les grandeurs manquantes avec des 0
152 // = false: on ne complète pas
153 // il faut que ptintmec comporte des tenseurs d'ordre 2 et this des tenseurs 3D
154 void Affectation_2D_a_3D(const PtIntegThermiInterne& ptinther, bool plusZero);
155 // l'inverse: comme le conteneur d'arrivée est plus petit, il n'y a pas de complétion
156 // il faut que ptintmec comporte des tenseurs d'ordre 3 et this des tenseurs 2D
157 void Affectation_3D_a_2D(const PtIntegThermiInterne& ptinther);
158
159 //==== méthode particulière pour un passage de l'ordre 1D à 3D des tenseurs et l'inverse
160 //====
161 // plusZero: = true: indique qu'il faut compléter les grandeurs manquantes avec des 0
162 // = false: on ne complète pas
163 // il faut que ptintmec comporte des tenseurs d'ordre 1 et this des tenseurs 3D
164 void Affectation_1D_a_3D(const PtIntegThermiInterne& ptinther, bool plusZero);
165 // l'inverse: comme le conteneur d'arrivée est plus petit, il n'y a pas de complétion
166 // il faut que ptintmec comporte des tenseurs d'ordre 3 et this des tenseurs 1D
167 void Affectation_3D_a_1D(const PtIntegThermiInterne& ptinther);
168
169 //==== lecture écriture dans base info ====
170 // cas donne le niveau de la récupération
171 // = 1 : on récupère tout

```

```

170 // = 2 : on récupère uniquement les données variables (supposées comme telles)
171 void Lecture_base_info (ifstream& ent,const int cas);
172 // cas donne le niveau de sauvegarde
173 // = 1 : on sauvegarde tout
174 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
175 void Ecriture_base_info(ofstream& sort,const int cas);
176
177
178 protected :
179
180 // VARIABLES PROTÉGÉES :
181
182 double temperature,temperature_t;
183 CoordonneeB gradTB; // gradient thermique finale
184 CoordonneeB dgradTB; // vitesse finale du gradient thermique
185 CoordonneeB deltaGradTB; // variation du gradient thermique entre t et t + delta t
186 CoordonneeH fluxH; // vecteur densité du flux thermique finale
187 CoordonneeH fluxH_t; // vecteur densité du flux thermique en début d'incrément
188
189 // ---- les invariants
190 double norme_gradT ; // norme du gradient thermique = invariant
191 double norme_dGradT ; // norme de la vitesse du gradient thermique = invariant
192 double norme_flux ; // norme du vecteur l densité de flux thermique
193
194 // --- temps cpu
195 Temps_CPU_HZpp tpsMetricque; // tps cpu relatif à la métrique uniquement
196 Temps_CPU_HZpp tps_cpu_loi_comp; // temps cumulé relatif à la loi de comportement
197
198 };
199 /// @} // end of group
200
201 #endif

```

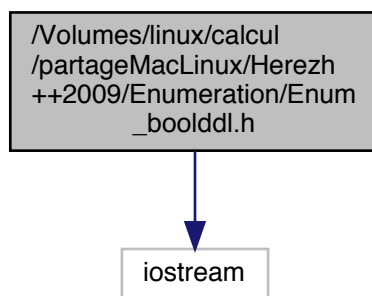
## 7.246 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/Enum\_boolddl.h

def de l'enuméré concernant les booléens ddl.

```
#include <iostream>
```

```
#include "Enum_boolddl.cc"
```

Graphe des dépendances par inclusion de Enum\_boolddl.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_boolddl` {  
`LIBRE = 0`, `FIXE`, `HSLIBRE`, `HSFIXE`,  
`SOUSLIBRE`, `SURFIXE`, `LISIBLE_LIBRE`, `LISIBLE_FIXE`,  
`LISIBLE_SOUSLIBRE`, `LISIBLE_SURFIXE`, `HS_LISIBLE_LIBRE`, `HS_LISIBLE_FIXE`,  
`HS_SOUSLIBRE`, `HS_SURFIXE`, `HS_LISIBLE_SOUSLIBRE`, `HS_LISIBLE_SURFIXE` }  
*Définition de l'enuméré concernant les booléens ddl.*

## Fonctions

- string `Nom_boolddl` (`Enum_boolddl id_ddl`)  
*Retourne le nom d'un degre de liberte a partir de son identificateur de type enumere id\_ddl correspondant.*
- `Enum_boolddl Id_nom_boolddl` (const string &nom\_ddl)  
*Retourne l'identificateur de type enumere associe au nom du degre de liberte nom\_ddl.*
- bool `Est_HS` (`Enum_boolddl id_ddl`)  
*dit si c'est HS ou pas ramène true si HS, false sinon*
- `istream &operator>>` (`istream &entree`, `Enum_boolddl &a`)  
*surcharge de l'operator de lecture*
- `ostream &operator<<` (`ostream &sort`, const `Enum_boolddl &a`)  
*surcharge de l'operator d'écriture*

### 7.246.1 Description détaillée

def de l'enuméré concernant les booléens ddl.

## 7.247 Enum\_boolddl.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_boolddl.h
2  \brief def de l'enuméré concernant les booléens ddl.
3  */
4  // FICHER : Enum_boolddl.h
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, et une vérification de type plus aisé qu' avec
36 // les entiers
37 // 10 caractere maxi
38
39
40 #ifndef ENUM_BOLDDDL_H
41 #define ENUM_BOLDDDL_H
42
43 // #include "Debug.h"
44 #include <iostream>

```

```

45 using namespace std;
46
47
48 /// @addtogroup Group_types_enumeres
49 /// @{
50
51 /// Définition de l'enuméré concernant les booléens ddl.
52
53 enum Enum_boolddl { LIBRE = 0, FIXE , HSLIBRE,HSFIXE,SOUSLIBRE,SURFIXE
54                   ,LISIBLE_LIBRE,LISIBLE_FIXE,LISIBLE_SOUSLIBRE, LISIBLE_SURFIXE
55                   ,HS_LISIBLE_LIBRE,HS_LISIBLE_FIXE
56                   ,HS_SOUSLIBRE , HS_SURFIXE , HS_LISIBLE_SOUSLIBRE ,
                    HS_LISIBLE_SURFIXE};
57 /// @} // end of group
58
59 /// Retourne le nom d'un degre de liberte a partir de son identificateur de
60 /// type enumere id_ddl correspondant
61 string Nom_boolddl ( Enum_boolddl id_ddl);
62
63 /// Retourne l'identificateur de type enumere associe au nom du degre de liberte
64 /// nom_ddl
65 Enum_boolddl Id_nom_boolddl (const string& nom_ddl);
66
67 /// dit si c'est HS ou pas
68 /// ramène true si HS, false sinon
69 bool Est_HS(Enum_boolddl id_ddl);
70
71 /// surcharge de l'operator de lecture
72 istream & operator » (istream & entree, Enum_boolddl& a);
73 /// surcharge de l'operator d'écriture
74 ostream & operator « (ostream & sort, const Enum_boolddl& a);
75
76 // pour faire de l'inline
77 #ifndef MISE_AU_POINT
78 #include "Enum_boolddl.cc"
79 #define Enum_boolddl_deja_inclus
80 #endif
81
82 #endif

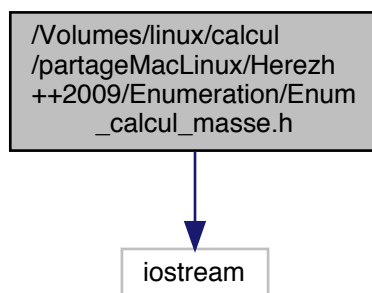
```

## 7.248 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/Enum\_calcul\_masse.h

def de l'enuméré concernant les types de calcul de masse

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_calcul\_masse.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :





## Énumérations

- enum `Enum_calcul_masse` { `MASSE_DIAG_COEF_EGAUX = 1` , `MASSE_DIAG_COEF_VAR` , `MASSE_←_CONSISTANTE` }  
*def de l'enuméré concernant les types de calcul de masse*

## Fonctions

- `char * Nom_calcul_masse (Enum_calcul_masse id_calcul_masse)`
- `Enum_calcul_masse Id_nom_calcul_masse (char *nom_calcul_masse)`
- `istream & operator>>` (`istream &entree`, `Enum_calcul_masse &a`)
- `ostream & operator<<` (`ostream &sort`, `const Enum_calcul_masse &a`)

### 7.248.1 Description détaillée

def de l'enuméré concernant les types de calcul de masse

Date

26/12/00

## 7.249 Enum\_calcul\_masse.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_calcul_masse.h
2  \brief def de l'enuméré concernant les types de calcul de masse
3  * \date      26/12/00
4  */
5  // FICHER : Enum_calcul_masse.h
6
7  // This file is part of the Herezh++ application.
8  //
9  // The finite element software Herezh++ is dedicated to the field
10 // of mechanics for large transformations of solid structures.
11 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
12 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
13 //
14 // Herezh++ is distributed under GPL 3 license ou ultérieure.
15 //
16 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
17 // AUTHOR : Gérard Rio
18 // E-MAIL : gerardrio56@free.fr
19 //
20 // This program is free software: you can redistribute it and/or modify
21 // it under the terms of the GNU General Public License as published by
22 // the Free Software Foundation, either version 3 of the License,
23 // or (at your option) any later version.
24 //
25 // This program is distributed in the hope that it will be useful,
26 // but WITHOUT ANY WARRANTY; without even the implied warranty
27 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
28 // See the GNU General Public License for more details.
29 //
30 // You should have received a copy of the GNU General Public License
31 // along with this program. If not, see <https://www.gnu.org/licenses/>.
32 //
33 // For more information, please consult: <https://herezh.irdl.fr/>.
34
35 /*****
36 *   DATE:      26/12/00
37 *
38 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
39 *
40 *   PROJET:    Herezh++
41 *
42 *   *****/
43 *   BUT: Enumeration des differents type de calcul de la masse.
44 *
45 *   *****
46 *
47 *   VERIFICATION:
48 *
49 *   ! date ! auteur ! but
50 *   -----
51 *
52 *   *****

```

```

53 *      MODIFICATIONS:
54 *      ! date ! auteur ! but
55 *      -----
56 *
57 *      $
58 *      *****/
59 #ifndef ENUM_CALCUL_MASSE_H
60 #define ENUM_CALCUL_MASSE_H
61
62 #include <iostream>
63 using namespace std;
64
65 /// @addtogroup Group_types_enumeres
66 /// @
67 ///
68
69 /// def de l'enuméré concernant les types de calcul de masse
70
71 enum Enum_calcul_masse { MASSE_DIAG_COEF_EGAUX = 1, MASSE_DIAG_COEF_VAR , MASSE_CONSISTANTE };
72 /// @ // end of group
73
74
75 // Retourne un nom de type de calcul_masse a partir de son identificateur de
76 // type enumere id_calcul_masse correspondant
77 char* Nom_calcul_masse (Enum_calcul_masse id_calcul_masse);
78
79 // Retourne l'identificateur de type enumere associe au nom du type
80 // de calcul_masse nom_calcul_masse
81 Enum_calcul_masse Id_nom_calcul_masse (char* nom_calcul_masse);
82
83 // surcharge de l'operator de lecture
84 istream & operator » (istream & entree, Enum_calcul_masse& a);
85 // surcharge de l'operator d'écriture
86 ostream & operator « (ostream & sort, const Enum_calcul_masse& a);
87
88
89 #endif
90
91

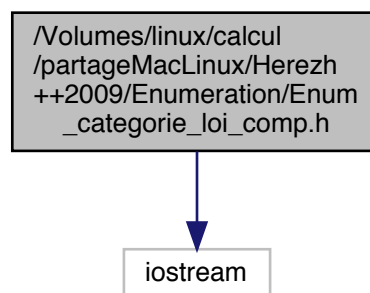
```

## 7.250 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/Enum\_categorie\_loi\_comp.h

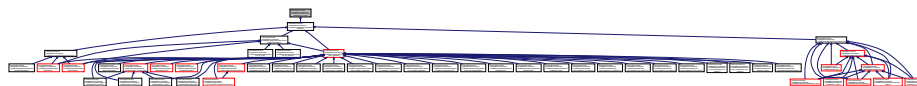
def de l'enuméré concernant les booléens ddl.

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_categorie\_loi\_comp.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

```
— enum Enum_categorie_loi_comp {
    CAT_MECANIQUE =1 , CAT_THERMO_MECANIQUE , CAT_THERMO_PHYSIQUE , CAT_FROTTEMENT
,
    RIEN_CATEGORIE_LOI_COMP }

```

*Définition de l'enuméré concernant catégories de lois de comportement.*

## Fonctions

```
— char * Nom_categorie_loi_comp (const Enum_categorie_loi_comp id_comport)
— Enum_categorie_loi_comp Id_nom_categorie_loi_comp (const char *nom_comport)
— bool GroupeMecanique (const Enum_categorie_loi_comp id_comport)
— bool GroupeThermique (const Enum_categorie_loi_comp id_comport)
— bool GroupeFrottement (const Enum_categorie_loi_comp id_comport)
— Enum_categorie_loi_comp Categorie_loi_comp_la_plus_complete (const Enum_categorie_loi_comp id←
_comport1, const Enum_categorie_loi_comp id_comport2)
— bool MemeCategorie (const Enum_categorie_loi_comp id_comport1, const Enum_categorie_loi_comp id←
_comport2)
— istream & operator>> (istream &entree, Enum_categorie_loi_comp &a)
— ostream & operator<< (ostream &sort, const Enum_categorie_loi_comp &a)

```

## Variables

```
— const int nbmax_caractere_Enum_categorie_loi_comp = 26

```

### 7.250.1 Description détaillée

def de l'enuméré concernant les booléens ddl.

## 7.251 Enum\_categorie\_loi\_comp.h

[Aller à la documentation de ce fichier.](#)

```
1 /*! \file Enum_categorie_loi_comp.h
2   \brief def de l'enuméré concernant les booléens ddl.
3 */
4 // FICHER : Enum_categorie_loi_comp.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //

```

```

29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, le type de categorie des lois de comportement sont
36 // stockes a l'aide d'un type enumere. Les fonctions Nom_categorie_comp et Id_nom_categorie_comp
37 // rendent possible le lien entre les noms des lois de comportement et les identificateurs
38 // de type enumere correspondants.
39
40
41 #ifndef ENUM_CATEGORIE_LOI_COMP_H
42 #define ENUM_CATEGORIE_LOI_COMP_H
43 #include <iostream>
44 using namespace std;
45
46 /// @addtogroup Group_types_enumeres
47 /// @{
48
49 /// Définition de l'énuméré concernant catégories de lois de comportement.
50
51 // *** important ****, pour chaque groupe de loi il faut que les catégories qui sont les plus
52 // générales, donc qui englobent les autres, doivent avoir une position plus haute
53 // ceci par exemple pour que la fonction Categorie_loi_comp_la_plus_complete fonctionne
54
55 enum Enum_categorie_loi_comp { CAT_MECANIQUE=1,CAT_THERMO_MECANIQUE, CAT_THERMO_PHYSIQUE
56                               ,CAT_FROTTEMENT
57                               ,RIEN_CATEGORIE_LOI_COMP};
58 /// @} // end of group
59
60
61 // ***** !!!! penser à changer *****
62 //***** nbmax_caractere_Enum_categorie_loi_comp si nécessaire *****
63 const int nbmax_caractere_Enum_categorie_loi_comp = 26;
64
65 // Retourne le nom d'une loi de comportement a partir de son identificateur de
66 // type enumere id_categorie_loi_comp correspondant
67 char* Nom_categorie_loi_comp (const Enum_categorie_loi_comp id_compport);
68
69 // Retourne l'identificateur de type enumere associe au nom de la loi de
70 // comportement nom_compport
71 Enum_categorie_loi_comp Id_nom_categorie_loi_comp (const char* nom_compport);
72
73 // indique si oui ou non la categorie est mécanique c'est-à-dire satisfait à Loi_comp_abstraite
74 bool GroupeMecanique(const Enum_categorie_loi_comp id_compport);
75
76 // indique si oui ou non la categorie est thermique c'est-à-dire satisfait à CompThermoPhysiqueAbstraite
77 bool GroupeThermique(const Enum_categorie_loi_comp id_compport);
78
79 // indique si oui ou non la categorie est frottement (de contact) c-a-d lié à CompFrotAbstraite
80 bool GroupeFrottement(const Enum_categorie_loi_comp id_compport);
81
82 // ramène un énuméré qui correspond à la catégorie la plus complète des deux passés en argument
83 // par exemple la catégorie CAT_THERMO_MECANIQUE englobe la catégorie CAT_MECANIQUE
84 // au paravant on test si les deux sont du même groupe, sinon erreur
85 Enum_categorie_loi_comp Categorie_loi_comp_la_plus_complete(const Enum_categorie_loi_comp id_compport1,
86                                                            const Enum_categorie_loi_comp id_compport2);
87 // test si deux énumérés correspondent à la même catégorie
88 bool MemeCategorie(const Enum_categorie_loi_comp id_compport1,const Enum_categorie_loi_comp id_compport2);
89 // surcharge de l'operator de lecture
90 istream & operator » (istream & entree, Enum_categorie_loi_comp& a);
91 // surcharge de l'operator d'écriture
92 ostream & operator « (ostream & sort, const Enum_categorie_loi_comp& a);
93
94 #endif

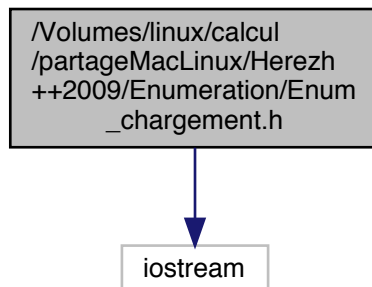
```

## 7.252 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/Enum\_chargement.h

def de l'énuméré concernant les types de chargement

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_chargement.h:



## Énumérations

```
— enum Enum_chargement {
    RIEN_CHARGEMENT = 1, UNIFORME, PRESSION, PONCTUELLE,
    PRESSDIR, PHYDRO, LINEIQUE, VOLUMIQUE,
    LINEIC_SUIVEUSE, P_HYDRODYNA, TORSEUR_PONCT }
    Définition de l'enuméré concernant les types de chargement.
```

## Fonctions

```
— string Nom_chargement (const Enum_chargement id_chargement)
— Enum_chargement Id_nom_chargement (const string &nom_chargement)
— istream & operator>> (istream &entree, Enum_chargement &a)
— ostream & operator<< (ostream &sort, const Enum_chargement &a)
```

### 7.252.1 Description détaillée

def de l'enuméré concernant les types de chargement

## 7.253 Enum\_chargement.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_chargement.h
2    \brief def de l'enuméré concernant les types de chargement
3 */
4 // FICHER : Enum_chargement.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
  
```

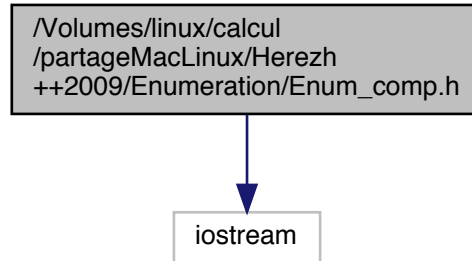
```
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms de chargement
36 // sont stockes a l'aide d'un type enumere. Les fonctions Nom_chargement et
37 // Id_nom_chargement rendent possible le lien entre les noms des types de chargement
38 // et les identificateurs de type enumere correspondants.
39
40
41 #ifndef ENUM_CHARGEMENT_H
42 #define ENUM_CHARGEMENT_H
43 #include <iostream>
44 using namespace std;
45
46 /// @addtogroup Group_types_enumeres
47 /// @{
48
49 /// Définition de l'énuméré concernant les types de chargement
50
51 enum Enum_chargement { RIEN_CHARGEMENT = 1, UNIFORME
52     , PRESSION, PONCTUELLE, PRESSDIR, PHYDRO
53     , LINEIQUE, VOLUMIQUE, LINEIC_SUIVEUSE, P_HYDRODYNA, TORSEUR_PONCT};
54 /// @} // end of group
55
56
57
58 // Retourne un nom du chargement a partir de son identificateur de
59 // type enumere id_chargement correspondant
60 string Nom_chargement (const Enum_chargement id_chargement) ;
61
62 // Retourne l'identificateur de type enumere associe au nom du type
63 // de chargement nom_chargement
64 Enum_chargement Id_nom_chargement (const string& nom_chargement);
65
66 // surcharge de l'operator de lecture
67 istream & operator » (istream & entree, Enum_chargement& a);
68 // surcharge de l'operator d'écriture
69 ostream & operator « (ostream & sort, const Enum_chargement& a);
70
71
72 #endif
73
74
```

## 7.254 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/Enum\_comp.h

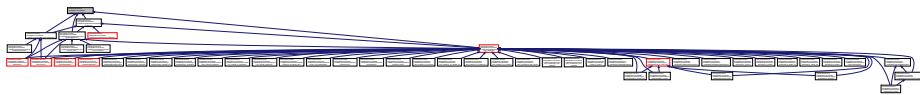
def de l'énuméré permettant d'identifier chaque loi de comportement

```
#include <iostream>
```

Graphes des dépendances par inclusion de Enum\_comp.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_comp` {
  - ISOELAS =1 , ISOELAS1D , ISOELAS2D\_D , ISOELAS2D\_C ,
  - ISO\_ELAS\_ESPO1D , ISO\_ELAS\_SE1D , ISO\_ELAS\_ESPO3D , ORTHOELA3D ,
  - ORTHOELA2D\_D , ORTHOELA2D\_C , HYPO\_ORTHO3D , HYPO\_ORTHO2D\_D ,
  - HYPO\_ORTHO2D\_C , PROJECTION\_ANISOTROPE\_3D , VISCOELA , ISOHYPER ,
  - ISOHYPER1 , ISOHYPER10 , ISOHYSTE , TRELOAR ,
  - ISOHYPER3DFAVIER1 , ISOHYPER3DFAVIER2 , ISOHYPER3DFAVIER3 , ISOHYPER3DFAVIER4 ,
  - ISOHYPER3DORGEAS1 , ISOHYPER3DORGEAS2 , ISOHYPERBULK3 , ISOHYPERBULK\_GENE ,
  - PRANDTL\_REUSS , PRANDTL\_REUSS2D\_D , PRANDTL\_REUSS2D\_C , PRANDTL\_REUSS1D ,
  - NEWTON1D , NEWTON2D\_C , NEWTON2D\_D , NEWTON3D ,
  - HYPO\_ELAS3D , HYPO\_ELAS2D\_C , HYPO\_ELAS2D\_D , HYPO\_ELAS1D ,
  - MAXWELL1D , MAXWELL2D\_C , MAXWELL2D\_D , MAXWELL3D ,
  - LOI\_ADDITIVE\_EN\_SIGMA , LOI\_DES\_MELANGES\_EN\_SIGMA , LOI\_CRITERE , LOI\_CONTRAINTE↔
  - \_PLANES ,
  - LOI\_CONTRAINTE↔\_PLANES\_DOUBLE , LOI\_DEFORMATIONS\_PLANES , HYSTERESIS\_1D ,
  - HYSTERESIS\_3D ,
  - HYSTERESIS\_BULK , LOI\_ISO\_THERMO , MOONEY\_RIVLIN\_1D , MOONEY\_RIVLIN\_3D ,
  - POLY\_HYPER3D , HART\_SMITH3D , MAHEO\_HYPER , HYPER\_EXTERNE\_W ,
  - LOI\_DE\_TAIT , LOI\_VIA\_UMAT , LOI\_VIA\_UMAT\_CP , LOI\_COULOMB ,
  - LOI\_RIEN1D , LOI\_RIEN2D\_D , LOI\_RIEN2D\_C , LOI\_RIEN3D ,
  - RIEN\_COMP }

*énuméré permettant d'identifier chaque loi de comportement*
- enum `Enum_comp_3D_CP_DP_1D` {
  - COMP\_1D =1 , COMP\_CONTRAINTE↔\_PLANES , COMP\_DEFORMATIONS\_PLANES , COMP\_3D ,
  - RIEN\_COMP\_3D\_CP\_DP\_1D }

*énuméré permettant de savoir si une loi est : 1D, 2D en déformations planes ou contraintes planes, 3D générale*

## Fonctions

- string `Nom_comp` (const `Enum_comp` id\_comport)

- `Enum_comp Id_nom_comp` (const string &nom\_comport)
- `bool Loi_rien` (const `Enum_comp` id\_comport)
- `istream & operator>>` (istream &entree, `Enum_comp` &a)
- `ostream & operator<<` (ostream &sort, const `Enum_comp` &a)
- `string Nom_comp_3D_CP_DP_1D` (const `Enum_comp_3D_CP_DP_1D` id\_Enum\_comp\_3D\_CP\_DP\_1D)
- `Enum_comp_3D_CP_DP_1D Id_nom_comp_3D_CP_DP_1D` (const string nom\_comp\_3D\_CP\_DP\_1D)
- `Enum_comp_3D_CP_DP_1D Comp_3D_CP_DP_1D` (const `Enum_comp` id\_comport)
- `istream & operator>>` (istream &entree, `Enum_comp_3D_CP_DP_1D` &a)
- `ostream & operator<<` (ostream &sort, const `Enum_comp_3D_CP_DP_1D` &a)

## Variables

- `const int nbmax_caractere_Enum_comp = 26`

### 7.254.1 Description détaillée

def de l'énuméré permettant d'identifier chaque loi de comportement

## 7.255 Enum\_comp.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_comp.h
2     \brief def de l'énuméré permettant d'identifier chaque loi de comportement
3  */
4  // FICHER : Enum_comp.h
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des lois de comportement sont
36 // stockes a l'aide d'un type enumere. Les fonctions Nom_comp et Id_nom_comp rendent
37 // possible le lien entre les noms des lois de comportement et les identificateurs
38 // de type enumere correspondants.
39
40
41 #ifndef ENUM_COMP_H
42 #define ENUM_COMP_H
43 #include <iostream>
44 using namespace std;
45
46 /// @addtogroup Group_types_enumeres
47 /// @{
48
49
50 /// énuméré permettant d'identifier chaque loi de comportement
51 enum Enum_comp { ISOELAS=1, ISOELAS1D, ISOELAS2D_D, ISOELAS2D_C, ISO_ELAS_ESPO1D
52                 , ISO_ELAS_SE1D, ISO_ELAS_ESPO3D
53                 , ORTHOELA3D, ORTHOELA2D_D, ORTHOELA2D_C
54                 , HYPO_ORTHO3D, HYPO_ORTHO2D_D, HYPO_ORTHO2D_C
55                 , PROJECTION_ANISOTROPE_3D
56                 , VISCOELA, ISOHYPER, ISOHYPER1, ISOHYPER10, ISOHYSTE

```



## 7.256 Référence du fichier

**/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_contrainte\_mathematique.t2975**

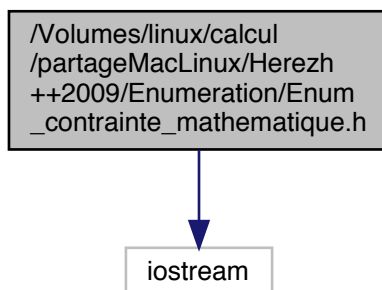
```
57         ,TRELOAR, ISOHYPER3DFAVIER1, ISOHYPER3DFAVIER2, ISOHYPER3DFAVIER3
58         , ISOHYPER3DFAVIER4, ISOHYPER3DORGEAS1, ISOHYPER3DORGEAS2
59         , ISOHYPERBULK3, ISOHYPERBULK_GENE
60         , PRANDTL_REUSS, PRANDTL_REUSS2D_D, PRANDTL_REUSS2D_C
61         , PRANDTL_REUSS1D
62         , NEWTON1D, NEWTON2D_C, NEWTON2D_D, NEWTON3D
63         , HYPO_ELAS3D, HYPO_ELAS2D_C, HYPO_ELAS2D_D, HYPO_ELAS1D
64         , MAXWELL1D, MAXWELL2D_C, MAXWELL2D_D, MAXWELL3D
65         , LOI_ADDITIVE_EN_SIGMA, LOI_DES_MELANGES_EN_SIGMA, LOI_CRITERE
66
67         , LOI_CONTRAINTES_PLANES, LOI_CONTRAINTES_PLANES_DOUBLE, LOI_DEFORMATIONS_PLANES
68         , HYSTERESIS_1D, HYSTERESIS_3D, HYSTERESIS_BULK, LOI_ISO_THERMO
69         , MOONEY_RIVLIN_1D, MOONEY_RIVLIN_3D, POLY_HYPER3D, HART_SMITH3D, MAHEO_HYPER
70         , HYPER_EXTERNE_W
71         , LOI_DE_TAIT, LOI_VIA_UMAT, LOI_VIA_UMAT_CP
72         , LOI_COULOMB
73         , LOI_RIEN1D, LOI_RIEN2D_D, LOI_RIEN2D_C, LOI_RIEN3D
74         , RIEN_COMP};
75
76
77 /// @} // end of group
78
79
80 /// @addtogroup Group_types_enumeres
81 /// @{
82
83 // énuméré permettant de savoir si une loi est : 1D, 2D en deformations planes ou contraintes planes, 3D
84 // générale
85 enum Enum_comp_3D_CP_DP_1D { COMP_1D =1, COMP_CONTRAINTES_PLANES, COMP_DEFORMATIONS_PLANES, COMP_3D ,
86     RIEN_COMP_3D_CP_DP_1D};
87
88 /// @} // end of group
89
90
91 // ***** !!!! penser à changer *****
92 //***** nbmax_caractere_Enum_comp si nécessaire *****
93 const int nbmax_caractere_Enum_comp = 26;
94
95 // ----- 1) concernant Enum_comp -----
96
97 // Retourne le nom d'une loi de comportement a partir de son identificateur de
98 // type enumere id_comport correspondant
99 string Nom_comp(const Enum_comp id_comport);
100
101 // Retourne l'identificateur de type enumere associe au nom de la loi de
102 // comportement nom_comport
103 Enum_comp Id_nom_comp(const string& nom_comport);
104
105 // indique si la loi est inactive mécaniquement
106 // typiquement de type : LOI_RIEN... ou RIEN_COMP
107 bool Loi_rien(const Enum_comp id_comport);
108
109 // surcharge de l'operator de lecture
110 istream & operator » (istream & entree, Enum_comp& a);
111 // surcharge de l'operator d'écriture
112 ostream & operator « (ostream & sort, const Enum_comp& a);
113
114 // ----- 2) concernant Enum_comp_3D_CP_DP_1D -----
115
116 // Retourne le nom a partir de son identificateur du type enumere id_Enum_comp_3D_CP_DP_1D
117 // correspondant
118 string Nom_comp_3D_CP_DP_1D(const Enum_comp_3D_CP_DP_1D id_Enum_comp_3D_CP_DP_1D);
119
120 // Retourne l'identificateur de type enumere associe à un nom nom_comp_3D_CP_DP_1D
121 Enum_comp_3D_CP_DP_1D Id_nom_comp_3D_CP_DP_1D(const string nom_comp_3D_CP_DP_1D);
122
123 // indique le type Enum_comp_3D_CP_DP_1D correspondant à une loi de comportement
124 Enum_comp_3D_CP_DP_1D Comp_3D_CP_DP_1D(const Enum_comp id_comport);
125
126 // surcharge de l'operator de lecture
127 istream & operator » (istream & entree, Enum_comp_3D_CP_DP_1D& a);
128 // surcharge de l'operator d'écriture
129 ostream & operator « (ostream & sort, const Enum_comp_3D_CP_DP_1D& a);
130
131
132
133 #endif
```

## 7.256 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_contrainte\_mathematique.h

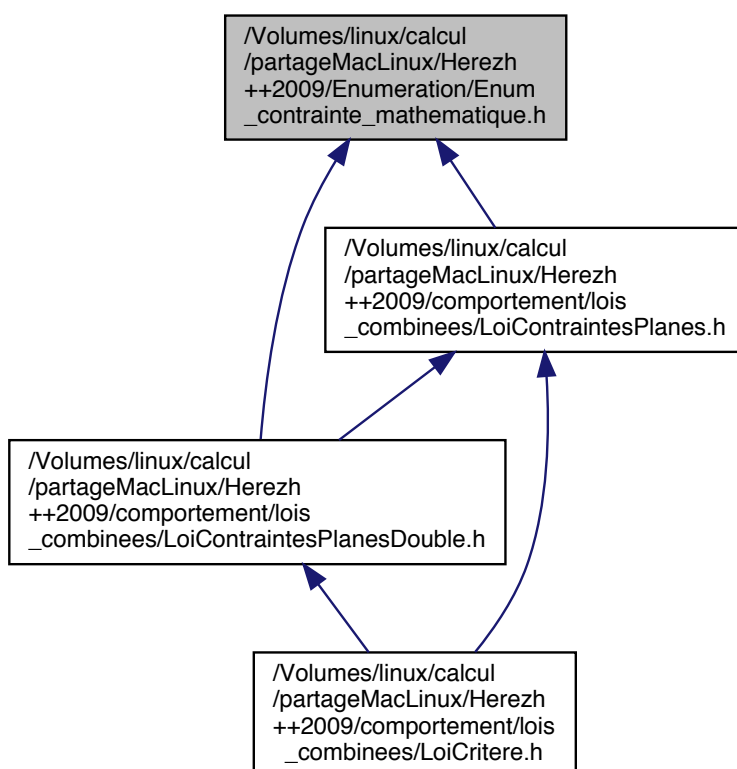
def de l'énuméré concernant les types de contraintes mathématiques

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_contrainte\_mathematique.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

```
— enum Enum_contrainte_mathematique {
    MULTIPLICATEUR_DE_LAGRANGE =1 , PENALISATION , NEWTON_LOCAL , PERTURBATION ,
```

**RIEN\_CONTRAINTE\_MATHEMATIQUE }**

*Définition de l'enuméré concernant les types de contraintes mathématiques.*

**Fonctions**

- string **Nom\_contrainte\_mathematique** (const [Enum\\_contrainte\\_mathematique](#) id\_contrainte\_↔  
mathematique)
- [Enum\\_contrainte\\_mathematique](#) **Id\_nom\_contrainte\_mathematique** (const string &nom\_contrainte\_↔  
mathematique)
- istream & **operator**>> (istream &entree, [Enum\\_contrainte\\_mathematique](#) &a)
- ostream & **operator**<< (ostream &sort, const [Enum\\_contrainte\\_mathematique](#) &a)

**Variables**

- const int **nbmax\_caractere\_Enum\_contrainte\_mathematique** = 50

**7.256.1 Description détaillée**

def de l'enuméré concernant les types de contraintes mathématiques

**7.257 Enum\_contrainte\_mathematique.h**

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_contrainte_mathematique.h
2  \brief def de l'enuméré concernant les types de contraintes mathématiques
3  */
4  // FICHER : Enum_contrainte_mathematique.h
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des methodes permettant de mettre en place une
   contrainte
36 // mathematique sont
37 // stockes a l'aide d'un type enumere. Les fonctions Nom_contrainte_mathematique et
   Id_nom_contrainte_mathematique rendent
38 // possible le lien entre les noms des methodes et les identificateurs
39 // de type enumere correspondants.
40
41
42 #ifndef ENUM_CONTRAINTE_MATHEMATIQUE_H
43 #define ENUM_CONTRAINTE_MATHEMATIQUE_H
44 #include <iostream>
45 using namespace std;
46
47 /// @addtogroup Group_types_enumeres
48 /// @{
49
50 /// Définition de l'enuméré concernant les types de contraintes mathématiques

```

```

51
52 enum Enum_contrainte_mathematique { MULTIPLICATEUR_DE_LAGRANGE=1, PENALISATION,
    NEWTON_LOCAL, PERTURBATION, RIEN_CONTRAINTTE_MATHEMATIQUE};
53 /// @} // end of group
54
55
56 // ***** !!!! penser à changer *****
57 //***** nbmax_caractere_Enum_contrainte_mathematique si nécessaire *****
58 const int nbmax_caractere_Enum_contrainte_mathematique = 50;
59
60 // Retourne le nom d'une méthode a partir de son identificateur de
61 // type enumere id_contrainte_mathematique correspondant
62 string Nom_contrainte_mathematique (const Enum_contrainte_mathematique id_contrainte_mathematique);
63
64 // Retourne l'identificateur de type enumere associe au nom de la méthode
65 // nom_contrainte_mathematique
66 Enum_contrainte_mathematique Id_nom_contrainte_mathematique (const string& nom_contrainte_mathematique);
67
68
69 // surcharge de l'operator de lecture
70 istream & operator » (istream & entree, Enum_contrainte_mathematique& a);
71 // surcharge de l'operator d'écriture
72 ostream & operator « (ostream & sort, const Enum_contrainte_mathematique& a);
73
74 #endif

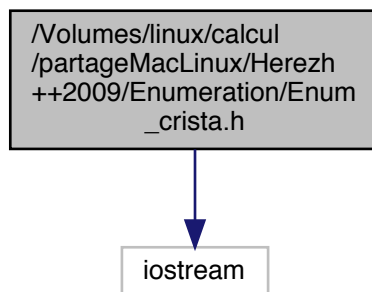
```

## 7.258 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_crista.h

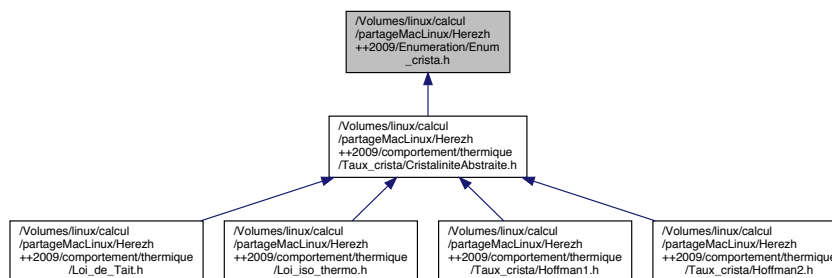
def Enumération des différents calcul de cristallinité

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_crista.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_crista` { `HOFFMAN = 1` , `HOFFMAN2` , `CRISTA_PAS_DEFINI` }  
*Énumération des différents calcul de cristallinité*

## Fonctions

- char \* `Nom_Enum_crista` (`Enum_crista id_Enum_crista`)
- `Enum_crista Id_nom_Enum_crista` (const char \*`nom_Enum_crista`)
- `istream & operator>>` (`istream &entree`, `Enum_crista &a`)
- `ostream & operator<<` (`ostream &sort`, const `Enum_crista &a`)

## Variables

- const int `nombre_maxi_de_type_de_Enum_crista` = 2

### 7.258.1 Description détaillée

def Énumération des différents calcul de cristallinité

Date

04/03/2008

## 7.259 Enum\_crista.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_crista.h
2   \brief def Énumération des différents calcul de cristallinité
3 * \date      04/03/2008
4 */
5
6 // FICHER : Enum_crista.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37 *      DATE:          04/03/2008
38 *
39 *      AUTEUR:        G RIO    (mailto:gerardrio56@free.fr)
40 *
41 *      PROJET:        Herezh++
42 *
43 *      ****
44 *      BUT:          Énumération des différents calcul de cristallinité
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *
50 *****/

```

```

49 *      ! date !   auteur !           but           !      *
50 *      -----
51 *      !           !           !           !           !      *
52 *      !           !           !           !           !      *
53 *      !           !           !           !           !      *
54 *      !           !           !           !           !      *
55 *      ! date !   auteur !           but           !      *
56 *      -----
57 *      !           !           !           !           !      *
58 *      !           !           !           !           !      *
59 *      !           !           !           !           !      *
60 #ifndef ENUM_CRISTA_H
61 #define ENUM_CRISTA_H
62
63 #include <iostream>
64 using namespace std;
65
66 /// @addtogroup Group_types_enumeres
67 /// @{
68
69 /// Énumération des différents calcul de cristallinité
70
71 enum Enum_crista { HOFFMAN = 1,HOFFMAN2, CRISTA_PAS_DEFINI };
72 const int nombre_maxi_de_type_de_Enum_crista = 2;
73 /// @} // end of group
74
75
76
77
78 // Retourne un nom de type de Enum_crista a partir de son identificateur de
79 // type enumere id_Enum_crista correspondant
80 char* Nom_Enum_crista (Enum_crista id_Enum_crista);
81
82 // Retourne l'identificateur de type enumere associe au nom du type
83 // de Enum_crista nom_Enum_crista
84 Enum_crista Id_nom_Enum_crista (const char* nom_Enum_crista) ;
85
86 // surcharge de l'operator de lecture
87 istream & operator » (istream & entree, Enum_crista& a);
88 // surcharge de l'operator d'écriture
89 ostream & operator « (ostream & sort, const Enum_crista& a);
90
91
92 #endif
93
94

```

## 7.260 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/Enum\_Critere\_loi.h

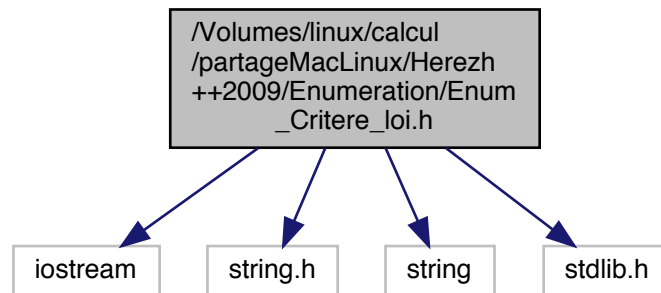
Enumeration des différentes méthodes permettant d'utiliser des critères sur les lois de comportement.

```

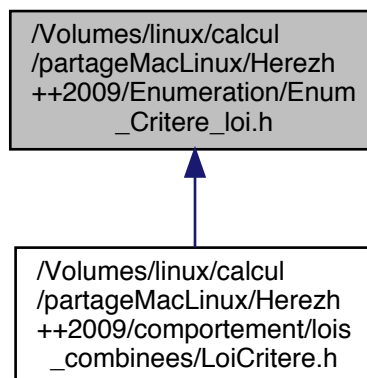
#include <iostream>
#include <string.h>
#include <string>
#include <stdlib.h>

```

Grappe des dépendances par inclusion de Enum\_Critere\_loi.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

```

— enum Enum_Critere_Loi {
    AUCUN_CRITERE = 0 , PLISSEMENT_MEMBRANE , PLISSEMENT_BIEL , RUPTURE_SIGMA_PRINC ,
    RUPTURE_EPS_PRINC }
  
```

*Enumeration des différentes méthodes permettant d'utiliser des critères sur les lois de comportement.*

## Fonctions

```

— string Nom_Critere_Loi (Enum_Critere_Loi id_Critere_Loi)
— Enum_Critere_Loi Id_Nom_Critere_Loi (const string &nom_Critere_Loi)
— bool Type_Enum_Critere_Loi_existe (const string &nom)
— istream & operator>> (istream &entree, Enum_Critere_Loi &a)
— ostream & operator<< (ostream &sort, const Enum_Critere_Loi &a)
  
```

## 7.260.1 Description détaillée

Enumeration des différentes méthodes permettant d'utiliser des critères sur les lois de comportement.

Date

11/06/2014

## 7.261 Enum\_Critere\_loi.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_Critere_loi.h
2     \brief Enumeration des différentes méthodes permettant d'utiliser des critères sur les lois de
           comportement
3  * \date      11/06/2014
4  */
5
6  // FICHER : Enum_Critere_loi.h
7
8  // This file is part of the Herezh++ application.
9  //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPIY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37 *      DATE:      11/06/2014
38 *
39 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
40 *
41 *      PROJET:     Herezh++
42 *
43 *      BUT:        Enumeration des différentes méthodes permettant
44 *                  d'utiliser des critères sur les lois de comportement
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *
50 *      ! date !   auteur !           but
51 *      -----
52 *      !           !           !
53 *
54 *      *****
55 *      MODIFICATIONS:
56 *      ! date !   auteur !           but
57 *      -----
58 *
59 *      *****/
60
61
62 #ifndef ENUM_CRITERE_LOI_H
63 #define ENUM_CRITERE_LOI_H
64 #include <iostream>
65 #include <string.h>
66 #include <string>
67 using namespace std; //introduces namespace std
68 #include <stdlib.h>
69
70 /// @addtogroup Group_types_enumeres
71 /// @{
72

```



```

73 // Enumeration des différentes méthodes permettant d'utiliser des critères sur les lois de comportement
74
75 enum Enum_Critere_Loi { AUCUN_CRITERE = 0, PLISSEMENT_MEMBRANE, PLISSEMENT_BIEL
76                        , RUPTURE_SIGMA_PRINC, RUPTURE_EPS_PRINC };
77 // @} // end of group
78
79
80
81 // Retourne le nom a partir de son identificateur de type enumere
82 string Nom_Critere_Loi(Enum_Critere_Loi id_Critere_Loi);
83
84 // Retourne l'identificateur de type enumere associe au nom d'un Critere_Loi
85 Enum_Critere_Loi Id_Nom_Critere_Loi(const string& nom_Critere_Loi) ;
86
87 // Retourne vrai si le nom passé en argument représente un type de Critere_Loi reconnu
88 // sinon false
89 bool Type_Enum_Critere_Loi_existe(const string& nom);
90
91 // surcharge de l'operator de lecture
92 istream & operator » (istream & entree, Enum_Critere_Loi& a);
93 // surcharge de l'operator d'écriture
94 ostream & operator « (ostream & sort, const Enum_Critere_Loi& a);
95
96 #endif

```

## 7.262 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_ddl.h

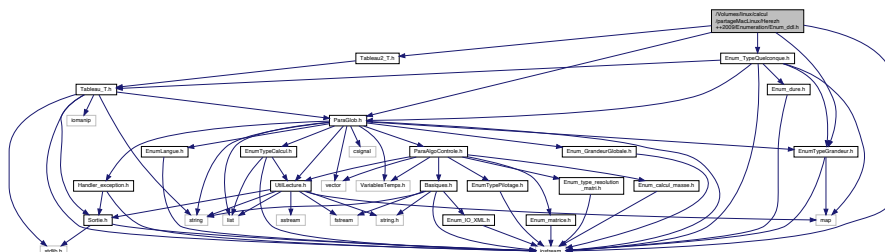
def de l'enuméré concernant les ddl.

```

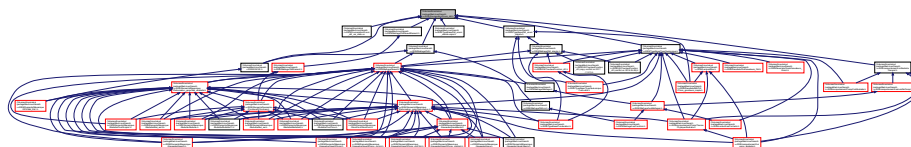
#include "Tableau2_T.h"
#include <iostream>
#include "EnumTypeGrandeur.h"
#include "ParaGlob.h"
#include "Enum_TypeQuelconque.h"
#include "Enum_ddl.cc"

```

Grappe des dépendances par inclusion de Enum\_ddl.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Classes

- class [ClassPourEnum\\_ddl](#)  
classe utilitaire entre Enum\_ddl et une map
- class [Deuxentiers\\_enu](#)

## Énumérations

```

— enum Enum_ddl {
    X1 = 1 , X2 , X3 , EPAIS ,
    TEMP , UX , UY , UZ ,
    V1 , V2 , V3 , PR ,
    GAMMA1 , GAMMA2 , GAMMA3 , SIG11 ,
    SIG22 , SIG33 , SIG12 , SIG23 ,
    SIG13 , ERREUR , EPS11 , EPS22 ,
    EPS33 , EPS12 , EPS23 , EPS13 ,
    DEPS11 , DEPS22 , DEPS33 , DEPS12 ,
    DEPS23 , DEPS13 , PROP_CRISTA , DELTA_TEMP ,
    FLUXD1 , FLUXD2 , FLUXD3 , R_TEMP ,
    GRADT1 , GRADT2 , GRADT3 , DGRADT1 ,
    DGRADT2 , DGRADT3 , R_X1 , R_X2 ,
    R_X3 , R_EPAIS , R_V1 , R_V2 ,
    R_V3 , R_GAMMA1 , R_GAMMA2 , R_GAMMA3 ,
    NU_DDL }

```

## Fonctions

- string **Nom\_ddl** (Enum\_ddl id\_ddl)  
*Retourne le nom d'un degre de liberte a partir de son identificateur de type enumere id\_ddl correspondant.*
- Enum\_ddl **Id\_nom\_ddl** (const string &nom\_ddl)  
*Retourne l'identificateur de type enumere associe au nom du degre de liberte nom\_ddl Enum\_ddl Id\_nom\_ddl (const char\* nom\_ddl);.*
- bool **ExisteEnum\_ddl** (const string &nom\_ddl)  
*retourne true si l'identificateur existe, false sinon bool ExisteEnum\_ddl(const char\* nom\_ddl);*
- int **NbEnum\_ddl** ()  
*retourne le nombre maxi de ddl existant*
- bool **FoncDim** (const string &nom\_ddl)  
*ramene true si le ddl fait parti d'une liste fonction de la dimension du pb: par exemple : X1 ou UY ou VZ dans le cas contraire , exemple: T ou EPAIS ramene false*
- bool **FoncDim** (Enum\_ddl id\_ddl)  
*idem mais avec l'énumération*
- **Tableau**< Enum\_ddl > **TableauTypeDdl** (Enum\_ddl id\_ddl)  
*ramene le tableau de tous les ddl correspondant au type du ddl passé en paramètre ceci en fonction de la dimension (par exemple pour un COORDONNEE, ramène les dim ddl correspondant aux composantes*
- **Tableau**< Enum\_ddl > **TableauTypeDdlmobileAvecAxi** (Enum\_ddl id\_ddl)  
*ramene le tableau des ddl mobile correspondant au type du ddl passé en paramètre ceci en fonction de la dimension (par exemple pour un COORDONNEE, ramène les dim ddl correspondant aux composantes par rapport à [TableauTypeDdl\(\)](#) la différence concerne le cas axi pour ce cas: X3=constant, U3=V3=Gamma3=0 ==> donc ne font pas partie du tableau retour*
- bool **CompatDim** (const string &nom\_ddl)  
*indique si le ddl est compatible avec la dimension c'est-à-dire qu'il peut exister avec la dimension actuelle : par exemple en 1D on ne peut pas avoir de UY, mais UX c'est ok*
- bool **CompatDim** (const Enum\_ddl id\_ddl)
- **EnumTypeGrandeur** **TypeGrandeur** (Enum\_ddl id\_ddl)  
*retour le type de grandeur auquel appartient l'énumération par exemple : UY : appartient à un COORDONNEE SIG12 : à un tenseur, TEMP : à un scalaire*
- bool **Meme\_famille** (Enum\_ddl a, Enum\_ddl b)  
*test si les deux ddl sont de la même famille dimensionnelle, par exemple X1 et X3, ou SIG11 et SIG22 ramène true si oui, false sinon*
- Enum\_ddl **PremierDdlFamille** (Enum\_ddl a)  
*ramène le premier ddl de la même famille*
- string **NomGeneric** (Enum\_ddl a)  
*ramène un nom générique pour la famille de ddl du même type*
- int **Nombre\_de\_famille\_de\_ddl** ()  
*ramène de nombre de famille différente qui existe*
- bool **Dans\_combinaison** (int cas, Enum\_ddl a)

- test si le ddl appartient à la combinaison donné par cas cas : spécifie la combinaison : =1 -> combinaison X V GAMMA ramène false si cas=0 c'est-à-dire pas de combinaison*
- **Tableau**< Enum\_ddl > **MemeCombinaison** (int cas, Enum\_ddl a)  
*ramène tous les membres d'une même combinaison de la même dimension y compris "a" cas : spécifie la combinaison : =0 -> pas de combinaison, ramène "a" =1 -> combinaison X V GAMMA*
  - **Tableau**< Enum\_ddl > **Combinaison** (int cas)  
*ramène tous les membres d'une même combinaison, pour i=1 à dim cas = 1 -> combinaison X V GAMMA cas = 0 -> pas de combinaison, ramène un tableau de dimension 0*
  - **EnumTypeQuelconque** **Equi\_Enum\_ddl\_en\_enu\_quelconque** (Enum\_ddl a)  
*recupération d'un enum de grandeurs quelconques équivalentes -> retour particulier si il y a une équivalence particulière en grandeur évoluée par exemple sinon, un retour sur un scalaire de type enum\_évolué, qui contient de toute manière les ddl de base -> ramène UN\_DDL\_ENUM\_ETENDUE s'il n'y a pas d'équivalent spécifique*
  - Enum\_ddl **UxyzXi** (Enum\_ddl a)  
*passage de Ui en Xi, c-a-d : UX -> X1, UY -> X2, UZ -> X3*
  - Enum\_ddl **XiUxyz** (Enum\_ddl a)  
*passage inverse*
  - istream & **operator**>> (istream &entree, Enum\_ddl &a)  
*surcharge de l'operator de lecture*
  - ostream & **operator**<< (ostream &sort, const Enum\_ddl &a)  
*surcharge de l'operator d'écriture*
  - int **Indice\_coor** (Enum\_ddl a, int nbcomposantes)  
*----- pour les Coordonnee ----- retourne l'indice en fonction de l'enum et du nbcomposante pour une grandeur de type Coordonnee, sinon erreur*
  - **Tableau2**< int > **OrdreContrainte** (int nbcomposantes)  
*----- pour les tenseurs ----- fonction donnant dans l'ordre des ddl de contraintes, les indices correspondant des tenseurs, ceci en fonction du nombre de composante de tenseur en retour un tableau : (i)(1) et (i)(2) -> les indices (en entier) du tenseur correspondant au i ième ddl de containte à partir de SIG11 mais en sautant les ddl qui ne font pas parti de la dimension !!!!!*
  - const **Tableau2**< int > & **OrdreContrainteR** (int nbcomposantes)  
*idem en fonction du nombre de composantes fonction plus rapide qui pointe sur des tableaux déjà construits à partir d'OrdreContrainte*
  - **Deuxentiers\_enu** **IJind** (Enum\_ddl a, int nbcomposantes)
  - Enum\_ddl **Vers\_enum\_reac** (Enum\_ddl a)  
*passage de enum\_ddl vers des réactions enum\_ddl correspondantes exemple X1 -> R\_X1 etc..*
  - Enum\_ddl **Enum\_reac\_vers\_enum** (Enum\_ddl a)  
*opération inverse: exemple R\_X1 -> X1*
  - bool **Ddl\_reaction** (Enum\_ddl a)  
*indique si un ddl est un ddl de réaction on non*

## Variables

- const int **nbmax\_caractere\_enum\_ddl** = 11  
*definition du maximum de caractères de la chaine équivalente à l'énumération*
- const int **nombre\_maxi\_de\_famille\_de\_ddl** = 21
- const int **nombre\_maxi\_de\_type\_de\_ddl** = 57
- const **Tableau2**< int > **OrdreContrainte1** = **OrdreContrainte**(1)  
*definition de tableau constant pour éviter de les recalculer à chaque appel*
- const **Tableau2**< int > **OrdreContrainte3** = **OrdreContrainte**(3)
- const **Tableau2**< int > **OrdreContrainte6** = **OrdreContrainte**(6)

### 7.262.1 Description détaillée

def de l'enuméré concernant les ddl.

## 7.263 Enum\_ddl.h

[Aller à la documentation de ce fichier.](#)

```
1 /*! \file Enum_ddl.h
2     \brief def de l'enuméré concernant les ddl.
3 */
4 // FICHER : Enum_ddl.h
```

```

5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34 /** @defgroup Group_types_enumeres Group_types_enumeres
35 *
36 * \author Gérard Rio
37 * \version 1.0
38 * \date 23/01/97
39 * \brief Def de grandeurs énumérées: permet une meilleure lisibilité du code et éventuellement un
    gain de place
40 *
41 */
42
43
44 /// @addtogroup Group_types_enumeres
45 /// @{
46 ///
47
48
49 /// Afin de realiser un gain en place memoire, les noms des degres de liberte sont
50 /// stockes a l'aide d'un type enumere. Les fonctions Nom_ddl et Id_nom_ddl rendent
51 /// possible le lien entre les noms des degres de liberte et les identificateurs
52 /// de type enumere correspondants.
53 /// 10 caractere maxi
54 /// NU_DDL correspond au cas ou pas de ddl est defini
55 /// les ddl X1 X2 X3 doivent se suivrent
56 /// les ddl UX, UY, UZ doivent se suivrent
57 /// les ddl VX , VY , VZ, doivent se suivrent
58 /// ne pas changer l'ordre des 3 premiers ddl dans l'enumeration!!!! car on se
59 /// sert du fait que Enum_ddl(1) = X1 etc pour initialiser les ddl des noeud elements
60 ///
61 /// une famille de ddl: se sont des ddl scalaires qui representent les composantes d'un ddl vectoriel
62 /// ou tensoriel, ou scalaire s'il n'y a qu'une seule grandeur, qui ne depend pas de la dimension (ex:
    pression
63 /// ou température)
64 /// une combinaison , c'est un groupe de ddl qui sont relié par le calcul: ex: position, vitesse et
    accélération
65 /// en dynamique.
66
67
68 #ifndef ENUM_DDL_H
69 #define ENUM_DDL_H
70
71 // #include "Debug.h"
72 # include "Tableau2_T.h"
73 #include <iostream>
74 using namespace std;
75 #include "EnumTypeGrandeur.h"
76 #include "ParaGlob.h"
77 #include "Enum_TypeQuelconque.h"
78
79 //=====
80 // l'énumération de degré de liberté de base, celle qui sert pour les calculs
81 // nombre de ddl limité
82 //=====
83 // ***** !!!! penser à changer la fonction "NbEnum_ddl" et *****
84 // ***** nbmax_caractere_enum_ddl si nécessaire *****
85 // le type énuméré doit commencer à 1, c'est utilisé dans Ddl_enum_etendu par exemple !!
86 enum Enum_ddl { X1 = 1, X2 , X3, EPAIS , TEMP , UX, UY, UZ , V1 , V2 , V3,
87 PR, GAMMA1, GAMMA2, GAMMA3,
88 SIG11, SIG22, SIG33, SIG12, SIG23, SIG13, ERREUR,

```

```

89             EPS11, EPS22, EPS33, EPS12, EPS23, EPS13,
90             DEPS11, DEPS22, DEPS33, DEPS12, DEPS23, DEPS13,
91             PROP_CRISTA, DELTA_TEMP, FLUXD1, FLUXD2, FLUXD3, R_TEMP,
92             GRADT1, GRADT2, GRADT3, DGRADT1, DGRADT2, DGRADT3,
93             R_X1, R_X2, R_X3, R_EPAIS, R_V1, R_V2, R_V3, R_GAMMA1, R_GAMMA2, R_GAMMA3,
94             NU_DDL };
95 // ***** !!!! penser à changer la fonction "NbEnum_ddl" et *****
96 //***** nbmax_caractere_enum_ddl si nécessaire *****
97 //***** Nombre_de_famille_de_ddl si nécessaire *****
98 // @} // end of group
99
100
101 /// @addtogroup Group_types_enumeres
102 /// @{
103
104 /// classe utilitaire entre Enum_ddl et une map
105 class ClassPourEnum_ddl
106 { public:
107     friend Enum_ddl Id_nom_ddl (const char* nom);
108     friend Enum_ddl Id_nom_ddl (const string& nom);
109     friend bool ExisteEnum_ddl(const char* nom);
110     friend bool ExisteEnum_ddl(const string& nom);
111     /// def de la map qui fait la liaison entre les string et les énumérés
112     static map < string, Enum_ddl , std::less < string> > map_Enum_ddl;
113     /// def de la grandeur statique qui permet de remplir la map
114     static ClassPourEnum_ddl remplir_map;
115     /// le constructeur qui remplit effectivement la map
116     ClassPourEnum_ddl();
117     /// variable de travail
118     protected:
119     map < string, Enum_ddl , std::less < string> >::iterator il, ilfin;
120 };
121 // @} // end of group
122
123 //-----
124 // def des fonctions de manipulation des Enum_ddl
125 //-----
126
127 /// definition du maximum de caractères de la chaine équivalente à l'énumération
128 const int nbmax_caractere_enum_ddl = 11;
129 const int nombre_maxi_de_famille_de_ddl = 21;
130 const int nombre_maxi_de_type_de_ddl = 57;
131
132 /// Retourne le nom d'un degre de liberte a partir de son identificateur de
133 /// type enumere id_ddl correspondant
134 string Nom_ddl (Enum_ddl id_ddl);
135
136 /// Retourne l'identificateur de type enumere associe au nom du degre de liberte
137 /// nom_ddl
138 ///Enum_ddl Id_nom_ddl (const char* nom_ddl);
139 Enum_ddl Id_nom_ddl (const string& nom_ddl);
140
141 /// retourne true si l'identificateur existe, false sinon
142 ///bool ExisteEnum_ddl(const char* nom_ddl);
143 bool ExisteEnum_ddl(const string& nom_ddl);
144
145 /// retourne le nombre maxi de ddl existant
146 inline int NbEnum_ddl() {return nombre_maxi_de_type_de_ddl;};
147
148 /// ramene true si le ddl fait parti d'une liste
149 /// fonction de la dimension du pb: par exemple : X1 ou UY ou VZ
150 /// dans le cas contraire , exemple: T ou EPAIS ramene false
151 bool FoncDim(const string& nom_ddl);
152 /// idem mais avec l'énumération
153 bool FoncDim(Enum_ddl id_ddl);
154
155 /// ramene le tableau de tous les ddl correspondant au type du ddl passé en paramètre
156 /// ceci en fonction de la dimension (par exemple pour un COORDONNEE, ramène
157 /// les dim ddl correspondant aux composantes
158 Tableau<Enum_ddl> TableauTypeDdl (Enum_ddl id_ddl);
159
160 /// ramene le tableau des ddl mobile correspondant au type du ddl passé en paramètre
161 /// ceci en fonction de la dimension (par exemple pour un COORDONNEE, ramène
162 /// les dim ddl correspondant aux composantes
163 /// par rapport à TableauTypeDdl() la différence concerne le cas axi
164 /// pour ce cas: X3=constant, U3=V3=Gamma3=0 ==> donc ne font pas partie du tableau retour
165 Tableau<Enum_ddl> TableauTypeDdlmobileAvecAxi (Enum_ddl id_ddl);
166
167
168 /// indique si le ddl est compatible avec la dimension
169 /// c'est-à-dire qu'il peut exister avec la dimension actuelle :
170 /// par exemple en 1D on ne peut pas avoir de UY, mais UX c'est ok
171 bool CompatDim(const string& nom_ddl);
172 bool CompatDim(const Enum_ddl id_ddl);
173
174 /// retour le type de grandeur auquel appartient l'énumération
175 /// par exemple : UY : appartient à un COORDONNEE

```

```

176 /// SIG12 : à un tenseur, TEMP : à un scalaire
177 EnumTypeGrandeur TypeGrandeur(Enum_ddl id_ddl);
178
179 /// test si les deux ddl sont de la même famille dimensionnelle,
180 /// par exemple X1 et X3, ou SIG11 et SIG22
181 /// ramène true si oui, false sinon
182 bool Meme_famille(Enum_ddl a,Enum_ddl b);
183
184 /// ramène le premier ddl de la même famille
185 Enum_ddl PremierDdlFamille(Enum_ddl a);
186 /// ramène un nom générique pour la famille de ddl du même type
187 string NomGeneric(Enum_ddl a);
188
189 /// ramène de nombre de famille différente qui existe
190 inline int Nombre_de_famille_de_ddl() {return nombre_maxi_de_famille_de_ddl;};
191
192 /// test si le ddl appartient à la combinaison donné par cas
193 /// cas : spécifie la combinaison : =1 -> combinaison X V GAMMA
194 /// ramène false si cas=0 c'est-à-dire pas de combinaison
195 bool Dans_combinaison(int cas,Enum_ddl a);
196
197 /// ramène tous les membres d'une même combinaison de la même dimension
198 /// y compris "a"
199 /// cas : spécifie la combinaison :
200 /// =0 -> pas de combinaison, ramène "a"
201 /// =1 -> combinaison X V GAMMA
202 Tableau <Enum_ddl> MemeCombinaison(int cas,Enum_ddl a);
203
204 /// ramène tous les membres d'une même combinaison, pour i=1 à dim
205 /// cas = 1 -> combinaison X V GAMMA
206 /// cas = 0 -> pas de combinaison, ramène un tableau de dimension 0
207 Tableau <Enum_ddl> Combinaison(int cas);
208
209 /// récupération d'un enum de grandeurs quelconques équivalentes
210 /// -> retour particulier si il y a une équivalence particulière en grandeur évoluée par exemple
211 /// sinon, un retour sur un scalaire de type enum_évolué, qui contient de toute manière
212 /// les ddl de base
213 /// -> ramène UN_DDL_ENUM_ETENDUE s'il n'y a pas d'équivalent spécifique
214 EnumTypeQuelconque Equi_Enum_ddl_en_enu_quelconque(Enum_ddl a);
215
216 /// passage de Ui en Xi, c-a-d : UX -> X1, UY -> X2, UZ -> X3
217 Enum_ddl UxyzXi(Enum_ddl a);
218 /// passage inverse
219 Enum_ddl XiUxyz(Enum_ddl a);
220 /// surcharge de l'operator de lecture
221 istream & operator » (istream & entree, Enum_ddl& a);
222 /// surcharge de l'operator d'écriture
223 ostream & operator « (ostream & sort, const Enum_ddl& a);
224
225 ///----- pour les Coordonnee -----
226 /// retourne l'indice en fonction de l'enum et du nbcomposante
227 /// pour une grandeur de type Coordonnee, sinon erreur
228 int Indice_coor(Enum_ddl a,int nbcomposantes);
229
230 ///----- pour les tenseurs -----
231 /// fonction donnant dans l'ordre des ddl de contraintes, les indices correspondant
232 /// des tenseurs, ceci en fonction du nombre de composante de tenseur
233 /// en retour un tableau : (i)(1) et (i)(2) -> les indices (en entier) du tenseur correspondant
234 /// au i ième ddl de contrainte à partir de SIG11
235 /// mais en sautant les ddl qui ne font pas parti de la dimension !!!!!
236 Tableau2 <int> OrdreContrainte(int nbcomposantes);
237 /// idem en fonction du nombre de composantes
238 /// fonction plus rapide qui pointe sur des tableaux déjà construits à partir d'OrdreContrainte
239 const Tableau2 <int>& OrdreContrainteR(int nbcomposantes);
240 /// donne directement le premier et le second indice en fonction de l'enum et du nbcomposante
241 /// pour les tenseurs sigma ou pour le tenseur epsilon ou encore pour le tenseur Dpsilon
242 /// @addtogroup Group_types_enumeres
243 /// @{
244 /// classe utilitaire pour les enum_ddl
245 class Deuxentiers_enu {public: int i;int j;};
246 /// @} // end of group
247
248 Deuxentiers_enu IJind(Enum_ddl a,int nbcomposantes);
249
250 /// definition de tableau constant pour éviter de les recalculer à chaque appel
251 const Tableau2 <int> OrdreContrainte1 = OrdreContrainte(1);
252 const Tableau2 <int> OrdreContrainte3 = OrdreContrainte(3);
253 const Tableau2 <int> OrdreContrainte6 = OrdreContrainte(6);
254
255 //----- fin spécifique tenseur -----
256
257 /// passage de enum_ddl vers des réactions enum_ddl correspondantes
258 /// exemple X1 -> R_X1 etc..
259 Enum_ddl Vers_enum_reac(Enum_ddl a);
260 /// opération inverse: exemple R_X1 -> X1
261 Enum_ddl Enum_reac_vers_enum(Enum_ddl a);
262 /// indique si un ddl est un ddl de réaction on non

```

```

263 bool Ddl_reaction(Enum_ddl a);
264
265 // pour faire de l'inline
266 #ifndef MISE_AU_POINT
267 #include "Enum_ddl.cc"
268 #define Enum_ddl_deja_inclus
269 #endif
270
271
272 #endif

```

## 7.264 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_ddl\_var\_static.cc

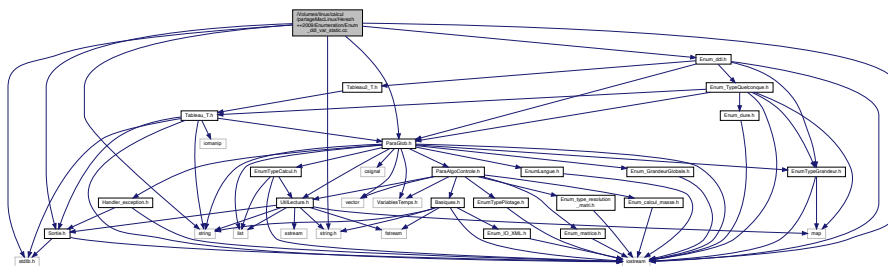
def de grandeurs statiques relatives aux Enum\_ddl.

```

#include "Enum_ddl.h"
#include "ParaGlob.h"
#include <iostream>
#include <stdlib.h>
#include "Sortie.h"
#include <string.h>
#include <string>

```

Graphe des dépendances par inclusion de Enum\_ddl\_var\_static.cc:



### 7.264.1 Description détaillée

def de grandeurs statiques relatives aux Enum\_ddl.

## 7.265 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_dure.h

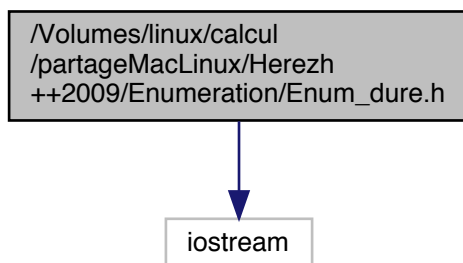
Défini une énumération en temps.

```

#include <iostream>
#include "Enum_dure.cc"

```

Grappe des dépendances par inclusion de Enum\_dure.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_dure` { `TEMPS_0 = 0` , `TEMPS_t` , `TEMPS_tdt` }
- Défini une énumération en temps.*

## Fonctions

- string `Nom_dure` (`Enum_dure id_ddl`)
- `Enum_dure Id_nom_dure` (`const string &nom_ddl`)
- bool `Existe_nom_dure` (`const string &nom_ddl`)
- `istream &operator>>` (`istream &entree`, `Enum_dure &a`)
- `ostream &operator<<` (`ostream &sort`, `const Enum_dure &a`)

### 7.265.1 Description détaillée

Défini une énumération en temps.

Date

19/01/2001

## 7.266 Enum\_dure.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_dure.h
2   \brief Définit une énumération en temps.
3 * \date      19/01/2001
4 */
5
6 // FICHER : Enum_dure.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
  
```



```

15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36
37 /*****
38 *   DATE:      19/01/2001
39 *
40 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
41 *
42 *   PROJET:    Herezh++
43 *
44 *   BUT:       Défini une énumération en temps.
45 *
46 *   *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !       but
51 *   -----
52 *   !       !       !
53 *
54 *   *****
55 *   MODIFICATIONS:
56 *   ! date !   auteur !       but
57 *   -----
58 *
59 *   *****/
60 // Afin de realiser un gain en place memoire, et une vérification de type plus aisé qu' avec
61 // les entiers
62 // 10 caractere maxi
63
64
65 #ifndef ENUM_DURE_H
66 #define ENUM_DURE_H
67
68 // #include "Debug.h"
69 #include <iostream>
70 using namespace std;
71
72 /// @addtogroup Group_types_enumeres
73 /// @{
74
75 /// Défini une énumération en temps.
76
77 enum Enum_dure { TEMPS_0 = 0, TEMPS_t , TEMPS_tdt };
78 /// @} // end of group
79
80
81 // Retourne le nom du temps a partir de son identificateur de
82 // type enumere id_ddl correspondant
83 string Nom_dure ( Enum_dure id_ddl);
84
85 // Retourne l'identificateur de type enumere associe au nom du temps
86 Enum_dure Id_nom_dure (const string& nom_ddl);
87
88 // test si l'identificateur de type enumere associe au nom du temps
89 // existe bien : ramène true s'il s'agit bien d'un type reconnu
90 bool Existe_nom_dure (const string& nom_ddl);
91
92 // surcharge de l'operator de lecture
93 istream & operator » (istream & entree, Enum_dure& a);
94 // surcharge de l'operator d'écriture
95 ostream & operator « (ostream & sort, const Enum_dure& a);
96
97 // pour faire de l'inline
98 #ifndef MISE_AU_POINT
99 #include "Enum_dure.cc"
100 #define Enum_dure_deja_inclus

```

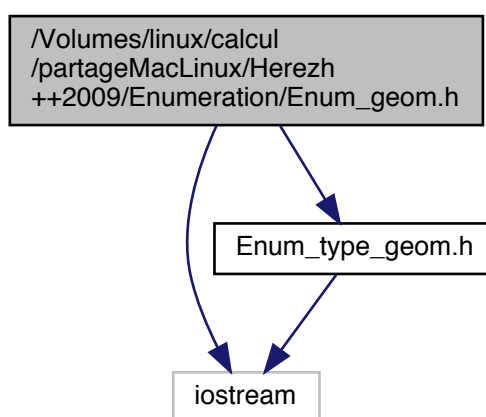
```
101 #endif
102
103 #endif
```

## 7.267 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_geom.h

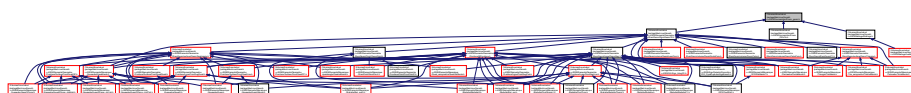
Définition de l'enuméré concernant les types de modélisations de géométrie .

```
#include <iostream>
#include "Enum_type_geom.h"
```

Grappe des dépendances par inclusion de Enum\_geom.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Énumérations

```
— enum Enum_geom {
    RIEN_GEOM = 1 , TRIANGLE , QUADRANGLE , TETRAEDRE ,
    PENTAEDRE , HEXAEDRE , SEGMENT , TRIA_AXI ,
    QUAD_AXI , SEG_AXI , POINT , POINT_CP ,
    POUT , PS1 }
```

*Définition de l'enuméré concernant les types de modélisations de géométrie .*

### Fonctions

```
— string Nom_geom (const Enum_geom id_geom)
— Enum_geom Id_nom_geom (const string &nom_geom)
— Enum_type_geom Type_geom_generique (const Enum_geom id_geom)
— bool TestEnum_geom_axisymetrique (const Enum_geom id_geom)
— istream & operator>> (istream &entree, Enum_geom &a)
— ostream & operator<< (ostream &sort, const Enum_geom &a)
```

## 7.267.1 Description détaillée

Définition de l'enuméré concernant les types de modélisations de géométrie .

## 7.268 Enum\_geom.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_geom.h
2  \brief Définition de l'enuméré concernant les types de modélisations de géométrie .
3 */
4 // FICHIER : Enum_geom.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des geometrie des elements sont
36 // stockes a l'aide d'un type enumere. Les fonctions Nom_geom et Id_nom_geom rendent
37 // possible le lien entre les noms des geometries et les identificateurs
38 // de type enumere correspondants.
39
40
41 #ifndef ENUM_GEOM_H
42 #define ENUM_GEOM_H
43 #include <iostream>
44 using namespace std;
45 #include "Enum_type_geom.h"
46
47
48 /// @addtogroup Group_types_enumeres
49 /// @{
50
51 /// Définition de l'enuméré concernant les types de modélisations de géométrie .
52
53 enum Enum_geom { RIEN_GEOM = 1, TRIANGLE , QUADRANGLE, TETRAEDRE, PENTAEDRE, HEXAEDRE, SEGMENT
54                 ,TRIA_AXI,QUAD_AXI,SEG_AXI
55                 ,POINT,POINT_CP,POUT,PS1 };
56 /// @} // end of group
57
58
59 /*** si on ajoute un élément, il faut penser à compléter l'énumération: Enum_PiPoCo
60
61 // Retourne le nom d'une geometrie a partir de son identificateur de
62 // type enumere id_geom correspondant
63 string Nom_geom (const Enum_geom id_geom);
64
65 // Retourne l'identificateur de type enumere associe au nom de geometrie
66 // nom_geom
67 Enum_geom Id_nom_geom (const string& nom_geom);
68
69 // Retourne le type de géométrie générique: POINT_G, LIGNE, SURFACE, VOLUME,
70 // associée à l'Enum_geom
71 Enum_type_geom Type_geom_generique(const Enum_geom id_geom);
72 // retourne vrai s'il s'agit d'un type axisymétrique
73 bool TestEnum_geom_axisymetrique(const Enum_geom id_geom);
74
75 // surcharge de l'operator de lecture
76 istream & operator » (istream & entree, Enum_geom& a);

```

```

77 // surcharge de l'operator d'écriture
78 ostream & operator « (ostream & sort, const Enum_geom& a);
79
80 #endif

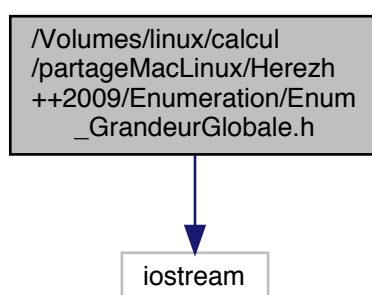
```

## 7.269 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_GrandeurGlobale.h

Définition de l'enuméré concernant les grandeurs globales.

```
#include <iostream>
```

Grphe des dépendances par inclusion de Enum\_GrandeurGlobale.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Énumérations

```

— enum Enum_GrandeurGlobale {
    ENERGIE_CINETIQUE = 0 , ENERGIE_INTERNE , ENERGIE_EXTERNE , ENERGIE_BILAN ,
    QUANTITE_MOUVEMENT , PUISSANCE_ACCELERATION , PUISSANCE_INTERNE , PUISSANCE_↔
    EXTERNE ,
    PUISSANCE_BILAN , ENERGIE_ELASTIQUE , ENERGIE_PLASTIQUE , ENERGIE_VISQUEUSE ,
    ENERGIE_HOURGLASS_ , ENERGIE_PENALISATION , ENERGIE_FROT_ELAST , ENERGIE_FROT_↔
    PLAST ,
    ENERGIE_FROT_VISQ , ENERGIE_VISCO_NUMERIQUE , ENERGIE_BULK_VISCOSITY , PUISSANCE_↔
    _BULK_VISCOSITY ,
    VOLUME_TOTAL_MATIERE , ENERGIE_STABILISATION_MEMB_BIEL , VOL_TOTAL2D_AVEC_↔
    PLAN_YZ , VOL_TOTAL2D_AVEC_PLAN_XZ ,
    VOL_TOTAL2D_AVEC_PLAN_XY , NORME_CONVERGENCE , COMPTEUR_ITERATION_ALGO_↔
    GLOBAL , MAXPUISSEXT ,
    MAXPUISSINT , MAXREACTION , MAXRESIDUGLOBAL , MAXdeltaX ,
    MAXvarDeltaX , MAXvarDdl , COMPTEUR_INCREMENT_CHARGE_ALGO_GLOBAL , AMOR_CINET_↔
    _VISQUEUX ,
    TEMPS_COURANT , ALGO_GLOBAL_ACTUEL }

```

*Définition de l'enuméré concernant les grandeurs globales.*

## Fonctions

- string **Nom\_GrandeurGlobale** (const [Enum\\_GrandeurGlobale](#) id\_GrandeurGlobale)
- [Enum\\_GrandeurGlobale](#) **Id\_nom\_GrandeurGlobale** (const string &nom\_GrandeurGlobale)
- bool **EstUneGrandeurGlobale** (const string &nom\_GrandeurGlobale)
- istream & **operator**>> (istream &entree, [Enum\\_GrandeurGlobale](#) &a)
- ostream & **operator**<< (ostream &sort, const [Enum\\_GrandeurGlobale](#) &a)

## Variables

- const int **taille\_Enum\_GrandeurGlobale** = 38

### 7.269.1 Description détaillée

Définition de l'enuméré concernant les grandeurs globales.

## 7.270 Enum\_GrandeurGlobale.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_GrandeurGlobale.h
2     \brief Définition de l'enuméré concernant les grandeurs globales
3  */
4  // FICHIER : Enum_GrandeurGlobale.h
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDLE) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des grandeurs globales
36 // sont stockes a l'aide d'un type enumere. Les fonctions Nom_GrandeurGlobale et
37 // Id_nom_GrandeurGlobale rendent possible le lien entre les noms des types d'interpolation
38 // et les identificateurs de type enumere correspondants.
39
40
41 #ifndef ENUM_GRANDEUR_GLOBALE_H
42 #define ENUM_GRANDEUR_GLOBALE_H
43 #include <iostream>
44 using namespace std;
45
46 /// @addtogroup Group_types_enumeres
47 /// @{
48
49 /// Définition de l'enuméré concernant les grandeurs globales
50
51 enum Enum_GrandeurGlobale { ENERGIE_CINETIQUE = 0, ENERGIE_INTERNE
52     , ENERGIE_EXTERNE, ENERGIE_BILAN, QUANTITE_MOUVEMENT
53     , PUISSANCE_ACCELERATION, PUISSANCE_INTERNE, PUISSANCE_EXTERNE
54     , PUISSANCE_BILAN, ENERGIE_ELASTIQUE, ENERGIE_PLASTIQUE
55     , ENERGIE_VISQUEUSE, ENERGIE_HOURGLASS_, ENERGIE_PENALISATION
56     , ENERGIE_FROT_ELAST, ENERGIE_FROT_PLAST, ENERGIE_FROT_VISQ
57     , ENERGIE_VISCO_NUMERIQUE, ENERGIE_BULK_VISCOSITY
58     , PUISSANCE_BULK_VISCOSITY, VOLUME_TOTAL_MATIERE
59     , ENERGIE_STABILISATION_MEMB_BIEL

```

```

60         ,VOL_TOTAL2D_AVEC_PLAN_YZ
61         ,VOL_TOTAL2D_AVEC_PLAN_XZ,VOL_TOTAL2D_AVEC_PLAN_XY
62         ,NORME_CONVERGENCE,COMPTEUR_ITERATION_ALGO_GLOBAL
63         ,MAXPUISSEXT,MAXPUISSINT,MAXREACTION,MAXRESIDUGLOBAL
64         ,MAXdeltaX,MAXvarDeltaX,MAXvarDdl
65         ,COMPTEUR_INCREMENT_CHARGE_ALGO_GLOBAL
66         ,AMOR_CINET_VISQUEUX
67         ,TEMPS_COURANT
68         ,ALGO_GLOBAL_ACTUEL
69     };
70 /// @} // end of group
71
72 /****** ne pas oublier de changer la taille maxi -> taille_Enum_GrandeurGlobale
73 const int taille_Enum_GrandeurGlobale = 38;
74
75 // Retourne un nom a partir de son identificateur de
76 // type enumere id_GrandeurGlobale correspondant
77 string Nom_GrandeurGlobale (const Enum_GrandeurGlobale id_GrandeurGlobale) ;
78
79 // Retourne l'identificateur de type enumere associe au nom du type
80 // nom_GrandeurGlobale
81 Enum_GrandeurGlobale Id_nom_GrandeurGlobale (const string& nom_GrandeurGlobale);
82
83 // test si le string est une grandeur globale
84 bool EstUneGrandeurGlobale(const string& nom_GrandeurGlobale);
85
86 // surcharge de l'operator de lecture
87 istream & operator » (istream & entree, Enum_GrandeurGlobale& a);
88 // surcharge de l'operator d'écriture
89 ostream & operator « (ostream & sort, const Enum_GrandeurGlobale& a);
90
91
92 #endif
93
94

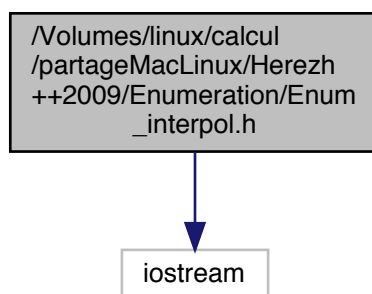
```

## 7.271 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/Enum\_interpol.h

Définition de l'enuméré concernant les différents types d'interpolation.

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_interpol.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

```
— enum Enum_interpol {
    RIEN_INTERPOL = 1, CONSTANT, LINEAIRE, QUADRATIQUE,
    QUADRACOMPL, CUBIQUE, CUBIQUE_INCOMPL, LINQUAD,
    HERMITE, SFE1, SFE2, SFE3,
    SFE3C, QSFE3, QSFE1, SFE1_3D,
    SFE2_3D, SFE3_3D, SFE3C_3D, SEG1,
    BIE1, BIE2 }
```

*Définition de l'enuméré concernant les différents types d'interpolation.*

## Fonctions

```
— string Nom_interpol (const Enum_interpol id_interpol)
— Enum_interpol Id_nom_interpol (const string &nom_interpol)
— istream & operator>>> (istream &entree, Enum_interpol &a)
— ostream & operator<<< (ostream &sort, const Enum_interpol &a)
```

### 7.271.1 Description détaillée

Définition de l'enuméré concernant les différents types d'interpolation.

## 7.272 Enum\_interpol.h

[Aller à la documentation de ce fichier.](#)

```
1 /*! \file Enum_interpol.h
2    \brief Définition de l'enuméré concernant les différents types d'interpolation
3 */
4 // FICHER : Enum_interpol.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms d'interpolation des elements
36 // sont stockes a l'aide d'un type enumere. Les fonctions Nom_interpol et
37 // Id_nom_interpol rendent possible le lien entre les noms des types d'interpolation
38 // et les identificateurs de type enumere correspondants.
39
40
41 #ifndef ENUM_INTERPOL_H
42 #define ENUM_INTERPOL_H
43 #include <iostream>
44 using namespace std;
45
46 /// @addtogroup Group_types_enumeres
47 /// @{
48
49 /// Définition de l'enuméré concernant les différents types d'interpolation
50
```

```

51 enum Enum_interpol { RIEN_INTERPOL = 1, CONSTANT
52     , LINEAIRE, QUADRATIQUE, QUADRACOMPL, CUBIQUE, CUBIQUE_INCOMPL
53     , LINQUAD, HERMITE, SFE1,SFE2,SFE3, SFE3C, QSFE3, QSFE1
54     , SFE1_3D,SFE2_3D,SFE3_3D, SFE3C_3D, SEG1, BIE1, BIE2};
55 /// @} // end of group
56
57
58 /*** si on ajoute un élément, il faut penser à compléter l'énumération: Enum_PiPoCo
59
60 // Retourne un nom d'interpolation a partir de son identificateur de
61 // type enumere id_interpol correspondant
62 string Nom_interpol (const Enum_interpol id_interpol) ;
63
64 // Retourne l'identificateur de type enumere associe au nom du type
65 // d'interpolation nom_interpol
66 Enum_interpol Id_nom_interpol (const string& nom_interpol);
67
68 // surcharge de l'operator de lecture
69 istream & operator » (istream & entree, Enum_interpol& a);
70 // surcharge de l'operator d'écriture
71 ostream & operator « (ostream & sort, const Enum_interpol& a);
72
73
74 #endif
75
76

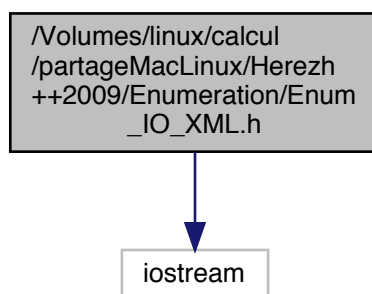
```

## 7.273 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/Enum\_IO\_XML.h

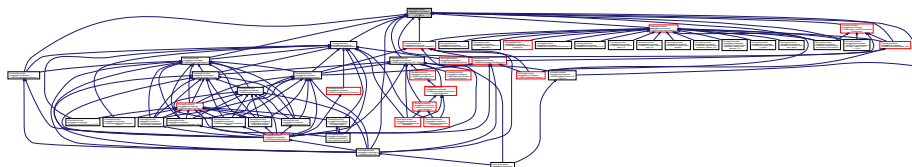
Énumération des différentes entree/sortie XML.

```
#include <iostream>
```

Grappe des dépendances par inclusion de Enum\_IO\_XML.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Énumérations

```

— enum Enum_IO_XML {
    XML_TYPE_GLOBAUX = 1 , XML_IO_POINT_INFO , XML_IO_POINT_BI , XML_IO_ELEMENT_FINI ,

```



**XML\_ACTION\_INTERACTIVE , XML\_STRUCTURE\_DONNEE }***Enumération des différentes entree/sortie XML.***Fonctions**

- string **Nom\_Enum\_IO\_XML** ([Enum\\_IO\\_XML](#) id\_Enum\_IO\_XML)
- [Enum\\_IO\\_XML](#) **Id\_nom\_Enum\_IO\_XML** (string nom\_Enum\_IO\_XML)
- istream & **operator**>> (istream &entree, [Enum\\_IO\\_XML](#) &a)
- ostream & **operator**<< (ostream &sort, const [Enum\\_IO\\_XML](#) &a)

**Variables**

- const int **nombre\_maxi\_de\_type\_de\_Enum\_IO\_XML** = 6

**7.273.1 Description détaillée**

Enumération des différentes entree/sortie XML.

Date

20/01/2005

**7.274 Enum\_IO\_XML.h**[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_IO_XML.h
2      \brief Enumération des différentes entree/sortie XML
3  * \date      20/01/2005
4  */
5
6  // FICHER : Enum_IO_XML.h
7
8  // This file is part of the Herezh++ application.
9  //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37  *      DATE:      20/01/2005
38  *
39  *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
40  *
41  *      PROJET:     Herezh++
42  *
43  *      ****
44  *      BUT: Enumération des différentes entree/sortie XML
45  *
46  *      *****
47  *
48  *      VERIFICATION:
49  *      ! date ! auteur ! but
50  *      -----
51  *      ! ! !

```

```

52 *                                     $ *
53 *      ***** *
54 *      MODIFICATIONS: *
55 *      ! date ! auteur ! but *
56 *      ----- *
57 *                                     $ *
58 *      *****/
59
60 #ifndef ENUM_ENTREE_SORTIE_XML_H
61 #define ENUM_ENTREE_SORTIE_XML_H
62
63 #include <iostream>
64 using namespace std;
65
66 /// @addtogroup Group_types_enumeres
67 /// @{
68
69 /// Énumération des différentes entree/sortie XML
70
71 enum Enum_IO_XML { XML_TYPE_GLOBAUX = 1, XML_IO_POINT_INFO, XML_IO_POINT_BI, XML_IO_ELEMENT_FINI
72                   , XML_ACTION_INTERACTIVE, XML_STRUCTURE_DONNEE};
73 /// @} // end of group
74
75 const int nombre_maxi_de_type_de_Enum_IO_XML = 6;
76
77
78
79 // Retourne un nom de type de Enum_IO_XML a partir de son identificateur de
80 // type enumere id_Enum_IO_XML correspondant
81 string Nom_Enum_IO_XML (Enum_IO_XML id_Enum_IO_XML);
82
83 // Retourne l'identificateur de type enumere associe au nom du type
84 // de Enum_IO_XML nom_Enum_IO_XML
85 Enum_IO_XML Id_nom_Enum_IO_XML (string nom_Enum_IO_XML);
86
87 // surcharge de l'operator de lecture
88 istream & operator » (istream & entree, Enum_IO_XML& a);
89 // surcharge de l'operator d'écriture
90 ostream & operator « (ostream & sort, const Enum_IO_XML& a);
91
92
93 #endif
94
95

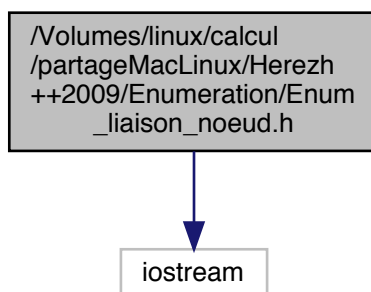
```

## 7.275 Référence du fichier /Volumes/linux/calcul/partageMacLinux/← Herezh++2009/Enumeration/Enum\_liaison\_noeud.h

Énuméré pour définir les différents types de liaison entre noeuds.

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_liaison\_noeud.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_liaison_noeud` { `PAS_LIER = 0` , `LIER_COMPLET` }  
*Enuméré pour définir les différents types de liaison entre noeuds.*

## Fonctions

- string `Nom_liaison_noeud` (`Enum_liaison_noeud id_ddl`)
- `Enum_liaison_noeud Id_nom_liaison_noeud` (const string &nom)
- istream & `operator>>` (istream &entree, `Enum_liaison_noeud &a`)
- ostream & `operator<<` (ostream &sort, const `Enum_liaison_noeud &a`)

### 7.275.1 Description détaillée

Enuméré pour définir les différents types de liaison entre noeuds.

Date

02/09/2016

## 7.276 Enum\_liaison\_noeud.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_liaison_noeud.h
2      \brief Enuméré pour définir les différents types de liaison entre noeuds.
3  * \date      02/09/2016
4  */
5
6  // FICHER : Enum_liaison_noeud.h
7
8  // This file is part of the Herezh++ application.
9  //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37  *      DATE:      02/09/2016
38  *
39  *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
40  *
41  *      PROJET:    Herezh++
42  *
43  *      ****
44  *      BUT:      Enuméré pour définir les différents types de liaison
45  *
46  *****/

```

```

45 *           entre noeuds.
46 *
47 *           *****
48 *
49 * VERIFICATION:
50 *
51 * ! date ! auteur ! but
52 * -----
53 * ! ! !
54 *
55 *           *****
56 * MODIFICATIONS:
57 * ! date ! auteur ! but
58 * -----
59 *
60 *****/
61
62 // Afin de realiser un gain en place memoire, et une vérification de type plus aisé qu' avec
63 // les entiers
64
65
66 #ifndef ENUM_LIAISON_NOEUD_H
67 #define ENUM_LIAISON_NOEUD_H
68
69 //include "Debug.h"
70 #include <iostream>
71 using namespace std;
72
73 /// @addtogroup Group_types_enumeres
74 /// @{
75
76 /// Enuméré pour définir les différents types de liaison entre noeuds.
77
78 enum Enum_liaison_noeud { PAS_LIER = 0, LIER_COMPLET };
79 /// @} // end of group
80
81
82 // Retourne le nom a partir de son identificateur de
83 // type enumere correspondant
84 string Nom_liaison_noeud ( Enum_liaison_noeud id_ddl);
85
86 // Retourne l'identificateur de type enumere associe au nom
87 Enum_liaison_noeud Id_nom_liaison_noeud (const string& nom);
88
89 // surcharge de l'operator de lecture
90 istream & operator » (istream & entree, Enum_liaison_noeud& a);
91 // surcharge de l'operator d'écriture
92 ostream & operator « (ostream & sort, const Enum_liaison_noeud& a);
93
94
95 #endif

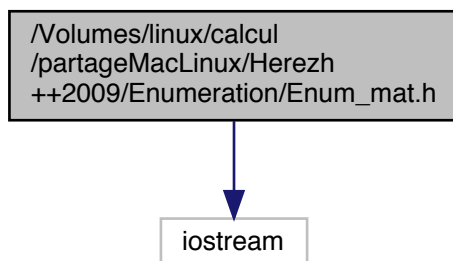
```

## 7.277 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/Enum\_mat.h

Définition de l'enuméré concernant les types de matériau.

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_mat.h:



## Énumérations

- enum `Enum_mat` { `ACIER =1` , `BETON` , `COMPOSITE` }  
*Définition de l'enuméré concernant les types de matériau.*

## Fonctions

- char \* `Nom_mat` (`Enum_mat id_mater`)
- `Enum_mat Id_nom_mat` (char \*nom\_mater)
- `istream & operator>>` (`istream &entree`, `Enum_mat &a`)
- `ostream & operator<<` (`ostream &sort`, const `Enum_mat &a`)

### 7.277.1 Description détaillée

Définition de l'enuméré concernant les types de matériau.

## 7.278 Enum\_mat.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_mat.h
2    \brief Définition de l'enuméré concernant les types de matériau.
3 */
4 // FICHER : Enum_mat.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
  
```

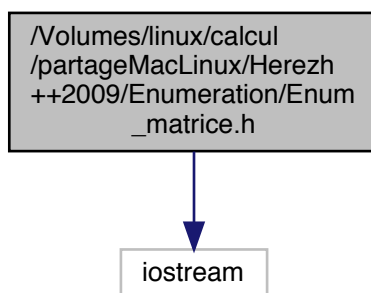
```
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms de materiaux sont
36 // stockes a l'aide d'un type enumere. Les fonctions Nom_mat et Id_nom_mat rendent
37 // possible le lien entre le nom des materiaux et les identificateurs
38 // de type enumere correspondants.
39
40
41 #ifndef ENUM_MAT_H
42 #define ENUM_MAT_H
43 #include <iostream>
44 using namespace std;
45
46
47 /// @addtogroup Group_types_enumeres
48 /// @{
49
50 /// Définition de l'enuméré concernant les types de matériau.
51
52 enum Enum_mat { ACIER=1, BETON, COMPOSITE };
53 /// @} // end of group
54
55
56
57 // Retourne le nom d'un materiau a partir de son identificateur de
58 // type enumere id_mater correspondant
59 char* Nom_mat (Enum_mat id_mater);
60
61 // Retourne l'identificateur de type enumere associe au nom de
62 // materiau nom_mater
63 Enum_mat Id_nom_mat (char* nom_mater);
64
65 // surcharge de l'operator de lecture
66 istream & operator » (istream & entree, Enum_mat& a);
67 // surcharge de l'operator d'ecriture
68 ostream & operator « (ostream & sort, const Enum_mat& a);
69
70 #endif
```

## 7.279 Référence du fichier /Volumes/linux/calcul/partageMacLinux/← Herezh++2009/Enumeration/Enum\_matrice.h

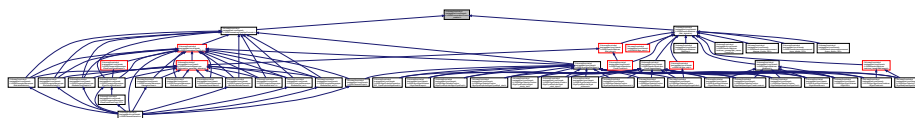
Enumeration des differents type de matrice possible.

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_matrice.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

```
— enum Enum_matrice {
    RIEN_MATRICE = 1, CARREE, CARREE_SYMETRIQUE, RECTANGLE,
    BANDE_SYMETRIQUE, BANDE_NON_SYMETRIQUE, CARREE_LAPACK, CARREE_SYMETRIQUE←
    _LAPACK,
    RECTANGLE_LAPACK, BANDE_SYMETRIQUE_LAPACK, BANDE_NON_SYMETRIQUE_LAPACK,
    TRIDIAGONALE_GENE_LAPACK,
    TRIDIAGONALE_DEF_POSITIVE_LAPACK, CREUSE_NON_COMPRESSEE, CREUSE_COMPRESSEE←
    _COLONNE, CREUSE_COMPRESSEE_LIGNE,
    DIAGONALE }
```

*Enumeration des differents type de matrice possible.*

## Fonctions

```
— string Nom_matrice (Enum_matrice id_matrice)
— Enum_matrice Id_nom_matrice (const string &nom_matrice)
— int Existe_Enum_matrice (string &nom_matrice)
— bool Symetrique_Enum_matrice (Enum_matrice id_matrice)
— istream & operator>> (istream &entree, Enum_matrice &a)
— ostream & operator<< (ostream &sort, const Enum_matrice &a)
```

### 7.279.1 Description détaillée

Enumeration des differents type de matrice possible.

Date

26/12/00

## 7.280 Enum\_matrice.h

[Aller à la documentation de ce fichier.](#)

```
1 /*! \file Enum_matrice.h
2 \brief Enumeration des differents type de matrice possible.
3 * \date 26/12/00
4 */
5
6 // FICHER : Enum_matrice.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```

29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37 *   DATE:      26/12/00
38 *
39 *   AUTEUR:    G RIO
40 *
41 *   PROJET:    Herezh++
42 *
43 * *****/
44 *   BUT: Enumeration des differents type de matrice possible.
45 *
46 *   *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !           !           !
53 *   *****
54 *
55 *   MODIFICATIONS:
56 *
57 *   ! date !   auteur !           but
58 *   -----
59 *
60 #ifndef ENUMTYPEMATRICE_H
61 #define ENUMTYPEMATRICE_H
62
63 #include <iostream>
64 using namespace std;
65
66 /// @addtogroup Group_types_enumeres
67 /// @{
68
69 /// Enumeration des differents type de matrice possible.
70
71 enum Enum_matrice { RIEN_MATRICE = 1,CARREE, CARREE_SYMETRIQUE, RECTANGLE
72 , BANDE_SYMETRIQUE, BANDE_NON_SYMETRIQUE
73 ,CARREE_LAPACK,CARREE_SYMETRIQUE_LAPACK, RECTANGLE_LAPACK,BANDE_SYMETRIQUE_LAPACK,
74 BANDE_NON_SYMETRIQUE_LAPACK
75 ,TRIDIAGONALE_GENE_LAPACK , TRIDIAGONALE_DEF_POSITIVE_LAPACK
76 ,CREUSE_NON_COMPRESSEE, CREUSE_COMPRESSEE_COLONNE, CREUSE_COMPRESSEE_LIGNE, DIAGONALE };
77 /// @} // end of group
78
79
80 // Retourne un nom de type de matrice a partir de son identificateur de
81 // type enumere id_matrice correspondant
82 string Nom_matrice(Enum_matrice id_matrice);
83
84 // Retourne l'identificateur de type enumere associe au nom du type
85 // de matrice nom_matrice
86 Enum_matrice Id_nom_matrice (const string& nom_matrice);
87
88 // indique si le type existe ou pas, -> 1 ou 0
89 int Existe_Enum_matrice(string& nom_matrice);
90
91 // indique si la matrice est de part son type, symétrique ou pas
92 bool Symetrique_Enum_matrice(Enum_matrice id_matrice);
93
94 // surcharge de l'operator de lecture
95 istream & operator » (istream & entree, Enum_matrice& a);
96 // surcharge de l'operator d'écriture
97 ostream & operator « (ostream & sort, const Enum_matrice& a);
98
99
100 #endif
101
102

```

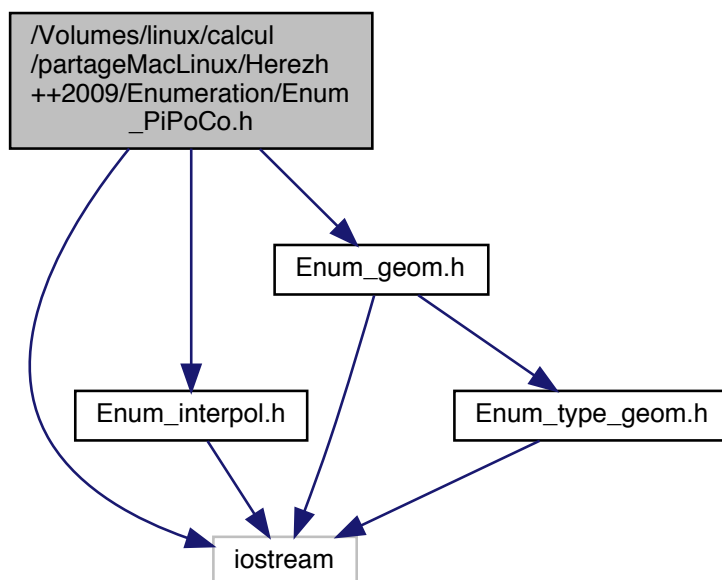
## 7.281 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/Enum\_PiPoCo.h

Définir un type énuméré pour la différenciation entre des éléments classiques et le cas poutre plaque ou coque.



```
#include <iostream>
#include "Enum_interpol.h"
#include "Enum_geom.h"
```

Graphe des dépendances par inclusion de Enum\_PiPoCo.h:



## Énumérations

- enum `Enum_PiPoCo` { `NON_PoutrePlaqueCoque = 0` , `POUTRE` , `PLAQUE` , `COQUE` }  
*Définir un type énuméré pour la différenciation entre des éléments classiques et le cas poutre plaque ou coque.*

## Fonctions

- string `Nom_Enum_PiPoCo` (const `Enum_PiPoCo` id\_Enum)
- `Enum_PiPoCo` `Id_Enum_PiPoCo` (const string &nom\_Enum)
- bool `ElementSfe` (`Enum_interpol` id\_interpol)
- `Enum_PiPoCo` `TypePiPoCo` (`Enum_interpol` id\_interpol, `Enum_geom` id\_geom)
- `istream & operator>>` (`istream` &entree, `Enum_PiPoCo` &a)
- `ostream & operator<<` (`ostream` &sort, const `Enum_PiPoCo` &a)

### 7.281.1 Description détaillée

Définir un type énuméré pour la différenciation entre des éléments classiques et le cas poutre plaque ou coque.

Date

20/06/2007

## 7.282 Enum\_PiPoCo.h

[Aller à la documentation de ce fichier.](#)

```
1 /*! \file Enum_PiPoCo.h
2   \brief Définir un type énuméré pour la différenciation entre des éléments classiques et le cas poutre
3   \date 20/06/2007
```

```

4 */
5
6
7 // This file is part of the Herezh++ application.
8 //
9 // The finite element software Herezh++ is dedicated to the field
10 // of mechanics for large transformations of solid structures.
11 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
12 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
13 //
14 // Herezh++ is distributed under GPL 3 license ou ultérieure.
15 //
16 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
17 // AUTHOR : Gérard Rio
18 // E-MAIL : gerardrio56@free.fr
19 //
20 // This program is free software: you can redistribute it and/or modify
21 // it under the terms of the GNU General Public License as published by
22 // the Free Software Foundation, either version 3 of the License,
23 // or (at your option) any later version.
24 //
25 // This program is distributed in the hope that it will be useful,
26 // but WITHOUT ANY WARRANTY; without even the implied warranty
27 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
28 // See the GNU General Public License for more details.
29 //
30 // You should have received a copy of the GNU General Public License
31 // along with this program. If not, see <https://www.gnu.org/licenses/>.
32 //
33 // For more information, please consult: <https://herezh.irdl.fr/>.
34
35 /*****
36 *   DATE:           20/06/2007                               *
37 *                                                         $ *
38 *   AUTEUR:        G RIO   (mailto:gerard.rio@univ-ubs.fr)   *
39 *                 Tel 0297874576   fax : 02.97.87.45.72     *
40 *                                                         $ *
41 *   PROJET:        Herezh++                                  *
42 *                                                         $ *
43 *****/
44 *   BUT: Définir un type énuméré pour la différenciation entre des *
45 *   éléments classiques et le cas poutre plaque ou coque.     *
46 *                                                         $ *
47 *   ***** *
48 *   *
49 *   VERIFICATION:                                           *
50 *   ! date ! auteur ! but ! *
51 *   ----- *
52 *   ! ! ! ! ! *
53 *   ! ! ! ! ! *
54 *   ***** *
55 *   MODIFICATIONS:                                           *
56 *   ! date ! auteur ! but ! *
57 *   ----- *
58 *   ! ! ! ! ! *
59 *****/
60
61
62
63 #ifndef ENUM_PIPOCO_H
64 #define ENUM_PIPOCO_H
65 #include <iostream>
66 using namespace std;
67 #include "Enum_interpol.h"
68 #include "Enum_geom.h"
69
70 /// @addtogroup Group_types_enumeres
71 /// @{
72
73 /// Définir un type énuméré pour la différenciation entre des éléments classiques et le cas poutre
74 /// plaque ou coque.
75 enum Enum_PiPoCo { NON_PoutrePlaqueCoque = 0, POUTRE , PLAQUE, COQUE };
76 /// @} // end of group
77
78
79
80
81 // Retourne un nom a partir de son identificateur de
82 // type enumere Enum_PiPoCo correspondant
83 string Nom_Enum_PiPoCo (const Enum_PiPoCo id_Enum) ;
84
85 // Retourne l'identificateur de type enumere associe au nom du type nom_Enum
86 Enum_PiPoCo Id_Enum_PiPoCo (const string& nom_Enum);
87
88 // indique si c'est un élément sfe ou non

```

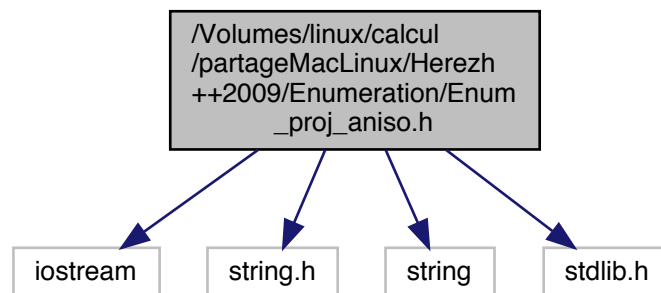
```
89 bool ElementSfe(Enum_interpol id_interpol);
90
91 // indique le type en fonction de l'interpolation et du découpage
92 Enum_PiPoCo TypePiPoCo(Enum_interpol id_interpol, Enum_geom id_geom);
93
94 // surcharge de l'operator de lecture
95 istream & operator » (istream & entree, Enum_PiPoCo& a);
96 // surcharge de l'operator d'écriture
97 ostream & operator « (ostream & sort, const Enum_PiPoCo& a);
98
99
100 #endif
101
```

## 7.283 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_proj\_aniso.h

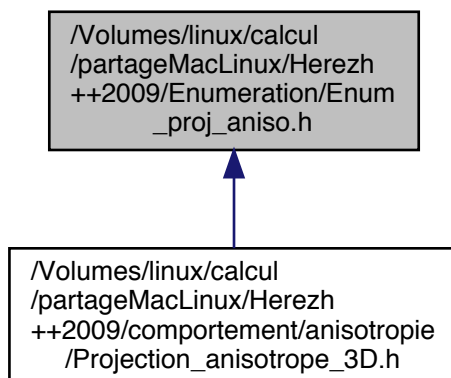
Enumeration des différentes méthodes concernant les techniques de projection anisotrope.

```
#include <iostream>
#include <string.h>
#include <string>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de Enum\_proj\_aniso.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_proj_aniso` { `AUCUNE_PROJ_ANISO = 0` , `PROJ_ORTHO` }  
*Enumeration des différentes méthodes concernant les techniques de projection anisotrope.*

## Fonctions

- string `Nom_proj_aniso` (`Enum_proj_aniso` id\_proj\_aniso)
- `Enum_proj_aniso` `Id_Nom_proj_aniso` (const string &nom\_proj\_aniso)
- bool `Type_Enum_proj_aniso_existe` (const string &nom)
- istream & `operator`>> (istream &entree, `Enum_proj_aniso` &a)
- ostream & `operator`<< (ostream &sort, const `Enum_proj_aniso` &a)

### 7.283.1 Description détaillée

Enumeration des différentes méthodes concernant les techniques de projection anisotrope.

Date

11/06/2019

## 7.284 Enum\_proj\_aniso.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_Proj_aniso.h
2   \brief Enumeration des différentes méthodes concernant les techniques de projection anisotrope
3   * \date      11/06/2019
4   */
5
6 // FICHER : Enum_Proj_aniso.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
  
```

```

20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37 *   DATE:      11/06/2019
38 *
39 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
40 *
41 *   PROJET:    Herezh++
42 *
43 *   BUT:       Enumeration des différentes méthodes concernant
44 *              les techniques de projection anisotrope
45 *
46 *   *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !           !           !
53 *   $
54 *   *****
55 *   MODIFICATIONS:
56 *   ! date !   auteur !           but
57 *   -----
58 *   $
59 *   *****/
60
61
62 #ifndef ENUM_PROJ_ANISO_H
63 #define ENUM_PROJ_ANISO_H
64 #include <iostream>
65 #include <string.h>
66 #include <string>
67 using namespace std; //introduces namespace std
68 #include <stdlib.h>
69
70 /// @addtogroup Group_types_enumeres
71 /// @{
72
73 /// Enumeration des différentes méthodes concernant les techniques de projection anisotrope
74
75 enum Enum_proj_aniso { AUCUNE_PROJ_ANISO = 0, PROJ_ORTHO };
76 /// @} // end of group
77
78
79
80 // Retourne le nom a partir de son identificateur de type enumere
81 string Nom_proj_aniso(Enum_proj_aniso id_proj_aniso);
82
83 // Retourne l'identificateur de type enumere associe au nom d'un proj_aniso
84 Enum_proj_aniso Id_Nom_proj_aniso(const string& nom_proj_aniso) ;
85
86 // Retourne vrai si le nom passé en argument représente un type de proj_aniso reconnu
87 // sinon false
88 bool Type_Enum_proj_aniso_existe(const string& nom);
89
90 // surcharge de l'operator de lecture
91 istream & operator » (istream & entree, Enum_proj_aniso& a);
92 // surcharge de l'operator d'écriture
93 ostream & operator « (ostream & sort, const Enum_proj_aniso& a);
94
95 #endif

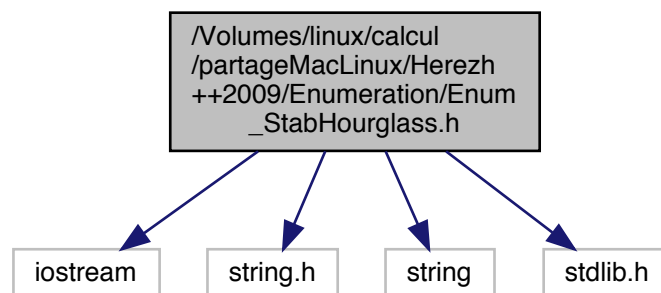
```

## 7.285 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/Enum\_StabHourglass.h

Enumeration des différentes méthodes permettant de stabiliser les modes d'hourglass.

```
#include <iostream>
#include <string.h>
#include <string>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de Enum\_StabHourglass.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_StabHourglass` { `STABHOURGLASS_NON_DEFINIE = 0` , `STABHOURGLASS_PAR_COMPORTEMENT` , `STABHOURGLASS_PAR_COMPORTEMENT_REDUIT` }
- Enumeration des différentes méthodes permettant de stabiliser les modes d'hourglass.*

## Fonctions

- string `Nom_StabHourglass` (`Enum_StabHourglass` id\_StabHourglass)
- `Enum_StabHourglass` `Id_Nom_StabHourglass` (const char \*nom\_StabHourglass)
- bool `Type_Enum_StabHourglass_existe` (const string &nom)
- istream & `operator>>` (istream &entree, `Enum_StabHourglass` &a)
- ostream & `operator<<` (ostream &sort, const `Enum_StabHourglass` &a)

### 7.285.1 Description détaillée

Enumeration des différentes méthodes permettant de stabiliser les modes d'hourglass.

Date

02/10/2010

## 7.286 Enum\_StabHourglass.h

[Aller à la documentation de ce fichier.](#)

```
1 /*! \file Enum_StabHourglass.h
2 \brief Enumeration des différentes méthodes permettant de stabiliser les modes d'hourglass
3 * \date 02/10/2010
4 */
5
6 // FICHER : Enum_StabHourglass.h
7
```

```

8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPIY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37 *   DATE:           02/10/2010
38 *
39 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
40 *
41 *   PROJET:        Herezh++
42 *
43 *   BUT:           Enumeration des différentes méthodes permettant
44 *                 de stabiliser les modes d'hourglass
45 *
46 *                 $
47 *   *****
48 *
49 *   VERIFICATION:
50 *
51 *   ! date ! auteur ! but
52 *   -----
53 *   ! ! !
54 *   $
55 *   *****
56 *   MODIFICATIONS:
57 *
58 *   ! date ! auteur ! but
59 *   -----
60 *   $
61 *   *****/
62 #ifndef ENUM_STABHOURGLASS_H
63 #define ENUM_STABHOURGLASS_H
64 #include <iostream>
65 #include <string.h>
66 #include <string>
67 using namespace std; //introduces namespace std
68 #include <stdlib.h>
69
70 /// @addtogroup Group_types_enumeres
71 /// @{
72
73 /// Enumeration des différentes méthodes permettant de stabiliser les modes d'hourglass
74
75 enum Enum_StabHourglass { STABHOURGLASS_NON_DEFINIE = 0, STABHOURGLASS_PAR_COMPORTEMENT
76 , STABHOURGLASS_PAR_COMPORTEMENT_REDUIT};
77 /// @} // end of group
78
79
80
81 // Retourne le nom a partir de son identificateur de type enumere
82 string Nom_StabHourglass (Enum_StabHourglass id_StabHourglass);
83
84 // Retourne l'identificateur de type enumere associe au nom d'une StabHourglass
85 Enum_StabHourglass Id_Nom_StabHourglass (const char* nom_StabHourglass) ;
86
87 // Retourne vrai si le nom passé en argument représente un type de StabHourglass reconnu
88 // sinon false
89 bool Type_Enum_StabHourglass_existe(const string& nom);
90
91 // surcharge de l'operator de lecture
92 istream & operator » (istream & entree, Enum_StabHourglass& a);
93 // surcharge de l'operator d'écriture

```

```

94 ostream & operator « (ostream & sort, const Enum_StabHourglass& a);
95
96 #endif

```

## 7.287 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_StabMembrane.h

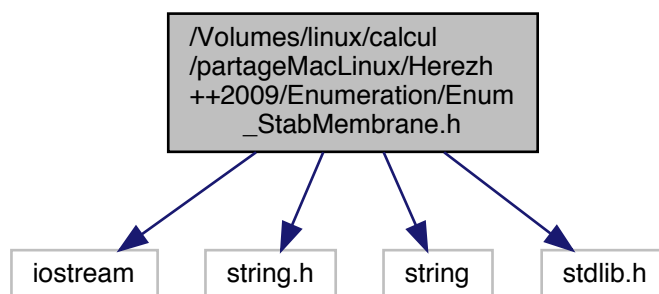
Enumeration des différentes méthodes permettant de stabiliser les membranes transversalement.

```

#include <iostream>
#include <string.h>
#include <string>
#include <stdlib.h>

```

Graphes des dépendances par inclusion de Enum\_StabMembrane.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Énumérations

```

— enum Enum_StabMembraneBiel {
    STABMEMBRANE_BIEL_NON_DEFINIE = 0, STABMEMBRANE_BIEL_PREMIER_ITER, STABMEMBRANE_
    _BIEL_PREMIER_ITER_INCR, STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER,
    STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER_INCR, STABMEMBRANE_BIEL_PREMIER_
    _ITER_NORMALE_AU_NOEUD, STABMEMBRANE_BIEL_PREMIER_ITER_INCR_NORMALE_AU_
    NOEUD, STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER_NORMALE_AU_NOEUD,
    STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER_INCR_NORMALE_AU_NOEUD, STAB_M_
    B_ITER_1_VIA_F_EXT, STAB_M_B_ITER_INCR_1_VIA_F_EXT, STAB_M_B_ITER_1_VIA_F_EXT_
    NORMALE_AU_NOEUD,
    STAB_M_B_ITER_INCR_1_VIA_F_EXT_NORMALE_AU_NOEUD }

```

*Enumeration des différentes méthodes permettant de stabiliser les membranes transversalement.*

### Fonctions

```

— string Nom_StabMembraneBiel (Enum_StabMembraneBiel id_StabMembraneBiel)
— Enum_StabMembraneBiel Id_Enum_StabMembraneBiel (const string &nom_StabMembraneBiel)
— bool Type_Enum_StabMembraneBiel_existe (const string &nom)
— bool Contient_Normale_au_noeud (Enum_StabMembraneBiel id_StabMembraneBiel)

```



- `istream & operator>>` (`istream &entree`, [Enum\\_StabMembraneBiel &a](#))
- `ostream & operator<<` (`ostream &sort`, `const Enum_StabMembraneBiel &a`)

### 7.287.1 Description détaillée

Enumeration des différentes méthodes permettant de stabiliser les membranes transversalement.

Date

30/06/2017

## 7.288 Enum\_StabMembrane.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_StabMembrane.h
2  \brief Enumeration des différentes méthodes permettant de stabiliser les membranes transversalement
3  * \date      30/06/2017
4  */
5
6
7  // This file is part of the Herezh++ application.
8  //
9  // The finite element software Herezh++ is dedicated to the field
10 // of mechanics for large transformations of solid structures.
11 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
12 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
13 //
14 // Herezh++ is distributed under GPL 3 license ou ultérieure.
15 //
16 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
17 // AUTHOR : Gérard Rio
18 // E-MAIL : gerardrio56@free.fr
19 //
20 // This program is free software: you can redistribute it and/or modify
21 // it under the terms of the GNU General Public License as published by
22 // the Free Software Foundation, either version 3 of the License,
23 // or (at your option) any later version.
24 //
25 // This program is distributed in the hope that it will be useful,
26 // but WITHOUT ANY WARRANTY; without even the implied warranty
27 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
28 // See the GNU General Public License for more details.
29 //
30 // You should have received a copy of the GNU General Public License
31 // along with this program. If not, see <https://www.gnu.org/licenses/>.
32 //
33 // For more information, please consult: <https://herezh.irdl.fr/>.
34
35 /*****
36 *      DATE:      30/06/2017
37 *
38 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
39 *
40 *      PROJET:     Herezh++
41 *
42 *      BUT:        Enumeration des différentes méthodes permettant
43 *                  de stabiliser les membranes transversalement
44 *
45 *                  $
46 *
47 *
48 *      VERIFICATION:
49 *
50 *      ! date ! auteur ! but
51 *
52 *      ! ! !
53 *
54 *      $
55 *
56 *      MODIFICATIONS:
57 *      ! date ! auteur ! but
58 *
59 *      $
60 *
61 *****/
62 #ifndef ENUM_STABMEMBRANE_BIEL_H
63 #define ENUM_STABMEMBRANE_BIEL_H
64 #include <iostream>
65 #include <string.h>
66 #include <string>
67 using namespace std; //introduces namespace std
68 #include <stdlib.h>

```

```

69
70 /// @addtogroup Group_types_enumeres
71 /// @{
72
73 /// Enumeration des différentes méthodes permettant de stabiliser les membranes transversalement
74
75 enum Enum_StabMembraneBiel { STABMEMBRANE_BIEL_NON_DEFINIE = 0
76                             , STABMEMBRANE_BIEL_PREMIER_ITER
77                             , STABMEMBRANE_BIEL_PREMIER_ITER_INCR
78                             , STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER
79                             , STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER_INCR
80                             , STABMEMBRANE_BIEL_PREMIER_ITER_NORMALE_AU_NOEUD
81                             , STABMEMBRANE_BIEL_PREMIER_ITER_INCR_NORMALE_AU_NOEUD
82                             , STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER_NORMALE_AU_NOEUD
83                             , STABMEMBRANE_BIEL_Gerschgorin_PREMIER_ITER_INCR_NORMALE_AU_NOEUD
84                             , STAB_M_B_ITER_1_VIA_F_EXT
85                             , STAB_M_B_ITER_INCR_1_VIA_F_EXT
86                             , STAB_M_B_ITER_1_VIA_F_EXT_NORMALE_AU_NOEUD
87                             , STAB_M_B_ITER_INCR_1_VIA_F_EXT_NORMALE_AU_NOEUD
88                             };
89 /// @} // end of group
90
91
92
93 // Retourne le nom a partir de son identificateur de type enumere
94 string Nom_StabMembraneBiel (Enum_StabMembraneBiel id_StabMembraneBiel);
95
96 // Retourne l'identificateur de type enumere associe au nom d'une StabMembraneBiel
97 Enum_StabMembraneBiel Id_Enum_StabMembraneBiel(const string& nom_StabMembraneBiel) ;
98
99 // Retourne vrai si le nom passé en argument représente un type de StabMembraneBiel reconnu
100 // sinon false
101 bool Type_Enum_StabMembraneBiel_existe(const string& nom);
102
103 // retourne vrai si le type enumere se termine par _NORMALE_AU_NOEUD
104 bool Contient_Normale_au_noeud(Enum_StabMembraneBiel id_StabMembraneBiel);
105
106 // surcharge de l'operator de lecture
107 istream & operator » (istream & entree, Enum_StabMembraneBiel& a);
108 // surcharge de l'operator d'écriture
109 ostream & operator « (ostream & sort, const Enum_StabMembraneBiel& a);
110
111 #endif

```

## 7.289 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/Enum\_Suite.h

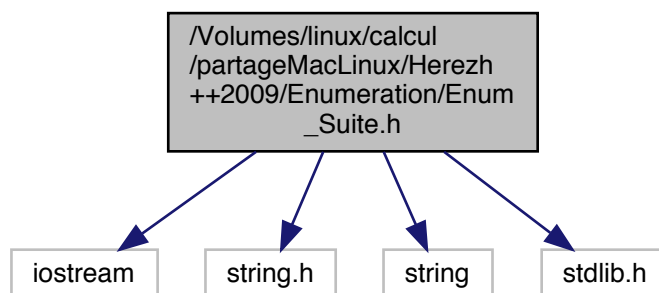
Enumeration des différentes Suites existantes.

```

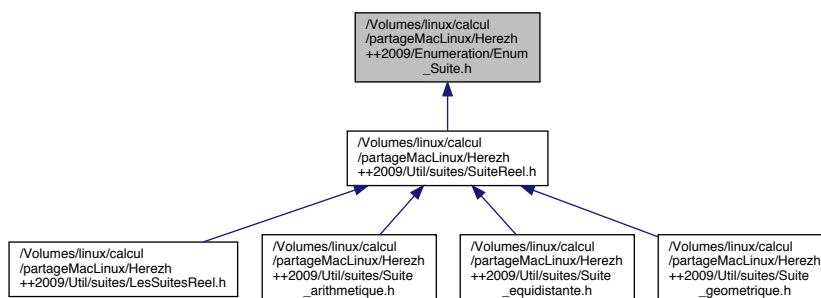
#include <iostream>
#include <string.h>
#include <string>
#include <stdlib.h>

```

Graphe des dépendances par inclusion de Enum\_Suite.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

— enum `Enum_Suite` { `SUITE_EQUIDISTANTE = 1` , `SUITE_ARITHMETIQUE` , `SUITE_GEOMETRIQUE` , `SUITE_NON_DEFINIE` }

*Enumeration des différentes Suites existantes.*

## Fonctions

- char \* `Nom_Suite` (`Enum_Suite id_Suite`)
- `Enum_Suite Id_Nom_Suite` (const char \*nom\_Suite)
- bool `Type_Enum_Suite_existe` (const string &nom)
- istream & `operator>>` (istream &entree, `Enum_Suite &a`)
- ostream & `operator<<` (ostream &sort, const `Enum_Suite &a`)

### 7.289.1 Description détaillée

Enumeration des différentes Suites existantes.

Date

19/01/2001

## 7.290 Enum\_Suite.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_Suite.h
2    \brief Enumeration des différentes Suites existantes
3 * \date    19/01/2001
4 */
5
6 // FICHER : Enum_Suite.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
  
```

```

28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37 *   DATE:      19/01/2001
38 *
39 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
40 *
41 *   PROJET:    Herezh++
42 *
43 *
44 *   BUT:      Enumeration des différentes Suites existantes
45 *
46 *   ****
47 *
48 *   VERIFICATION:
49 *   ! date !   auteur !       but
50 *   -----
51 *   !       !       !
52 *
53 *   ****
54 *   MODIFICATIONS:
55 *   ! date !   auteur !       but
56 *   -----
57 *
58 *   *****/
59
60
61 #ifndef ENUM_SUITE_H
62 #define ENUM_SUITE_H
63 #include <iostream>
64 #include <string.h>
65 #include <string>
66 using namespace std; //introduces namespace std
67 #include <stdlib.h>
68
69 /// @addtogroup Group_types_enumeres
70 /// @{
71
72 /// Enumeration des différentes Suites existantes
73
74 enum Enum_Suite { SUITE_EQUIDISTANTE = 1, SUITE_ARITHMETIQUE, SUITE_GEOMETRIQUE
75                 ,SUITE_NON_DEFINIE};
76 /// @} // end of group
77
78
79
80 // Retourne le nom d'une Suite a partir de son identificateur de
81 // type enumere id_Suite correspondant
82 char* Nom_Suite (Enum_Suite id_Suite);
83
84 // Retourne l'identificateur de type enumere associe au nom d'une Suite
85 Enum_Suite Id_Nom_Suite (const char* nom_Suite) ;
86
87 // Retourne vrai si le nom passé en argument représente un type de suite reconnu
88 // sinon false
89 bool Type_Enum_Suite_existe(const string& nom);
90
91 // surcharge de l'operator de lecture
92 istream & operator » (istream & entree, Enum_Suite& a);
93 // surcharge de l'operator d'écriture
94 ostream & operator « (ostream & sort, const Enum_Suite& a);
95
96 #endif

```

## 7.291 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/Enum\_type\_deformation.h

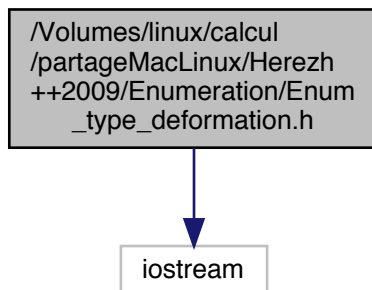
Définit une énumération des différents types de déformations.

```

#include <iostream>
#include "Enum_type_deformation.cc"

```

Graphe des dépendances par inclusion de Enum\_type\_deformation.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_type_deformation` {  
**DEFORMATION\_STANDART =1 , DEFORMATION\_POUTRE\_PLAQUE\_STANDART , DEFORMATION\_↔  
 LOGARITHMIQUE , DEF\_CUMUL\_CORROTATIONNEL ,  
 DEF\_CUMUL\_ROTATION\_PROPRE , DEFORMATION\_CUMU\_LOGARITHMIQUE }**  
*Défini une énumération des différents types de déformations.*

## Fonctions

- string **Nom\_type\_deformation** (const `Enum_type_deformation` id\_ddl)
- `Enum_type_deformation` **Id\_nom\_type\_deformation** (const string &nom\_ddl)
- istream & **operator**>> (istream &entree, `Enum_type_deformation` &a)
- ostream & **operator**<< (ostream &sort, const `Enum_type_deformation` &a)

### 7.291.1 Description détaillée

Défini une énumération des différents types de déformations.

Date

28/03/2003

## 7.292 Enum\_type\_deformation.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_type_deformation.h
2     \brief Défini une énumération des différents types de déformations
3 * \date      28/03/2003
4 */
5
6 // FICHER : Enum_type_deformation.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
  
```

```

11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPLY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36
37 /*****
38 *   DATE:      28/03/2003
39 *
40 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
41 *
42 *   PROJET:    Herezh++
43 *
44 *   *****/
45 *   BUT: Défini une énumération des différents types de déformations.*
46 *
47 *   *****/
48 *
49 *   VERIFICATION:
50 *
51 *   ! date ! auteur ! but
52 *   -----
53 *   ! ! !
54 *   *****/
55 *   MODIFICATIONS:
56 *   ! date ! auteur ! but
57 *   -----
58 *
59 *****/
60 // Afin de realiser un gain en place memoire, et une vérification de type plus aisé qu' avec
61 // les entiers
62
63
64 #ifndef ENUM_TYPE_DEFORMATION_H
65 #define ENUM_TYPE_DEFORMATION_H
66
67 // #include "Debug.h"
68 #include <iostream>
69 using namespace std;
70
71 /// @addtogroup Group_types_enumeres
72 /// @{
73
74 /// Défini une énumération des différents types de déformations
75
76 enum Enum_type_deformation { DEFORMATION_STANDART=1, DEFORMATION_POUTRE_PLAQUE_STANDART
77     ,DEFORMATION_LOGARITHMIQUE,DEF_CUMUL_CORROTATIONNEL,DEF_CUMUL_ROTATION_PROPRE
78     ,DEFORMATION_CUMU_LOGARITHMIQUE};
79 /// @} // end of group
80
81
82 // Retourne le nom du type de deformation a partir de son identificateur de
83 // type enumere id_ddl correspondant
84 string Nom_type_deformation ( const Enum_type_deformation id_ddl) ;
85
86 // Retourne l'identificateur de type enumere associe au nom du type de deformation
87 Enum_type_deformation Id_nom_type_deformation (const string& nom_ddl) ;
88
89 // surcharge de l'operator de lecture
90 istream & operator » (istream & entree, Enum_type_deformation& a);
91 // surcharge de l'operator d'écriture
92 ostream & operator « (ostream & sort, const Enum_type_deformation& a);
93
94 // pour faire de l'inline
95 #ifndef MISE_AU_POINT
96 #include "Enum_type_deformation.cc"

```

```

97  #define  Enum_type_deformation_deja_inclus
98 #endif
99
100 #endif

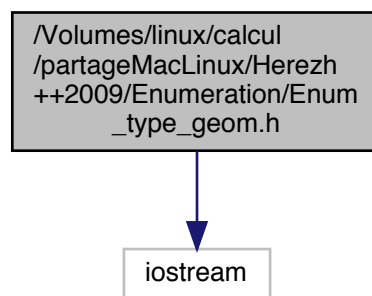
```

## 7.293 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_type\_geom.h

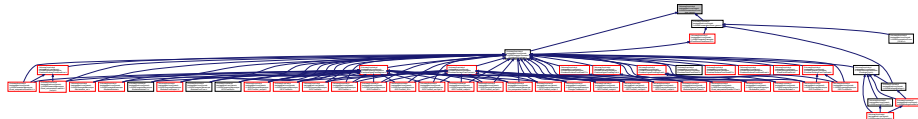
def de l'enuméré concernant les types de géométrie

```
#include <iostream>
```

Grphe des dépendances par inclusion de Enum\_type\_geom.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Énumérations

```

— enum Enum_type_geom {
    POINT_G = 1 , LIGNE , SURFACE , VOLUME ,
    RIEN_TYPE_GEOM }

```

*Définition de l'enuméré concernant les types de géométrie.*

### Fonctions

```

— string Nom_type_geom (const Enum_type_geom id_type_geom)
— Enum_type_geom Id_nom_type_geom (const string &nom_type_geom)
— istream & operator>> (istream &entree, Enum_type_geom &a)
— ostream & operator<< (ostream &sort, const Enum_type_geom &a)

```

#### 7.293.1 Description détaillée

def de l'enuméré concernant les types de géométrie

## 7.294 Enum\_type\_geom.h

Aller à la documentation de ce fichier.

```

1  /*! \file Enum_type_geom.h
2     \brief def de l'enuméré concernant les types de géométrie
3  */
4  // FICHER : Enum_type_geom.h
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software; you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des geometrie des elements sont
36 // stockes a l'aide d'un type enumere. Les fonctions Nom_type_geom et Id_nom_type_geom rendent
37 // possible le lien entre les noms des geometries et les identificateurs
38 // de type enumere correspondants.
39
40
41 #ifndef ENUM_TYPE_GEOM_H
42 #define ENUM_TYPE_GEOM_H
43 #include <iostream>
44 using namespace std;
45
46
47 /// @addtogroup Group_types_enumeres
48 /// @{
49
50 /// Définition de l'enuméré concernant les types de géométrie
51
52 enum Enum_type_geom { POINT_G = 1 , LIGNE , SURFACE, VOLUME, RIEN_TYPE_GEOM };
53 /// @} // end of group
54
55
56 // Retourne le nom d'une geometrie a partir de son identificateur de
57 // type enumere id_type_geom correspondant
58 string Nom_type_geom (const Enum_type_geom id_type_geom);
59
60 // Retourne l'identificateur de type enumere associe au nom de geometrie
61 // nom_type_geom
62 Enum_type_geom Id_nom_type_geom (const string& nom_type_geom);
63
64 // surcharge de l'operator de lecture
65 istream & operator » (istream & entree, Enum_type_geom& a);
66 // surcharge de l'operator d'écriture
67 ostream & operator « (ostream & sort, const Enum_type_geom& a);
68
69 #endif

```

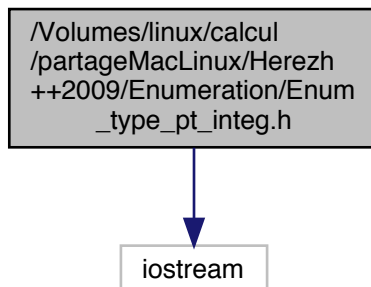
## 7.295 Référence du fichier /Volumes/linux/calcul/partageMacLinux/← Herezh++2009/Enumeration/Enum\_type\_pt\_integ.h

def de l'enuméré concernant les types de point d'intégration



```
#include <iostream>
```

Grappe des dépendances par inclusion de Enum\_type\_pt\_integ.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_type_pt_integ` { `PTI_GAUSS = 1` , `PTI_GAUSS_LOBATTO` }  
*Définition de l'enuméré concernant les types de point d'intégration.*

## Fonctions

- string `Nom_Enum_type_pt_integ` (const `Enum_type_pt_integ` id\_type\_pt\_integ)
- `Enum_type_pt_integ` `ld_nom_type_pt_integ` (const string &nom\_type\_pt\_integ)
- `istream & operator>>` (`istream &entree`, `Enum_type_pt_integ &a`)
- `ostream & operator<<` (`ostream &sort`, const `Enum_type_pt_integ &a`)

### 7.295.1 Description détaillée

def de l'enuméré concernant les types de point d'intégration

## 7.296 Enum\_type\_pt\_integ.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_type_pt_integ.h
2    \brief def de l'enuméré concernant les types de point d'intégration
3 */
4 // FICHER : Enum_type_pt_integ.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
  
```

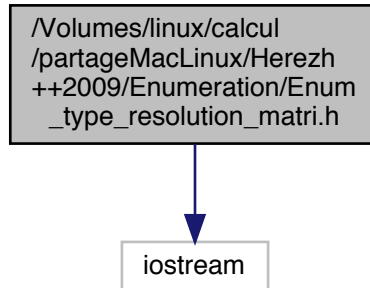
```
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms d'interpolation des elements
36 // sont stockes a l'aide d'un type enumere. Les fonctions Nom_Enum_type_pt_integ et
37 // Id_nom_type_pt_integ rendent possible le lien entre les noms des types
38 // et les identificateurs de type enumere correspondants.
39
40
41 #ifndef ENUM_TYPE_PT_INTEG_H
42 #define ENUM_TYPE_PT_INTEG_H
43 #include <iostream>
44 using namespace std;
45
46 /// @addtogroup Group_types_enumeres
47 /// @{
48
49 /// Définition de l'enuméré concernant les types de point d'intégration
50
51 enum Enum_type_pt_integ{ PTI_GAUSS = 1,PTI_GAUSS_LOBATTO};
52 /// @} // end of group
53
54
55 // Retourne un nom de type de point d'integ a partir de son identificateur de
56 // type enumere correspondant
57 string Nom_Enum_type_pt_integ (const Enum_type_pt_integ id_type_pt_integ) ;
58
59 // Retourne l'identificateur de type enumere associe au nom du type
60 // d'interpolation
61 Enum_type_pt_integ Id_nom_type_pt_integ (const string& nom_type_pt_integ);
62
63 // surcharge de l'operator de lecture
64 istream & operator » (istream & entree, Enum_type_pt_integ& a);
65 // surcharge de l'operator d'écriture
66 ostream & operator « (ostream & sort, const Enum_type_pt_integ& a);
67
68
69 #endif
70
71
```

## 7.297 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/Enum\_type\_resolution\_matri.h

Enumeration des differents type de résolution de systèmes matricielle possible, ainsi que du préconditionnement éventuel.

```
#include <iostream>
```

Graphes des dépendances par inclusion de Enum\_type\_resolution\_matri.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_type_resolution_matri` {  
**RIEN\_TYPE\_RESOLUTION\_MATRI** = 1 , **CHOLESKY** , **SYMETRISATION\_PUIS\_CHOLESKY** , **GAUSS** ,  
**GAUSS\_EXPERT** , **BI\_CONJUG** , **BI\_CONJUG\_STAB** , **CONJUG\_GRAD** ,  
**CONJUG\_GRAD\_SQUARE** , **CHEBYSHEV** , **GENE\_MINI\_RESIDUAL** , **ITERATION\_RICHARSON** ,  
**QUASI\_MINI\_RESIDUAL** , **DIRECT\_DIAGONAL** , **CRAMER** , **LU\_EQUILIBRE** }  
*Enumeration des differents type de résolution de systèmes matricielle possible.*
- enum `Enum_preconditionnement` { **RIEN\_PRECONDITIONNEMENT** = 1 , **DIAGONAL** , **ICP** , **ILU** }  
*Enumeration des differents type de préconditionnement éventuel.*

## Fonctions

- string **Nom\_resolution** (`Enum_type_resolution_matri` id\_resolution)
- `Enum_type_resolution_matri` **Id\_nom\_resolution** (string &nom\_resolution)
- int **Existe\_Enum\_type\_resolution\_matri** (string &nom\_resolution)
- istream & **operator**>> (istream &entree, `Enum_type_resolution_matri` &a)
- ostream & **operator**<< (ostream &sort, const `Enum_type_resolution_matri` &a)
- string **Nom\_preconditionnement** (`Enum_preconditionnement` id\_preconditionnement)
- `Enum_preconditionnement` **Id\_nom\_preconditionnement** (string nom\_preconditionnement)
- int **Existe\_Enum\_preconditionnement** (string &nom\_preconditionnement)
- istream & **operator**>> (istream &entree, `Enum_preconditionnement` &a)
- ostream & **operator**<< (ostream &sort, const `Enum_preconditionnement` &a)

### 7.297.1 Description détaillée

Enumeration des differents type de résolution de systèmes matricielle possible, ainsi que du préconditionnement éventuel.

Date

03/01/01

## 7.298 Enum\_type\_resolution\_matri.h

Aller à la documentation de ce fichier.

```

1  /*! \file Enum_type_resolution_matri.h
2      \brief Enumeration des differents type de résolution de systèmes matricielle possible, ainsi que du
           préconditionnement éventuel.
3  * \date      03/01/01
4  */
5
6  // FICHER : Enum_type_resolution_matri.h
7
8  // This file is part of the Herezh++ application.
9  //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37 *      DATE:      03/01/01
38 *
39 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
40 *
41 *      PROJET:    Herezh++
42 *
43 *      BUT:      Enumeration des differents type de résolution de systèmes
44 *               matricielle possible, ainsi que du préconditionnement
45 *               éventuel.
46 *
47 *               $
48 *
49 *      VERIFICATION:
50 *
51 *      ! date ! auteur ! but
52 *      -----
53 *      ! ! !
54 *               $
55 *
56 *      MODIFICATIONS:
57 *      ! date ! auteur ! but
58 *      -----
59 *               $
60 *****/
61
62 #ifndef ENUM_TYPE_RESOLUTION_MATRI_H
63 #define ENUM_TYPE_RESOLUTION_MATRI_H
64
65 #include <iostream>
66 using namespace std;
67
68 /// @addtogroup Group_types_enumeres
69 /// @{
70
71 /// Enumeration des differents type de résolution de systèmes matricielle possible.
72
73 enum Enum_type_resolution_matri { RIEN_TYPE_RESOLUTION_MATRI = 1, CHOLESKY, SYMETRISATION_PUIS_CHOLESKY,
74     GAUSS, GAUSS_EXPERT,
75     BI_CONJUG, BI_CONJUG_STAB, CONJUG_GRAD, CONJUG_GRAD_SQUARE, CHEBYSHEV, GENE_MINI_RESIDUAL
76     , ITERATION_RICHARSON, QUASI_MINI_RESIDUAL , DIRECT_DIAGONAL, CRAMER, LU_EQUILIBRE};
77     /// @} // end of group
78
79 /// @addtogroup Group_types_enumeres
80 /// @{

```

## 7.299 Référence du fichier

/Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_type\_stocke\_deformation.h3027

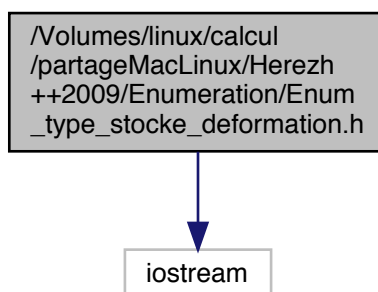
```
81
82 /// Enumeration des differents type de préconditionnement éventuel.
83
84 enum Enum_preconditionnement { RIEN_PRECONDITIONNEMENT = 1, DIAGONAL, ICP, ILU };
85     /// @} // end of group
86
87
88
89 // =====
90 //   cas du type de résolution
91 // =====
92
93 // Retourne un nom de type de résolution a partir de son identificateur de
94 // type enumere id_resolution correspondant
95 string Nom_resolution (Enum_type_resolution_matri id_resolution);
96
97 // Retourne l'identificateur de type enumere associe au nom du type
98 // de resolution nom_resolution
99 Enum_type_resolution_matri Id_nom_resolution (string& nom_resolution);
100
101 // indique si le type existe ou pas, -> 1 ou 0
102 int Existe_Enum_type_resolution_matri(string& nom_resolution);
103
104 // surcharge de l'operator de lecture
105 istream & operator » (istream & entree, Enum_type_resolution_matri& a);
106 // surcharge de l'operator d'écriture
107 ostream & operator « (ostream & sort, const Enum_type_resolution_matri& a);
108
109 // =====
110 //   cas du type de préconditionnement
111 // =====
112
113 // Retourne un nom de type de preconditionnement a partir de son identificateur de
114 // type enumere id_preconditionnement correspondant
115 string Nom_preconditionnement (Enum_preconditionnement id_preconditionnement);
116
117 // Retourne l'identificateur de type enumere associe au nom du type
118 // de preconditionnement nom_preconditionnement
119 Enum_preconditionnement Id_nom_preconditionnement (string nom_preconditionnement);
120
121 // indique si le type existe ou pas, -> 1 ou 0
122 int Existe_Enum_preconditionnement(string& nom_preconditionnement);
123
124 // surcharge de l'operator de lecture
125 istream & operator » (istream & entree, Enum_preconditionnement& a);
126 // surcharge de l'operator d'écriture
127 ostream & operator « (ostream & sort, const Enum_preconditionnement& a);
128
129 #endif
130
131
```

## 7.299 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/Enum\_type\_stocke\_deformation.h

Définir une énumération pour les type de stockage pour les données de déformations à chaque pt de gauss.

```
#include <iostream>
#include "Enum_type_stocke_deformation.cc"
```

Graphe des dépendances par inclusion de Enum\_type\_stocke\_deformation.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `Enum_type_stocke_deformation` { `SAVEDEFRESUL_GENERAL = 0` , `SAVEDEFRESUL_SFE1` }  
*Définir une énumération pour les type de stockage pour les données de déformations à chaque pt de gauss.*

## Fonctions

- `char * Nom_type_stockage_def (Enum_type_stocke_deformation id_ddl)`
- `Enum_type_stocke_deformation Id_nom_type_stockage_def (char *nom_ddl)`
- `istream & operator>> (istream &entree, Enum_type_stocke_deformation &a)`
- `ostream & operator<< (ostream &sort, const Enum_type_stocke_deformation &a)`

### 7.299.1 Description détaillée

Définir une énumération pour les type de stockage pour les données de déformations à chaque pt de gauss.

Date

25/mai/2007

## 7.300 Enum\_type\_stocke\_deformation.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_type_stocke_deformation.h
2   \brief Définir une énumération pour les type de stockage pour les données de déformations à chaque pt
3     de gauss.
4 * \date      25/mai/2007
5 */
6 // FICHER : Enum_type_stocke_deformation.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
  
```

```

15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36
37 /*****
38 *   DATE:      25/mai/2007
39 *
40 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
41 *
42 *   PROJET:    Herezh++
43 *
44 * *****/
45 *   BUT:   Définir une énumération pour les type de stockage pour
46 *          les données de déformations à chaque pt de gauss.
47 *
48 *   *****
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !
55 *   *****
56 *   MODIFICATIONS:
57 *
58 *   ! date !   auteur !           but
59 *   -----
60 *   *****/
61 // Afin de realiser un gain en place memoire, et une vérification de type plus aisé qu' avec
62 // les entiers
63 // 30 caracteres maxi
64
65
66 #ifndef ENUM_TYPE_STOCKE_DEFORMATION_H
67 #define ENUM_TYPE_STOCKE_DEFORMATION_H
68
69 // #include "Debug.h"
70 #include <iostream>
71 using namespace std;
72
73 /// @addtogroup Group_types_enumeres
74 /// @{
75
76 /// Définir une énumération pour les type de stockage pour les données de déformations à chaque pt de
77 /// gauss.
78 enum Enum_type_stocke_deformation { SAVEDEFRESUL_GENERAL = 0, SAVEDEFRESUL_SFE1 };
79 /// @} // end of group
80
81
82 // Retourne le nom du type a partir de son identificateur de
83 // type enumere id_ddl correspondant
84 char* Nom_type_stockage_def ( Enum_type_stocke_deformation id_ddl);
85
86 // Retourne l'identificateur de type enumere associe au nom du type
87 Enum_type_stocke_deformation Id_nom_type_stockage_def (char* nom_ddl);
88
89 // surcharge de l'operator de lecture
90 istream & operator » (istream & entree, Enum_type_stocke_deformation& a);
91 // surcharge de l'operator d'écriture
92 ostream & operator « (ostream & sort, const Enum_type_stocke_deformation& a);
93
94 // pour faire de l'inline
95 #ifndef MISE_AU_POINT
96 #include "Enum_type_stocke_deformation.cc"
97 #define Enum_type_stocke_deformation_deja_inclus
98 #endif
99

```

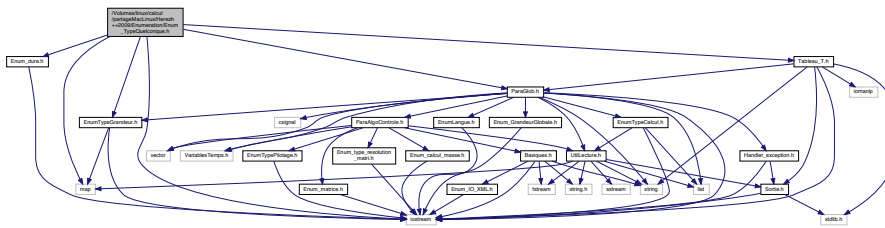
```
100 #endif
```

### 7.301 Référence du fichier /Volumes/linux/calcul/partageMacLinux/← Herezh++2009/Enumeration/Enum\_TypeQuelconque.h

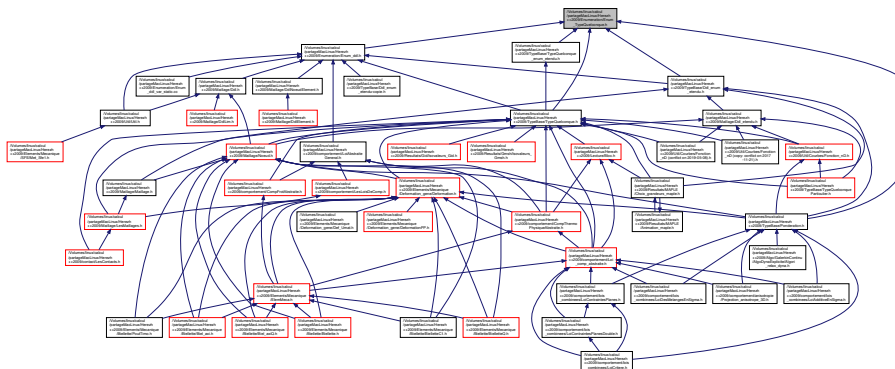
Définition de l'énuméré pour repérer un type quelconque.

```
#include <iostream>
#include <map>
#include "ParaGlob.h"
#include "EnumTypeGrandeur.h"
#include "Enum_dure.h"
#include "Tableau_T.h"
```

Graphes des dépendances par inclusion de Enum\_TypeQuelconque.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Classes

- class [ClassPourEnumTypeQuelconque](#)  
def de la map qui fait la liaison entre les string et les énumérés

## Énumérations

- enum [EnumTypeQuelconque](#) {  
RIEN\_TYPEQUELCONQUE = 1, SIGMA\_BARRE\_BH\_T, CONTRAINTE\_INDIVIDUELLE\_A\_CHAQUE\_←  
LOI\_A\_T, CONTRAINTE\_INDIVIDUELLE\_A\_CHAQUE\_LOI\_A\_T\_SANS\_PROPORTION,  
CONTRAINTE\_COURANTE, DEFORMATION\_COURANTE, VITESSE\_DEFORMATION\_COURANTE,  
ALMANSI,  
GREEN\_LAGRANGE, LOGARITHMIQUE, DELTA\_DEF, ALMANSI\_TOTAL,  
GREEN\_LAGRANGE\_TOTAL, LOGARITHMIQUE\_TOTALE, DEF\_PRINCIPALES, SIGMA\_←  
PRINCIPALES,  
VIT\_PRINCIPALES, DEF\_DUALE\_MISES, DEF\_DUALE\_MISES\_MAXI, CONTRAINTE\_MISES,



CONTRAINTE\_MISES\_T , CONTRAINTE\_TRESCA , CONTRAINTE\_TRESCA\_T , ERREUR\_Q ,  
DEF\_PLASTIQUE\_CUMULEE , ERREUR\_SIG\_RELATIVE , TEMPERATURE\_LOI\_THERMO\_PHYSIQUE  
, PRESSION\_LOI\_THERMO\_PHYSIQUE ,  
TEMPERATURE\_TRANSITION , VOLUME\_SPECIFIQUE , FLUXD , GRADT ,  
DGRADT , DELTAGRADT , COEFF\_DILATATION\_LINEAIRE , CONDUCTIVITE ,  
CAPACITE\_CALORIFIQUE , MODULE\_COMPRESSIBILITE , MODULE\_CISAILLEMENT , COEFF↵  
COMPRESSIBILITE ,  
MODULE\_COMPRESSIBILITE\_TOTAL , MODULE\_CISAILLEMENT\_TOTAL , E\_YOUNG , NU\_YOUNG ,  
MU\_VISCO , MU\_VISCO\_SPHERIQUE , MODULE\_TANGENT\_1D , COMPRESSIBILITE\_TANGENTE ,  
NB\_INVERSION , HYPER\_CENTRE\_HYSTERESIS , SIGMA\_REF , Q\_SIG\_HYST\_Oi\_A\_R ,  
Q\_SIG\_HYST\_R\_A\_T , Q\_DELTA\_SIG\_HYST , COS\_ALPHA\_HYSTERESIS , COS3PHI\_SIG↵  
HYSTERESIS ,  
COS3PHI\_DELTA\_SIG\_HYSTERESIS , FCT\_AIDE , NB\_ITER\_TOTAL\_RESIDU , NB\_INCRE\_TOTAL↵  
RESIDU ,  
NB\_APPEL\_FCT , NB\_STEP , ERREUR\_RK , PRESSION\_HYST\_REF ,  
PRESSION\_HYST , PRESSION\_HYST\_REF\_M1 , PRESSION\_HYST\_T , UN\_DDL\_ENUM\_ETENDUE ,  
ENERGIE\_ELASTIQUE\_INDIVIDUELLE\_A\_CHAQUE\_LOI\_A\_T , ENERGIE\_PLASTIQUE\_INDIVIDUELLE↵  
\_A\_CHAQUE\_LOI\_A\_T , ENERGIE\_VISQUEUSE\_INDIVIDUELLE\_A\_CHAQUE\_LOI\_A\_T , PROPORTION↵  
\_LOI\_MELANGE ,  
FONC\_PONDERATION , POSITION\_GEOMETRIQUE , POSITION\_GEOMETRIQUE\_t , POSITION↵  
GEOMETRIQUE\_t0 ,  
CRISTALINITE , VOLUME\_ELEMENT , VOLUME\_PTI , EPAISSEUR\_MOY\_INITIALE ,  
EPAISSEUR\_MOY\_FINALE , SECTION\_MOY\_INITIALE , SECTION\_MOY\_FINALE , EPAISSEUR↵  
INITIALE ,  
EPAISSEUR\_FINALE , SECTION\_INITIALE , SECTION\_FINALE , VOL\_ELEM\_AVEC\_PLAN\_REF ,  
INTEG\_SUR\_VOLUME , INTEG\_SUR\_VOLUME\_ET\_TEMPS , STATISTIQUE , STATISTIQUE\_ET\_TEMPS  
,  
ENERGIE\_HOURGLASS , PUISSANCE\_BULK , ENERGIE\_BULK , ENERGIE\_STABMEMB\_BIEL ,  
FORCE\_STABMEMB\_BIEL , TENSEUR\_COURBURE , COURBURES\_PRINCIPALES , DIRECTIONS↵  
PRINC\_COURBURE ,  
DIRECTIONS\_PRINC\_SIGMA , DIRECTIONS\_PRINC\_DEF , DIRECTIONS\_PRINC\_D , REPERE↵  
LOCAL\_ORTHO ,  
REPERE\_LOCAL\_H , REPERE\_LOCAL\_B , REPERE\_D\_ANISOTROPIE , EPS\_TRANSPORTEE\_ANISO  
,  
SIGMA\_DANS\_ANISO , DELTA\_EPS\_TRANSPORTEE\_ANISO , DELTA\_SIGMA\_DANS\_ANISO , PARA↵  
\_ORTHO ,  
SPHERIQUE\_EPS , Q\_EPS , COS3PHI\_EPS , SPHERIQUE\_SIG ,  
Q\_SIG , COS3PHI\_SIG , SPHERIQUE\_DEPS , V\_vol ,  
Q\_DEPS , COS3PHI\_DEPS , POTENTIEL , FCT\_POTENTIEL\_ND ,  
INVAR\_B1 , INVAR\_B2 , INVAR\_B3 , INVAR\_J1 ,  
INVAR\_J2 , INVAR\_J3 , DEF\_EQUIVALENTE , DEF\_EPAISSEUR ,  
D\_EPAISSEUR , DEF\_LARGEUR , D\_LARGEUR , DEF\_MECHANIQUE ,  
DEF ASSO\_LOI , DEF\_P\_DANS\_V\_A , SIG\_EPAISSEUR , SIG\_LARGEUR ,  
FORCE\_GENE\_EXT , FORCE\_GENE\_INT , FORCE\_GENE\_TOT , RESIDU\_GLOBAL ,  
FORCE\_GENE\_EXT\_t , FORCE\_GENE\_INT\_t , VECT\_PRESSION , PRESSION\_SCALEIRE ,  
VECT\_FORCE\_VOLUM , VECT\_DIR\_FIXE , VECT\_SURF\_SUIV , VECT\_HYDRODYNA\_Fn ,  
VECT\_HYDRODYNA\_Ft , VECT\_HYDRODYNA\_T , VECT\_LINE , VECT\_LINE\_SUIV ,  
VECT\_REAC , VECT\_REAC\_N , NN\_11 , NN\_22 ,  
NN\_33 , NN\_12 , NN\_13 , NN\_23 ,  
MM\_11 , MM\_22 , MM\_33 , MM\_12 ,  
MM\_13 , MM\_23 , DIRECTION\_PLI , DIRECTION\_PLI\_NORMEE ,  
INDIC\_CAL\_PLIS , NN\_SURF , NN\_SURF\_t , NN\_SURF\_t0 ,  
NOEUD\_PROJECTILE\_EN\_CONTACT , NOEUD\_FACETTE\_EN\_CONTACT , GLISSEMENT\_CONTACT ,  
PENETRATION\_CONTACT ,  
GLISSEMENT\_CONTACT\_T , PENETRATION\_CONTACT\_T , FORCE\_CONTACT , FORCE\_CONTACT↵  
\_T ,  
CONTACT\_NB\_PENET , CONTACT\_NB\_DECOL , CONTACT\_CAS\_SOLIDE , CONTACT\_ENERG↵  
PENAL ,

```

CONTACT_COLLANT , NUM_ZONE_CONTACT , CONTACT_ENERG_GLISSSE_ELAS , CONTACT_↵
ENERG_GLISSSE_PLAS ,
CONTACT_ENERG_GLISSSE_VISQ , CONTACT_PENALISATION_N , CONTACT_PENALISATION_T ,
NORMALE_CONTACT ,
TEMPS_CPU_USER , TEMPS_CPU_LOI_COMP , TEMPS_CPU_METRIQUE , GENERIQUE_UNE_↵
GRANDEUR_GLOBALE ,
GENERIQUE_UNE_CONSTANTE_GLOB_INT_UTILISATEUR , GENERIQUE_UNE_CONSTANTE_↵
GLOB_DOUBLE_UTILISATEUR , GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_0
, GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_T ,
GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_TDT , DEPLACEMENT , VITESSE ,
DELTA_XI ,
XI_ITER_0 , MASSE_RELAX_DYN , COMP_TORSEUR_REACTION , NUM_NOEUD ,
NUM_MAIL_NOEUD , NUM_ELEMENT , NUM_MAIL_ELEM , NUM_PTI ,
NUM_FACE , NUM_ARETE }

```

*Définition de l'énuméré pour repérer un type quelconque.*

## Fonctions

- string **NomTypeQuelconque** ([EnumTypeQuelconque](#) id\_TypeQuelconque)
- string **NomTypeQuelconque\_court** ([EnumTypeQuelconque](#) id\_TypeQuelconque)
- string **NomGeneriqueTypeQuelconque** ([EnumTypeQuelconque](#) id\_TypeQuelconque)
- bool **Existe\_typeQuelconque** (const string &nom)
- [EnumTypeQuelconque](#) **Id\_nomTypeQuelconque** (const char \*nom\_TypeQuelconque)
- [EnumTypeQuelconque](#) **Id\_nomTypeQuelconque** (const string &nom\_TypeQuelconque)
- [EnumTypeGrandeur](#) **Type\_de\_grandeur\_associee** ([EnumTypeQuelconque](#) typa)
- [Enum\\_dure](#) **EnumTypeQuelconqueTemps** ([EnumTypeQuelconque](#) typa)
- int **EnumTypeQuelconqueGlobale** ([EnumTypeQuelconque](#) typa)
- istream & **operator**>> (istream &entree, [EnumTypeQuelconque](#) &a)
- ostream & **operator**<< (ostream &sort, const [EnumTypeQuelconque](#) &a)

### 7.301.1 Description détaillée

Définition de l'énuméré pour repérer un type quelconque.

## 7.302 Enum\_TypeQuelconque.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Enum_TypeQuelconque.h
2    \brief Définition de l'énuméré pour repérer un type quelconque
3 */
4 // FICHER : Enum_TypeQuelconque.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33

```

```

34
35 // Afin de realiser un gain en place memoire, les noms des types de grandeurs sont
36 // stockes a l'aide d'un type enumere. Les fonctions NomTypeQuelconque et Id_nomTypeQuelconque rendent
37 // possible le lien entre les noms des geometries et les identificateurs
38 // de type enumere correspondants.
39
40
41 #ifndef ENUMTYPEQUELCONQUE_H
42 #define ENUMTYPEQUELCONQUE_H
43 #include <iostream>
44 using namespace std;
45 #include <map>
46 #include "ParaGlob.h"
47 #include "EnumTypeGrandeur.h"
48 #include "Enum_dure.h"
49 #include "Tableau_T.h"
50
51 /// @addtogroup Group_types_enumeres
52 /// @{
53
54 /// Définition de l'enuméré pour repérer un type quelconque
55
56 enum EnumTypeQuelconque { RIEN_TYPEQUELCONQUE = 1, SIGMA_BARRE_BH_T
57     , CONTRAINTE_INDIVIDUELLE_A_CHAQUE_LOI_A_T
58     , CONTRAINTE_INDIVIDUELLE_A_CHAQUE_LOI_A_T_SANS_PROPORTION
59     , CONTRAINTE_COURANTE, DEFORMATION_COURANTE, VITESSE_DEFORMATION_COURANTE
60     , ALMANSI, GREEN_LAGRANGE, LOGARITHMIQUE, DELTA_DEF
61     , ALMANSI_TOTAL, GREEN_LAGRANGE_TOTAL, LOGARITHMIQUE_TOTALE
62
63     , DEF_PRINCIPALES, SIGMA_PRINCIPALES, VIT_PRINCIPALES, DEF_DUALE_MISES, DEF_DUALE_MISES_MAXI
64     , CONTRAINTE_MISES, CONTRAINTE_MISES_T, CONTRAINTE_TRESCA, CONTRAINTE_TRESCA_T
65     , ERREUR_Q, DEF_PLASTIQUE_CUMULEE
66     , ERREUR_SIG_RELATIVE
67     , TEMPERATURE_LOI_THERMO_PHYSIQUE, PRESSION_LOI_THERMO_PHYSIQUE
68     , TEMPERATURE_TRANSITION, VOLUME_SPECIFIQUE
69     , FLUXD, GRADT, DGRADT, DELTAGRADT
70     , COEFF_DILATATION_LINEAIRE, CONDUCTIVITE, CAPACITE_CALORIFIQUE
71     , MODULE_COMPRESSIBILITE, MODULE_CISAILLEMENT, COEFF_COMPRESSIBILITE
72     , MODULE_COMPRESSIBILITE_TOTAL, MODULE_CISAILLEMENT_TOTAL
73     , E_YOUNG, NU_YOUNG, MU_VISCO, MU_VISCO_SPHERIQUE
74     , MODULE_TANGENT_1D, COMPRESSIBILITE_TANGENTE
75
76     , NB_INVERSION, HYPER_CENTRE_HYSTERESIS, SIGMA_REF, Q_SIG_HYST_Oi_A_R, Q_SIG_HYST_R_A_T
77
78     , Q_DELTA_SIG_HYST, COS_ALPHA_HYSTERESIS, COS3PHI_SIG_HYSTERESIS, COS3PHI_DELTA_SIG_HYSTERESIS
79     , FCT_AIDE
80     , NB_ITER_TOTAL_RESIDU, NB_INCRE_TOTAL_RESIDU, NB_APPEL_FCT, NB_STEP, ERREUR_RK
81     , PRESSION_HYST_REF, PRESSION_HYST, PRESSION_HYST_REF_M1, PRESSION_HYST_T
82
83     , UN_DDL_ENUM_ETENDUE, ENERGIE_ELASTIQUE_INDIVIDUELLE_A_CHAQUE_LOI_A_T
84     , ENERGIE_PLASTIQUE_INDIVIDUELLE_A_CHAQUE_LOI_A_T
85     , ENERGIE_VISQUEUSE_INDIVIDUELLE_A_CHAQUE_LOI_A_T
86     , PROPORTION_LOI_MELEGE, FONC_PONDERATION
87
88     , POSITION_GEOMETRIQUE, POSITION_GEOMETRIQUE_t, POSITION_GEOMETRIQUE_t0
89     , CRISTALINITE
90     , VOLUME_ELEMENT, VOLUME_PTI, EPAISSEUR_MOY_INITIALE, EPAISSEUR_MOY_FINALE
91     , SECTION_MOY_INITIALE, SECTION_MOY_FINALE
92     , EPAISSEUR_INITIALE, EPAISSEUR_FINALE, SECTION_INITIALE, SECTION_FINALE
93
94     , VOL_ELEM_AVEC_PLAN_REF, INTEG_SUR_VOLUME, INTEG_SUR_VOLUME_ET_TEMPS
95     , STATISTIQUE, STATISTIQUE_ET_TEMPS
96
97     , ENERGIE_HOURLASS, PUISSANCE_BULK, ENERGIE_BULK, ENERGIE_STABMEMB_BIEL, FORCE_STABMEMB_BIEL
98     , TENSEUR_COURBURE, COURBURES_PRINCIPALES, DIRECTIONS_PRINC_COURBURE
99     , DIRECTIONS_PRINC_SIGMA, DIRECTIONS_PRINC_DEF, DIRECTIONS_PRINC_D
100     , REPERE_LOCAL_ORTHO, REPERE_LOCAL_H, REPERE_LOCAL_B
101     , REPERE_D_ANISOTROPIE, EPS_TRANSPORTEE_ANISO, SIGMA_DANS_ANISO
102     , DELTA_EPS_TRANSPORTEE_ANISO, DELTA_SIGMA_DANS_ANISO
103     , PARA_ORTHO
104     , SPHERIQUE_EPS, Q_EPS, COS3PHI_EPS, SPHERIQUE_SIG, Q_SIG, COS3PHI_SIG
105
106     , SPHERIQUE_DEPS, V_vol, Q_DEPS, COS3PHI_DEPS, POTENTIEL, FCT_POTENTIEL_ND
107     , INVAR_B1, INVAR_B2, INVAR_B3, INVAR_J1, INVAR_J2, INVAR_J3
108     , DEF_EQUIVALENTE, DEF_EPAISSEUR, D_EPAISSEUR, DEF_LARGEUR, D_LARGEUR
109     , DEF_MECAENIQUE, DEF ASSO_LOI, DEF_P_DANS_V_A
110     , SIG_EPAISSEUR, SIG_LARGEUR
111     , FORCE_GENE_EXT, FORCE_GENE_INT, FORCE_GENE_TOT, RESIDU_GLOBAL
112     , FORCE_GENE_EXT_t, FORCE_GENE_INT_t
113     , VECT_PRESSION, PRESSION_SCALAIRE, VECT_FORCE_VOLUM
114
115     , VECT_DIR_FIXE, VECT_SURF_SUIV, VECT_HYDRODYNA_Fn, VECT_HYDRODYNA_Ft, VECT_HYDRODYNA_T
116     , VECT_LINE, VECT_LINE_SUIV, VECT_REAC, VECT_REAC_N
117     , NN_11, NN_22, NN_33, NN_12, NN_13, NN_23, MM_11, MM_22, MM_33, MM_12, MM_13, MM_23
118     , DIRECTION_PLI, DIRECTION_PLI_NORMEE, INDIC_CAL_PLIS
119     , NN_SURF, NN_SURF_t, NN_SURF_t0
120     , NOEUD_PROJECTILE_EN_CONTACT, NOEUD_FACETTE_EN_CONTACT

```

```

112         ,GLISSEMENT_CONTACT,PENETRATION_CONTACT
113         ,GLISSEMENT_CONTACT_T,PENETRATION_CONTACT_T
114         ,FORCE_CONTACT,FORCE_CONTACT_T,CONTACT_NB_PENET
115         ,CONTACT_NB_DECOL,CONTACT_CAS_SOLIDE
116         ,CONTACT_ENERG_PENAL,CONTACT_COLLANT,NUM_ZONE_CONTACT
117         ,CONTACT_ENERG_GLISSSE_ELAS,CONTACT_ENERG_GLISSSE_PLAS,CONTACT_ENERG_GLISSSE_VISQ
118         ,CONTACT_PENALISATION_N,CONTACT_PENALISATION_T
119         ,NORMALE_CONTACT
120         ,TEMPS_CPU_USER,TEMPS_CPU_LOI_COMP,TEMPS_CPU_METRIQUE
121         ,GENERIQUE_UNE_Grandeur_GLOBALE
122         ,GENERIQUE_UNE_Constante_GLOB_INT_UTILISATEUR
123         ,GENERIQUE_UNE_Constante_GLOB_DOUBLE_UTILISATEUR
124         ,GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_0
125         ,GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_T
126         ,GENERIQUE_UNE_VARIABLE_GLOB_DOUBLE_UTILISATEUR_TDT
127         ,DEPLACEMENT, VITESSE
128         ,DELTA_XI,XI_ITER_0,MASSE_RELAX_DYN
129         ,COMP_TORSEUR_REACTION
130         ,NUM_NOEUD,NUM_MAIL_NOEUD,NUM_ELEMENT,NUM_MAIL_ELEM,NUM_PTI
131         ,NUM_FACE,NUM_ARETE
132     };
133 /// @} // end of group
134
135 //-----*****-----
136 // en fonction des ajouts voir pour mettre à jour éventuellement
137 // EnumTypeQuelconque Equi_Enum_ddl_en_enu_quelconque(Enum_ddl a);
138 //-----*****-----
139
140 /// @addtogroup Group_types_enumeres
141 /// @{
142
143 /// def de la map qui fait la liaison entre les string et les énumérés
144
145 class ClassPourEnumTypeQuelconque
146 { public:
147     // def de la map qui fait la liaison entre les string et les énumérés
148     static map < string, EnumTypeQuelconque , std::less < string> > map_EnumTypeQuelconque;
149     // def de la grandeur statique qui permet de remplir la map
150     static ClassPourEnumTypeQuelconque remplir_map;
151     // le constructeur qui rempli effectivement la map
152     ClassPourEnumTypeQuelconque ();
153     // indique le type d'expression de la grandeur
154     static Tableau < int > tt_GLOB; // = 1 indique si la grandeur est naturellement exprimé dans le
        repère globale
155                                     // = 0 indique que la grandeur est dans le repère naturelle (ou une
        combinaison)
156     // indique à quel temps la grandeur est calculée
157     static Tableau < Enum_dure > tt_TQ_temps;
158     // nombre de grandeur énumérée existante
159     static int NombreEnumTypeQuelconque () {return tt_GLOB.Taille();};
160 };
161 /// @} // end of group
162
163
164 // Retourne le nom du type de grandeur a partir de son identificateur de
165 // type enumere id_TypeQuelconque correspondant
166 string NomTypeQuelconque (EnumTypeQuelconque id_TypeQuelconque);
167 // idem mais en version courte (sur quelques caractères)
168 string NomTypeQuelconque_court (EnumTypeQuelconque id_TypeQuelconque);
169 // nom générique, lorsqu'il s'agit de composantes
170 string NomGeneriqueTypeQuelconque (EnumTypeQuelconque id_TypeQuelconque);
171
172 // indique si le string correspond à un type quelconque reconnu ou non
173 bool Existe_typeQuelconque(const string& nom);
174
175 // Retourne l'identificateur de type enumere associe au nom du type de grandeur
176 // nom_TypeQuelconque
177 EnumTypeQuelconque Id_nomTypeQuelconque (const char* nom_TypeQuelconque);
178 EnumTypeQuelconque Id_nomTypeQuelconque (const string& nom_TypeQuelconque);
179
180 // Retourne le type de grandeur associée au type quelconque
181 EnumTypeGrandeur Type_de_grandeur_associee(EnumTypeQuelconque typa);
182 // retourne le temps auquel la grandeur est définie
183 Enum_dure EnumTypeQuelconqueTemps (EnumTypeQuelconque typa);
184 // retour du type d'expression de la grandeur
185 // = 1 indique si la grandeur est naturellement exprimé dans le repère globale
186 // = 0 indique que la grandeur est dans le repère naturelle (ou une combinaison)
187 int EnumTypeQuelconqueGlobale (EnumTypeQuelconque typa);
188 // surcharge de l'operator de lecture
189 istream & operator » (istream & entree, EnumTypeQuelconque& a);
190 // surcharge de l'operator d'écriture
191 ostream & operator « (ostream & sort, const EnumTypeQuelconque& a);
192
193 #endif

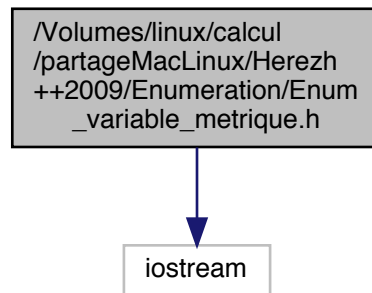
```

## 7.303 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/Enum\_variable\_metrique.h

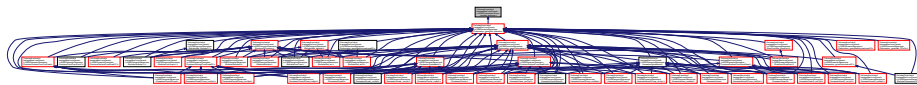
Définition de l'enuméré concernant les grandeurs gérées par les classes métriques.

```
#include <iostream>
```

Graphe des dépendances par inclusion de Enum\_variable\_metrique.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Énumérations

```

— enum Enum_variable_metrique {
    iM0 = 1, iMt, idMt, iMtdt,
    idMtdt, iV0, iVt, idVt,
    iVtdt, idVtdt, igiB_0, igiB_t,
    igiB_tdt, igiH_0, igiH_t, igiH_tdt,
    igijBB_0, igijBB_t, igijBB_tdt, igijHH_0,
    igijHH_t, igijHH_tdt, id_giB_t, id_giB_tdt,
    id_giH_t, id_giH_tdt, id_gijBB_t, id_gijBB_tdt,
    id_gijHH_t, id_gijHH_tdt, id_jacobien_t, id_jacobien_tdt,
    id2_gijBB_tdt, igradVmoyBB_t, igradVmoyBB_tdt, igradVBB_t,
    igradVBB_tdt, id_gradVmoyBB_t, id_gradVmoyBB_tdt, id_gradVBB_t,
    id_gradVBB_tdt }
  
```

*Définition de l'enuméré concernant les grandeurs gérées par les classes métriques générales.*

```

— enum Enum_variable_metsfe {
    iP0 = 1, iPt, idPt, iPtdt,
    idPtdt, iaiB_0, iaiB_t, iaiB_tdt,
    iaiH_0, iaiH_t, iaiH_tdt, iaijBB_0,
    iaijBB_t, iaijBB_tdt, iaijHH_0, iaijHH_t,
    iaijHH_tdt, id_aiB_t, id_aiB_tdt, id_aiH_t,
    id_aiH_tdt, id_aijBB_t, id_aijBB_tdt, id_aijHH_t,
    id_aijHH_tdt, id_ajacobien_t, id_ajacobien_tdt }
  
```

*Définition de l'enuméré concernant les grandeurs spécifiques gérées par les classes métriques plaques, coques et poutres.*

## Fonctions

- `char * Nom_Enum_variable_metrique (Enum_variable_metrique id_nom)`
- `Enum_variable_metrique Id_nom_Enum_variable_metrique (char *nom)`
- `char * Nom_Enum_variable_metsfe (Enum_variable_metsfe id_nom)`
- `Enum_variable_metsfe Id_nom_Enum_variable_metsfe (char *nom)`
- `istream & operator>> (istream &entree, Enum_variable_metrique &a)`
- `ostream & operator<< (ostream &sort, const Enum_variable_metrique &a)`
- `istream & operator>> (istream &entree, Enum_variable_metsfe &a)`
- `ostream & operator<< (ostream &sort, const Enum_variable_metsfe &a)`

### 7.303.1 Description détaillée

Définition de l'enuméré concernant les grandeurs gérées par les classes métriques.

## 7.304 Enum\_variable\_metrique.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Enum_variable_metrique.h
2     \brief Définition de l'enuméré concernant les grandeurs gérées par les classes métriques
3  */
4
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // FICHER : Enum_variable_metrique.h
36
37 // Afin de realiser un gain en place memoire, les noms des variables privees utilisees
38 // dans la classe Met_abstraite sont
39 // stockes a l'aide d'un type enumere. Les fonctions Nom_Enum_variable_metrique
40 // et Id_nom_Enum_variable_metrique rendent
41 // possible le lien entre le nom en caractère et les identificateurs
42 // de type enumere correspondants.
43
44
45 #ifndef ENUM_ENUM_VARIABLE_METRIQUE_H
46 #define ENUM_ENUM_VARIABLE_METRIQUE_H
47 #include <iostream>
48 using namespace std;
49
50
51 /// @addtogroup Group_types_enumeres
52 /// @{
53
54 /// Définition de l'enuméré concernant les grandeurs gérées par les classes métriques générales
55
56 enum Enum_variable_metrique { iM0 =1, iMt, idMt, iMtdt, idMtdt, iV0, iVt, idVt, iVtdt, idVtdt
57     , igiB_0 , igiB_t, igiB_tdt,
58     igiH_0, igiH_t, igiH_tdt, igijBB_0, igijBB_t, igijBB_tdt, igijHH_0, igijHH_t,
59     igijHH_tdt, id_giB_t, id_giB_tdt,
60     id_giH_t, id_giH_tdt, id_gijBB_tdt, id_gijHH_t, id_gijHH_tdt,
61     id_jacobien_t , id_jacobien_tdt

```

```

62         , id2_gijBB_tdt // variation seconde de la déformation
63         , igradVmoyBB_t, igradVmoyBB_tdt, igradVBB_t, igradVBB_tdt // gradient de vitesse moyen et
instantannée
64         , id_gradVmoyBB_t, id_gradVmoyBB_tdt, id_gradVBB_t, id_gradVBB_tdt // variation du gradient de
vitesse
65     };
66 /// @} // end of group
67
68 /// @addtogroup Group_types_enumeres
69 /// @{
70
71 /// Définition de l'enuméré concernant les grandeurs spécifiques gérées par les classes métriques
plaques, coques et poutres
72
73 enum Enum_variable_metsfe { iP0 =1, iPt, idPt, iPtdt, idPtdt
74     , iaiB_0 , iaiB_t, iaiB_tdt,
75     iaiH_0, iaiH_t, iaiH_tdt, iaijBB_0, iaijBB_t, iaijBB_tdt, iaijHH_0, iaijHH_t,
76     iaijHH_tdt, id_aiB_t, id_aiB_tdt,
77     id_aiH_t, id_aiH_tdt, id_aijBB_t, id_aijBB_tdt, id_aijHH_t, id_aijHH_tdt,
78     id_ajacobien_t , id_ajacobien_tdt
79     };
80 /// @} // end of group
81
82
83 // Retourne la chaine a partir de son identificateur de
84 // type enumere id_nom correspondant
85 char* Nom_Enum_variable_metrrique (Enum_variable_metrrique id_nom);
86
87 // Retourne l'identificateur de type enumere associe au nom
88 Enum_variable_metrrique Id_nom_Enum_variable_metrrique (char* nom);
89
90 // Retourne la chaine a partir de son identificateur de
91 // type enumere id_nom correspondant
92 char* Nom_Enum_variable_metsfe (Enum_variable_metsfe id_nom);
93
94 // Retourne l'identificateur de type enumere associe au nom
95 Enum_variable_metsfe Id_nom_Enum_variable_metsfe (char* nom);
96
97 // surcharge de l'operator de lecture
98 istream & operator » (istream & entree, Enum_variable_metrrique& a);
99 // surcharge de l'operator d'écriture
100 ostream & operator « (ostream & sort, const Enum_variable_metrrique& a);
101
102 // surcharge de l'operator de lecture
103 istream & operator » (istream & entree, Enum_variable_metsfe& a);
104 // surcharge de l'operator d'écriture
105 ostream & operator « (ostream & sort, const Enum_variable_metsfe& a);
106
107 #endif

```

## 7.305 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/EnumCourbe1D.h

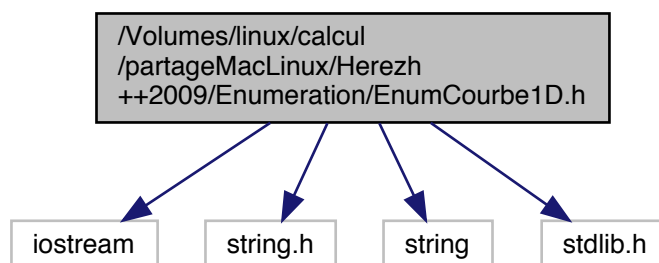
Enumeration des différentes courbes 1D existantes.

```

#include <iostream>
#include <string.h>
#include <string>
#include <stdlib.h>

```

Grappe des dépendances par inclusion de EnumCourbe1D.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

```

— enum EnumCourbe1D {
    COURBEPOLYLINEAIRE_1_D = 1, COURBE_EXPOAFF, COURBE_UN_MOINS_COS, CPL1D,
    COURBEPOLYNOMIALE, F1 Rond_F2, F1 PLUS_F2, F CYCLIQUE,
    F CYCLE_ADD, F UNION_1D, COURBE TRIPODECOS3PHI, COURBE SIXPODECOS3PHI,
    COURBE POLY_LAGRANGE, COURBE_EXPO_N, COURBE_EXPO2_N, COURBE_RELAX_EXPO,
    COURBE_COS, COURBE_SIN, COURBE_TANH, COURBEPOLYHERMITE_1_D,
    COURBE_LN_COSH, COURBE_EXPRESSION_LITTERALE_1D, COURBE_EXPRESSION_↔
    LITTERALE_AVEC_DERIVEE_1D, AUCUNE_COURBE1D }
  
```

*Enumeration des différentes courbes 1D existantes.*

## Fonctions

```

— string Nom_Courbe1D (EnumCourbe1D id_Courbe1D)
— EnumCourbe1D Id_Nom_Courbe1D (const string &nom_Courbe1D)
— bool Type_EnumCourbe1D_existe (const std::string &nom)
— istream & operator>> (istream &entree, EnumCourbe1D &a)
— ostream & operator<< (ostream &sort, const EnumCourbe1D &a)
  
```

### 7.305.1 Description détaillée

Enumeration des différentes courbes 1D existantes.

Date

19/01/2001

## 7.306 EnumCourbe1D.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file EnumCourbe1D.h
2 \brief Enumeration des différentes courbes 1D existantes
3 * \date 19/01/2001
4 */
5
  
```



```

6 // FICHER : EnumCourbe1D.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37 *   DATE:      19/01/2001
38 *
39 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
40 *
41 *   PROJET:    Herezh++
42 *
43 * *****/
44 *   BUT:      Enumeration des différentes courbes 1D existantes
45 *
46 *   *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !       !           !
53 *   *****
54 *
55 *   MODIFICATIONS:
56 *
57 *   ! date !   auteur !           but
58 *   -----
59 *   $
60 *
61 * *****/
62
63 #ifndef ENUMCOURBE_1_D_H
64 #define ENUMCOURBE_1_D_H
65 #include <iostream>
66 using namespace std;
67 #include <string.h>
68 #include <string>
69 #include <stdlib.h>
70
71 /// @addtogroup Group_types_enumeres
72 /// @{
73
74 /// Enumeration des différentes courbes 1D existantes
75
76 enum EnumCourbe1D { COURBEPOLYLINEAIRE_1_D=1, COURBE_EXPOAFF, COURBE_UN_MOINS_COS
77 , CPL1D, COURBEPOLYNOMIALE, F1_ROND_F2, F1_PLUS_F2
78 , F_CYCLIQUE, F_CYCLE_ADD, F_UNION_1D
79 , COURBE_TRIPODECOS3PHI, COURBE_SIXPODECOS3PHI
80 , COURBE_POLY_LAGRANGE, COURBE_EXPO_N, COURBE_EXPO2_N
81 , COURBE_RELAX_EXPO, COURBE_COS, COURBE_SIN, COURBE_TANH
82 , COURBEPOLYHERMITE_1_D, COURBE_LN_COSH
83 , COURBE_EXPRESSION_LITTERALE_1D, COURBE_EXPRESSION_LITTERALE_AVEC_DERIVEE_1D
84 , AUCUNE_COURBE1D};
85
86 /// @} // end of group
87
88 // Retourne le nom d'une courbe1D a partir de son identificateur de
89 // type enumere id_Courbe1D correspondant
90 string Nom_Courbe1D (EnumCourbe1D id_Courbe1D);
91
92 // Retourne l'identificateur de type enumere associe au nom d'une courbe1D

```

```

92 EnumCourbe1D Id_Nom_Courbe1D (const string& nom_Courbe1D) ;
93
94 // Retourne vrai si le nom passé en argument représente un type de courbe reconnu
95 // sinon false
96 bool Type_EnumCourbe1D_existe(const std::string& nom);
97
98 // surcharge de l'operator de lecture
99 istream & operator » (istream & entree, EnumCourbe1D& a);
100 // surcharge de l'operator d'écriture
101 ostream & operator « (ostream & sort, const EnumCourbe1D& a);
102
103 #endif

```

## 7.307 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumElemTypeProblem.h

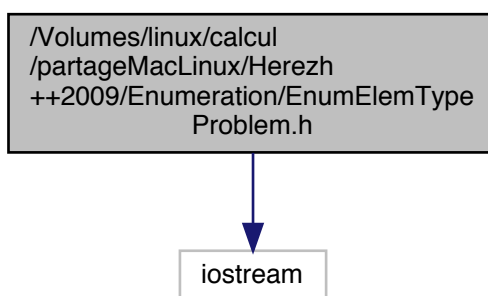
def de l'enuméré concernant les types de problème

```

#include <iostream>
#include "EnumElemTypeProblem.cc"

```

Graphes des dépendances par inclusion de EnumElemTypeProblem.h:



### Énumérations

```

— enum EnumElemTypeProblem {
    MECA_SOLIDE_DEFORMABLE = 1 , MECA_SOLIDE_INDEFORMABLE , MECA_FLUIDE , THERMIQUE
    ,
    ELECTROMAGNETIQUE , RIEN_PROBLEM }

```

*Définition de l'enuméré concernant les types de problème.*

### Fonctions

```

— int Nombre_de_EnumElemTypeProblem ()
— string NomElemTypeProblem (const EnumElemTypeProblem id_ElemTypeProblem)
— EnumElemTypeProblem Id_nom_ElemTypeProblem (const string &nom_ElemTypeProblem)
— bool ExisteEnum_ElemTypeProblem (const string &nom_ElemTypeProblem)
— int NbEnum_ElemTypeProblem ()
— istream & operator>> (istream &entree, EnumElemTypeProblem &a)
— ostream & operator<< (ostream &sort, const EnumElemTypeProblem &a)

```

#### 7.307.1 Description détaillée

def de l'enuméré concernant les types de problème

## 7.308 EnumElemTypeProblem.h

Aller à la documentation de ce fichier.

```
1  /*! \file EnumElemTypeProblem.h
2     \brief def de l'enuméré concernant les types de problème
3  */
4  // FICHER : EnumElemTypeProblem.h
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software; you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // liste des différentes types de problème que l'on peut trouver au niveau d'un élément
36 // fini
37
38
39 #ifndef ENUMELEMTYPEPROBLEM_H
40 #define ENUMELEMTYPEPROBLEM_H
41
42 ##include "Debug.h"
43 ##include <bool.h>
44 #include <iostream>
45 using namespace std;
46
47
48
49 /// @addtogroup Group_types_enumeres
50 /// @{
51
52 /// Définition de l'enuméré concernant les types de problème
53
54 enum EnumElemTypeProblem { MECA_SOLIDE_DEFORMABLE = 1, MECA_SOLIDE_INDEFORMABLE,
55                            MECA_FLUIDE, THERMIQUE, ELECTROMAGNETIQUE , RIEN_PROBLEM};
56 /// @} // end of group
57
58
59 int Nombre_de_EnumElemTypeProblem() ; // nombre maxi d'enum
60 // Retourne le nom a partir de son identificateur de
61 // type enumere id_ElemTypeProbleme correspondant
62 string NomElemTypeProblem (const EnumElemTypeProblem id_ElemTypeProblem);
63
64 // Retourne l'identificateur de type enumere associe au nom
65 EnumElemTypeProblem Id_nom_ElemTypeProblem (const string& nom_ElemTypeProblem);
66
67 // retourne true si l'identificateur existe, false sinon
68 bool ExisteEnum_ElemTypeProblem(const string& nom_ElemTypeProblem);
69
70 // retourne le nombre maxi de Type de problème appréhendé
71 int NbEnum_ElemTypeProblem();
72 // surcharge de l'operator de lecture
73 istream & operator » (istream & entree, EnumElemTypeProblem& a);
74 // surcharge de l'operator d'écriture
75 ostream & operator « (ostream & sort, const EnumElemTypeProblem& a);
76
77 // pour faire de l'inline
78 #ifndef MISE_AU_POINT
79 #include "EnumElemTypeProblem.cc"
80 #define EnumElemTypeProblem_deja_inclus
81 #endif
82
83
```

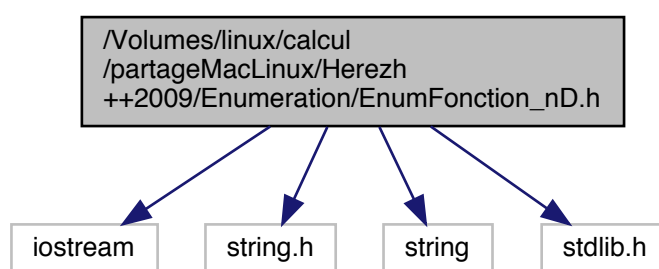
```
84 #endif
```

## 7.309 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumFonction\_nD.h

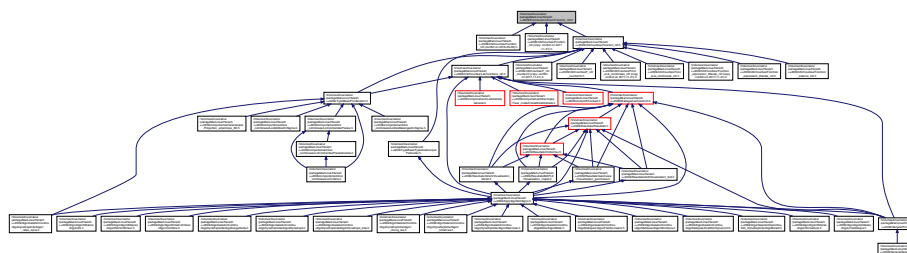
Enumeration des différentes fonctions nD existantes.

```
#include <iostream>
#include <string.h>
#include <string>
#include <stdlib.h>
```

Graphe des dépendances par inclusion de EnumFonction\_nD.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Énumérations

```
— enum EnumFonction_nD {
    FONCTION_EXPRESSION_LITTERALE_nD = 1 , FONCTION_COURBE1D , FONC_SCAL_COMBINEES↔
    _ND , FONCTION_EXTERNE_ND ,
    AUCUNE_FONCTION_nD }
```

*Enumeration des différentes fonctions nD existantes.*

### Fonctions

```
— string Nom_Fonction_nD (EnumFonction_nD id_Fonction_nD)
— EnumFonction_nD Id_Nom_Fonction_nD (const string &nom_Fonction_nD)
— bool Type_EnumFonction_nD_existe (const std::string &nom)
— istream & operator>> (istream &entree, EnumFonction_nD &a)
— ostream & operator<< (ostream &sort, const EnumFonction_nD &a)
```

### 7.309.1 Description détaillée

Enumeration des différentes fonctions nD existantes.

Date

06/03/2023

## 7.310 EnumFonction\_nD.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file EnumFonction_nD.h
2      \brief Enumeration des différentes fonctions nD existantes
3  * \date      06/03/2023
4  */
5
6
7  // This file is part of the Herezh++ application.
8  //
9  // The finite element software Herezh++ is dedicated to the field
10 // of mechanics for large transformations of solid structures.
11 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
12 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
13 //
14 // Herezh++ is distributed under GPL 3 license ou ultérieure.
15 //
16 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
17 // AUTHOR : Gérard Rio
18 // E-MAIL : gerardrio56@free.fr
19 //
20 // This program is free software: you can redistribute it and/or modify
21 // it under the terms of the GNU General Public License as published by
22 // the Free Software Foundation, either version 3 of the License,
23 // or (at your option) any later version.
24 //
25 // This program is distributed in the hope that it will be useful,
26 // but WITHOUT ANY WARRANTY; without even the implied warranty
27 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
28 // See the GNU General Public License for more details.
29 //
30 // You should have received a copy of the GNU General Public License
31 // along with this program. If not, see <https://www.gnu.org/licenses/>.
32 //
33 // For more information, please consult: <https://herezh.irdl.fr/>.
34
35 /*****
36 *      DATE:          06/03/2023
37 *
38 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
39 *
40 *      PROJET:        Herezh++
41 *
42 *
43 *      BUT:           Enumeration des différentes fonctions nD existantes
44 *
45 *      *****
46 *
47 *      VERIFICATION:
48 *
49 *      ! date !   auteur !           but
50 *      -----
51 *      !       !           !
52 *
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date !   auteur !           but
56 *      -----
57 *
58 *      *****/
59
60
61 #ifndef ENUMFONCTION_ND_H
62 #define ENUMFONCTION_ND_H
63 #include <iostream>
64 using namespace std;
65 #include <string.h>
66 #include <string>
67 #include <stdlib.h>
68
69 /// @addtogroup Group_types_enumeres
70 /// @{
71
72 /// Enumeration des différentes fonctions nD existantes
73

```

```

74 enum EnumFonction_nD { FONCTION_EXPRESSION_LITTERALE_nD=1
75                       ,FONCTION_COURBEID,FONC_SCAL_COMBINEES_nD
76                       ,FONCTION_EXTERNE_nD
77                       ,AUCUNE_FONCTION_nD};
78 /// @} // end of group
79
80
81
82 // Retourne le nom d'une Fonction_nD a partir de son identificateur de
83 // type enumere id_Fonction_nD correspondant
84 string Nom_Fonction_nD (EnumFonction_nD id_Fonction_nD);
85
86 // Retourne l'identificateur de type enumere associe au nom d'une Fonction_nD
87 EnumFonction_nD Id_Nom_Fonction_nD (const string& nom_Fonction_nD) ;
88
89 // Retourne vrai si le nom passé en argument représente un type de courbe reconnu
90 // sinon false
91 bool Type_EnumFonction_nD_existe(const std::string& nom);
92
93 // surcharge de l'operator de lecture
94 istream & operator » (istream & entree, EnumFonction_nD& a);
95 // surcharge de l'operator d'écriture
96 ostream & operator « (ostream & sort, const EnumFonction_nD& a);
97
98 #endif

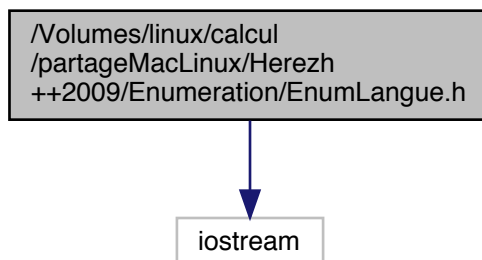
```

### 7.311 Référence du fichier /Volumes/linux/calcul/partageMacLinux/← Herezh++2009/Enumeration/EnumLangue.h

def Enumeration des différents langages

```
#include <iostream>
```

Graphes des dépendances par inclusion de EnumLangue.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



#### Énumérations

- enum `EnumLangue` { `FRANCAIS = 1` , `ENGLISH` }
- Énumération des différents langages.*

#### Fonctions

- `char * Nom_EnumLangue (EnumLangue id_EnumLangue)`
- `EnumLangue Id_nom_EnumLangue (char *nom_EnumLangue)`

- istream & operator>> (istream &entree, EnumLangue &a)
- ostream & operator<< (ostream &sort, const EnumLangue &a)

## Variables

- const int nombre\_maxi\_de\_type\_de\_EnumLangue = 2

### 7.311.1 Description détaillée

def Enumeration des différents langages

Date

30/05/2009

## 7.312 EnumLangue.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file EnumLangue.h
2  \brief def Enumeration des différents langages
3  * \date      30/05/2009
4  */
5
6 // FICHER : EnumLangue.h
7
8 // This file is part of the Herezh++ application.
9 //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36 /*****
37  *      DATE:          30/05/2009
38  *
39  *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
40  *
41  *      PROJET:        Herezh++
42  *
43  *****/
44  *      BUT:  Enumération des différents langages
45  *
46  *      *****
47  *
48  *      VERIFICATION:
49  *      ! date ! auteur ! but
50  *      -----
51  *      ! ! !
52  *
53  *      *****
54  *      MODIFICATIONS:
55  *      ! date ! auteur ! but
56  *      -----
57  *
58  *****/
59
60 #ifndef ENUM_LANGAGE_H
61 #define ENUM_LANGAGE_H

```

```

62
63 #include <iostream>
64 using namespace std;
65
66
67 /// @addtogroup Group_types_enumeres
68 /// @{
69
70 /// Énumération des différents langages
71
72 enum EnumLangue { FRANCAIS = 1, ENGLISH };
73 /// @} // end of group
74
75 const int nombre_maxi_de_type_de_EnumLangue = 2;
76
77 // Retourne un nom de type de EnumLangue a partir de son identificateur de
78 // type enumere id_EnumLangue correspondant
79 char* Nom_EnumLangue (EnumLangue id_EnumLangue);
80
81 // Retourne l'identificateur de type enumere associe au nom du type
82 // de EnumLangue nom_EnumLangue
83 EnumLangue Id_nom_EnumLangue (char* nom_EnumLangue);
84
85 // surcharge de l'operator de lecture
86 istream & operator » (istream & entree, EnumLangue& a);
87 // surcharge de l'operator d'écriture
88 ostream & operator « (ostream & sort, const EnumLangue& a);
89
90
91 #endif
92
93

```

### 7.313 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/EnumTypeCalcul.h

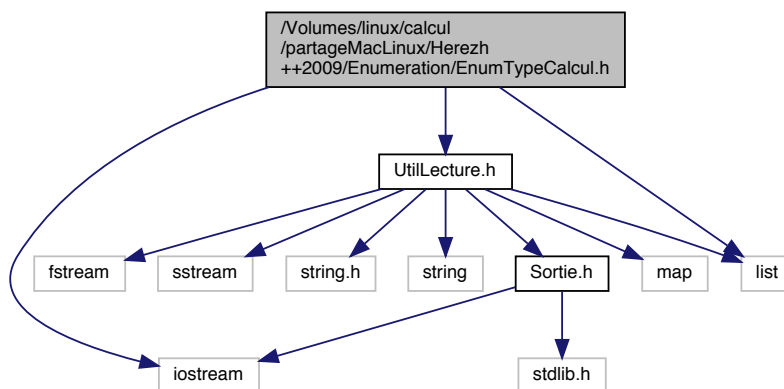
def Enumeration concernant les différents types de calcul globaux

```

#include <iostream>
#include "UtilLecture.h"
#include <list>

```

Graphe des dépendances par inclusion de EnumTypeCalcul.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :





## Énumérations

- enum `EnumTypeCalcul` {  
`DYNA_IMP = 1`, `DYNA_EXP`, `DYNA_EXP_TCHAMWA`, `DYNA_EXP_CHUNG_LEE`,  
`DYNA_EXP_ZHAI`, `DYNA_RUNGE_KUTTA`, `NON_DYNA`, `NON_DYNA_CONT`,  
`FLAMB_LINEAIRE`, `INFORMATIONS`, `UTILITAIRES`, `DEF_SCHEMA_XML`,  
`RIEN_TYPECALCUL`, `UMAT_ABAQUS`, `DYNA_EXP_BONELLI`, `RELAX_DYNA`,  
`STAT_DYNA_EXP`, `COMBINER` }  
*identificateur principal du type de Calcul*
- enum `EnumSousTypeCalcul` {  
`aucun_soustypedecalcul = 1`, `avec_remonte`, `avec_remonte_erreur`, `avec_remonte_erreur_relocation`,  
`avec_remonte_erreur_raffinement_relocation`, `remonte`, `remonte_erreur`, `remonte_erreur_relocation`,  
`remonte_erreur_raffinement_relocation`, `frontieres`, `visualisation`, `LinVersQuad`,  
`QuadIncVersQuadComp`, `relocPtMilieuQuad`, `sauveCommandesVisu`, `lectureCommandesVisu`,  
`commandeInteractive`, `sauveMaillagesEnCours`, `extrusion2D3D`, `creation_reference`,  
`prevision_visu_sigma`, `prevision_visu_epsilon`, `prevision_visu_erreur`, `modif_orientation_element`,  
`creationMaillageSFE`, `suppression_noeud_non_references`, `renumerotation_des_noeuds`, `fusion_↵`  
`_de_noeuds`,  
`fusion_elements`, `fusion_maillages`, `cree_sous_maillage`, `calcul_geometrique` }  
*identificateur secondaire optionnel du type de Calcul renseigne par exemple sur le fait de calculer l'erreur, une relocation, un raffinement, etc ..*
- enum `Enum_evolution_temporelle` { `STATIQUE = 1`, `TRANSITOIRE`, `DYNAMIQUE`, `AUCUNE_↵`  
`EVOLUTION` }

## Fonctions

- string `Nom_TypeCalcul` (const `EnumTypeCalcul` id\_TypeCalcul)
- string `Nom_SousTypeCalcul` (const `EnumSousTypeCalcul` id\_SousTypeCalcul)
- `Enum_evolution_temporelle` `Evolution_temporelle_du_calcul` (const `EnumTypeCalcul` id\_TypeCalcul)
- `EnumTypeCalcul` `Id_nom_TypeCalcul` (const string &nom\_TypeCalcul)
- `EnumSousTypeCalcul` `Id_nom_SousTypeCalcul` (const string &nom\_SousTypeCalcul)
- bool `DynamiqueExplicite` (const `EnumTypeCalcul` id\_TypeCalcul)
- bool `Implicite` (const `EnumTypeCalcul` id\_TypeCalcul)
- void `LectureTypeCalcul` (`UtilLecture` &lec, `EnumTypeCalcul` &typeCalcul, bool &avec\_Calcul, list<  
`EnumSousTypeCalcul` > &sousTypeCalcul, list< bool > &avec\_sousCalcul)
- void `Info_commande_type_calcul` (`UtilLecture` &lec, `EnumTypeCalcul` &typeCalcul, bool &avec\_Calcul,  
list< `EnumSousTypeCalcul` > &sousTypeCalcul, list< bool > &avec\_sousCalcul)
- bool `SousType` (const `EnumSousTypeCalcul` sousTypeCalcul)
- bool `Remonte_in` (const `EnumSousTypeCalcul` sousTypeCalcul)
- bool `Avec_in` (const `EnumSousTypeCalcul` sousTypeCalcul)
- bool `Erreur_in` (const `EnumSousTypeCalcul` sousTypeCalcul)
- `istream` & `operator`>> (`istream` &entree, `EnumTypeCalcul` &a)
- `ostream` & `operator`<< (`ostream` &sort, const `EnumTypeCalcul` &a)
- `istream` & `operator`>> (`istream` &entree, `EnumSousTypeCalcul` &a)
- `ostream` & `operator`<< (`ostream` &sort, const `EnumSousTypeCalcul` &a)
- string `Nom_evolution_temporelle` (const `Enum_evolution_temporelle` id\_evolution\_temporelle)
- `Enum_evolution_temporelle` `Id_nom_evolution_temporelle` (const string &nom\_evolution\_temporelle)
- `istream` & `operator`>> (`istream` &entree, `Enum_evolution_temporelle` &a)
- `ostream` & `operator`<< (`ostream` &sort, const `Enum_evolution_temporelle` &a)

### 7.313.1 Description détaillée

def Enumeration concernant les différents types de calcul globaux

Date

23/01/97

## 7.314 EnumTypeCalcul.h

Aller à la documentation de ce fichier.

```

1  /*! \file EnumTypeCalcul.h
2      \brief def Enumeration concernant les différents types de calcul globaux
3  * \date      23/01/97
4  */
5
6
7  // This file is part of the Herezh++ application.
8  //
9  // The finite element software Herezh++ is dedicated to the field
10 // of mechanics for large transformations of solid structures.
11 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
12 // INSTITUT DE RECHERCHE DUPIY DE LÔME (IRDL) <https://www.irdl.fr/>.
13 //
14 // Herezh++ is distributed under GPL 3 license ou ultérieure.
15 //
16 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
17 // AUTHOR : Gérard Rio
18 // E-MAIL : gerardrio56@free.fr
19 //
20 // This program is free software: you can redistribute it and/or modify
21 // it under the terms of the GNU General Public License as published by
22 // the Free Software Foundation, either version 3 of the License,
23 // or (at your option) any later version.
24 //
25 // This program is distributed in the hope that it will be useful,
26 // but WITHOUT ANY WARRANTY; without even the implied warranty
27 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
28 // See the GNU General Public License for more details.
29 //
30 // You should have received a copy of the GNU General Public License
31 // along with this program. If not, see <https://www.gnu.org/licenses/>.
32 //
33 // For more information, please consult: <https://herezh.irdl.fr/>.
34
35 /*****
36 *      DATE:          23/01/97
37 *
38 *      AUTEUR:        G RIO (mailto:gerardrio56@free.fr)
39 *
40 *      PROJET:        Herezh++
41 *
42 *      *****
43 *      BUT: Enumeration des différents type de calcul possible.
44 *
45 *      *****
46 *
47 *      VERIFICATION:
48 *
49 *      ! date ! auteur ! but
50 *      -----
51 *      ! ! !
52 *
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *
58 *      *****/
59 #ifndef ENUMTYPECALCUL_H
60 #define ENUMTYPECALCUL_H
61
62 #include <iostream>
63 using namespace std;
64 #include "UtilLecture.h"
65 #include <list>
66
67 /// @addtogroup Group_types_enumeres
68 /// @{
69
70 /// identificateur principal du type de Calcul
71 enum EnumTypeCalcul { DYNA_IMP = 1, DYNA_EXP, DYNA_EXP_TCHAMWA, DYNA_EXP_CHUNG_LEE, DYNA_EXP_ZHAI,
72                      DYNA_RUNGE_KUTTA, NON_DYNA, NON_DYNA_CONT,
73                      FLAMB_LINEAIRE, INFORMATIONS, UTILITAIRES, DEF_SCHEMA_XML, RIEN_TYPECALCUL,
74                      UMAT_ABAQUS, DYNA_EXP_BONELLI, RELAX_DYNA, STAT_DYNA_EXP,
75                      COMBINER
76                      };
77 /// @} // end of group
78
79 /// @addtogroup Group_types_enumeres
80 /// @{
81
82 /// identificateur secondaire optionnel du type de Calcul renseigné par exemple
83 /// sur le fait de calculer l'erreur, une relocation, un raffinement, etc ..

```

```

84 enum EnumSousTypeCalcul { aucun_soustypedecalcul =1, avec_remonte , avec_remonte_erreur,
85     avec_remonte_erreur_relocation, avec_remonte_erreur_raffinement_relocation,
86     remonte_erreur_remonte_erreur_relocation, remonte_erreur_raffinement_relocation,
87     frontieres, visualisation, LinVersQuad, QuadIncVersQuadComp, relocPtMilieuQuad, sauveCommandesVisu,
88
89     lectureCommandesVisu, commandeInteractive, sauveMaillagesEnCours, extrusion2D3D, creation_reference,
90     prevision_visu_sigma, prevision_visu_epsilon, prevision_visu_erreur,
91
92     modif_orientation_element, creationMaillageSFE, suppression_noeud_non_references, renumerotation_des_noeuds,
93     fusion_de_noeuds, fusion_elements, fusion_maillages, cree_sous_maillage, calcul_geometrique
94 };
95 // @} // end of group
96 // identificateur pour décrire si le calcul est statique, transitoire, dynamique et autre ...
97 enum Enum_evolution_temporelle{STATIQUE=1, TRANSITOIRE, DYNAMIQUE, AUCUNE_EVOLUTION};
98
99 // Retourne le nom d'un type de calcul a partir de son identificateur de
100 // type enumere id_TypeCalcul correspondant
101 string Nom_TypeCalcul (const EnumTypeCalcul id_TypeCalcul);
102 // Retourne le nom du sous type de calcul a partir de son identificateur de
103 // type enumere id_SousTypeCalcul correspondant, si il existe sinon retourne : aucun_soustypedecalcul
104 string Nom_SousTypeCalcul (const EnumSousTypeCalcul id_SousTypeCalcul);
105 // retourne le type d'évolution temporelle du calcul
106 Enum_evolution_temporelle Evolution_temporelle_du_calcul(const EnumTypeCalcul id_TypeCalcul);
107
108 // Retourne l'identificateur de type enumere associe au nom du type de calcul
109 EnumTypeCalcul Id_nom_TypeCalcul (const string& nom_TypeCalcul);
110 // Retourne l'identificateur de type enumere associe au nom du sous type de calcul
111 EnumSousTypeCalcul Id_nom_SousTypeCalcul (const string& nom_SousTypeCalcul);
112
113 // indique si le calcul est de type dynamique explicite ou non
114 bool DynamiqueExplicite(const EnumTypeCalcul id_TypeCalcul);
115
116 // indique si le calcul est de type implicite ou non
117 bool Implicite(const EnumTypeCalcul id_TypeCalcul);
118
119 //lecture du type de calcul et d'une liste de sous-type eventuel
120 // la lecture se fait via UtilLecture.
121 // Le booléen indique si le type_calcul ou le sous_type_calcul est actif ou pas
122 // au niveau de syntaxe on a :
123 // - tout d'abord le type de calcul suivi ou non de la chaine "avec"
124 // dans le premier cas le booléen avec_Calcul est mis a true sinon false
125 // - puis une liste de sous type qui chacun sont précédé ou non de la chaine "plus"
126 // dans le premier cas le booléen avec_sousCalcul correspondant est true sinon false
127 // les sous_types sont stocké par ordre d'arrivée
128 void LectureTypeCalcul(UtilLecture& lec, EnumTypeCalcul& typeCalcul, bool& avec_Calcul
129     , list<EnumSousTypeCalcul> & sousTypeCalcul
130     , list<bool>& avec_sousCalcul);
131 // affichage des différents type de calcul et choix du type et éventuellement
132 // de sous type
133 void Info_commande_type_calcul(UtilLecture& lec, EnumTypeCalcul& typeCalcul, bool& avec_Calcul
134     , list<EnumSousTypeCalcul> & sousTypeCalcul
135     , list<bool>& avec_sousCalcul);
136
137 // indique s'il y a un sous-type ou pas -> retour vrai s'il y a un sous type
138 bool SousType(const EnumSousTypeCalcul sousTypeCalcul);
139
140 // indique s'il y a remonte ou pas dans le sous-type
141 bool Remonte_in(const EnumSousTypeCalcul sousTypeCalcul);
142
143 // indique s'il y a "avec" ou pas dans le sous-type
144 bool Avec_in(const EnumSousTypeCalcul sousTypeCalcul);
145
146 // indique s'il y a erreur dans le sous-type
147 bool Erreur_in(const EnumSousTypeCalcul sousTypeCalcul);
148
149 // surcharge de l'operator de lecture
150 istream & operator » (istream & entree, EnumTypeCalcul& a);
151 // surcharge de l'operator d'écriture
152 ostream & operator « (ostream & sort, const EnumTypeCalcul& a);
153
154 // surcharge de l'operator de lecture
155 istream & operator » (istream & entree, EnumSousTypeCalcul& a);
156 // surcharge de l'operator d'écriture
157 ostream & operator « (ostream & sort, const EnumSousTypeCalcul& a);
158
159 //----- cas de Enum_evolution_temporelle-----
160
161 // Retourne le nom d'un type de Enum_evolution_temporelle a partir de son identificateur de
162 // type enumere id_Enum_evolution_temporelle correspondant
163 string Nom_evolution_temporelle (const Enum_evolution_temporelle id_evolution_temporelle);
164 // Retourne l'identificateur de type enumere associe au nom du type de Enum_evolution_temporelle
165 Enum_evolution_temporelle Id_nom_evolution_temporelle (const string& nom_evolution_temporelle);
166 // surcharge de l'operator de lecture
167 istream & operator » (istream & entree, Enum_evolution_temporelle& a);
168 // surcharge de l'operator d'écriture

```

```

169 ostream & operator << (ostream & sort, const Enum_evolution_temporelle& a);
170
171
172 #endif

```

## 7.315 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnumTypeGradient.h

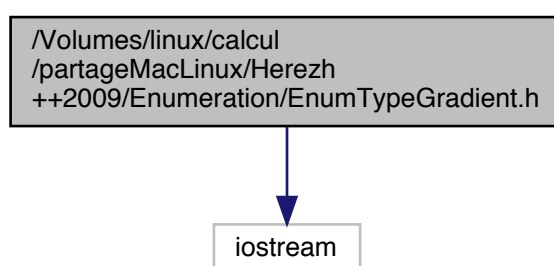
def du gradient de vitesse pour différentes conditions

```

#include <iostream>
#include "EnumTypeGradient.cc"

```

Grphe des dépendances par inclusion de EnumTypeGradient.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Énumérations

```

— enum Enum_type_gradient { GRADVITESSE_V_TDT =1 , GRADVITESSE_VCONST , GRADVITESSE_VT_VTDT }

```

*def du gradient de vitesse pour différentes conditions*

### Fonctions

```

— char * Nom_type_gradient (Enum_type_gradient id_ddl)
— Enum_type_gradient Id_nom_type_gradient (char *nom_ddl)
— ostream & operator>> (ostream &entree, Enum_type_gradient &a)
— ostream & operator<< (ostream &sort, const Enum_type_gradient &a)

```

#### 7.315.1 Description détaillée

def du gradient de vitesse pour différentes conditions

Date

28/03/2003

## 7.316 EnumTypeGradient.h

Aller à la documentation de ce fichier.

```

1  /*! \file EnumTypeGradient.h
2      \brief def du gradient de vitesse pour différentes conditions
3  * \date      28/03/2003
4  */
5
6  // FICHER : EnumTypeGradient.h
7
8  // This file is part of the Herezh++ application.
9  //
10 // The finite element software Herezh++ is dedicated to the field
11 // of mechanics for large transformations of solid structures.
12 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
13 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
14 //
15 // Herezh++ is distributed under GPL 3 license ou ultérieure.
16 //
17 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
18 // AUTHOR : Gérard Rio
19 // E-MAIL : gerardrio56@free.fr
20 //
21 // This program is free software: you can redistribute it and/or modify
22 // it under the terms of the GNU General Public License as published by
23 // the Free Software Foundation, either version 3 of the License,
24 // or (at your option) any later version.
25 //
26 // This program is distributed in the hope that it will be useful,
27 // but WITHOUT ANY WARRANTY; without even the implied warranty
28 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
29 // See the GNU General Public License for more details.
30 //
31 // You should have received a copy of the GNU General Public License
32 // along with this program. If not, see <https://www.gnu.org/licenses/>.
33 //
34 // For more information, please consult: <https://herezh.irdl.fr/>.
35
36
37 /*****
38 *      DATE:          28/03/2003
39 *
40 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
41 *
42 *      PROJET:        Herezh++
43 *
44 *      *****
45 *      BUT: Défini une énumération des différents types de calcul du
46 *      gradient de vitesse.
47 *
48 *      *****
49 *
50 *      VERIFICATION:
51 *
52 *      ! date !   auteur !           but
53 *      -----
54 *      !           !           !
55 *      *****
56 *      MODIFICATIONS:
57 *      ! date !   auteur !           but
58 *      -----
59 *
60 *      *****/
61 // Afin de réaliser un gain en place memoire, et une vérification de type plus aisé qu' avec
62 // les entiers
63
64
65 #ifndef ENUM_TYPE_GRADIENT_V_H
66 #define ENUM_TYPE_GRADIENT_V_H
67
68 // #include "Debug.h"
69 #include <iostream>
70 using namespace std;
71
72 /// @addtogroup Group_types_enumeres
73 /// @{
74
75
76 // ----- info -----
77 /// def du gradient de vitesse pour différentes conditions
78 //
79 // GRADVITESSE_V_TDT : le gradient est déterminé à partir de ddl de vitesse à t+dt
80 // GRADVITESSE_VCONST : calcul en considérant V constant =delta X/delta t dans la config à t+(delta
81 // t)/2
82 // GRADVITESSE_VT_VTDT : calcul à partir des ddl de vitesse : moyenne de t et t+dt, dans la config

```

```

82 //          à t+(delta t)/2
83
84
85 enum Enum_type_gradient { GRADVITESSE_V_TDT=1, GRADVITESSE_VCONST, GRADVITESSE_VT_VTDT};
86 /// @} // end of group
87
88 // Retourne le nom du type de gradient a partir de son identificateur de
89 // type enumere id_ddl correspondant
90 char* Nom_type_gradient ( Enum_type_gradient id_ddl);
91
92 // Retourne l'identificateur de type enumere associe au nom du type de gradient
93 Enum_type_gradient Id_nom_type_gradient (char* nom_ddl);
94
95 // surcharge de l'operator de lecture
96 istream & operator » (istream & entree, Enum_type_gradient& a);
97 // surcharge de l'operator d'écriture
98 ostream & operator « (ostream & sort, const Enum_type_gradient& a);
99
100 // pour faire de l'inline
101 #ifndef MISE_AU_POINT
102 #include "EnumTypeGradient.cc"
103 #define Enum_type_gradient_deja_inclus
104 #endif
105
106 #endif

```

### 7.317 Référence du fichier /Volumes/linux/calcul/partageMacLinux/← Herezh++2009/Enumeration/EnumTypeGrandeur.h

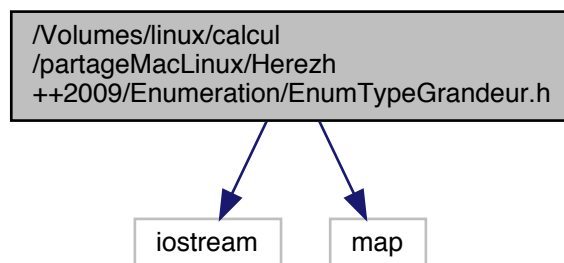
def de l'enuméré concernant les types de structures de grandeurs

```

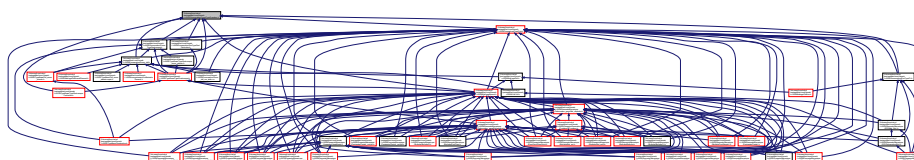
#include <iostream>
#include <map>

```

Graphe des dépendances par inclusion de EnumTypeGrandeur.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



#### Classes

— class [ClassPourEnumTypeGrandeur](#)

*def de map qui fait la liaison entre les string et les énumérés*

## Énumérations

- enum `EnumTypeGrandeur` {  
**RIEN\_TYPEGRANDEUR** = 0, **SCALAIRE**, **VECTEUR**, **TENSEUR**,  
**TENSEUR\_NON\_SYM**, **SCALAIRE\_ENTIER**, **SCALAIRE\_DOUBLE**, **TENSEURBB**,  
**TENSEURHH**, **TENSEURBH**, **TENSEURHB**, **TENSEUR\_NON\_SYM\_BB**,  
**TENSEUR\_NON\_SYM\_HH**, **COORDONNEE**, **COORDONNEEB**, **COORDONNEEH**,  
**BASE\_H**, **BASE\_B**, **CHAINE\_CAR**, **GRANDEUR\_QUELCONQUE** }  
*un énuméré pour les types de base*
- enum `EnumType2Niveau` {  
**TYPE\_SIMPLE** = 0, **TABLEAU\_T**, **TABLEAU2\_T**, **LISTE\_T**,  
**LISTE\_IO\_T**, **MAP\_T**, **VECTOR\_T** }  
*un énuméré pour les types de structure*

## Fonctions

- string **NomTypeGrandeur** (`EnumTypeGrandeur` id\_typeGrandeur)
- string **NomType2Niveau** (`EnumType2Niveau` id\_Type2Niveau)
- `EnumTypeGrandeur` **Id\_nomTypeGrandeur** (string nom\_typeGrandeur)
- `EnumType2Niveau` **Id\_nomType2Niveau** (string nom\_Type2Niveau)
- int **NombreElementFoncDim** (`EnumTypeGrandeur` id\_typeGrandeur)
- bool **Type\_grandeur\_numerique** (`EnumTypeGrandeur` id\_typeGrandeur)
- istream & **operator**>> (istream &entree, `EnumTypeGrandeur` &a)
- ostream & **operator**<< (ostream &sort, const `EnumTypeGrandeur` &a)

### 7.317.1 Description détaillée

def de l'énuméré concernant les types de structures de grandeurs

## 7.318 EnumTypeGrandeur.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file EnumTypeGrandeur.h
2   \brief def de l'énuméré concernant les types de structures de grandeurs
3 */
4 // FICHER : EnumTypeGrandeur.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des types de grandeurs sont
36 // stockes a l'aide d'un type enumere. Les fonctions NomTypeGrandeur et Id_nomTypeGrandeur rendent
37 // possible le lien entre des noms grandeur et les identificateurs
38 // de type enumere correspondants.
39
40

```

```

41 #ifndef ENUMTYPEGRANDEUR_H
42 #define ENUMTYPEGRANDEUR_H
43 #include <iostream>
44 using namespace std;
45 #include <map>
46
47 /// @addtogroup Group_types_enumeres
48 /// @{
49
50 /// un énuméré pour les types de base
51 enum EnumTypeGrandeur { RIEN_TYPEGRANDEUR = 0, SCALAIRE, VECTEUR, TENSEUR, TENSEUR_NON_SYM
52 , SCALAIRE_ENTIER, SCALAIRE_DOUBLE, TENSEURBB, TENSEURHH, TENSEURBH, TENSEURHB
53 , TENSEUR_NON_SYM_BB, TENSEUR_NON_SYM_HH
54 , COORDONNEE, COORDONNEEB, COORDONNEEH
55 , BASE__H, BASE__B
56 , CHAINE_CAR, GRANDEUR_QUELCONQUE};
57 /// @} // end of group
58
59 /// @addtogroup Group_types_enumeres
60 /// @{
61
62 /// un énuméré pour les types de structure
63 enum EnumType2Niveau { TYPE_SIMPLE = 0, TABLEAU_T, TABLEAU2_T, LISTE_T, LISTE_IO_T, MAP_T, VECTOR_T };
64 /// @} // end of group
65
66 /// @addtogroup Group_types_enumeres
67 /// @{
68
69 /// def de map qui fait la liaison entre les string et les énumérés
70
71 class ClassPourEnumTypeGrandeur
72 { public:
73 /// def de map qui fait la liaison entre les string et les énumérés
74 static map < string, EnumTypeGrandeur , std::less < string> > map_EnumTypeGrandeur;
75 static map < string, EnumType2Niveau , std::less < string> > map_EnumType2Niveau;
76 // def de la grandeur statique qui permet de remplir les map
77 static ClassPourEnumTypeGrandeur remplir_map;
78 // le constructeur qui remplit effectivement la map
79 ClassPourEnumTypeGrandeur();
80 };
81 /// @} // end of group
82
83 // Retourne le nom du type de grandeur a partir de son identificateur de
84 // type enumere id_typeGrandeur correspondant
85 string NomTypeGrandeur (EnumTypeGrandeur id_typeGrandeur);
86 // idem pour le type secondaire
87 string NomType2Niveau (EnumType2Niveau id_Type2Niveau );
88
89 // Retourne l'identificateur de type enumere associe au nom du type de grandeur
90 // nom_typeGrandeur
91 EnumTypeGrandeur Id_nomTypeGrandeur (string nom_typeGrandeur);
92 // item pour le type secondaire
93 EnumType2Niveau Id_nomType2Niveau (string nom_Type2Niveau );
94
95 // Retourne le nombre d'éléments de la grandeur en fonction de la dimension
96 // exemple en dimension 3, pour un scalaire retourne 1, pour un vecteur 3
97 // pour un tenseur (a priori symétrique) 6
98 // pour un tenseur non symétrique 9
99 // RIEN_TYPEGRANDEUR 0
100 // pour le cas particulier GRANDEUR_QUELCONQUE : retourne -1
101 int NombreElementFoncDim(EnumTypeGrandeur id_typeGrandeur);
102
103 // indique si c'est un type numérique ou non
104 bool Type_grandeur_numerique(EnumTypeGrandeur id_typeGrandeur);
105
106 // surcharge de l'operator de lecture
107 istream & operator » (istream & entree, EnumTypeGrandeur& a);
108 // surcharge de l'operator d'écriture
109 ostream & operator « (ostream & sort, const EnumTypeGrandeur& a);
110
111 #endif

```

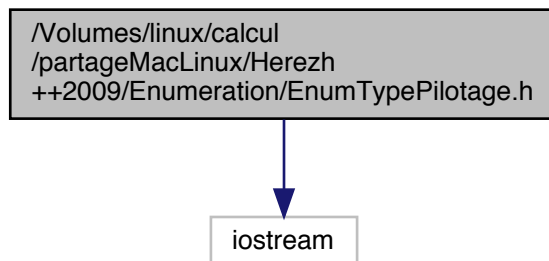
### 7.319 Référence du fichier /Volumes/linux/calcul/partageMacLinux/← Herezh++2009/Enumeration/EnumTypePilotage.h

def de l'énuméré concernant les types de pilotage

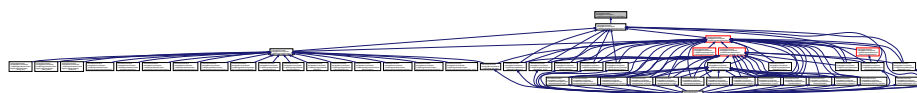


```
#include <iostream>
```

Graphe des dépendances par inclusion de EnumTypePilotage.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

- enum `EnumTypePilotage` { `PILOTAGE_BASIQUE =1` , `PILOT_GRADIENT` , `AUCUN_PILOTAGE` }  
*Définition de l'enuméré concernant les types de pilotage.*

## Fonctions

- char \* `Nom_TypePilotage` (const `EnumTypePilotage` id\_TypePilotage)
- `EnumTypePilotage` `Id_nom_TypePilotage` (const char \*nom\_TypePilotage)
- bool `Existe_dans_TypePilotage` (const char \*nom\_TypePilotage)
- istream & `operator`>> (istream &entree, `EnumTypePilotage` &a)
- ostream & `operator`<< (ostream &sort, const `EnumTypePilotage` &a)

### 7.319.1 Description détaillée

def de l'enuméré concernant les types de pilotage

## 7.320 EnumTypePilotage.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file EnumTypePilotage.h
2   \brief def de l'enuméré concernant les types de pilotage
3 */
4 // FICHER : EnumTypePilotage.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
  
```

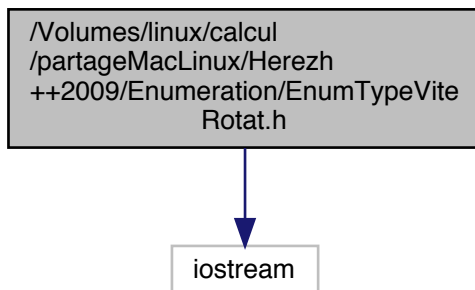
```
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des types de pilotage sont
36 // stockes a l'aide d'un type enumere. Les fonctions Nom_TypePilotage et Id_nom_TypePilotage
37 // facilitent la liaison entre type énuméré et string
38
39
40 #ifndef ENUM_TYPEPILOTAGE_H
41 #define ENUM_TYPEPILOTAGE_H
42 #include <iostream>
43 using namespace std;
44
45 /// @addtogroup Group_types_enumeres
46 /// @{
47
48 /// Définition de l'énuméré concernant les types de pilotage
49
50 enum EnumTypePilotage { PILOTAGE_BASIQUE=1, PILOT_GRADIENT, AUCUN_PILOTAGE};
51 /// @} // end of group
52
53
54 // Retourne le nom du type de pilotage a partir de son identificateur de
55 // type enumere id_TypePilotage correspondant
56 char* Nom_TypePilotage (const EnumTypePilotage id_TypePilotage);
57
58 // Retourne l'identificateur de type enumere associe au nom du type de pilotage
59 EnumTypePilotage Id_nom_TypePilotage (const char* nom_TypePilotage);
60
61 // retourne si la chaine de caractère existe ou pas en tant que EnumTypePilotage
62 bool Existe_dans_TypePilotage(const char* nom_TypePilotage);
63
64 // surcharge de l'operator de lecture
65 istream & operator » (istream & entree, EnumTypePilotage& a);
66 // surcharge de l'operator d'écriture
67 ostream & operator « (ostream & sort, const EnumTypePilotage& a);
68
69 #endif
```

### 7.321 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Enumeration/EnumTypeViteRotat.h

def de l'énuméré concernant les type de vitesse de rotation

```
#include <iostream>
```

Graphe des dépendances par inclusion de EnumTypeViteRotat.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Énumérations

```
— enum EnumTypeViteRotat { R_CORROTATIONNEL = 1 , R_REF_ROT_PROPRE , R_ROT_↔  
LOGARITHMIQUE }
```

*Définition de l'énuméré concernant les type de vitesse de rotation.*

## Fonctions

- char \* **Nom\_TypeViteRotat** (const EnumTypeViteRotat id\_TypeViteRotat)
- EnumTypeViteRotat **Id\_nom\_TypeViteRotat** (const char \*nom\_TypeViteRotat)
- istream & **operator**>> (istream &entree, EnumTypeViteRotat &a)
- ostream & **operator**<< (ostream &sort, const EnumTypeViteRotat &a)

### 7.321.1 Description détaillée

def de l'énuméré concernant les type de vitesse de rotation

## 7.322 EnumTypeViteRotat.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file EnumTypeViteRotat.h
2 \brief def de l'énuméré concernant les type de vitesse de rotation
3 */
4 // FICHER : EnumTypeViteRotat.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
  
```

```

17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des différents type de calcul de vitesse
36 // de rotation sont stockes a l'aide d'un type enumere. Les fonctions Nom_TypeViteRotat et
37 // Id_nom_TypeViteRotat rendent possible le lien entre les noms des types de vitesse de rotation
38 // et les identificateurs de type enumere correspondants.
39
40
41 #ifndef ENUMTYPEVITEROTAT_H
42 #define ENUMTYPEVITEROTAT_H
43 #include <iostream>
44 using namespace std;
45
46 /// @addtogroup Group_types_enumeres
47 /// @{
48
49 /// Définition de l'enuméré concernant les type de vitesse de rotation
50
51 enum EnumTypeViteRotat { R_CORROTATIONNEL = 1,R_REF_ROT_PROPRE,R_ROT_LOGARITHMIQUE};
52 /// @} // end of group
53
54
55 // Retourne un nom de type de vitesse de rotation a partir de son identificateur de
56 // type enumere id_TypeViteRotat correspondant
57 char* Nom_TypeViteRotat (const EnumTypeViteRotat id_TypeViteRotat) ;
58
59 // Retourne l'identificateur de type enumere associe au nom du type
60 // de vitesse de rotation nom_TypeViteRotat
61 EnumTypeViteRotat Id_nom_TypeViteRotat (const char* nom_TypeViteRotat);
62
63 // surcharge de l'operator de lecture
64 istream & operator » (istream & entree, EnumTypeViteRotat& a);
65 // surcharge de l'operator d'écriture
66 ostream & operator « (ostream & sort, const EnumTypeViteRotat& a);
67
68
69 #endif
70
71

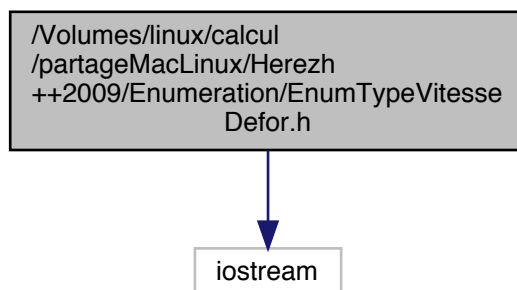
```

### 7.323 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/EnumTypeVitesseDeform.h

Définition de l'enuméré concernant les types de vitesse de déformation.

```
#include <iostream>
```

Graphe des dépendances par inclusion de EnumTypeVitesseDefor.h:



## Énumérations

- enum `Enum_TypeVitDef` { `D_MOY_EPS_SUR_DT = 1`, `D_AVEC_V_A_T_PLUS_DT`, `D_AVEC_DX_SUR_DT`, `D_A_T_PLUS_DT_SUR2`, `D_AVEC_V_MOY_A_T_PLUS_DT_SUR2` }
- Définition de l'enuméré concernant les types de vitesse de déformation.*

## Fonctions

- char \* `Nom_TypeVitDef` (const `Enum_TypeVitDef` id\_TypeVitDef)
- `Enum_TypeVitDef` `Id_nom_TypeVitDef` (const char \*nom\_TypeVitDef)
- `istream & operator>>` (`istream &entree`, `Enum_TypeVitDef &a`)
- `ostream & operator<<` (`ostream &sort`, const `Enum_TypeVitDef &a`)

### 7.323.1 Description détaillée

Définition de l'enuméré concernant les types de vitesse de déformation.

## 7.324 EnumTypeVitesseDefor.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file EnumTypeVitesseDefor.h
2    \brief Définition de l'enuméré concernant les types de vitesse de déformation
3 */
4 // FICHER : EnumTypeVitesseDefor.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
  
```

```

27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des différents type de calcul de vitesse
36 // de déformation sont stockes a l'aide d'un type enumere. Les fonctions Nom_TypeVitDef et
37 // Id_nom_TypeVitDef rendent possible le lien entre les noms des types de vitesse de déformation
38 // et les identificateurs de type enumere correspondants.
39
40
41 #ifndef ENUM_TYPEVITESSEDEFOR_H
42 #define ENUM_TYPEVITESSEDEFOR_H
43 #include <iostream>
44 using namespace std;
45
46 /// @addtogroup Group_types_enumeres
47 /// @{
48
49 /// Définition de l'enumeré concernant les types de vitesse de déformation
50
51 enum Enum_TypeVitDef { D_MOY_EPS_SUR_DT = 1,D_AVEC_V_A_T_PLUS_DT,
52 D_AVEC_DX_SUR_DT_A_T_PLUS_DT_SUR2, D_AVEC_V_MOY_A_T_PLUS_DT_SUR2};
53 /// @} // end of group
54
55
56 // Retourne un nom de type de vitesse de déformation a partir de son identificateur de
57 // type enumere id_TypeVitDef correspondant
58 char* Nom_TypeVitDef (const Enum_TypeVitDef id_TypeVitDef) ;
59
60 // Retourne l'identificateur de type enumere associe au nom du type
61 // de vitesse de déformation nom_TypeVitDef
62 Enum_TypeVitDef Id_nom_TypeVitDef (const char* nom_TypeVitDef);
63
64 // surcharge de l'operator de lecture
65 istream & operator » (istream & entree, Enum_TypeVitDef& a);
66 // surcharge de l'operator d'écriture
67 ostream & operator « (ostream & sort, const Enum_TypeVitDef& a);
68
69
70 #endif
71
72

```

## 7.325 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Enumeration/EnumTypeCL.h

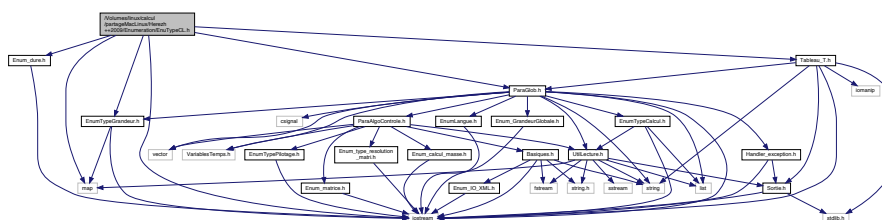
Type énuméré pour définir les différentes sous-classes de conditions limites.

```

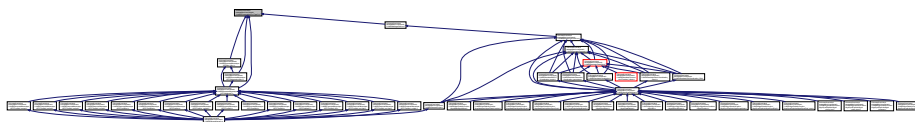
#include <iostream>
#include <map>
#include "ParaGlob.h"
#include "EnumTypeGrandeur.h"
#include "Enum_dure.h"
#include "Tableau_T.h"

```

Graphe des dépendances par inclusion de EnumTypeCL.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Classes

- class [ClassPourEnuTypeCL](#)  
def de la map qui fait la liaison entre les string et les énumérés

## Énumérations

- enum [EnuTypeCL](#) { **TANGENTE\_CL** , **RIEN\_TYPE\_CL** }  
Type énuméré pour définir les différentes sous-classes de conditions limites.

## Fonctions

- char \* **NomTypeCL** ([EnuTypeCL](#) id\_TypeCL)
- bool **Existe\_typeCL** (string &nom)
- [EnuTypeCL](#) **Id\_nomTypeCL** (char \*nom\_TypeCL)
- [EnuTypeCL](#) **Id\_nomTypeCL** (string &nom\_TypeCL)
- istream & **operator**>> (istream &entree, [EnuTypeCL](#) &a)
- ostream & **operator**<< (ostream &sort, const [EnuTypeCL](#) &a)

### 7.325.1 Description détaillée

Type énuméré pour définir les différentes sous-classes de conditions limites.

Date

04/10/2007

## 7.326 EnuTypeCL.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file EnuTypeCL.h
2   \brief Type énuméré pour définir les différentes sous-classes de conditions limites.
3 * \date    04/10/2007
4 */
5
6
7 // This file is part of the Herezh++ application.
8 //
9 // The finite element software Herezh++ is dedicated to the field
10 // of mechanics for large transformations of solid structures.
11 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
12 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
13 //
14 // Herezh++ is distributed under GPL 3 license ou ultérieure.
15 //
16 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
17 // AUTHOR : Gérard Rio
18 // E-MAIL : gerardrio56@free.fr
19 //
20 // This program is free software: you can redistribute it and/or modify
21 // it under the terms of the GNU General Public License as published by
22 // the Free Software Foundation, either version 3 of the License,
23 // or (at your option) any later version.
24 //
25 // This program is distributed in the hope that it will be useful,
26 // but WITHOUT ANY WARRANTY; without even the implied warranty
27 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
28 // See the GNU General Public License for more details.
29 //
30 // You should have received a copy of the GNU General Public License
31 // along with this program. If not, see <https://www.gnu.org/licenses/>.
32 //

```

```

33 // For more information, please consult: <https://herezh.irdl.fr/>.
34
35 /*****
36 *   DATE:      04/10/2007
37 *
38 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
39 *
40 *   PROJET:    Herezh++
41 *
42 *   $
43 *   BUT:      Type énuméré pour définir les différentes sous-classes
44 *             de conditions limites.
45 *
46 *   $
47 *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !           !           !
53 *   $
54 *
55 *   MODIFICATIONS:
56 *
57 *   ! date !   auteur !           but
58 *   -----
59 *   $
60 */***/
61 // Afin de realiser un gain en place memoire, les noms des types de grandeurs sont
62 // stockes a l'aide d'un type enumere.
63
64
65 #ifndef ENUTYPECL_H
66 #define ENUTYPECL_H
67 #include <iostream>
68 using namespace std;
69 #include <map>
70 #include "ParaGlob.h"
71 #include "EnumTypeGrandeur.h"
72 #include "Enum_dure.h"
73 #include "Tableau_T.h"
74
75 /// @addtogroup Group_types_enumeres
76 /// @{
77
78 /// Type énuméré pour définir les différentes sous-classes de conditions limites.
79 enum EnuTypeCL { TANGENTE_CL, RIEN_TYPE_CL};
80 /// @} // end of group
81
82 /// @addtogroup Group_types_enumeres
83 /// @{
84
85 /// def de la map qui fait la liaison entre les string et les énumérés
86
87 class ClassPourEnuTypeCL
88 { public:
89     /// def de la map qui fait la liaison entre les string et les énumérés
90     static map < string, EnuTypeCL , std::less < string> > map_EnuTypeCL;
91     // def de la grandeur statique qui permet de remplir la map
92     static ClassPourEnuTypeCL remplir_map;
93     // le constructeur qui rempli effectivement la map
94     ClassPourEnuTypeCL();
95 };
96 /// @} // end of group
97
98 // Retourne le nom du type de grandeur a partir de son identificateur de
99 // type enumere id_TypeCL correspondant
100 char* NomTypeCL (EnuTypeCL id_TypeCL);
101
102 // indique si le string correspond à un type CL reconnu ou non
103 bool Existe_typeCL(string& nom);
104
105 // Retourne l'identificateur de type enumere associe au nom du type de grandeur
106 // nom_TypeCL
107 EnuTypeCL Id_nomTypeCL (char* nom_TypeCL);
108 EnuTypeCL Id_nomTypeCL (string& nom_TypeCL);
109
110 // surcharge de l'operator de lecture
111 istream & operator » (istream & entree, EnuTypeCL& a);
112 // surcharge de l'operator d'écriture
113 ostream & operator « (ostream & sort, const EnuTypeCL& a);
114
115 #endif

```

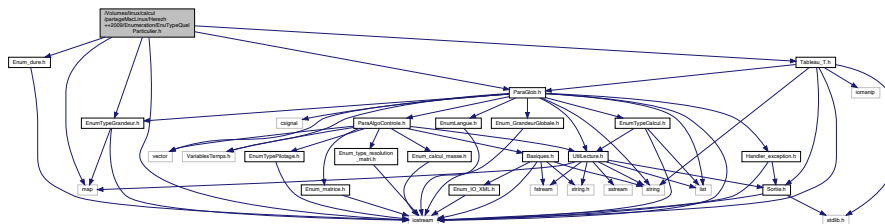


## 7.327 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/EnuTypeQuelParticulier.h

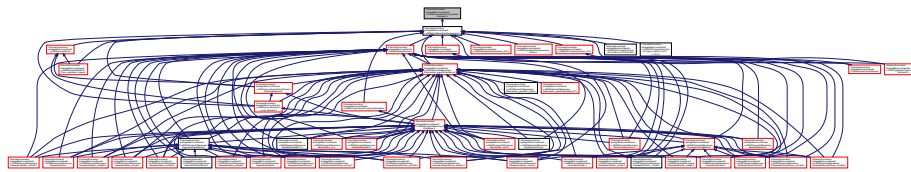
def de l'enuméré concernant les types quelconques particuliers.

```
#include <iostream>
#include <map>
#include "ParaGlob.h"
#include "EnumTypeGrandeur.h"
#include "Enum_dure.h"
#include "Tableau_T.h"
```

Graphe des dépendances par inclusion de EnuTypeQuelParticulier.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Classes

- class [ClassPourEnuTypeQuelParticulier](#)  
def de la map qui fait la liaison entre les string et les énumérés

### Énumérations

- enum [EnuTypeQuelParticulier](#) {  
**RIEN\_TYPE\_QUELCONQUE\_PARTICULIER** = 1 , **PARTICULIER\_TENSEURHH** , **PARTICULIER\_TENSEURBB** , **PARTICULIER\_COORDONNEE** ,  
**PARTICULIER\_TABLEAU\_COORDONNEE** , **PARTICULIER\_VECTEUR** , **PARTICULIER\_TABLEAU\_VECTEUR** , **PARTICULIER\_BASE\_H** ,  
**PARTICULIER\_BASE\_B** , **PARTICULIER\_TABLEAU\_BASE\_H** , **PARTICULIER\_TABLEAU\_BASE\_B** ,  
**PARTICULIER\_TABLEAU\_TENSEURHH** ,  
**PARTICULIER\_TABLEAU2\_TENSEURHH** , **PARTICULIER\_TENSEURBH** , **PARTICULIER\_SCALAIRE\_DOUBLE** ,  
**PARTICULIER\_TABLEAU\_SCALAIRE\_DOUBLE** ,  
**PARTICULIER\_SCALAIRE\_ENTIER** , **PARTICULIER\_TABLEAU\_SCALAIRE\_ENTIER** , **PARTICULIER\_TABLEAU\_QUELCONQUE** ,  
**PARTICULIER\_TABLEAU\_TENSEURBH** ,  
**PARTICULIER\_TABLEAU\_TENSEURBB** , **PARTICULIER\_TENSEURHB** , **PARTICULIER\_TABLEAU\_TENSEURHB** ,  
**PARTICULIER\_DDL\_ETENDU** ,  
**PARTICULIER\_TABLEAU\_DDL\_ETENDU** , **PARTICULIER\_VECTEUR\_NOMMER** , **PARTICULIER\_TABLEAU\_VECTEUR\_NOMMER** ,  
**PARTICULIER\_SCALAIRE\_DOUBLE\_NOMMER\_INDICER** }  
*Définition de l'enuméré concernant les types quelconques particuliers.*

### Fonctions

- string **NomTypeQuelParticulier** ([EnuTypeQuelParticulier](#) id\_TypeQuelParticulier)

- bool **Existe\_typeQuelParticulier** (string &nom)
- **EnuTypeQuelParticulier Id\_nomTypeQuelParticulier** (char \*nom\_TypeQuelParticulier)
- **EnuTypeQuelParticulier Id\_nomTypeQuelParticulier** (string &nom\_TypeQuelParticulier)
- istream & **operator**>> (istream &entree, **EnuTypeQuelParticulier** &a)
- ostream & **operator**<< (ostream &sort, const **EnuTypeQuelParticulier** &a)

### 7.327.1 Description détaillée

def de l'enuméré concernant les types quelconques particuliers.

## 7.328 EnuTypeQuelParticulier.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file EnuTypeQuelParticulier.h
2  \brief def de l'enuméré concernant les types quelconques particuliers.
3  */
4  // FICHER : EnuTypeQuelParticulier.h
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 // Afin de realiser un gain en place memoire, les noms des types de grandeurs sont
36 // stockes a l'aide d'un type enumere. Les fonctions NomTypeQuelParticulier et Id_nomTypeQuelParticulier
37 // rendent
38 // possible le lien entre les noms des geometries et les identificateurs
39 // de type enumere correspondants.
40
41 #ifndef ENUTYPEQUELPARTICULIER_H
42 #define ENUTYPEQUELPARTICULIER_H
43 #include <iostream>
44 using namespace std;
45 #include <map>
46 #include "ParaGlob.h"
47 #include "EnumTypeGrandeur.h"
48 #include "Enum_dure.h"
49 #include "Tableau_T.h"
50
51
52 /// @addtogroup Group_types_enumeres
53 /// @{
54
55 /// Définition de l'enuméré concernant les types quelconques particuliers.
56
57 enum EnuTypeQuelParticulier { RIEN_TYPE_QUELCONQUE_PARTICULIER = 1, PARTICULIER_TENSEURHH
58 , PARTICULIER_TENSEURBB, PARTICULIER_COORDONNEE
59 , PARTICULIER_TABLEAU_COORDONNEE
60 , PARTICULIER_VECTEUR, PARTICULIER_TABLEAU_VECTEUR
61 , PARTICULIER_BASE_H, PARTICULIER_BASE_B
62 , PARTICULIER_TABLEAU_BASE_H, PARTICULIER_TABLEAU_BASE_B
63 , PARTICULIER_TABLEAU_TENSEURHH, PARTICULIER_TABLEAU2_TENSEURHH
64 , PARTICULIER_TENSEURBH, PARTICULIER_SCALAIRE_DOUBLE
65 , PARTICULIER_TABLEAU_SCALAIRE_DOUBLE
66 , PARTICULIER_SCALAIRE_ENTIER, PARTICULIER_TABLEAU_SCALAIRE_ENTIER
67 , PARTICULIER_TABLEAU_QUELCONQUE

```

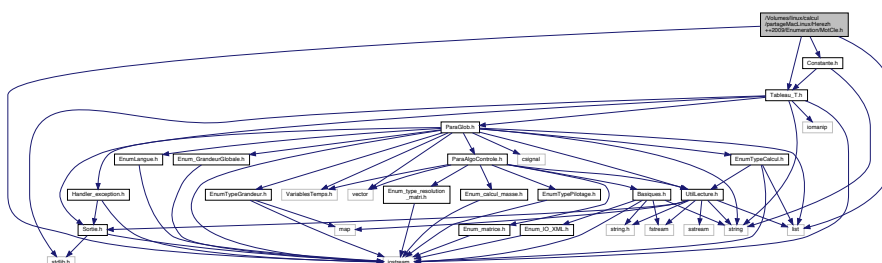
```
68     ,PARTICULIER_TABLEAU_TENSEURBH,PARTICULIER_TABLEAU_TENSEURBB
69     ,PARTICULIER_TENSEURHB,PARTICULIER_TABLEAU_TENSEURHB
70     ,PARTICULIER_DDL_ETENDU,PARTICULIER_TABLEAU_DDL_ETENDU
71     ,PARTICULIER_VECTEUR_NOMMER,PARTICULIER_TABLEAU_VECTEUR_NOMMER
72     ,PARTICULIER_SCALAIRE_DOUBLE_NOMMER_INDICER
73     };
74 // @} // end of group
75
76
77 // @addtogroup Group_types_enumeres
78 // @{
79
80 // def de la map qui fait la liaison entre les string et les énumérés
81
82 class ClassPourEnuTypeQuelParticulier
83 { public:
84     // def de la map qui fait la liaison entre les string et les énumérés
85     static map < string, EnuTypeQuelParticulier , std::less < string> > map_EnuTypeQuelParticulier;
86     // def de la grandeur statique qui permet de remplir la map
87     static ClassPourEnuTypeQuelParticulier remplir_map;
88     // le constructeur qui remplit effectivement la map
89     ClassPourEnuTypeQuelParticulier();
90 };
91 // @} // end of group
92
93 // Retourne le nom du type de grandeur a partir de son identificateur de
94 // type enumere id_TypeQuelParticulier correspondant
95 string NomTypeQuelParticulier (EnuTypeQuelParticulier id_TypeQuelParticulier);
96
97 // indique si le string correspond à un type quelParticulier reconnu ou non
98 bool Existe_typeQuelParticulier(string& nom);
99
100 // Retourne l'identificateur de type enumere associe au nom du type de grandeur
101 // nom_TypeQuelParticulier
102 EnuTypeQuelParticulier Id_nomTypeQuelParticulier (char* nom_TypeQuelParticulier);
103 EnuTypeQuelParticulier Id_nomTypeQuelParticulier (string& nom_TypeQuelParticulier);
104
105 // surcharge de l'operator de lecture
106 istream & operator » (istream & entree, EnuTypeQuelParticulier& a);
107 // surcharge de l'operator d'écriture
108 ostream & operator « (ostream & sort, const EnuTypeQuelParticulier& a);
109
110 #endif
```

## 7.329 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Enumeration/MotCle.h

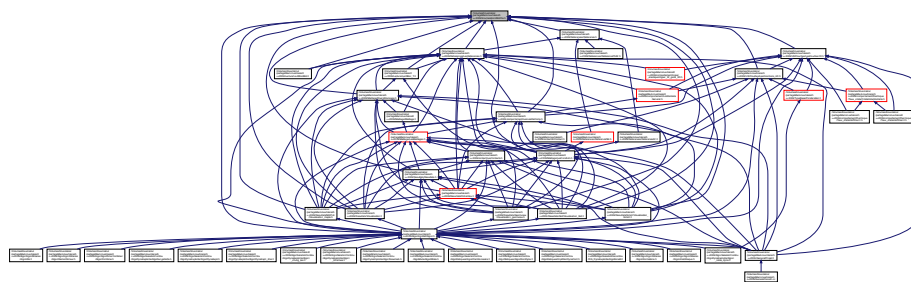
def Enumeration des differents mot cle dans le fichier d'entree.

```
#include <iostream>
#include "Tableau_T.h"
#include "Constante.h"
#include <list>
```

Grappe des dépendances par inclusion de MotCle.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Classes

- class [MotCle](#)  
*ici l'énuméré est remplacé par une classe*

### 7.329.1 Description détaillée

def Enumeration des differents mot cle dans le fichier d'entree.

Date

23/01/97

## 7.330 MotCle.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file MotCle.h
2  \brief def Enumeration des differents mot cle dans le fichier d'entree.
3  * \date      23/01/97
4  */
5
6  // This file is part of the Herezh++ application.
7  //
8  // The finite element software Herezh++ is dedicated to the field
9  // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34 /*****
35 *      DATE:          23/01/97
36 *
37 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *      PROJET:        Herezh++
40 *
41 *****/
42 *      BUT: Enumeration des differents mot cle dans le fichier d'entree.*
43 *
44 *      *****

```

```

45 *      VERIFICATION:                                *
46 *      ! date !   auteur !           but           ! *
47 *      ----- *
48 *      !           !           !           !           ! *
49 *      !           !           !           !           ! *
50 *      !           !           !           !           ! *
51 *      !           !           !           !           ! *
52 *      MODIFICATIONS:                                *
53 *      ! date !   auteur !           but           ! *
54 *      ----- *
55 *      !           !           !           !           ! *
56 *      !           !           !           !           ! *
57 *      !           !           !           !           ! *
57 #ifndef ENUM_MOTCLE_H
58 #define ENUM_MOTCLE_H
59
60 #include <iostream>
61 using namespace std;
62 // #include "bool.h"
63 #include "Tableau_T.h"
64 #include "Constante.h"
65 #include <list>
66
67 /// @addtogroup Group_types_enumeres
68 /// @{
69
70 /// ici l'énuméré est remplacé par une classe
71
72 class MotCle
73 {
74 public :
75     // VARIABLES PUBLIQUES :
76
77     // CONSTRUCTEURS :
78     MotCle ( const Tableau<string>& TsousMot = tab_Zero_string );
79     // DESTRUCTEUR :
80     ~MotCle () {};
81     // METHODES PUBLIQUES :
82
83     /// retourne true s'il y a un mot cle dans la chaine de character
84     /// false sinon
85     bool SimotCle(char* tabcar);
86
87 private :
88     // VARIABLES PROTEGEES :
89     list <string> lesmotscles; // liste des mots clés
90     Tableau <string> tabSous;
91     int TabSous_taille;
92     int lesmotscles_taille;
93     list <string>::iterator itdeb,itfin,it;
94 };
95 /// @} // end of group
96
97
98
99 #endif

```

## 7.331 LesValVecPropres.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //

```

```

28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:      Gestion des differents vecteurs et valeurs propres.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !       but
45 *   -----
46 *   !       !       !
47 *   *****
48 *   MODIFICATIONS:
49 *
50 *   ! date !   auteur !       but
51 *   -----
52 *   *****/
53 #ifndef LESVALVECPROPRES_H
54 #define LESVALVECPROPRES_H
55
56 #include "Tableau_T.h"
57 #include "VeurPropre.h"
58
59 class LesValVecPropres
60 {
61 public :
62     // CONSTRUCTEURS :
63     LesValVecPropres (); // par default
64     // fonction d'un tableau de valeurs propres
65     LesValVecPropres (Tableau <VeurPropre>& VP);
66     // DESTRUCTEUR :
67     ~LesValVecPropres ();
68
69     // constructeur de copie
70     LesValVecPropres (const LesValVecPropres & a);
71
72     // METHODES PUBLIQUES :
73     // surcharge de l'affectation
74     LesValVecPropres& operator= (const LesValVecPropres& a);
75
76     // affichage des valeurs et vecteurs propres
77     void Affiche(ofstream& sort) const ;
78
79 private :
80     // VARIABLES PROTEGEES :
81     Tableau <VeurPropre> VaeP;
82 };
83
84 #endif

```

### 7.332 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/General/herezh.cc

programme principal herezh++.

```

#include <iostream>
#include "herezh.h"
#include <math.h>
#include "ConstantePrinc.h"
#include "ParaGlob.h"
#include "Tenseur3.h"
#include "Tenseur2.h"
#include "Tenseur1.h"
#include "TenseurQ-3.h"
#include "TenseurQ-2.h"
#include "TenseurQ-1.h"
#include "Tenseur1_TroisSym.h"
#include "Tenseur2_TroisSym.h"

```

```
#include "Tenseur3_TroisSym.h"
#include "EnteteTenseur.h"
#include "NevezTenseur.h"
#include "TypeConstensPrinc.h"
#include "Projet.h"
#include "Biellette.h"
#include "BielletteQ.h"
#include "Biel_axi.h"
#include "Biel_axiQ.h"
#include "TriaMembL1.h"
#include "TriaMembQ3.h"
#include "TriaMembQ3_cm1pti.h"
#include "TriaQ3_cmpti1003.h"
#include "TriaCub.h"
#include "TriaCub_cm4pti.h"
#include "Quad.h"
#include "Quad_cm1pti.h"
#include "QuadQ.h"
#include "QuadQCom.h"
#include "QuadQCom_cm4pti.h"
#include "Hexa.h"
#include "Hexa_cm1pti.h"
#include "Hexa_cm27pti.h"
#include "Hexa_cm64pti.h"
#include "HexaQ.h"
#include "HexaQ_cm1pti.h"
#include "HexaQ_cm27pti.h"
#include "HexaQ_cm64pti.h"
#include "HexaQComp.h"
#include "HexaQComp_cm1pti.h"
#include "HexaQComp_cm27pti.h"
#include "HexaQComp_cm64pti.h"
#include "Tetra.h"
#include "TetraQ.h"
#include "TetraQ_cm1pti.h"
#include "TetraQ_cm15pti.h"
#include "PentaL.h"
#include "PentaL_cm1pti.h"
#include "PentaL_cm6pti.h"
#include "PentaQ.h"
#include "PentaQ_cm3pti.h"
#include "PentaQ_cm9pti.h"
#include "PentaQ_cm12pti.h"
#include "PentaQ_cm18pti.h"
#include "PentaQComp.h"
#include "PentaQComp_cm9pti.h"
#include "PentaQComp_cm12pti.h"
#include "PentaQComp_cm18pti.h"
#include "TriaSfe1.h"
#include "TriaSfe1_cm5pti.h"
#include "TriaSfe2.h"
#include "TriaSfe3.h"
#include "TriaSfe3_3D.h"
#include "TriaSfe3_cm3pti.h"
#include "TriaSfe3_cm4pti.h"
#include "TriaSfe3_cm5pti.h"
#include "TriaSfe3_cm6pti.h"
#include "TriaSfe3_cm7pti.h"
```

```

#include "TriaSfe3_cm12pti.h"
#include "TriaSfe3_cm13pti.h"
#include "TriaSfe3C.h"
#include "TriaQSfe3.h"
#include "TriaQSfe1.h"
#include "PoutSimple1.h"
#include "QuadCCom.h"
#include "QuadCCom_cm9pti.h"
#include "TriaAxiL1.h"
#include "TriaAxiQ3.h"
#include "TriaAxiQ3_cm1pti.h"
#include "TriaAxiQ3_cmpti1003.h"
#include "QuadAxiL1.h"
#include "QuadAxiL1_cm1pti.h"
#include "QuadAxiQ.h"
#include "QuadAxiQComp.h"
#include "QuadAxiQComp_cm4pti.h"
#include "QuadAxiCCom.h"
#include "QuadAxiCCom_cm9pti.h"
#include "ElemPoint.h"
#include "ElemPoint_CP.h"
#include "BielleTteThermi.h"
#include "Temps_CPU_HZpp.h"
#include "Temps_CPU_HZpp_3.h"

```

## Fonctions

— int **main** ()

### 7.332.1 Description détaillée

programme principal herezh++.

## 7.333 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/General/herezh.h

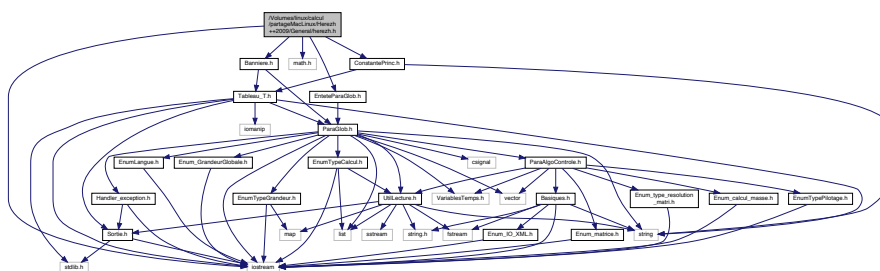
Entête du programme herezh++.

```

#include <iostream>
#include "Banniere.h"
#include <math.h>
#include "EnteteParaGlob.h"
#include "ConstantePrinc.h"

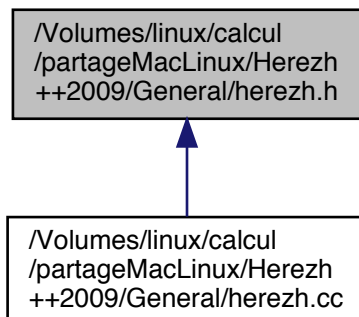
```

Graphe des dépendances par inclusion de herezh.h:





Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

— void **Presentation** ()

### 7.333.1 Description détaillée

Entête du programme herezh++.

## 7.334 herezh.h

[Aller à la documentation de ce fichier.](#)

```
1 /*! \file herezh.h
2    \brief Entête du programme herezh++.
3 */
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32 //
33 /*****
34 *
35 *   DATE:      18/10/2001
36 *
37 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
38 *
39 *   PROJET:    Herezh++
40 *
41 * *****/
```

```

40 *
41 *
42 * BUT: Entête du programme herezh++
43 *
44 *
45 * VERIFICATION:
46 *
47 * ! date ! auteur ! but
48 * -----
49 * ! ! !
50 *
51 *
52 *
53 * MODIFICATIONS:
54 *
55 * ! date ! auteur ! but
56 * -----
57 *
58 *
59 *
60 *****/
61 # include <iostream>
62
63 // #include "Debug.h"
64 #include "Banniere.h"
65
66 #include <math.h>
67 # include "EnteteParaGlob.h"
68
69 #include "ConstantePrinc.h"
70
71 void Presentation();
72
73 void Presentation()
74 { // affichage de l'entête du programme
75 // sortie de la banniere
76 Banniere::Sortie_banniere();
77 // puis affichage de la version
78 ParaGlob::Sortie_Version();
79 };

```

## 7.335 Projet.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *
32 * DATE: 15/01/97
33 *
34 * AUTEUR: G RIO (mailto:gerardrio56@free.fr)
35 *
36 * PROJET: Herezh++
37 *
38 *****/
39 * CLASSE: Projet
40 *
41 * BUT: definition des operations de haut niveau :

```

```

42 *           Lecture, Calcul, Sortie des Resultats.           *
43 *                                                                 $ *
44 *  ***** *
45 *  VERIFICATION: *
46 * * *
47 *  ! date ! auteur ! but ! *
48 * ----- *
49 *  ! ! ! ! *
50 * $ *
51 *  ***** *
52 *  MODIFICATIONS: *
53 * * *
54 *  ! date ! auteur ! but ! *
55 * ----- *
56 * * *
57 * $ *
58 * *
59 *****/
60 #ifndef PROJET_H
61 #define PROJET_H
62 #include <iostream>
63 //include "Debug.h"
64 #ifndef ENLINUX_STREAM
65 #include <sstream> // pour le flot en memoire centrale
66 #else
67 #include <stringstream> // pour le flot en memoire centrale
68 #endif
69 #include "UtilLecture.h"
70 #include "LesMaillages.h"
71 #include "ParaGlob.h"
72 // pour la definition de nouveaux tenseurs
73 #include "NevezTenseur.h"
74 // "DefValConstTens.h"
75 // est a placer dans le fichier qui defini les valeurs des constantes
76 // des tenseurs
77 #include "DefValConstTens.h"
78
79 #include "Algori.h"
80 #include "LesReferences.h"
81 #include "LesLoisDeComp.h"
82 #include "LesCourbes1D.h"
83 #include "LesFonctions_nD.h"
84 #include "DiversStockage.h"
85 #include "Charge.h"
86 #include "LesCondLim.h"
87 #include "Resultats.h"
88 #include "LesContacts.h"
89 #include "Enum_IO_XML.h"
90 #include "VariablesExporter.h"
91
92 //-----
93 //! definition des operations de haut niveau : Lecture, Calcul, Sortie des Resultats.
94 //-----
95 /// \author Gérard Rio
96 /// \version 1.0
97 /// \date 15/01/97
98
99
100 class Projet
101 {
102 public:
103 // ----- constructeurs ----- :
104 // défaut inutilisable
105 Projet () {cout << "\n erreur constructeur de Projet"; Sortie(1);};
106
107 // ++ le constructeur officiel ++
108 // il y a passage éventuelle d'argument : argc=le nombre d'argument
109 // argv : donne un tableau correspondant de mots clés
110 Projet (string& retour,int argc=0, const char * argv[] = NULL);
111 // ----- destructeur ----- :
112 ~Projet ();
113 // ----- methodes ----- :
114 // lecture initiale des informations
115 void Lecture();
116 // lectures secondaires éventuelles: ramène true s'il y a effectivement
117 // une lecture secondaire, qui donc modifie la lecture initiale
118 bool Lecture_secondaire();
119
120 // calculs
121 void Calcul ();
122
123 // type de calcul
124 EnumTypeCalcul TypeDeCalcul() const {return algori->TypeDeCalcul();};
125
126 // sortie des resultats
127 void SortieDesResultats ();

```

```

128
129 // visualisation (éventuelle)
130 void Visualisation_interactive ();
131
132 // définition d'un fichier de commandes
133 void Def_fichier_commande(string& retour);
134
135 // définition du fichier schema XML
136 void Def_fichier_SchemaXML(string& );
137
138 // fermeture des fichiers de visualisation liés au projet
139 void FermetureFichiersVisualisation();
140
141 // sortie sur fichier des temps cpu
142 // retourne un pointeur sur UtilLecture pour la récupération du fichier
143 // de temps, qui peut ainsi être utilisé par la méthode appelant
144 UtilLecture * Sortie_temps_cpu();
145
146 // def du fichier des temps cpu
147 // puis retourne un pointeur sur UtilLecture pour la récupération du fichier
148 // de temps, qui peut ainsi être utilisé par la méthode appelant
149 UtilLecture * Def_sortie_temps_cpu();
150
151 // une méthode qui a pour objectif de terminer tous les comptages, utile
152 // dans le cas d'un arrêt imprévu
153 void Arrêt_du_comptage_CPU();
154
155 private :
156
157 // VARIABLES PROTEGEES :
158 ParaGlob * paraGlob; // parametres globaux
159 UtilLecture * entreePrinc; // acces a la lecture du fichier principal
160 LesMaillages * lesMaillages; // description des maillages
161 Algori* algori; // algorithme de calcul
162 LesReferences* lesRef; // references des maillages
163 LesCourbesID* lesCourbesID; // courbes ID
164 LesFonctions_nD* lesFonctionsnD; // les fonctions multi-dimensionnelles
165 LesLoisDeComp* lesLoisDeComp; // lois de comportement
166 DiversStockage* diversStockage; // stockage divers
167 Charge* charge; // chargement
168 LesCondLim* lesCondLim; // conditions limites
169 LesContacts* lescontacts; // le contact eventuel
170 Resultats* resultats; // sortie des resultats
171 VariablesExporter* varExpor; // variables exportées globalement
172
173 Tableau <Algori* > tabAlgo; // pour le schema XML
174
175 Temps_CPU_HZpp tempsMiseEnDonnees; // TempsCpu
176
177 // METHODES PROTEGEES :
178 // lecture eventuelle de la dimension sinon elle est mise a trois
179 int LectureDimension();
180 // lecture eventuelle du niveau de sortie des commentaires
181 int LectureNiveauCommentaire();
182 // affichage de toutes les informations lue
183 void Affiche() const ;
184 // procedure qui definit un exemplaire de chaque element
185 // pour que l'edition de lien est lieu effectivement
186 // void DefElement();
187
188 // lecture des données externes pour chaque entité gérée par le projet
189 // type indique le type de données externes à lire
190 void LectureDonneesExternes();
191
192
193 // -- def de la dimension pour une sortie sur un fichier de commande
194 int Info_commande_dimension(UtilLecture * entreePrinc);
195 // -- def du niveau de commentaire pour une sortie sur un fichier de commande
196 int Info_commande_niveauCommentaire(UtilLecture * entreePrinc);
197
198 // -- def de l'entête pour une sortie schema XML
199 void SchemaXML_entete(UtilLecture * entreePrinc);
200 // -- def de la fin pour une sortie schema XML
201 void SchemaXML_fin(UtilLecture * entreePrinc);
202 // -- def de la dimension pour une sortie schema XML suivant le type d'io
203 void SchemaXML_dim(UtilLecture * entreePrinc, Enum_IO_XML enuIO_XML);
204 // sortie du schemaXML: en fonction de enu
205 // ceci pour tous les types de base
206 void SchemaXML_Type_de_base(ofstream& sort,const Enum_IO_XML enu);
207
208 void LectureVersion(); // lecture éventuelle de l'indicateur de version
209 void LectureLangue(); // lecture éventuelle de l'indicateur de langue
210
211 // ---- fonctions d'aides pour le constructeur de Projet ----
212 // démarrage: cas de la construction du schema XML
213 void InitConstructionSchemaXml(string & retour);
214 // démarrage: cas d'une écriture d'un fichier de commande

```

```

215 void InitConstructionFichierCommande(string & retour);
216 // démarrage: cas d'une entrée via un fichier .info
217 void InitEntreeFichierInfo(string & retour);
218 // démarrage: cas d'une entrée via un fichier .base-info
219 void InitEntreeFichierBaseInfo(string & retour);
220 };
221
222 #endif

```

## 7.336 Bloc.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Gestion de divers bloc courant du fichier de lecture .
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *
44 *      ! date !   auteur !           but
45 *      -----
46 *      !           !           !
47 *      *****
48 *      MODIFICATIONS:
49 *
50 *      ! date !   auteur !           but
51 *      -----
52 *      $
53 *****/
54 #ifndef BLOC_H
55 #define BLOC_H
56
57 #include <iostream>
58 #include <stdlib.h>
59 #include "Sortie.h"
60 #include "UtilLecture.h"
61 #include "string"
62 #include "string.h"
63 // #include "bool.h"
64 #include "ParaGlob.h"
65 #include "Tableau_T.h"
66 #include "Coordonnee.h"
67 #include "TypeQuelconque.h"
68
69 /** @defgroup Goupe_conteneurs_bloc
70 *
71 *      BUT:      groupe concernant la définition de bloc de stockage divers associant des opérations d'I/O
72 *
73 *

```

```

74 * \author    Gérard Rio
75 * \version  1.0
76 * \date     23/01/97
77 * \brief    groupe concernant la définition de bloc de stockage divers associant des opérations d'I/O
78 *
79 */
80
81
82 /// @addtogroup Goupe_conteneurs_bloc
83 /// @{
84 ///
85
86 //=====
87 //   cas d'un bloc scalaire normal
88 //=====
89 // un bloc scalaire normal correspond a une reference
90 // et une seule valeur
91
92 class BlocScal
93 {
94     // surcharge de l'operator de lecture
95     friend istream & operator » (istream &, BlocScal &);
96     // surcharge de l'operator d'écriture
97     friend ostream & operator « (ostream &, const BlocScal &);
98 public :
99     // VARIABLES PUBLIQUES :
100    // stockage d'un element
101    // class conforme a la specif de T de la class LectBloc_T
102
103    // Constructeur
104    BlocScal () ; // par défaut
105    BlocScal (const string nom, const double valeur)
106        : nomref(nom),val(valeur) {} ;
107    BlocScal (const BlocScal& a) ; // de copie
108    // destructeur
109    ~BlocScal () ;
110
111    //----- les méthodes constantes -----
112    // retourne le nom de reference
113    const string & NomRef() const;
114    // la méthode MemeCibleMaisDataDifférents, ne sert pas ici,
115    // mais elle est définit pour des raisons de compatibilité
116    bool MemeCibleMaisDataDifférents(BlocScal& ) const {return false;};
117    // affichage des informations
118    void Affiche() const ;
119    // surcharge des operateurs
120    bool operator == (const BlocScal& a) const;
121    // retourne la valeur en lecture
122    const double Val() const { return val;};
123    bool operator != (const BlocScal& a) const;
124    bool operator < (const BlocScal& a) const;
125
126    //----- les méthodes qui modifient -----
127    // modifie le nom de reference
128    void Change_Nomref(const string newnom) { nomref = newnom;};
129    // lecture d'un bloc
130    void Lecture(UtilLecture & entreePrinc);
131    // nom de maillage: par défaut null, surchargé ensuite si l'on veut
132    string * NomMaillage() {return NULL;};
133    BlocScal& operator = (const BlocScal& a);
134    // modifie la valeur
135    void Change_val(const double a) {val = a;};
136
137 protected :
138     double val; // valeur
139     string nomref; // nom de la ref
140
141 };
142 /// @} // end of group
143
144
145 /// @addtogroup Goupe_conteneurs_bloc
146 /// @{
147 ///
148
149 //=====
150 //   cas d'un bloc scalaire ou non de fonction nD
151 //=====
152 // un bloc qui correspond a une reference
153 // et soit une seule valeur ou soit le nom d'une fonction nD
154 // qui doit-être précédé par le mot clé une_fonction_nD_
155
156 class BlocScal_ou_fctnD
157 {
158     // surcharge de l'operator de lecture
159     friend istream & operator » (istream &, BlocScal_ou_fctnD &);
160     // surcharge de l'operator d'écriture

```

```

161     friend ostream & operator « (ostream &, const BlocScal_ou_fctnD &);
162 public :
163     // VARIABLES PUBLIQUES :
164     // stockage d'un element
165     // class conforme a la specif de T de la class LectBloc_T
166
167     // Constructeur
168     BlocScal_ou_fctnD () ; // par default
169     BlocScal_ou_fctnD (const string nom, const double valeur)
170         : nomref(nom),val(new double(valeur)) {} ;
171     BlocScal_ou_fctnD (const BlocScal_ou_fctnD& a) ; // de copie
172     // destructeur
173     ~BlocScal_ou_fctnD () ;
174
175     //----- les méthodes constantes -----
176     // retourne le nom de reference
177     const string & NomRef() const;
178     // la méthode MemeCibleMaisDataDifférents, ne sert pas ici,
179     // mais elle est définie pour des raisons de compatibilité
180     bool MemeCibleMaisDataDifférents(BlocScal_ou_fctnD& ) const {return false;};
181     // affichage des informations
182     void Affiche() const ;
183     // surcharge des operateurs
184     bool operator == (const BlocScal_ou_fctnD& a) const;
185     // retourne la valeur en lecture si elle existe
186     // sinon retourne un pointeur NULL
187     const double* Val() const { return val;};
188     // retourne le nom de la fonction nD en lecture si elle existe
189     // sinon retourne un pointeur NULL
190     const string* Fct_nD() const { return fctnD;};
191     bool operator != (const BlocScal_ou_fctnD& a) const;
192     bool operator < (const BlocScal_ou_fctnD& a) const;
193
194     //----- les méthodes qui modifient -----
195     // modifie le nom de reference
196     void Change_Nomref(const string newnom) { nomref = newnom;};
197     // lecture d'un bloc
198     void Lecture(UtilLecture & entreePrinc);
199     // nom de maillage: par défaut null, surchargé ensuite si l'on veut
200     string * NomMaillage() {return NULL;};
201     BlocScal_ou_fctnD& operator = (const BlocScal_ou_fctnD& a);
202     // modifie la valeur du double: possible uniquement s'il n'y a pas de fonction nD
203     void Change_val(const double a);
204     // modifie la valeur du nom de la fonction : possible uniquement s'il n'y a pas de valeur
205     void Change_fctnD(const string a);
206
207 protected :
208     double* val; // valeur si le pointeur est non nulle
209     string* fctnD; // nom de fonction si le pointeur est non nulle
210     string nomref; // nom de la ref
211
212 };
213 /// @} // end of group
214
215
216 /// @addtogroup Groupe_conteneurs_bloc
217 /// @{
218 ///
219
220 //=====
221 // cas d'un bloc scalaire type
222 //=====
223 // un bloc scalaire type correspond a une reference
224 // un mot cle, et une seule valeur
225
226 class BlocScalType
227 {
228     // surcharge de l'operator de lecture
229     friend istream & operator » (istream &, BlocScalType &);
230     // surcharge de l'operator d'écriture
231     friend ostream & operator « (ostream &, const BlocScalType &);
232 public :
233     // VARIABLES PUBLIQUES :
234     // stockage d'un element
235     // class conforme a la specif de T de la class LectBloc_T
236
237     // Constructeur
238     BlocScalType () ; // par default
239     BlocScalType (const BlocScalType& a) ; // de copie
240     // destructeur
241     ~BlocScalType () ;
242
243     //----- les méthodes constantes -----
244     // retourne le nom de reference
245     const string & NomRef() const;
246     // retourne le nom du mot clef
247     const string & Mot_clef() const {return motClef;};

```

```

248 // la méthode MemeCibleMaisDataDifferentes, ramène true si d et this on le même nom de ref et mot
    clé
249 // mais pas la même valeur
250 bool MemeCibleMaisDataDifferentes(BlocScalType& d) const
251     {if ((nomref == d.NomRef()) && (motClef == d.Mot_clef())
252         && (val != d.val))
253         {return true;}
254         else return false;
255     };
256 // affichage des informations
257 void Affiche() const ;
258 // surcharge des operateurs
259 bool operator == (const BlocScalType& a) const;
260 bool operator != (const BlocScalType& a) const;
261 // retourne la valeur en lecture
262 const double Val() const { return val;};
263
264 //----- les méthodes qui modifient -----
265 // nom de maillage: par défaut null, surchargé ensuite si l'on veut
266 string * NomMaillage() {return NULL;};
267 // modifie le nom de reference
268 void Change_Nomref(const string newnom) { nomref = newnom;};
269 // modifie le nom du mot clef
270 void Change_Mot_clef(const string newnom) { motClef = newnom;};
271 // lecture d'un bloc
272 void Lecture(UtilLecture & entreePrinc);
273 BlocScalType& operator = (const BlocScalType& a);
274 // modifie la valeur
275 void Change_val(const double a) {val = a;};
276
277 protected :
278     double val; // valeur
279     string nomref; // nom de la ref
280     string motClef; // nom du mot cle
281
282 };
283 /// @} // end of group
284
285
286 /// @addtogroup Groupe_conteneurs_bloc
287 /// @{
288 ///
289
290 //=====
291 // cas d'un bloc general de n chaines et de m scalaires
292 //=====
293 //
294
295 class BlocGen
296 {
297     // surcharge de l'operator de lecture
298     friend istream & operator » (istream &, BlocGen &);
299     // surcharge de l'operator d'écriture
300     friend ostream & operator « (ostream &, const BlocGen &);
301 public :
302     // VARIABLES PUBLIQUES :
303     // stockage d'un element
304     // class conforme a la specif de T de la class LectBloc_T
305     // Constructeur
306     // n= nombre de string, m nombre de double
307     BlocGen (int n=1, int m =1) ; // par default
308     BlocGen (const BlocGen& a) ; // de copie
309
310     // destructeur
311     ~BlocGen () ;
312
313     //----- les méthodes constantes -----
314     // retourne le nom de reference
315     // apriori c'est le premier nom stocke
316     string & NomRef() const;
317     // retourne le i ieme nom en lecture
318     inline const string& Nom(int i) const { return ptnom(i);};
319     // retourne la ieme valeur en lecture
320     inline const double& Val(int i) const { return pt(i);};
321     // retourne la dimension en nom
322     inline int DimNom() const{ return ptnom.Taille();};
323     // retourne la dimension en valeurs
324     inline int DimVal() const { return pt.Taille();};
325     // la méthode MemeCibleMaisDataDifferentes, ne sert pas ici,
326     // mais elle est définie pour des raisons de compatibilité
327     bool MemeCibleMaisDataDifferentes(BlocGen& )const {return false;};
328     // affichage des informations
329     void Affiche() const ;
330     // surcharge des operateurs
331     bool operator == (const BlocGen& a) const;
332     bool operator != (const BlocGen& a) const;
333

```



```

334 //----- les méthodes qui modifient -----
335 // change la ieme valeur
336 void Change_val(const int i, double val) {pt(i)=val;};
337 // change le i ieme nom
338 void Change_nom(const int i,const string& nom) { ptnom(i) = nom;};
339 // lecture d'un bloc
340 void Lecture(UtilLecture & entreePrinc);
341 // nom de maillage: par défaut null, surchargé ensuite si l'on veut
342 string * NomMaillage() {return NULL;};
343 BlocGen& operator = (const BlocGen& a);
344
345
346 protected :
347     Tableau<double> pt ; // les valeurs
348     Tableau<string> ptnom; // les noms
349 };
350 /// @} // end of group
351
352
353 /// @addtogroup Groupe_conteneurs_bloc
354 /// @{
355 ///
356
357 //=====
358 //     cas d'un bloc de 3 noms et 1 scalaires
359 //=====
360 //
361
362 class BlocGen_3_1 : public BlocGen
363 { // surcharge de l'operator de lecture
364     friend istream & operator » (istream & entree, BlocGen_3_1 & coo)
365     { entree » *((BlocGen*) &coo);return entree;};
366     // surcharge de l'operator d'écriture
367     friend ostream & operator « (ostream & sort, const BlocGen_3_1 & coo)
368     { sort « *((BlocGen*) &coo);return sort;};
369 public :
370     // Constructeur par défaut
371     BlocGen_3_1 () : BlocGen(3,1) { };
372 };
373 /// @} // end of group
374
375
376 /// @addtogroup Groupe_conteneurs_bloc
377 /// @{
378 ///
379
380 //=====
381 //     cas d'un bloc de 3 noms et 2 scalaires
382 //=====
383 //
384
385 class BlocGen_3_2 : public BlocGen
386 { // surcharge de l'operator de lecture
387     friend istream & operator » (istream & entree, BlocGen_3_2 & coo)
388     { entree » *((BlocGen*) &coo);return entree;};
389     // surcharge de l'operator d'écriture
390     friend ostream & operator « (ostream & sort, const BlocGen_3_2 & coo)
391     { sort « *((BlocGen*) &coo);return sort;};
392 public :
393     // Constructeur par défaut
394     BlocGen_3_2 () : BlocGen(3,2) { };
395 };
396 /// @} // end of group
397
398 /// @addtogroup Groupe_conteneurs_bloc
399 /// @{
400 ///
401
402 //=====
403 //     cas d'un bloc de 3 noms et 0 scalaires
404 //=====
405 //
406
407 class BlocGen_3_0 : public BlocGen
408 { // surcharge de l'operator de lecture
409     friend istream & operator » (istream & entree, BlocGen_3_0 & coo)
410     { entree » *((BlocGen*) &coo);return entree;};
411     // surcharge de l'operator d'écriture
412     friend ostream & operator « (ostream & sort, const BlocGen_3_0 & coo)
413     { sort « *((BlocGen*) &coo);return sort;};
414 public :
415     // Constructeur par défaut
416     BlocGen_3_0 () : BlocGen(3,0) { };
417 };
418 /// @} // end of group
419
420 /// @addtogroup Groupe_conteneurs_bloc

```

```

421 ///  

422 ///  

423 ///  

424 //=====
425 //   cas d'un bloc de 4 noms et 0 scalaires
426 //=====
427 ///  

428 ///  

429 class BlocGen_4_0 : public BlocGen
430 { // surcharge de l'operator de lecture
431   friend istream & operator » (istream & entree, BlocGen_4_0 & coo)
432     { entree » *((BlocGen*) &coo);return entree;} ;
433   // surcharge de l'operator d'écriture
434   friend ostream & operator « (ostream & sort, const BlocGen_4_0 & coo)
435     { sort « *((BlocGen*) &coo);return sort;} ;
436   public :
437     // Constructeur par défaut
438     BlocGen_4_0 () : BlocGen(4,0) { };
439 };
440 ///  

441 ///  

442 ///  

443 ///  

444 ///  

445 ///  

446 ///  

447 //=====
448 //   cas d'un bloc de 5 noms et 0 scalaires
449 //=====
450 ///  

451 ///  

452 class BlocGen_5_0 : public BlocGen
453 { // surcharge de l'operator de lecture
454   friend istream & operator » (istream & entree, BlocGen_5_0 & coo)
455     { entree » *((BlocGen*) &coo);return entree;} ;
456   // surcharge de l'operator d'écriture
457   friend ostream & operator « (ostream & sort, const BlocGen_5_0 & coo)
458     { sort « *((BlocGen*) &coo);return sort;} ;
459   public :
460     // Constructeur par défaut
461     BlocGen_5_0 () : BlocGen(5,0) { };
462 };
463 ///  

464 ///  

465 ///  

466 ///  

467 ///  

468 ///  

469 //=====
470 //   cas d'un bloc de 6 noms et 0 scalaires
471 //=====
472 ///  

473 ///  

474 class BlocGen_6_0 : public BlocGen
475 { // surcharge de l'operator de lecture
476   friend istream & operator » (istream & entree, BlocGen_6_0 & coo)
477     { entree » *((BlocGen*) &coo);return entree;} ;
478   // surcharge de l'operator d'écriture
479   friend ostream & operator « (ostream & sort, const BlocGen_6_0 & coo)
480     { sort « *((BlocGen*) &coo);return sort;} ;
481   public :
482     // Constructeur par défaut
483     BlocGen_6_0 () : BlocGen(6,0) { };
484 };
485 ///  

486 ///  

487 ///  

488 ///  

489 ///  

490 ///  

491 ///  

492 //=====
493 //   cas d'un bloc vecteur
494 //=====
495 // un bloc vecteur correspond a une reference et
496 // un vecteur, fonction de la dimension du pb
497 ///  

498 class BlocVec
499 {
500   // surcharge de l'operator de lecture
501   friend istream & operator » (istream &, BlocVec &);
502   // surcharge de l'operator d'écriture
503   friend ostream & operator « (ostream &, const BlocVec &);
504   public :
505     // VARIABLES PUBLIQUES :
506     // stockage d'un element
507     // class conforme a la specif de T de la class LectBloc_T

```

```

508
509 // Constructeur
510 BlocVec () ; // par défaut
511 BlocVec (const BlocVec & a) ; // de copie
512 // destructeur
513 ~BlocVec () ;
514
515 //----- les méthodes constantes -----
516 // retourne le nom de reference
517 const string & NomRef() const;
518 // la méthode MemeCibleMaisDataDifferents, ne sert pas ici,
519 // mais elle est définie pour des raisons de compatibilité
520 bool MemeCibleMaisDataDifferents(BlocVec& ) const {return false;};
521 // affichage des informations
522 void Affiche() const ;
523 // surcharge des operateurs
524 bool operator == (const BlocVec& a) const;
525 bool operator != (const BlocVec& a) const;
526 // retourne les coordonnées en constantes
527 const Coordonnee& Coord() const { return coorpt;};
528 // retourne une coordonnée
529 double Coord(int i) const { return coorpt(i);};
530
531 //----- les méthodes qui modifient -----
532 // nom de maillage: par défaut null, surchargé ensuite si l'on veut
533 string * NomMaillage() {return NULL;};
534 // lecture d'un bloc en fonction de la dimension
535 void Lecture(UtilLecture & entreePrinc);
536 BlocVec& operator = (const BlocVec& a);
537 // modifie les valeurs du vecteur
538 void Change_val(Coordonnee& a) {coorpt = a;};
539
540
541 protected :
542     Coordonnee coorpt; // les coordonnées du vecteur
543     string nomref; // nom de la ref
544 };
545 /// @} // end of group
546
547
548
549 /// @addtogroup Goupe_conteneurs_bloc
550 /// @{
551 ///
552
553 //=====
554 // cas d'un bloc vecteur type
555 //=====
556 // un bloc vecteur type correspond a une reference et
557 // un mot cle et un vecteur, fonction de la dimension du pb
558
559 class BlocVecType
560 {
561 // surcharge de l'operator de lecture
562 friend istream & operator » (istream &, BlocVecType &);
563 // surcharge de l'operator d'écriture
564 friend ostream & operator « (ostream &, const BlocVecType &);
565 public :
566 // VARIABLES PUBLIQUES :
567 // stockage d'un element
568 // class conforme a la specif de T de la class LectBloc_T
569
570 // Constructeur
571 BlocVecType () ; // par défaut
572 BlocVecType (const BlocVecType & a) ; // par défaut
573 // destructeur
574 ~BlocVecType () ;
575
576 //----- les méthodes constantes -----
577 // retourne le nom de reference
578 const string & NomRef() const;
579 // retourne le nom du mot clef
580 const string & Mot_clef() const {return motClef;};
581 // la méthode MemeCibleMaisDataDifferents, ramène true
582 // si d et this on le même nom de ref et mot clé
583 // mais pas les mêmes coordonnées
584 bool MemeCibleMaisDataDifferents(BlocVecType& d)const
585 {if ((nomref == d.NomRef()) && (motClef == d.Mot_clef())
586     && (coorpt != d.coorpt))
587     {return true;}
588     else return false;
589 };
590 // affichage des informations
591 void Affiche() const ;
592 // surcharge des operateurs
593 bool operator == (const BlocVecType& a) const;
594 bool operator != (const BlocVecType& a) const;

```

```

595         // retourne les coordonnées en constantes
596         const Coordonnee& Coord() const { return coorpt;};
597         // retourne une coordonnée
598         double Coord(int i) const { return coorpt(i);};
599
600         //----- les méthodes qui modifient -----
601         // nom de maillage: par défaut null, surchargé ensuite si l'on veut
602         string * NomMaillage() {return NULL;};
603         // lecture d'un bloc en fonction de la dimension
604         void Lecture(UtilLecture & entreePrinc);
605         // modifie les valeurs du vecteur
606         void Change_val(Coordonnee& a) {coorpt = a;};
607         BlocVecType& operator = (const BlocVecType& a);
608
609
610     protected :
611         Coordonnee coorpt; // les coordonnées du vecteur
612         string nomref; // nom de la ref
613         string motClef; // nom du mot cle
614     };
615     /// @} // end of group
616
617     /// @addtogroup Goupe_conteneurs_bloc
618     /// @{
619     ///
620     ///
621     ///=====
622     // cas d'un bloc de n vecteur et m scalaires type
623     //=====
624     // un bloc de plusieurs vecteurs et plusieurs scalaires type correspond a une reference et
625     // un mot cle et des vecteurs, fonction de la dimension du pb et des scalaires
626
627     class BlocVecMultType
628     {
629     {
630         // surcharge de l'operator de lecture
631         friend istream & operator » (istream &, BlocVecMultType &);
632         // surcharge de l'operator d'écriture
633         friend ostream & operator « (ostream &, const BlocVecMultType &);
634     public :
635         // VARIABLES PUBLIQUES :
636         // stockage d'un element
637         // class conforme a la specif de T de la class LectBloc_T
638
639         // Constructeur
640         // n est le nombre de vecteur à dimensionner
641         // m le nombre de scalaires
642         BlocVecMultType (int n=1,int m=1) ; // par défaut
643         BlocVecMultType (const BlocVecMultType & a) ; // par défaut
644         // destructeur
645         ~BlocVecMultType () ;
646
647         //----- les méthodes constantes -----
648         // retourne le nom de reference
649         const string & NomRef() const;
650         // retourne le nom du mot clef
651         const string & Mot_clef() const {return motClef;};
652         // la méthode MemeCibleMaisDataDifferentes, ramène true si d et this on le même nom de ref et mot
653         clé
654         // mais que le tableau de coordonnées ou de scalaire n'est pas identique
655         bool MemeCibleMaisDataDifferentes(BlocVecMultType& d) const
656         {if ((nomref == d.NomRef()) && (motClef == d.Mot_clef())
657             && (((this->pt)!=d.pt) || (this->vect_coorpt)!= d.vect_coorpt) )
658             {return true;}
659             else return false;
660         };
661         // affichage des informations
662         void Affiche() const ;
663         // surcharge des operateurs
664         bool operator == (const BlocVecMultType& a) const;
665         bool operator != (const BlocVecMultType& a) const;
666         // retourne les coordonnées du vecteur i en constantes
667         const Coordonnee& Vect_de_cooronnee (int i) const { return vect_coorpt(i);};
668         // retourne la ieme valeur scalaire en lecture
669         inline const double& Val(int i) const { return pt(i);};
670         // retourne la dimension en vecteur
671         inline int DimVect() const { return vect_coorpt.Taille();};
672         // retourne la dimension en scalaire
673         inline int DimVal() const { return pt.Taille();};
674
675         //----- les méthodes qui modifient -----
676         // nom de maillage: par défaut null, surchargé ensuite si l'on veut
677         string * NomMaillage() {return NULL;};
678         // lecture d'un bloc en fonction de la dimension
679         void Lecture(UtilLecture & entreePrinc);
680         BlocVecMultType& operator = (const BlocVecMultType& a);
681         // modifie les valeurs d'un vecteur i

```

```

681     void Change_vect_de_coordonnee(int i, const Coordonnee& a) {vect_coorpt(i) = a;};
682     // change la ieme valeur scalaire
683     void Change_val(const int i, double val) {pt(i)=val;};
684
685
686 protected :
687     Tableau <Coordonnee> vect_coorpt; // les coordonnées du vecteur
688     Tableau<double> pt ; // les scalaires
689     string nomref; // nom de la ref
690     string motClef; // nom du mot cle
691 };
692 /// @} // end of group
693
694
695 /// @addtogroup Groupe_conteneurs_bloc
696 /// @{
697 ///
698
699 //=====
700 // cas d'un bloc general de n vecteurs, m scalaire
701 // avec un choix en lecture et définition des coordonnees ou de chaine
702 //=====
703 //
704 class BlocGeneEtVecMultType
705 { // surcharge de l'operator de lecture
706     friend istream & operator » (istream & entree, BlocGeneEtVecMultType & );
707     // surcharge de l'operator d'écriture
708     friend ostream & operator « (ostream & sort, const BlocGeneEtVecMultType & );
709 public :
710     // constructeur par défaut
711     // n: le nombre de vecteurs, m le nombre de scalaires , n le nombre de ptnom
712     BlocGeneEtVecMultType(int n=1,int m=1);
713     BlocGeneEtVecMultType (const BlocGeneEtVecMultType & a) ; // constructeur par défaut
714     ~BlocGeneEtVecMultType () ; // destructeur
715
716     //----- les méthodes constantes -----
717     // retourne le nom de reference
718     const string & NomRef() const {return nomref;};
719     // retourne le nom du mot clef
720     const string & Mot_clef() const {return motClef;};
721     // la méthode MemeCibleMaisDataDifférents, ramène true si d et this on le même nom de ref et mot
722     clé // mais que le tableau de coordonnées ou de scalaire n'est pas identique
723     bool MemeCibleMaisDataDifférents(BlocGeneEtVecMultType& d) const
724     {if ((nomref == d.NomRef()) && (motClef == d.Mot_clef())
725         && (((this->tab_val)!=d.tab_val) || ((this->vect_coorpt)!= d.vect_coorpt)
726             || ((this->ptnom_vect)!=d.ptnom_vect))
727         )
728         {return true;}
729         else return false;
730     };
731     // affichage des informations
732     void Affiche() const ;
733     // surcharge des operateurs
734     bool operator == (const BlocGeneEtVecMultType& a) const;
735     bool operator != (const BlocGeneEtVecMultType& a) const;
736     // retourne les coordonnées du vecteur i en constantes
737     const Coordonnee& Vect_de_coordonnee(int i) const { return vect_coorpt(i);};
738     // retourne la ieme valeur scalaire en lecture
739     inline const double& Val(int i) const { return tab_val(i);};
740     // retourne la dimension en vecteur
741     inline int DimVect() const { return vect_coorpt.Taille();};
742     // retourne le nombre de scalaire
743     inline int DimVal() const { return tab_val.Taille();};
744     // retourne le (i,j) ieme nom associé aux vecteurs en lecture
745     inline const string& Nom_vect(int i,int j) const { return ptnom_vect(i)(j);};
746     // retourne le (i) ieme nom associé aux valeurs en lecture
747     inline const string& Nom_val(int i) const { return ptnom_tab_val(i);};
748
749     //----- les méthodes qui modifient -----
750     BlocGeneEtVecMultType& operator = (const BlocGeneEtVecMultType& a);
751     // lecture d'un bloc en fonction de la dimension
752     void Lecture(UtilLecture & entreePrinc);
753     // lecture uniquement de l'entête: nom ref et mot clef
754     void Lecture_entete(UtilLecture & entreePrinc)
755     { *(entreePrinc.entree) » nomref » motClef;};
756
757     // modifie les valeurs d'un vecteur i
758     void Change_vect_de_coordonnee(int i, const Coordonnee& a) {vect_coorpt(i) = a;};
759     // nom de maillage: par défaut null, surchargé ensuite si l'on veut
760     string * NomMaillage() {return NULL;};
761     // change la ieme valeur scalaire
762     void Change_val(const int i, double val) {tab_val(i)=val;};
763     // change le (i,j) ieme nom associé aux vecteurs
764     void Change_nom_val(int i,const string& nom) { ptnom_tab_val(i) = nom;};
765     // change le tableau des coordonnées, la taille des noms associés est modifié
766     // en conséquence

```

```

767 void Change_vect_coorpt(const Tableau <Coordonnee>& vect);
768 // change le tableau des coordonnées, la taille des noms associés est modifié
769 // en conséquence, ici à partir d'une liste
770 void Change_vect_coorpt(const list<Coordonnee>& livect);
771 // change le tableau des scalaires
772 // la taille des noms associés est modifié en conséquence
773 void Change_tab_val(const Tableau <double>& t_val)
774 {tab_val=t_val;ptnom_tab_val.Change_taille(t_val.Taille());};
775 // change le tableau de scalaires en fonction d'une liste
776 // la taille des noms associés est modifié en conséquence
777 void Change_tab_val(const list<double>& li);
778 // change le tableau des noms associés aux vecteurs
779 // la taille du tableau de vecteurs associés est modifié en conséquence
780 void Change_ptnom_vect(const Tableau <Tableau<string> >& t_ptnom_vect);
781 // change le (i,j) ieme nom associé aux vecteurs
782 void Change_nom_vect(int i,int j,const string& nom) { ptnom_vect(i)(j) = nom;};
783 // change le tableau des noms associés aux scalaires
784 // la taille du tableau de vecteurs associés est modifié en conséquence
785 void Change_ptnom_val(const Tableau<string> & t_ptnom_val);
786 // change le tableau des noms associé aux scalaires en fonction d'une liste
787 // la taille du tableau de scalaires associés est modifié en conséquence
788 void Change_ptnom_val(const list<string>& li);
789
790 protected :
791 Tableau <Coordonnee> vect_coorpt; // les coordonnées du vecteur
792 Tableau <Tableau<string> > ptnom_vect; // les noms associés aux vecteurs
793 Tableau<double> tab_val ; // les scalaires
794 Tableau <string > ptnom_tab_val; // les noms associés aux scalaires
795 string nomref; // nom de la ref
796 string motClef; // nom du mot cle
797 };
798 /// @} // end of group
799
800
801 #endif

```

## 7.337 LectBloc\_T.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 * DATE: 23/01/97 *
32 * * $ *
33 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
34 * * $ *
35 * PROJET: Herezh++ *
36 * * $ *
37 *****/
38 * BUT: Class template pour la lecture d'unbloc delimite par : *
39 * mots cles. *
40 * * $ *
41 * ***** *
42 * VERIFICATION: *
43 * * *
44 * ! date ! auteur ! but ! *
45 * ----- *
46 * ! ! ! ! *

```

```

47 *                                     $ *
48 *          ***** *
49 *      MODIFICATIONS: *
50 *      ! date ! auteur ! but *
51 *      ----- *
52 *                                     $ *
53 *****/
54 #ifndef LECTBLOC_T_H
55 #define LECTBLOC_T_H
56
57 #include <stdlib.h>
58 using namespace std; //introduces namespace std
59
60 #include <iostream>
61 #include <stdlib.h>
62 #include "Sortie.h"
63 #include "UtilLecture.h"
64 #include "LesReferences.h"
65 #include "string"
66 #include "MotCle.h"
67 #include <list>
68 #include "string.h"
69 #include "Tableau_T.h"
70 #include "Constante.h"
71 #include "ParaGlob.h"
72
73
74 template <class T>
75
76 /// @addtogroup Goupe_relatif_aux_entrees_sorties
77 /// @{
78 ///
79
80
81 class LectBloc
82 {
83 public :
84     // CONSTRUCTEURS :
85     LectBloc () {} ;
86     // DESTRUCTEUR :
87     ~LectBloc () {} ;
88     // METHODES PUBLIQUES :
89
90     // lecture d'un bloc
91     // entreePrinc : stream de lecture ( sur le fichier d'entree)
92     // lesRef : liste des references generale lue
93     // motcle de debut de lecture
94     // message : a afficher en cas d'erreur
95     // tab : tableau des information a lire
96     // il faut que la classe T possede les proprietees suivantes :
97     // 1) constructeur par default, 2) NomRef() fourni une reference sur un string contenant la
98     // reference d'identification de l'instance, 3) les operators == , !=
99     // et = doivent etre surcharges, 4) Lecture(UtilLecture &) une fonction qui permet de lire
100    // une instance de T sur l'unite UtilLecture , 5) Affiche () une fonction qui affiche les infos
101    // specifique a la classe T, 6) NomMaillage() qui ramene un pointeur sur un nom de maillage
102    // eventuelle, le pointeur peut sinon être null.
103    // les derniers arguments sont par default, il permet de specifier un tableau de sous mot cle
104    // delimitant des sous blocs.
105    // la lecture s'arrete lorsque l'on aura trouve soit un mot cle ou soit un sous mot cle
106    // saut : indique si l'instance T est a lire immediatement ou s'il faut demander une
107    // nouvelle ligne avant sa lecture
108
109    void Lecture(UtilLecture & entreePrinc,LesReferences& lesRef,
110                string motcle, string message,Tableau<T>& tab,
111                Tableau<string>& TsousMot = tab_Zero_string, bool saut = true);
112
113
114 private :
115     // VARIABLES PROTEGEES :
116     // CONSTRUCTEURS :
117
118     // DESTRUCTEUR :
119
120     // METHODES PROTEGEES :
121
122 };
123 /// @} // end of group
124
125 template <class T>
126 // lecture d'un bloc
127 void LectBloc<T>::Lecture(UtilLecture & entreePrinc,LesReferences& lesRef,
128                          string motcle, string message,Tableau<T>& tab,Tableau<string>& TsousMot,
129                          bool saut )
130 { MotCle motCle(TsousMot); // ref aux mots cle
131   if (strstr(entreePrinc.tablcar,motcle.c_str())!=NULL)
132   { if (ParaGlob::NiveauImpression() >= 7)
133     cout << " lecture " << motcle << flush;

```

```

134     if (saut) entreePrinc.NouvelleDonnee();
135     list <T> lili;
136     T elem;
137     while ( !motCle.SimotCle(entreePrinc.tablcar) )
138     { // on initialise elem à partir d'une variable nouvellement construite par défaut
139         T initial;
140         elem = initial;
141         // lecture
142         elem.Lecture(entreePrinc);
143         // on regarde si cela correspond bien a une reference existante
144         if (!lesRef.Existe(elem.NomRef(),elem.NomMaillage()))
145         { cout << "\n erreur, la ref " << elem.NomRef() << " de(s) " << motcle << " ne correspond a
aucun element"
146             << " de la liste de reference lue !! \n";
147             elem.Affiche();
148             cout << " LectBloc::Lecture( etc ..."
149                 << endl;
150             entreePrinc.MessageBuffer(message);
151             throw (UtilLecture::ErrNouvelleDonnee(-1));
152             Sortie (1);
153         };
154         // on verifie que l'on n'utilise pas deux fois la meme reference
155         // pour deux cibles differentes
156         typename list <T>::iterator ii;
157         for (ii=lili.begin() ; ii != lili.end(); ii++)
158         //         if ((*ii).NomRef() == elem.NomRef()) && ((*ii) != elem)    pour adaptation linux

159         if ((*ii).NomRef() == elem.NomRef()) && !(elem == (*ii))
160             && ((*ii).MemeCibleMaisDataDifferentes(elem) )
161         { // on regarde pour le nom de maillage
162             bool message = true;
163             // si les deux maillages sont différents il n'y a pas d'erreur
164             if ((*ii).NomMaillage() != NULL) && (elem.NomMaillage() != NULL)
165                 if ( *((*ii).NomMaillage()) != *(elem.NomMaillage()) ) message = false;
166             if (message) && (ParaGlob::NiveauImpression() > 0)
167                 { // on écrit un message pour avertir de l'utilisation d'une référence plusieurs
168                     // fois
169                     cout << "\n ---- remarque ---- , un meme nom de reference est utilisee pour"
170                         << " deux cibles differentes \n";
171                     cout << " premiere cible : "; (*ii).Affiche();
172                     cout << " seconde cible : "; elem.Affiche();
173                     if (ParaGlob::NiveauImpression() >= 4)
174                         cout << " LectBloc::Lecture( etc etc .." ;
175                     cout << endl;
176                 };
177             /*         cout << "\n erreur , un meme nom de reference est utilisee pour"
178                 << " deux cibles differentes \n";
179                 cout << " premiere cible : "; (*ii).Affiche();
180                 cout << " seconde cible : "; elem.Affiche();
181                 cout << " LectBloc::Lecture( etc etc .." << endl;
182                 entreePrinc.MessageBuffer(message);
183                 throw (UtilLecture::ErrNouvelleDonnee(-1));
184                 Sortie (1);          */
185         };
186         // stockage
187         lili.push_back(elem);
188         entreePrinc.NouvelleDonnee();
189     }
190     // enregistrement des infos
191     int ancien = tab.Taille();
192     tab.Change_taille(((int)lili.size()+ancien);
193     typename list <T>::iterator i,ifin=lili.end();
194     int j;
195     for (i=lili.begin(),j=1 ; i != ifin; i++,j++)
196         tab(j+ancien) = *i;
197 };
198 }
199
200
201
202 #endif

```

## 7.338 LectBlocMot.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)

```



```

12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *
38 *   BUT:   Class pour la lecture d'une reference et d'un tableau de
39 *          string devant appartenir a une liste identifiee.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *   *****/
54 #ifndef LECTBLOCMOT_H
55 #define LECTBLOCMOT_H
56
57 #include <iostream>
58 #include <stdlib.h>
59 #include "Sortie.h"
60 #include "UtilLecture.h"
61 #include "LesReferences.h"
62 #include "string"
63 #include "MotCle.h"
64 #include <list>
65 #include "string.h"
66 #include "Tableau_T.h"
67 #include "Constante.h"
68 #include "Bloc.h"
69 #include "CharUtil.h"
70 #include "BlocDdlLim.h"
71
72
73 /// @addtogroup Groupe_relatif_aux_entrees_sorties
74 /// @{
75 ///
76
77
78 class LectBlocmot
79 {
80 public :
81     // CONSTRUCTEURS :
82     LectBlocmot () {};
83     // DESTRUCTEUR :
84     ~LectBlocmot () {};
85     // METHODES PUBLIQUES :
86
87     // lecture d'un bloc de mot
88
89     // blocMot : contient la ref lue et la liste des mots lue
90     // entreePrinc : stream de lecture ( sur le fichier d'entree)
91     // lesRef : liste des references generale lue
92     // message : a afficher en cas d'erreur
93     // tabMot : contient la liste des mots acceptable pour la lecture
94     // TsousMot : tableau de sous mot cle permettant l'arret de la lecture
95     //
96     // la lecture s'arretera lorsque l'on aura trouve soit un mot cle ou soit un sous mot cle
97

```

```

98     BlocDdlLim< BlocGen> Lecture(UtilLecture & entreePrinc,LesReferences& lesRef,string message,
99         Tableau<string>& tabMot,Tableau<string>& TsousMot);
100
101
102 private :
103     // VARIABLES PROTEGEES :
104     // METHODES PROTEGEES :
105
106 };
107     /// @} // end of group
108
109 #endif

```

## 7.339 UtilLecture.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *
32 *     DATE:          23/01/97
33 *
34 *     AUTEUR:       G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *     PROJET:      Herezh++
37 *
38 *
39 *     BUT:          definir l'ensemble des outils relatifs aux
40 *                  operations basiques de lecture dans le fichier
41 *                  d'entree
42 *
43 *
44 *     VERIFICATION:
45 *
46 *     ! date !   auteur !           but
47 *     -----
48 *     !           !           !
49 *
50 *
51 *
52 *     MODIFICATIONS:
53 *
54 *     ! date !   auteur !           but
55 *     -----
56 *
57 *
58 *
59 *****/
60
61 #ifndef UtiLecture_H
62 #define UtiLecture_H
63
64 #include <fstream>
65 // #include "Debug.h"
66
67 #ifndef ENLINUX_STREAM
68 #include <sstream> // pour le flot en memoire centrale
69 #else

```

```

70 #include <sstream> // pour le flot en memoire centrale
71 #endif
72 #include <string.h>
73 #include <string>
74 #include <list>
75 #include "Sortie.h"
76 #include <map>
77     using namespace std;
78
79 /** @defgroup Goupe_relatif_aux_entrees_sorties
80 *
81 *     BUT:   groupe relatif aux méthodes de lectures et d'écritures sur fichiers, écrans
82 *
83 *
84 * \author   Gérard Rio
85 * \version  1.0
86 * \date    23/01/97
87 * \brief   groupe relatif aux méthodes de lectures et d'écritures sur fichiers, écrans
88 *
89 */
90
91 /// @addtogroup Goupe_relatif_aux_entrees_sorties
92 /// @{
93 ///
94
95
96 class UtilLecture
97 {
98     public :
99     // VARIABLES PUBLIQUES :
100
101     // ----- cas du fichier principal .info et dérivés -----
102
103     char* tablcar ; // buffer contenant les infos utiles courantes
104     const int longueur; // longueur maxi du tableau de travail
105     class ErrNouvelleDonnee // gestion d'exception pour nouvel_Enreg
106     // =1 lecture OK on est arrive a la fin de tous les fichiers, < 0 probleme de lecture
107     { public :
108         int lecture;
109         ErrNouvelleDonnee (int entrees) {lecture = entrees;} ; // CONSTRUCTEURS
110         ~ErrNouvelleDonnee () {};// DESTRUCTEUR :
111     };
112
113     // Flot d'entree
114     // pour la lecture en memoire centrale, permet la conversion de type
115     // dans le cas de lecture de tableau de character il faut tout
116     // d'abord preciser le nombre de caractere a lire avec la fonction
117     // membre width(nb de crarac)
118     // a utiliser pour la lecture d'entier, de reelle etc
119
120     #ifndef ENLINUX_STREAM
121         istringstream * entree;
122         ostringstream * sortie;
123     #else
124         istrstream * entree;
125         ostrstream * sortie;
126     #endif
127
128     // retour du fichier dans lequel on écrit les commandes .info
129     ofstream * Commande_pointInfo() {return ptrcommandefich;};
130     // retour du fichier dans lequel on écrit le schema XML
131     ofstream * ShemaXML() {return ptrschemaXMLfich;};
132
133     // fichier de recopie des informations lue par le programme
134     ofstream * copie;
135
136     // ----- fin du cas du fichier principal .info et dérivés -----
137     // ----- cas du fichier de commandes de visualisation -----
138
139     char* tablcarCVisu ; // buffer contenant les infos utiles courantes
140     class ErrNouvelleDonneeCVisu // gestion d'exception pour nouvel_EnregCVisu
141     // =1 lecture OK on est arrive a la fin de tous les fichiers, < 0 probleme de lecture
142     // = 0 probleme d'ouverture de fichier en lecture
143     // = 10 probleme en ouverture pour l'écriture
144     { public :
145         int lecture;
146         ErrNouvelleDonneeCVisu (int entrees) {lecture = entrees;} ; // CONSTRUCTEURS
147         ~ErrNouvelleDonneeCVisu () {};// DESTRUCTEUR :
148     };
149
150     // Flot d'entree
151     // pour la lecture en memoire centrale, permet la conversion de type
152     // dans le cas de lecture de tableau de character il faut tout
153     // d'abord preciser le nombre de caractere a lire avec la fonction
154     // membre width(nb de crarac)
155     // a utiliser pour la lecture d'entier, de reelle etc
156

```

```

157     #ifndef ENLINUX_STREAM
158         istringstream * entCVisu;
159     #else
160         istrstream * entCVisu;
161     #endif
162
163     // ----- fin du cas du fichier de commandes de visualisation -----
164
165     // CONSTRUCTEURS :
166     // ++ le constructeur officiel ++
167     // il y a passage éventuelle d'argument : argc=le nombre d'argument
168     // argv : donne un tableau correspondant de mots clés
169     // ---> lecture de la racine du nom du fichier princial
170     UtilLecture (int argc=0, const char * argv[] = NULL) ;
171     // DESTRUCTEUR :
172     ~UtilLecture () ;
173
174     // METHODES PUBLIQUES :
175     // ouverture du fichier de lecture. Les infos interessantes se trouvent
176     // dans la chaine pointee par tablcar :
177     //
178     // ou cela peut-être l'ouverture d'un nouveau fichier d'écriture de commande
179     // ceci en fonction des réponses de l'utilisateur lors de la création de l'instance
180     // UtilLecture. Dans ce cas le fichier .info n'est disponible qu'en écriture !!!
181     // aucune des fonctions de lecture sont activable :
182     //
183     // en retour idem = Lec_ent_info()
184     int OuvrirFichier () ;
185
186     // rafraichir le dernier fichier de lecture .info en cours
187     // il y a fermeture puis ouverture au même endroit du fichier .info
188     void Raffraichir_pointInfo();
189
190     // fermeture du fichier .info qui est en création donc en écriture
191     //(ce n'est pas celui qui serait en lecture !!),
192     void fermeture_pointInfo();
193     // fermeture du fichier SchemaXML
194     void fermetureSchemaXML();
195
196     // ouverture d'un fichier de copie des infos lue par le programme
197     // utilise pour verifier la bonne lecture
198     void OuvrirCopie();
199     // fermeture du fichier
200     void FermerCopie();
201
202     // redirection du flot d'entré vers un nouveau fichier
203     // nevezFich pointe sur le nom du nouveau fichier
204     void NouveauFichier( char* nevezFich) ;
205
206     // fermeture du flot actuel, et retour sur le flot précédent
207     // indic est un indicateur de retour
208     // = 0 il reste encore des fichiers intermédiaires ouverts
209     // =1 on se situe sur le flot primaire
210     void FermetureFichier(int& indic);
211
212     // lecture d'une nouvelle ligne de donnée utilisable pour les données
213     // lorsqu'apparait le signe < dans le fichier d'entrée il y a automatiquement
214     // ouverture d'un nouveau fichier
215     void NouvelleDonnee();
216
217     // idem mais sans ouverture de fichier automatique
218     // le signe < est conservé dans la ligne
219     void NouvelleDonneeSansInf();
220
221     // Retour du flot au debut de l'enregistrement
222     // a utiliser s'il y a eu une erreur de lecture sur le flot
223     // mais que l'on connait l'erreur et l'on veut relire
224     // par exemple : on essaie un string -> erreur, alors c'est un reel
225     // on veut relire un reel, on utilise FlotDebutDonnee() avant
226     void FlotDebutDonnee();
227
228     // gestion d'erreur d'entree
229     // ecriture d'un message d'erreur du a une mauvaise lecture
230     // le programme ecrit la chaine transmise puis le contenu des buffers
231     void ErreurLecture(UtilLecture::ErrNouvelleDonnee, char*);
232
233     // ecriture d'un message et du contenu des buffers courant
234     void MessageBuffer(string);
235
236     // affichage sur la sortie standard de l'enregistrement courant
237     void AfficheEnreg () const ;
238
239     // retourne la racine du nom du fichier principal
240     char* RacineNom() { return (char*)nomRacine.c_str() ;};
241
242     // indique de type d'entrée des données ou de sortie
243     // dans le cas ou c'est une création de fichier de commande

```

```

244 // = -11 indique que l'on cré un fichier de commande .info
245 // = -12 création du fichier schema XML
246 // = 0 indique que ce sera via le fichier info
247 // = 1 ce sera via un fichier base-info
248 // = 2 ce sera via un serveur base-info
249 int Lec_ent_info () {return lec_ent_info;};
250
251 // ----- cas de la base d'info -----
252 //fichier base_info dans lequel sont lus les enregistrements
253 // dans le cas ou celui-ci n'est pas ouvert il y a ouverture
254 ifstream * Ent_BI() ;
255 // et pour l'écriture
256 // dans le cas ou celui-ci n'est pas ouvert il y a ouverture
257 ofstream * Sort_BI() ;
258 // ouverture du fichier base_info après le .info dans le cas d'un restart par exemple
259 // dans le cas ou le fichier est déjà ouvert en lecture ou écriture dans le type demandé
260 // il n'y a aucune action
261 // a priori le fichier est ouvert en lecture : lecture
262 // mais il peut également être en écriture : ecriture
263 void Ouverture_base_info(string type_entree="lecture");
264 // fermeture du fichier base_info
265 void Fermeture_base_info();
266 // positionnement dans le fichier d'entrès au niveau d'un numéro d'incrément
267 // si l'on ne trouve pas le numéro on renvoi un indicateur à false
268 bool Positionnement_base_info(int inc_voulu);
269 // positionnement dans le fichier d'entrès au niveau du numéro d'incrément suivant
270 // si l'on ne trouve pas de numéro on renvoi un indicateur à false
271 // la méthode renvoie également la position du début de l'incrément
272 // et le numéro trouvé d'incrément
273 bool Increment_suivant_base_info(streampos& debut_increment,int& inc_lu);
274 // enregistrement de la position
275 // normalement correspond à un début d'incrément
276 void Enregistrement_position_increment_base_info(int incr);
277 // renvoi tous les incréments actuellement enregistré
278 list <int> Liste_increment();
279
280 // ----- cas des temps de calcul -----
281 // ouverture du fichier spécifique des temps cpu
282 void Ouverture_fichier_temps_cpu();
283 // fermeture du fichier des temps cpu
284 void Fermeture_fichier_temps_cpu();
285 // récupération du fichier des temps cpu
286 ofstream & Sort_temps_cpu() { return *sort_temps_cpu;};
287
288 // ----- cas de la visualisation -----
289 //      +++ vrml +++
290 // ouverture du fichier principal vrml
291 void Ouverture_fichier_principal_vrml();
292 // fermeture du fichier principal vrml
293 void Fermeture_fichier_principal_vrml();
294 // récupération du fichier principal vrml
295 ofstream & Sort_princ_vrml() { return *sort_princ_vrml;};
296 // ouverture du fichier de légendes vrml
297 void Ouverture_fichier_legende_vrml();
298 // fermeture du fichier légendes vrml
299 void Fermeture_fichier_legende_vrml();
300 // récupération du fichier légendes vrml
301 ofstream & Sort_legende_vrml() { return *sort_legende_vrml;};
302 // existence
303 bool Existe_princ_vrml() const {return (sort_princ_vrml==NULL ? false : true); };
304 //      +++ maple +++
305 // ouverture du fichier principal maple
306 void Ouverture_fichier_principal_maple();
307 // fermeture du fichier principal maple
308 void Fermeture_fichier_principal_maple();
309 // récupération du fichier principal maple
310 ofstream & Sort_princ_maple() { return *sort_princ_maple;};
311 // existence
312 bool Existe_princ_maple() const {return (sort_princ_maple==NULL ? false : true); };
313 //      +++ geomview +++
314 // ouverture du fichier principal geomview
315 void Ouverture_fichier_principal_geomview();
316 // fermeture du fichier principal geomview
317 void Fermeture_fichier_principal_geomview();
318 // récupération du fichier principal geomview
319 ofstream & Sort_princ_geomview() { return *sort_princ_geomview;};
320 // ouverture du fichier de légendes geomview
321 void Ouverture_fichier_legende_geomview();
322 // fermeture du fichier légendes geomview
323 void Fermeture_fichier_legende_geomview();
324 // récupération du fichier légendes geomview
325 ofstream & Sort_legende_geomview() { return *sort_legende_geomview;};
326 // existence
327 bool Existe_princ_geomview() const {return (sort_princ_geomview==NULL ? false : true); };
328 bool Existe_legende_geomview() const {return (sort_legende_geomview==NULL ? false : true); };
329 //      +++ Gid +++
330 // ouverture du fichier pour le maillage initial Gid

```

```

331 void Ouverture_fichier_initial_Gid();
332 // fermeture du fichier maillage initial Gid
333 void Fermeture_fichier_initial_Gid();
334 // récupération du fichier maillage initial Gid
335 ofstream & Sort_initial_Gid() { return *sort_initial_Gid;};
336 // ouverture du fichier résultat Gid
337 void Ouverture_fichier_resultat_Gid();
338 // fermeture du fichier résultat Gid
339 void Fermeture_fichier_resultat_Gid();
340 // récupération du fichier résultats Gid
341 ofstream & Sort_resultat_Gid() { return *sort_resultat_Gid;};
342 // existence
343 bool Existe_initial_Gid() const {return (sort_initial_Gid==NULL ? false : true); };
344 bool Existe_resultat_Gid() const {return (sort_resultat_Gid==NULL ? false : true); };
345
346 //      +++ Gmsh +++
347 // ouverture du fichier pour le maillage initial Gmsh
348 void Ouverture_fichier_initial_Gmsh();
349 // fermeture du fichier maillage initial Gmsh
350 void Fermeture_fichier_initial_Gmsh();
351 // récupération du fichier maillage initial Gmsh
352 ofstream & Sort_initial_Gmsh() { return *sort_initial_Gmsh;};
353 // création du répertoire contenant tous les fichiers résultats
354 void CreationRepertoireResultat();
355 // ouverture d'un fichier résultat Gmsh, comprenant un nom particulier
356 // si le fichier n'existe pas -> création
357 void Ouverture_fichier_resultat_Gmsh(const string& nom);
358 // ouverture d'un ensemble de fichiers: ces fichiers seront ensuite accessibles
359 // si un des fichiers n'existe pas -> création
360 void Ouverture_fichier_resultat_Gmsh(const list <string>& list_nom);
361 // récupération de la liste de noms de base qui servent pour la création des
362 // fichiers de base: cette liste = la somme des noms passés en paramètre via les
363 // méthode Ouverture_fichier_resultat_Gmsh et Ouverture_fichier_resultat_Gmsh
364 // chaque nom peut ensuite être utilisé dans les méthodes d'accès, ouverture, fermeture
365 const list <string>& Noms_base_fichiers_gmsh() const { return noms_base_fichiers_gmsh;};
366
367 // fermeture d'un fichier résultat Gmsh
368 void Fermeture_fichier_resultat_Gmsh(const string& nom);
369 // fermeture de tous les fichiers résultat
370 void Fermeture_TousLesFichiersResultats_Gmsh();
371 // récupération d'un fichier résultats Gmsh
372 ofstream & Sort_resultat_Gmsh(const string& nom);
373 // existence
374 bool Existe_initial_Gmsh() const {return (sort_initial_Gmsh==NULL ? false : true); };
375 // pour tester l'existence d'un fichier resultat on utilise la liste Noms_base_fichiers_gmsh()
376
377 // ----- automatization de la visualisation -----
378 // ouverture du fichier CommandeVisu s'il est déjà ouvert on ne fait rien
379 // a priori le fichier est ouvert en lecture : type_entree = "lecture"
380 // mais il peut également être en écriture : type_entree = "ecriture"
381 void Ouverture_CommandeVisu(string type_entree);
382 // fermeture du fichier CommandeVisu
383 void Fermeture_CommandeVisu();
384 // changement du nom de fichier .CVisu
385 // entraîne la fermeture du fichier en cours, en lecture et en écriture
386 // il n'y a pas de vérification à ce niveau d'existence éventuelle
387 // car on ne sait pas ce que l'on veut en faire
388 // si la chaîne de caractère passée en paramètre est non nulle on l'utilise
389 // sinon on passe en interactif et on demande le nom de la chaîne
390 void Changement_NomCVisu(string nouveauNomCvisu);
391 // fichier CommandeVisu dans lequel sont lus les enregistrements
392 // uniquement l'écriture est à accès directe
393 ofstream * Sort_CommandeVisu();
394 // lecture d'une nouvelle ligne de donnée utilisable pour l'entrée de donnée
395 // dans le fichier de commande de visualisation (idem NouvelleDonnee mais en plus simple
396 // car il n'y a pas de fichier inclus)
397 void NouvelleDonneeCVisu();
398 // recherche dans le fichier de commande la chaîne de caractère passée en paramètre
399 // retour false: si on ne trouve pas la chaîne
400 // retour true: si on la trouve, et dans ce cas le pointeur courant du fichier est
401 // positionné sur la ligne de la chaîne
402 bool PositionEtExisteCVisu(const string chaîne);
403
404 //----- utilitaire de lecture d'un paramètre précédé d'un mot clé, avec une restriction
405 // sur les valeurs
406 // val_defaut: si la grandeur n'est pas lue, elle est mise par défaut = val_defaut
407 // nom_class_methode: le nom de la méthode qui appelle l'utilitaire -> sortie dans le message si pb
408 // min max : les bornes entre lesquelles doit se trouver le paramètre lue sinon -> génération d'une
409 // erreur et arrêt
410 // si max < min alors la condition n'est pas prise en compte
411 // mot_cle : mot_cle devant précéder le paramètre
412 // parametre: le paramètre à lire
413 // retourne un boolean: indique si oui ou non il y a eu lecture
414 bool Lecture_un_parametre_int(int val_defaut,const string& nom_class_methode
415                               ,int min, int max, string& mot_cle, int& parametre ) const;
416 // idem pour un double
417 bool Lecture_un_parametre_double(double val_defaut,const string& nom_class_methode

```

```

417                                     ,double min, double max,const string& mot_cle, double&
418 parametre ) const;
419 // lecture et passage d'un mot clé: il y a simplement vérification que c'est le bon mot clé qui est
420 lue
421 // ramène si oui ou non le mot clé à été lue
422 bool Lecture_et_verif_mot_cle(const string& nom_class_methode,const string& mot_cle) const;
423 // idem pour un mot clé et un string; nom
424 bool Lecture_mot_cle_et_string(const string& nom_class_methode,const string& mot_cle,string& nom)
425 const;
426
427 //----- utilitaires de lecture d'une grandeurs qui peut-être une constante utilisateur ---
428 // int Lect_avec_const_int_utilisateur()const;
429 double lect_avec_const_double_utilisateur(string erreur) const;
430
431 // modification interactive d'une constante utilisateur
432 // la constante doit déjà exister
433 static void Modif_interactive_constante_utilisateur();
434
435 // ----- définition de conteneurs à usage interne -----
436 // cependant elles ne peuvent pas être déclaré en protégé ??
437 // protected :
438 //private :
439 // définition de type particulier
440 // def d'un maillon de liste chainee pour memoriser les differentes ouverture
441 // de fichier
442 class PointFich
443 { public :
444     PointFich* t1; // adresse du maillon precedent
445     string nom; // nom du fichier
446     ifstream * sauvePtr; // sauvegarde de pointeur de fichier
447     int numLigne; // numero de la ligne actuellement lue
448     PointFich ( ) : nom(0),numLigne(0),t1(NULL),sauvePtr(NULL) {}; // défaut, ne doit pas être
449 utilisé
450 // constructeur utile:
451 PointFich ( PointFich* x1,ifstream * x2,string n) : nom(n),numLigne(0),t1(x1),sauvePtr(x2)
452 {};
453 // constructeur de copie
454 PointFich (const PointFich& a) :
455 nom(a.nom),numLigne(a.numLigne),t1(a.t1),sauvePtr(a.sauvePtr) {};
456 ~PointFich () {};
457 // operateur =
458 PointFich& operator=(const PointFich & a)
459 {nom=a.nom;numLigne=a.numLigne;t1=a.t1;sauvePtr=a.sauvePtr;return *this;};
460 };
461 // gestion d'exception pour nouvel_Enreg
462 class ErrNouvelEnreg
463 // =0 lecture OK, sinon =1 -> fin de fichier, < 0 probleme de lecture
464 { public :
465     int lecture;
466     ErrNouvelEnreg (int entrees) {lecture = entrees;}; // CONSTRUCTEURS
467     ~ErrNouvelEnreg () {}; // DESTRUCTEUR :
468 };
469 // def d'une position d'incrément dans le fichier base_info
470 class Position_BI
471 {
472     // surcharge de l'operateur de lecture
473     friend istream & operator » (istream & entree, Position_BI& a);
474     // surcharge de l'operateur d'écriture
475     friend ostream & operator « (ostream & sort, const Position_BI& a);
476
477 public :
478     Position_BI(streampos posi=(ios::beg), int num = 0) : // constructeur
479     position (posi), num_incr(num)
480     {};
481     ~Position_BI() {}; // destructeur
482     Position_BI(const Position_BI& a) : // constructeur de copie
483     position (a.position),num_incr(a.num_incr)
484     {};
485     // surcharge des operateurs
486     bool operator == (const Position_BI& a) const
487     {if (a.num_incr == this->num_incr) return true; else return false;};
488     bool operator > (const Position_BI& a) const
489     { return (this->num_incr > a.num_incr);};
490     bool operator < (const Position_BI& a) const
491     { return (this->num_incr < a.num_incr);};
492
493     Position_BI& operator = (const Position_BI& a)
494     {position = a.position; num_incr = a.num_incr;return *this;};
495     bool operator != (const Position_BI& a)
496     { return !(*this == a);};
497     // données
498     streampos position; // position dans le fichier
499     int num_incr; // numéro de l'incrément
500 };
501 // gestion d'exception pour nouvel_EnregCVisu
502 class ErrNouvelEnregCVisu

```

```

497 // =0 lecture OK, sinon =1 -> fin de fichier, < 0 probleme de lecture
498 { public :
499     int lecture;
500     ErrNouvelEnregCVisu (int entrees) {lecture = entrees;} ; // CONSTRUCTEURS
501     ~ErrNouvelEnregCVisu () {};// DESTRUCTEUR :
502 };
503
504 // VARIABLES PROTEGEES :
505 private :
506
507     char* pteurnom; // pointeur sur la racine du nom du fichier principal
508     string nomRacine; // racine du nom du fichier principal
509     ifstream * ptrfich ; //fichier dans lequel sont lus les enregistrements
510     ofstream * ptrcommandefich;// fichier dans lequel on écrit les commandes .info
511     ofstream * ptrschemaXMLfich;// fichier dans lequel on écrit le schema XML
512     char* tableau ; // le tableau de travail intermediaire
513     char* tabMultiLignes; // stockage intermédiaire de plusieurs lignes
514     string nomCourant; // nom du fichier courant ouvert en lecture
515     int lec_ent_info; // type d'entrée des données
516     // = 0 indique que ce sera via le fichier info
517     // = 1 ce sera via un fichier base-info
518     // = 2 ce sera via un serveur base-info
519     // = -11 création d'un fichier de commande
520     // = -12 création du schema XML
521     fstream * ent_BI ; //fichier base_info dans lequel sont lus les enregistrements
522     fstream * sort_BI ; //fichier base_info de sauvegarde
523     fstream * ent_PI ; //fichier de la liste des pointeurs d'incréments de base info
524     fstream * sort_PI ; //fichier base_info de sauvegarde
525     // -----cas des temps de calcul -----
526     ofstream * sort_temps_cpu;
527     // ----- pour la visualisation -----
528     ofstream * sort_princ_vrml; // fichier principal vrml
529     ofstream * sort_legende_vrml; // fichier de legendes vrml
530     ofstream * sort_princ_maple; // fichier principal maple
531     ofstream * sort_princ_geomview; // fichier principal geomview
532     ofstream * sort_legende_geomview; // fichier de legendes geomview
533     ofstream * sort_initial_Gid; // fichier maillage initial pour Gid
534     ofstream * sort_resultat_Gid; // fichier des résultats pour Gid
535     ofstream * sort_initial_Gmsh; // fichier maillage initial pour Gmsh
536     map < string, ofstream * , std::less <string> > sort_resultat_Gmsh;
537     list <string> noms_base_fichiers_gmsh; // la liste des clés dans la map
538     // on double l'info, mais cela permet de récupérer la liste indépendamment de la map
539     // ----- pour l'automatisation de l'interactif -----
540     fstream * sort_CommandeVisu; // écriture des parametres de visualisation
541     fstream * ent_CommandeVisu; // lecture des parametres de visualisation
542     char* tableauCVisu ; // le tableau de travail intermediaire
543     char* ptnomCVisu; // le pointeur de nom pour le CVisu
544     string nomRacineCVisu; // racine du nom pour le CVisu
545
546     int dernier_increment_lu; // variable de travail pour Increment_suivant_base_info
547
548     // liste des positions déjà investiguées dans le fichier .base_info
549     list <Position_BI> liste_posi_BI;
550     // maillon courant des fichiers ouverts via .info
551     PointFich* maill1;
552
553
554 // METHODES PROTEGEES :
555
556     void Nouvel_enreg () ; // lecture d'une nouvelle enregistrement
557         // dans le fichier courant
558     void Nouvel_enregCvisu () ; // lecture d'une nouvelle enregistrement
559         // dans le fichier de commande de visualisation
560     // ouverture du fichier CommandeVisu dans lequel sont lus les enregistrements
561     ifstream * Ent_CommandeVisu();
562     // cherche l'apparition d'une chaine de caractere -> pas bon a priori
563     void Cherche(const char *);
564     // entrée par argument au niveau de la construction de l'objet UtilLecture
565     // dans le cas où il y a eu un pb lec_ent_info=-100, et rien n'est initialisé
566     bool EntreeParArgument(int argc, const char * argv[]);
567     // entrée interactive au niveau de la construction de l'objet UtilLecture
568     // dans le cas où il y a eu un pb lec_ent_info=-100, et rien n'est initialisé
569     void EntreeInteractive(string& ancien,bool& ouverture_ancien);
570
571
572 };
573 /// @} // end of group
574
575 #endif

```

## 7.340 UtilXML.h

```

1
2 // This file is part of the Herezh++ application.
3 //

```



```

4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *
32 *   DATE:          21/12/2005
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 * *****/
39 *   BUT:           definir l'ensemble des outils relatifs aux
40 *                 operations I/O pour le format XML
41 *                 d'entree
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   -----
49 *   !       !           !
50 *
51 *   *****
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *
58 *
59 *****/
60
61 #ifndef UTILXML_H
62 #define UTILXML_H
63
64 #include <fstream>
65 // #include "Debug.h"
66
67 #ifndef ENLINUX_STREAM
68 #include <sstream> // pour le flot en memoire centrale
69 #else
70 #include <strstream> // pour le flot en memoire centrale
71 #endif
72 #include <string.h>
73 #include <string>
74 #include <list>
75 #include "Sortie.h"
76     using namespace std;
77
78 /// @addtogroup Groupe_relatif_aux_entrees_sorties
79 /// @{
80 ///
81
82
83 class UtilXML
84 {
85     public :
86         // CONSTRUCTEURS :
87         // constructeur par défaut
88         UtilXML () ;
89         // DESTRUCTEUR :

```

```

90     ~UtilXML () ;
91
92     // METHODES PUBLIQUES :
93
94     // VARIABLES PROTEGEES :
95     private :
96
97
98
99     // METHODES PROTEGEES :
100
101
102 };
103 /// @} // end of group
104
105 #endif

```

## 7.341 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Maillage/BlocDdlLim.h

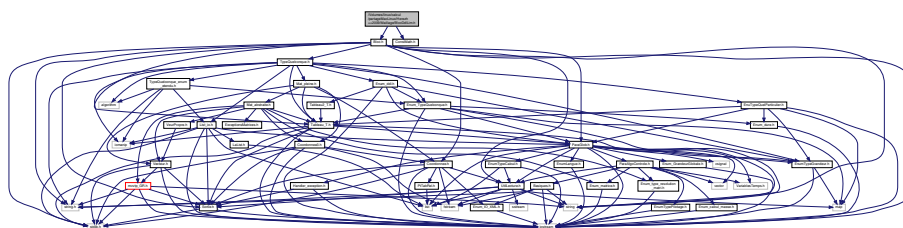
def classe et fonctions template pour la lecture de ddl lim.

```

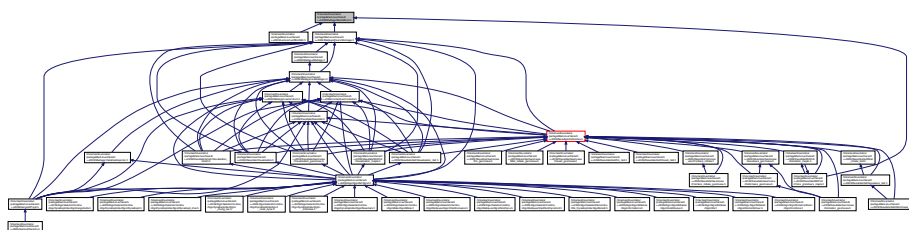
#include "Bloc.h"
#include "ConstMath.h"

```

Graphe des dépendances par inclusion de BlocDdlLim.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Classes

— class [BlocDdlLim](#) < [Bloc\\_particulier](#) >

#### 7.341.1 Description détaillée

def classe et fonctions template pour la lecture de ddl lim.

#### 7.342 BlocDdlLim.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file BlocDdlLim.h
2    \brief def classe et fonctions template pour la lecture de ddl lim.
3 */
4
5 // This file is part of the Herezh++ application.
6 //

```

```

7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32 //
33 /*****
34 *   DATE:      28/03/2004
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   $
41 *   *****/
42 *   BUT:   Gestion de divers bloc courant du fichier de lecture .
43 *          Cas où ces blocs sont relatif aux conditions limites,
44 *          et héritent de bloc classiques.
45 *          par rapport au bloc, l'apport est la gestion d'un nom_de maillage
46 *          éventuellement en plus du nom de référence.
47 *
48 *   $
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !           !
55 *   $
56 *   *****/
57 *   MODIFICATIONS:
58 *
59 *   ! date !   auteur !           but
60 *   -----
61 *   !           !           !           !
62 *   $
63 *   *****/
64 #ifndef BLOC_DDLIM_T_H
65 #define BLOC_DDLIM_T_H
66
67 #include "Bloc.h"
68 #include "ConstMath.h"
69
70 /// @addtogroup Les_classes_Ddl_en_tout_genre
71 /// @{
72 ///
73 ///=====
74 /// cas d'un bloc template avec conditions limites
75 ///=====
76 template <class Bloc_particulier>
77 class BlocDdlLim : public Bloc_particulier
78 {
79     // surcharge de l'operator de lecture
80     friend istream & operator » (istream & entree, BlocDdlLim<Bloc_particulier> & coo)
81     { // lecture d'un nom de maillage éventuel
82         bool existe_nom_maillage; entree » existe_nom_maillage;
83         if (existe_nom_maillage)
84             { string nom; entree » nom;
85               if (coo.nom_maillage == NULL)
86                 coo.nom_maillage = new string(nom);
87               else
88                 *(coo.nom_maillage) = nom;
89             };
90         // puis la classe mère
91         entree » ((Bloc_particulier&)(coo));
92         return entree;
93     }

```

```

94
95 // surcharge de l'operator d'écriture
96 friend ostream & operator << (ostream & sort, const BlocDdlLim<Bloc_particulier> & coo)
97 { // tout d'abord le maillage éventuelle
98   if (coo.nom_maillage == NULL)
99     sort << false << " ";
100   else
101     sort << true << " " << *(coo.nom_maillage) << " ";
102   // puis la classe mère
103   sort << ((const Bloc_particulier&)(coo));
104   return sort;
105 }
106
107 public :
108 // VARIABLES PUBLIQUES :
109 // stockage d'un element
110 // class conforme a la specif de T de la class LectBloc_T
111
112 // Constructeur
113 BlocDdlLim () : // par défaut
114   nom_maillage(NULL),Bloc_particulier()
115   {};
116 // fonction d'une instance de Bloc_particulier
117 BlocDdlLim (const Bloc_particulier& a) : Bloc_particulier(a),nom_maillage(NULL) {};
118 // fonction d'une instance et d'un string
119 BlocDdlLim (const string& nom_mail,const Bloc_particulier& a) :
120   Bloc_particulier(a),nom_maillage(new string (nom_mail)) {};
121 // fonction d'une instance et d'un pointeur de string
122 BlocDdlLim (const string* nom_mail,const Bloc_particulier& a) :
123   Bloc_particulier(a),nom_maillage(NULL)
124   { if (nom_mail != NULL) nom_maillage = new string (*nom_mail);};
125 // de copie
126 BlocDdlLim (const BlocDdlLim& a) :
127   nom_maillage(NULL),Bloc_particulier(a)
128   {if (a.nom_maillage != NULL) nom_maillage = new string (*(a.nom_maillage));};
129 // destructeur
130 ~BlocDdlLim () {if (nom_maillage != NULL) delete nom_maillage;};
131 // retourne un pointeur de string donnant le nom du maillage associé
132 // = NULL si aucun maillage
133 const string* NomMaillage() const {return nom_maillage;};
134 // change le nom de maillage
135 // ATTENTION : Les modifications liees au changement du nom de maillage
136 // sont a la charge de l'utilisateur.
137 // le nouveau nom ne doit pas être vide !! sinon erreur
138 void Change_nom_maillage(const string& nouveau);
139 // lecture d'un bloc
140 void Lecture(UtilLecture & entreePrinc);
141 // affichage des informations
142 void Affiche() const ;
143 // surcharge des operateurs
144 bool operator == ( const BlocDdlLim& a) const;
145 BlocDdlLim& operator = (const BlocDdlLim& a);
146 bool operator != ( const BlocDdlLim& a) const { return !(*this == a);};
147
148 protected :
149   string* nom_maillage; // nom d'un maillage
150 };
151 /// @} // end of group
152
153
154 /// lecture d'un bloc
155 template <class Bloc_particulier>
156 void BlocDdlLim<Bloc_particulier>::Lecture(UtilLecture & entreePrinc)
157 { // on regarde tout d'abord si il y a un nom de maillage, qui doit être le premier nom
158   string nom;
159   if (strstr(entreePrinc.tablcar,"nom_mail=")!=NULL)
160     { // cas où il y a un nom de maillage
161       *(entreePrinc.entree) >> nom >> nom; // lecture du mot_clé et du nom
162       if (nom_maillage == NULL) { nom_maillage = new string(nom);}
163       else { *nom_maillage = nom;};
164     }
165   // puis lecture de la classe mère
166   Bloc_particulier::Lecture(entreePrinc);
167 }
168
169 /// affichage des infos
170 template <class Bloc_particulier>
171 void BlocDdlLim<Bloc_particulier>::Affiche() const
172 { // affichage éventuelle du nom de maillage
173   if (nom_maillage != NULL)
174     { cout << "\n nom_mail= " << *nom_maillage;} else cout << "\n";
175   // puis affichage des infos de la classe mère
176   Bloc_particulier::Affiche();
177 }
178
179 /// surcharge de l'operateur ==
180 template <class Bloc_particulier>

```

```

181 bool BlocDdlLim<Bloc_particulier>::operator == ( const BlocDdlLim<Bloc_particulier>& a) const
182 { if (nom_maillage == NULL)
183   { if (a.nom_maillage != NULL) return false;
184   }
185   else
186   { if (a.nom_maillage == NULL) return false;
187     // sinon les deux sont non null: on test leur égalité
188     if (*nom_maillage != *(a.nom_maillage)) return false;
189   };
190   // puis la classe mère
191   if ((Bloc_particulier&)(*this)).operator==(a)
192     return true;
193   else
194   { return false; };
195 }
196
197 /// surcharge de l'operateur =
198 template <class Bloc_particulier>
199 BlocDdlLim<Bloc_particulier>& BlocDdlLim<Bloc_particulier>::operator
200 = ( const BlocDdlLim<Bloc_particulier>& a)
201 { if (a.nom_maillage == NULL)
202   { if (nom_maillage != NULL) {delete nom_maillage;nom_maillage=NULL;}}
203   else
204   { if (nom_maillage == NULL) nom_maillage = new string(*(a.nom_maillage));
205     else
206     *nom_maillage = *(a.nom_maillage);
207   };
208   // puis la classe mère
209   ((Bloc_particulier&)(*this)).operator=(a);
210   return *this;
211 }
212
213 /// change le nom de maillage
214 /// ATTENTION : Les modifications liees au changement du nom de maillage
215 /// sont a la charge de l'utilisateur.
216 /// le nouveau nom ne doit pas être vide !! sinon erreur
217 template <class Bloc_particulier>
218 void BlocDdlLim<Bloc_particulier>::Change_nom_maillage(const string& nouveau)
219 { if (nouveau.length() == 0)
220   { cout << "\nErreur : reference de nom de maillage non valide car de longueur nulle ! "
221     << " nom_mail= " << nouveau << "\n";
222     cout << "BlocDdlLim<Bloc_particulier>::Change_nom_maillage(...) \n";
223     Sortie(1);
224   }
225   if (nom_maillage == NULL)
226   { nom_maillage = new string(nouveau);}
227   else
228   { *nom_maillage = nouveau;};
229 }
230 #endif

```

## 7.343 Ddl.h

```

1 // FICHER : Ddl.h
2 // CLASSE : Ddl
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****

```

```

33 *      DATE:          23/01/97
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *****
40 * La classe Ddl permet de declarer des degres de liberte.
41 * soit c'est une variable ou soit c'est une données  qui peuvent être soit active
42 * soit hors-service
43 * ensuite soit il est fixe ou soit il est bloqué
44 * 1) libre, fixe,  ou 2) hors service libre ou hors service fixe ==> une variable
45 * 3) lisible libre ou fixe ou 4) hors service libre ou fixe ==> une donnée
46 * 1) c'est pour les ddl qui sont des variables que le calcul va déterminer,
47 * 2) sont les variables qui sont hors service: on ne les considères pas
48 * 3) sont des variables que l'on peut utiliser mais que le calcul ne
49 * cherche pas à déterminer: elle sont en lecture seule en quelque sorte
50 * 4) se sont des données qui ne sont pas dispon pour la consultation (par exemple hors temps)
51 *
52 *      *****
53 *
54 *      VERIFICATION:
55 *
56 *      ! date !   auteur   !           but
57 *      -----
58 *      !       !           !
59 *
60 *      *****
61 *      MODIFICATIONS:
62 *      ! date !   auteur   !           but
63 *      -----
64 *
65 *****/
66
67 // La classe Ddl permet de declarer des degres de liberte.
68 // soit c'est une variable ou soit c'est une données  qui peuvent être soit active
69 // soit hors-service
70 // ensuite soit il est fixe ou soit il est bloqué
71 // 1) libre, fixe,  ou 2) hors service libre ou hors service fixe ==> une variable
72 // 3) lisible libre ou fixe ou 4) hors service libre ou fixe ==> une donnée
73 // 1) c'est pour les ddl qui sont des variables que le calcul va déterminer,
74 // 2) sont les variables qui sont hors service: on ne les considères pas
75 // 3) sont des variables que l'on peut utiliser mais que le calcul ne
76 // cherche pas à déterminer: elle sont en lecture seule en quelque sorte
77 // 4) se sont des données qui ne sont pas dispon pour la consultation (par exemple hors temps)
78
79
80 #ifndef DDL_H
81 #define DDL_H
82
83
84 #include "Enum_ddl.h"
85 #include "Enum_boolddl.h"
86
87
88 #include <iostream>
89 #include <stdlib.h>
90 #include "Sortie.h"
91 #include <string>
92 // #include <bool.h>
93
94 /** @defgroup Les_classes_Ddl_en_tout_genre
95 *
96 *      BUT: def de classes relatives aux ddl en tout genre
97 *      Ddl, Ddl_etendu, DdlElement, DdlLim etc.
98 *
99 * \author Gérard Rio
100 * \version 1.0
101 * \date 23/01/97
102 * \brief Def de classes relatives aux ddl en tout genre
103 *
104 */
105
106 /// @addtogroup Les_classes_Ddl_en_tout_genre
107 /// @{
108 ///
109
110
111 ///      BUT: La classe Ddl permet de declarer des degres de liberte.
112 /// soit c'est une variable ou soit c'est une données  qui peuvent être soit active
113 /// soit hors-service
114 /// ensuite soit il est fixe ou soit il est bloqué
115 /// 1) libre, fixe,  ou 2) hors service libre ou hors service fixe ==> une variable
116 /// 3) lisible libre ou fixe ou 4) hors service libre ou fixe ==> une donnée
117 /// 1) c'est pour les ddl qui sont des variables que le calcul va déterminer,
118 /// 2) sont les variables qui sont hors service: on ne les considères pas
119 /// 3) sont des variables que l'on peut utiliser mais que le calcul ne

```

```
120 ///  
121 ///  
122 ///  
123 ///  
124 ///  
125 ///  
126 ///  
127 ///  
128 ///  
129 ///  
130 class Ddl  
131 {  
132     // surcharge de l'operator de lecture  
133     friend istream & operator » (istream & entree, Ddl& a);  
134     // surcharge de l'operator d'écriture  
135     friend ostream & operator « (ostream & sort, const Ddl& a);  
136  
137     public:  
138  
139  
140     // CONSTRUCTEURS :  
141  
142     // Constructeur eventuellement par défaut  
143     Ddl (Enum_ddl id_nom_ddl=NU_DDL,double val=0.0,Enum_boolddl val_fixe=HS_LISIBLE_LIBRE);  
144  
145     // Constructeur se servant d'un nom  
146     // ( N.B. : val et val_fixe ont des valeurs par défaut, il n'est par consequent  
147     // pas obligatoire de leur donner des valeurs a l'appel de ce constructeur )  
148     Ddl (char* nom,double val=0.0,Enum_boolddl val_fixe=HS_LISIBLE_LIBRE);  
149  
150  
151     // Constructeur de copie  
152     Ddl (const Ddl& d);  
153  
154  
155     // DESTRUCTEUR :  
156  
157     ~Ddl ();  
158  
159  
160     // METHODES :  
161  
162     // Retourne 1 si le degre de liberte a une valeur fixée tout au long du calcul  
163     // Retourne 0 sinon  
164     // ne concerne que les variables (par exemple ne concerne pas les données)  
165     bool Fixe() const;  
166  
167     // test pour savoir si le ddl est une variable ou une donnée  
168     bool UneVariable() const;  
169  
170     // changement du statut de variable à donnée  
171     void ChangeVariable_a_Donnee();  
172  
173     // changement du statut de donnée à variable  
174     void ChangeDonnee_a_Variable();  
175  
176     // Retourne la valeur de fixe en lecture uniquement  
177     inline const Enum_boolddl Retour_Fixe() const { return fixe ; };  
178  
179     // Modifie la valeur de fixe, si val est vrai -> blocage , sinon libre  
180     void Change_fixe(bool val);  
181  
182     // modifie toute les conditions sauf la valeur: c-a-d:  
183     // le fixage, le fait que c'est une variable ou une donnée, le fait  
184     // d'être en service ou non etc..  
185     void CopieToutesLesConditions( Enum_boolddl en) {fixe = en;};  
186  
187     // test pour savoir si le ddl est en service ou pas  
188     inline bool Service() const  
189     { return (!(Est_HS(fixe)));};  
190  
191     // test si le ddl est SOUSLIBRE ou SURFIXE, ramène true si oui  
192     bool SousSurFixe() const;  
193  
194     // remise à normal si le ddl est en souslibre ou surfixe  
195     // s'il est en souslibre il passe en libre et s'il est en surfixe il passe en fixe  
196     // sinon on ne fait rien  
197     void Retour_normal_sur_ou_sous_fixe();  
198  
199     // met hors_service le ddl ou la donnée, cela signifie que momentanément il ne sert plus  
200     void Met_hors_service();  
201     // met en service le ddl ou la donnée  
202     void Met_en_service();  
203  
204     // met hors_service le ddl, cela signifie que momentanément il ne sert plus  
205     // ne concerne pas les données  
206     void Met_hors_service_ddl();
```

```

207 // met en service le ddl, ne concerne pas les données
208 void Met_en_service_ddl();
209
210 // Retourne la valeur associee au degre de liberte
211 inline double& Valeur() { return val_ddl; };
212
213 // Retourne le nom associe au degre de liberte
214 inline const string Nom() const { return Nom_ddl(id_nom); };
215
216 // Retourne la variable de type enumere associee au nom du degre de liberte
217 inline Enum_ddl Id_nom() const { return id_nom; };
218
219 // Remplace le nom du degre de liberte par nouveau_nom
220 // (par l'intermediaire du type enumere associe)
221 // ATTENTION : Les modifications liees au changement de nom du degre
222 // de liberte sont a la charge de l'utilisateur.
223 inline void Change_nom(char* nouveau_nom)
224 { id_nom=Id_nom_ddl(nouveau_nom); };
225
226 // Remplace l'identificateur de nom du degre de liberte par
227 // nouveau_id
228 // ATTENTION : Les modifications liees au changement de nom du degre
229 // de liberte sont a la charge de l'utilisateur.
230 inline void Change_nom(Enum_ddl nouveau_id)
231 { id_nom=nouveau_id; };
232
233 // Affiche les donnees liees au degre de liberte
234 void Affiche() const ;
235
236 inline Ddl& operator= (const Ddl& d)
237 // Realise l'egalite entre deux degres de liberte
238 { id_nom=d.id_nom;
239   val_ddl=d.val_ddl;
240   fixe=d.fixe;
241   return (*this);
242 };
243
244 //Surcharge d'operateur logique
245 bool operator == ( const Ddl& a) const ;
246
247 bool operator != ( const Ddl& a) const;
248
249
250 // test si le degre de liberte est complet
251 // = 1 tout est ok, =0 element incomplet
252 int TestCompleet() const;
253
254
255 protected :
256
257   Enum_ddl id_nom; // identificateur du nom
258
259   // bool fixe; // booleen pour savoir si la valeur associee au
260 // degre de liberte est fixe pendant tout le calcul
261   Enum_booldddl fixe; // 1) libre, fixe, ou 2) hors service libre ou hors service fixe
262 // ou 3) lisible libre ou fixe
263 // cf. l'énuméré
264
265   double val_ddl; // valeur associee
266
267 };
268
269 /// @} // end of group
270
271 // pour faire de l'inline
272 #ifndef MISE_AU_POINT
273 #include "Ddl.cc"
274 #define Ddl_deja_inclus
275 #endif
276
277
278
279 #endif

```

## 7.344 Ddl\_etendu.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.

```



```

10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29 //
30 /*****
31 *   DATE:      19/01/2006
32 *
33 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:      class de stockage d'un ddl étendue, associé au type
39 *             Ddl_enum_etendu. En fait il s'agit du pendant des types
40 *             Ddl associé au type enum_ddl
41 *   Noter que la grandeur stockée est un scalaire !!
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   -----
49 *   !       !           !
50 *   *****
51 *   MODIFICATIONS:
52 *
53 *   ! date !   auteur !           but
54 *   -----
55 *   $
56 *****/
57 #ifndef DDL_ETENDU_H
58 #define DDL_ETENDU_H
59
60
61 #include "Ddl_enum_etendu.h"
62 #include "Enum_boolddl.h"
63
64
65 #include <iostream>
66 #include <stdlib.h>
67 #include "Sortie.h"
68 #include <string>
69
70 /// @addtogroup Les_classes_Ddl_en_tout_genre
71 /// @{
72 ///
73
74
75 ///   BUT:      class de stockage d'un ddl étendue, associé au type
76 ///             Ddl_enum_etendu. En fait il s'agit du pendant des types
77 ///             Ddl associé au type enum_ddl
78 ///   Noter que la grandeur stockée est un scalaire !!
79 ///
80 /// \author   Gérard Rio
81 /// \version  1.0
82 /// \date    19/01/2006
83 ///
84 ///
85
86 class Ddl_etendu
87 {
88     // surcharge de l'operator de lecture
89     friend istream & operator » (istream & entree, Ddl_etendu& a);
90     // surcharge de l'operator d'écriture
91     friend ostream & operator « (ostream & sort, const Ddl_etendu& a);
92
93     public:
94
95

```

```

96     // CONSTRUCTEURS :
97
98     // Constructeur par défaut
99     Ddl_etendu ();
100
101     // Constructeur fonction des variables, la seconde est facultative
102     Ddl_etendu (const Ddl_enum_etendu& ddl,double vale=0.0) ;
103
104     // Constructeur de copie
105     Ddl_etendu (const Ddl_etendu& d);
106
107
108     // DESTRUCTEUR :
109
110     ~Ddl_etendu ();
111
112
113     // METHODES :
114
115     // Retourne la valeur associee au degre de liberte
116     // en lecture écriture
117     inline double& Valeur() { return val_ddl; };
118     // idem en lecture seulement
119     inline const double& ConstValeur() const { return val_ddl; };
120
121     // Retourne le Ddl_enum_etendu associé
122     // en lecture écriture
123     inline Ddl_enum_etendu& DdlEnumEtendu() { return ddl_enum_etendu; };
124     // idem mais en constant
125     inline const Ddl_enum_etendu& Const_DdlEnumEtendu() const { return ddl_enum_etendu; };
126
127     // Affiche les donnees liees au degre de liberte
128 void Affiche() const { cout << *this; };
129 void Affiche(ostream& sort) const { sort << *this; };
130 void Affiche(ostream& sort,int nb) const { sort << setprecision(nb) << *this; };
131
132     inline Ddl_etendu& operator= (const Ddl_etendu& d)
133     // Realise l'egalite entre deux degres de liberte étendus
134     { ddl_enum_etendu=d.ddl_enum_etendu;
135       val_ddl=d.val_ddl;
136       return (*this);
137     };
138     // Surcharge de l'operateur +=
139     inline void operator+= (const Ddl_etendu& c)
140     {if (c.ddl_enum_etendu == ddl_enum_etendu)
141       {val_ddl += c.val_ddl;}
142     else
143     {cout << "\n *** erreur d'affectation void operator+= (const Ddl_enum_etendu& c)";
144       cout << "\n this: "; this->Affiche();
145       cout << "\n c: "; c.Affiche();
146       Sortie(1);
147     };
148     };
149     // Surcharge de l'operateur -=
150     inline void operator-= (const Ddl_etendu& c)
151     {if (c.ddl_enum_etendu == ddl_enum_etendu)
152       {val_ddl -= c.val_ddl;}
153     else
154     {cout << "\n *** erreur d'affectation void operator-= (const Ddl_enum_etendu& c)";
155       cout << "\n this: "; this->Affiche();
156       cout << "\n c: "; c.Affiche();
157       Sortie(1);
158     };
159     };
160
161     // Surcharge de l'operateur *=
162     inline void operator*= (double val) {val_ddl *= val;};
163     // Surcharge de l'operateur /= : division par un scalaire
164     inline void operator/= (double val) {val_ddl /= val;};
165
166     //Surcharge d'operateur logique
167     // ne concerne que le type et non la valeur
168     bool operator == ( Ddl_etendu& a) const
169     {return (ddl_enum_etendu == a.ddl_enum_etendu);};
170
171     bool operator != ( Ddl_etendu& a) const
172     {return (ddl_enum_etendu != a.ddl_enum_etendu);};
173
174     // test du type et de la valeur
175     bool Identique(Ddl_etendu& a)
176     { return ((*this == a)&&(val_ddl == a.val_ddl));};
177
178
179     protected :
180
181     Ddl_enum_etendu ddl_enum_etendu; // identificateur du nom
182     double val_ddl; // valeur associee

```

```

183
184
185 };
186 /// @} // end of group
187
188 // pour faire de l'inline
189 #ifndef MISE_AU_POINT
190     #include "Ddl_etendu.cc"
191     #define Ddl_etendu_deja_inclus
192 #endif
193
194
195
196 #endif

```

## 7.345 DdlElement.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:  Definition, gestion et stockage des ddl de l'element.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date ! auteur ! but
44 *      -----
45 *      ! ! !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date ! auteur ! but
50 *      -----
51 *
52 *****/
53 #ifndef DDLELEMENT_H
54 #define DDLELEMENT_H
55
56 #include "Tableau_T.h"
57 #include "DdlNoeudElement.h"
58 // #ifdef SYSTEM_MAC_OS_X
59 // #include <stringfwd.h> // a priori ce n'est pas portable
60 // #endif
61 #if defined SYSTEM_MAC_OS_CARBON
62 #include <stringfwd.h> // a priori ce n'est pas portable
63 #else
64 #include <string.h> // pour le flot en memoire centrale
65 #endif
66 #include <string>
67 #include <map>

```

```

68
69 /// @addtogroup Les_classes_Ddl_en_tout_genre
70 /// @{
71 ///
72
73
74 ///     BUT:     Definition, gestion et stockage des ddl de l'element.
75 ///
76 /// \author     Gérard Rio
77 /// \version   1.0
78 /// \date      23/01/97
79 ///
80 ///
81
82
83 class DdlElement
84 { // surcharge de l'operator de lecture
85     friend istream & operator » (istream & entree, DdlElement& a);
86     // surcharge de l'operator d'écriture
87     friend ostream & operator « (ostream & sort, const DdlElement& a);
88
89     public :
90
91         // CONSTRUCTEURS :
92         DdlElement (); // par défaut
93         // definition du tableau pour n noeuds vide
94         DdlElement (int n);
95         // definition du tableau pour n noeuds et m ddl par noeud
96         // utile lorsque tous les noeuds ont le meme nombre de ddl
97         // par contre ici aucun ddl n'est encore défini
98         DdlElement (int n, int m);
99         // definition du tableau pour n noeuds et un nombre de ddl par noeud
100        // stocke dans le tableau mddl qui doit avoir la dimension n
101        // par contre ici aucun ddl n'est encore défini
102        DdlElement (int n, const Tableau <int>& mddl);
103        // definition du tableau pour n noeuds et une meme liste de ddl pour chaque
104        // noeud identique au second argument
105        DdlElement (int n, DdlNoeudElement d);
106        // de copie
107        DdlElement (const DdlElement& a);
108        // DESTRUCTEUR :
109        ~DdlElement ();
110
111        // retourne nombre de noeud
112        int NbNoeud() const { return te.Taille();};
113        // acces en modification au ddl j du noeud i
114        void Change_Enum(int i, int j, const Enum_ddl enu);
115        // acces en lecture seulement au ddl j du noeud i
116        Enum_ddl operator() (const int i, const int j) const { return (te(i)).tb(j);};
117        // acces en lecture seulement au tableau de ddl du noeud i
118        const DdlNoeudElement& operator() (int i) const { return (te(i));};
119        // nb de ddl total contenu
120        int NbDdl() const { return nbddl;};
121        // nb de ddl du noeudElement i contenu
122        int NbDdl(int i) const ;
123        // retour du nombre de ddl d'une famille donnée
124        int NbDdl_famille(Enum_ddl enu) const;
125        // change la dimension du tableau de ddl pour n noeuds et m ddl par noeud
126        // dans le cas ou tous les noeuds ont le meme nombre de ddl
127        // si la nouvelle taille est plus petite: suppression
128        // si la nouvelle dimension est plus grande on met les ddl supplémentaires à NU_DDL
129        void Change_taille(int n, int m);
130        // change la dimension du tableau de ddl pour n noeuds et un nombre de ddl par noeud
131        // stocke dans le tableau mddl qui doit avoir la dimension n
132        // si la nouvelle taille est plus petite: suppression
133        // si la nouvelle dimension est plus grande on met les ddl supplémentaires à NU_DDL
134        void Change_taille(int n, Tableau <int>& mddl);
135        // change un des ddlNoeudElements
136        void Change_un_ddlNoeudElement(int i, const DdlNoeudElement& add);
137        // mais le tableau de ddl a zero
138        void TailleZero();
139        // surcharge de l'opérateur d'affectation
140        DdlElement& operator = (const DdlElement & a);
141        // surcharge des operator de test
142        bool operator == (const DdlElement & a) const ;
143        bool operator != (const DdlElement & a) const ;
144
145
146        // VARIABLES et méthodes protegees :
147    protected :
148        Tableau <DdlNoeudElement> te;
149        int nbddl;
150
151        // on définit un tableau associatif pour contenir le nombre de ddl de chaque famille
152        // on redéfinit une classe de int dont les éléments sont mis à 0 au début
153        class Int_initer {public: int vali; Int_initer(): vali(0) {};}
154        Int_initer(const Int_initer& a): vali(a.vali) {};}

```

```

155             Int_initer& operator =(const Int_initer& a){vali=a.vali;return *this;};
156             void operator ++(int){vali++;};
157             void operator --(int){vali--;};};
158     map < Enum_ddl, Int_initer , std::less <Enum_ddl> > tab_enum_famille;
159
160     // calcul du nb de ddl total contenu
161     void Calcul_NbDdl() ;
162     // calcul du nb de ddl de chaque type contenu
163     void Calcul_NbDdl_typer() ;
164
165
166 };
167 /// @} // end of group
168
169 #endif
170
171

```

## 7.346 DdlLim.h

```

1 // FICHER : DdlLim.h
2 // CLASSES : DdlLim
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      15/04/1997
34 *
35 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:     Herezh++
38 *
39 *****/
40 *      BUT: La classe DdlLim permet de declarer un ensemble de ddl coiffe *
41 *          par une reference utilise pour les conditions limites. $ *
42 *          ***** *
43 *
44 *      VERIFICATION:
45 *          ! date ! auteur ! but
46 *          -----
47 *          ! ! !
48 *          $
49 *          ***** *
50 *      MODIFICATIONS:
51 *          ! date ! auteur ! but
52 *          -----
53 *          $
54 *****/
55
56
57 #ifndef DDLLIM_H
58 #define DDLLIM_H
59
60
61 #include "Ddl.h"
62 #include "Tableau_T.h"
63 #include <iostream>
64 #include <stdlib.h>

```

```

65 #include "Sortie.h"
66 #include "UtilLecture.h"
67 #include "string.h"
68 #include "string"
69 #include <list>
70 #include "MvtSolide.h"
71 #include "EnumTypeCL.h"
72
73 /// @addtogroup Les_classes_Ddl_en_tout_genre
74 /// @{
75 ///
76 ///
77
78 /// BUT: La classe DdlLim permet de declarer un ensemble de ddl coiffe par une reference utilise
    pour les conditions limites.
79 ///
80 /// \author Gérard Rio
81 /// \version 1.0
82 /// \date 15/04/1997
83 ///
84 ///
85
86
87 class DdlLim
88 { // surcharge de l'operator de lecture
89 // les informations sont typées
90 friend istream & operator » (istream &, DdlLim &);
91 // surcharge de l'operator d'écriture typée
92 friend ostream & operator « (ostream &, const DdlLim &);
93
94 public:
95
96 // CONSTRUCTEURS :
97
98 // Constructeur default
99 DdlLim ();
100
101 // Constructeur de copie
102 DdlLim (const DdlLim& d);
103
104
105 // DESTRUCTEUR :
106 ~DdlLim ();
107
108
109 // METHODES :
110
111 // Retourne la reference attache au degre de liberte
112 const string& NomRef () const { return ref; };
113 // retourne un pointeur de string donnant le nom du maillage associé
114 // = NULL si aucun maillage
115 const string* NomMaillage() const {return nom_maillage;};
116 // retourne le type de condition limite associé
117 EnumTypeCL TypeDeCL() const {return typeCL;};
118
119 // retourne la taille du tableau de ddl
120 int Taille() const { return tab.Taille();};
121
122 // retourne le ddl nb i en lecture/écriture
123 Ddl& ElemLim(int i) { return tab(i).ddl;};
124 // idem en constant
125 const Ddl& ElemLim_const(int i) const { return tab(i).ddl;};
126
127 // retourne le nom de la courbe i ou "" s'il n'y en n'a pas
128 // dans le cas d'un mouvement solide, c'est uniquement le premier ddl
129 // qui contient la courbe de charge éventuelle
130 const string& Nom_courbe(int i) const { return tab(i).co_charge;};
131
132 // retourne le nom de la fonctionnD i ou "" s'il n'y en n'a pas
133 // dans le cas d'un mouvement solide, c'est uniquement le premier ddl
134 // qui contient la courbe de charge éventuelle
135 const string & NomF_charge(int i) const {return tab(i).f_charge;};
136
137 // retourne si la référence est celle d'un champ ou pas
138 bool Champ() const {return champ;};
139
140 // retourne si le blocage est relatif ou pas
141 // un blocage relatif signifie que le blocage s'effectue par rapport au pas de temps
142 // précédent i.e. t=t
143 // un blocage non relatif, donc absolue signifie que le blocage s'effectue par rapport à
144 // t = 0
145 bool BlocageRelatif() const {return blocage_relatif;};
146
147 // indique si c'est un mouvement solide ou pas
148 bool Mouvement_Solide() const {return (mvtsolide!=NULL);};
149 // récupération du mouvement solide
150

```

```

151     const MvtSolide* Const_MouvementSolide() const {return mvtsolide;};
152     // récupération du mouvement solide
153     MvtSolide* MouvementSolide() const {return mvtsolide;};
154
155     // retourne l'échelle pour le ddl nb i
156     // dans le cas d'un mouvement solide, c'est uniquement le premier ddl
157     // qui contient la courbe de charge éventuelle
158     double Echelle_courbe(int i) const {return tab(i).echelle;};
159
160     // indique si le temps mini et/ou le temps maxi dépend d'une fonction nD
161     // retour:
162     // 0 : pas de dépendance
163     // 1 : tmin uniquement
164     // 2 : tmax uniquement
165     // 3 : tmin et tmax
166
167     int Temps_depend_nD() const
168     {int retour = 0; if (nom_fnD_t_min != "") {if (nom_fnD_t_max != "") return 3;else return 1;}
169       else {if (nom_fnD_t_max != "") return 2;else return 0;};
170     };
171
172     // retour du nom de la fonction nD qui pilote tmin
173     // si == "", pas de fonction
174     string Nom_fctnD_tmin() const {return nom_fnD_t_min;};
175     // idem pour tmax
176     string Nom_fctnD_tmax() const {return nom_fnD_t_max;};
177
178     // mise à jour de tmin et/ou tmax
179     // uniquement s'ils dépendent d'une fct nD
180     void Mise_a_jour_tmin_tmax(double tmin, double tmax)
181     { if (nom_fnD_t_min != "") t_min = tmin;
182       if (nom_fnD_t_max != "") t_max = tmax;
183     };
184     // mise à jour de tmin
185     // uniquement s'il dépend d'une fct nD
186     void Mise_a_jour_tmin(double tmin)
187     { if (nom_fnD_t_min != "") t_min = tmin;
188     };
189     // mise à jour de tmax
190     // uniquement s'il dépend d'une fct nD
191     void Mise_a_jour_tmax(double tmax)
192     { if (nom_fnD_t_max != "") t_max = tmax;
193     };
194
195     // retourne un booléen qui indique si oui ou non le temps passé en paramètre
196     // est situé entre le temps min et maxi du ddlim
197     bool Temps_actif(const double& temps) const
198     {if ((t_min < temps) && (temps <= t_max))
199       return true;
200       else return false;
201     };
202
203     // ramène vrai si le temps est inférieur au temps actif
204     // ou si d'une part le temps n'est pas actif et qu'au pas précédent
205     // il n'était pas également actif
206     bool Pas_a_prendre_en_compte(const double& temps) const
207     {if ((temps <= t_min) || ((temps > t_max) && !precedent))
208       return true;
209       else return false;
210     };
211     // ramène vrai si le temps est inférieur ou égale au temps actif, ou supérieur au temps max
212     bool Pas_a_prendre_en_compte_dans_intervalle(const double& temps) const
213     {if ((temps <= t_min) || (temps > t_max))
214       return true;
215       else return false;
216     };
217
218     // affichage et definition interactive des commandes
219     // plusieurs_maillages : indique si oui ou non, il y a plusieurs maillage
220     void Info_commande_DdlLim(ofstream & sort,bool plusieurs_maillages);
221
222     // Validation on non de l'activité du ddlLim
223     void Validation(const double& temps) {precedent = Temps_actif(temps);};
224     // retour du statut de validation
225     // vrai signifie que l'état enregistré est actif
226     bool Etat_validation() const {return precedent;};
227
228
229     // lecture des degres de liberte
230     // sur le fichier d'entree
231     // util pour la lecture de ddl bloque
232     void Lecture(UtilLecture & entreePrinc);
233
234     // examine si le ddlmin passé en argument possède les mêmes cibles que this
235     // mais que les data associés sont différents
236     // ramène true si c'est vrai, false sinon
237     bool MemeCibleMaisDataDifferentes(const DdlLim& d) const;

```

```

238
239 // ramène true si le DdlLim contient l'enum_ddl passé en argument, false sinon
240 bool Existe_ici_leDdl(Enum_ddl enu) const;
241 // ramène true si le DdlLim contient au moins un déplacement, false sinon
242 bool Existe_ici_un_deplacement() const;
243
244 // Affiche les donnees liees aux degres de liberte
245 void Affiche () const ;
246
247 // Realise l'egalite
248 DdlLim& operator= (const DdlLim& d);
249
250 //Surcharge d'opérateur logique
251 bool operator == ( const DdlLim& a) const ;
252
253 bool operator != ( const DdlLim& a) const { return !(*this == a);};
254
255 // ----- définition de conteneurs à usage interne -----
256 // cependant elles ne peuvent pas être déclaré en protégé ??
257 // protected :
258
259 // le conteneur des ddl bloqués
260 class Ddlbloque
261 { public:
262 // surcharge de l'opérateur de lecture
263 friend istream & operator » (istream & entree, Ddlbloque& d);
264 // surcharge de l'opérateur d'écriture
265 friend ostream & operator « (ostream & sort, const Ddlbloque& d);
266 // constructeur
267 Ddlbloque() : co_charge(""), f_charge(""), ddl(), echelle(1.) {};
268 Ddlbloque(const Ddlbloque& d): co_charge(d.co_charge)
269 , f_charge(d.f_charge), ddl(d.ddl), echelle(d.echelle) {};
270 Ddlbloque(Ddl ddl_): co_charge(""), f_charge(""), ddl(ddl_), echelle(1.) {};
271 // destructeur
272 ~Ddlbloque() {};
273 // surcharge de l'opérateur d'affectation
274 Ddlbloque& operator= (const Ddlbloque& d);
275 //Surcharge d'opérateur logique
276 bool operator == ( Ddlbloque& a);
277 bool operator != ( Ddlbloque& a);
278 public:
279 Ddl ddl; // ddl bloqué
280 string co_charge; // nom d'une courbe de charge éventuelle
281 string f_charge; // nom d'une fonction nD utilisant des variables globales et autres
282 double echelle;
283 };
284
285 protected :
286
287 // données
288 string* nom_maillage; // nom de maillage associé, éventuellement!
289 string ref; // reference attachee aux ddl
290 EnumTypeCL typeCL; // indique éventuellement un type de condition limite associé
291 // = RIEN_TYPE_CL par exemple pour les noeuds (donc ne sert à rien dans ce cas)
292 // = TANGENTE_CL par exemple pour une arête, indique une condition de tangente
293
294 imposée
295 Tableau <Ddlbloque> tab; // tableau des ddl
296 double t_min,t_max; // temps mini et maxi de durée des ddl imposés
297 string nom_fnD_t_min; // nom éventuelle de la fonction associée
298 string nom_fnD_t_max; // nom éventuelle de la fonction associée
299 int precedent; // pour la description de l'évolution du ddlLim
300 bool champ; // indique si c'est une référence de champ ou pas
301 bool blocage_relatif; // indique si oui ou non le blocage à t+dt s'effectue
302 // par rapport à t ou par rapport à 0
303 MvtSolide* mvtsolide; // indique si diff de NULL qu'il s'agit d'un mouvement solide appliqué
304 // si oui, tab contient les 3 ddl de position, dont le premier contient éventuellement
305 // une courbe de charge si on veut un pilotage du mvt solide par courbe de charge,
306 // ainsi qu'une échelle éventuelle globale
307
308 // FONCTIONS PROTEGEES
309 // verification en fonction de la dimension du pb
310 // par exemple en dim 2 d'existence des UZ ou des X3 etc..
311 void VerifDimDdl();
312 // lecture dans le cas particulier d'un champ
313 list <Ddlbloque> Lecture_champ(UtilLecture & entreePrinc);
314 // lecture dans le cas particulier d'un mouvement solide
315 list <Ddlbloque> Lecture_mvtSolide(UtilLecture & entreePrinc);
316
317 // Remplace la reference du degre de
318 // liberte par nouveau
319 // ATTENTION : Les modifications liees au changement de la reference du degre
320 // de liberte sont a la charge de l'utilisateur, en particulier il n'y a pas
321 // de vérification par exemple du fait que c'est un champ ou pas .
322 void Change_ref (string nouveau);
323 // idem pour le changement de nom de maillage
324 // le nouveau nom ne doit pas être vide !! sinon erreur

```



```

324 void Change_nom_maillage(const string& nouveau);
325
326 };
327 /// @} // end of group
328
329
330
331 #endif

```

## 7.347 DdlNoeudElement.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *      $
38 *      BUT: Def des ddl lie a un noeud d'un element ( different des ddl
39 *           lies aux noeuds globaux).
40 *
41 *      $
42 *
43 *      VERIFICATION:
44 *
45 *      ! date !   auteur !           but
46 *      !           !           !
47 *      $
48 *
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      !           !           !
52 *      $
53 *****/
54 #ifndef DDLNOEUDELEMENT_T
55 #define DDLNOEUDELEMENT_T
56 #include "Enum_ddl.h"
57 #include "Tableau_T.h"
58 // #ifdef SYSTEM_MAC_OS_X
59 // #include <stringfwd.h> // a priori ce n'est pas portable
60 // #else
61 // #include <stringfwd.h> // a priori ce n'est pas portable
62 // #endif
63 #include <string.h> // pour le flot en memoire centrale
64 #endif
65 #include <string>
66
67
68
69 /// @addtogroup Les_classes_Ddl_en_tout_genre
70 /// @{
71 ///
72
73

```

```

74 ///     BUT:     Def des ddl lie a un noeud d'un element ( different des ddl lies aux noeuds globaux)
75 ///
76 /// \author     Gérard Rio
77 /// \version    1.0
78 /// \date      23/01/97
79 ///
80 ///
81
82 class DdlNoeudElement
83 {
84     // surcharge de l'operator de lecture
85     friend istream & operator » (istream & entree, DdlNoeudElement& a);
86     // surcharge de l'operator d'écriture
87     friend ostream & operator « (ostream & sort, const DdlNoeudElement& a);
88
89 public :
90     // VARIABLES PUBLIQUES :
91
92     // CONSTRUCTEURS :
93     DdlNoeudElement (); // par default
94     // def en fonction dunombre de ddl a stocker
95     DdlNoeudElement (int n);
96     // cas où il y a un seul identificateur de ddl a stocker
97     DdlNoeudElement (Enum_ddl e) :
98         tb(1,e)
99         { };
100     // de copie
101     DdlNoeudElement (const DdlNoeudElement& a);
102     // DESTRUCTEUR :
103     ~DdlNoeudElement ();
104     // Methodes
105     // surcharge de l'affectation
106     DdlNoeudElement& operator = (const DdlNoeudElement & a);
107     // surcharge des tests
108     bool operator == (const DdlNoeudElement & a) const ;
109     bool operator != (const DdlNoeudElement & a) const ;
110
111     // le tableau des identificateurs de ddl
112     Tableau <Enum_ddl> tb;
113     // pointeur du tableau de connection ddl noeud element/ ddl noeud global
114     // pour l'instant ne sert à rien donc en commentaire
115 protected :
116     // int * pt;
117 };
118 /// @} // end of group
119
120 #endif

```

## 7.348 DiversStockage.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           23/01/97
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++

```

```

36 *                                     $ *
37 *****
38 * BUT: Classe servant a stocker des informations intermediaires.*
39 *
40 *                                     $ *
41 * //*****
42 *
43 * VERIFICATION:
44 * ! date ! auteur ! but !
45 * -----
46 * ! ! ! !
47 * $
48 * //*****
49 * MODIFICATIONS:
50 * ! date ! auteur ! but !
51 * -----
52 * $
53 *****/
54 #ifndef DIVERSSTOCKAGE_H
55 #define DIVERSSTOCKAGE_H
56
57 #include <string.h>
58 #include <string>
59 #include <list>
60 #include "Tableau_T.h"
61 #include "UtilLecture.h"
62 #include "LesReferences.h"
63 #include "MotCle.h"
64 #include "LectBloc_T.h"
65 #include "Bloc.h"
66 #include "BlocDdlLim.h"
67
68 /**
69 *
70 * BUT:Classe servant a stocker des informations intermediaires.
71 *
72 * \author Gérard Rio
73 * \version 1.0
74 * \date 23/01/97
75 * \brief Classe servant a stocker des informations intermediaires.
76 *
77 */
78
79 class DiversStockage
80 {
81 public :
82 // VARIABLES PUBLIQUES :
83 // CONSTRUCTEURS :
84
85 DiversStockage ();
86 // DESTRUCTEUR :
87 ~DiversStockage () {};
88
89 // METHODES PUBLIQUES :
90 // affichage des infos
91 void Affiche () const ; // affichage de tous les infos
92 void Affiche1 () const ; // affichage des infos de la premiere lecture
93 void Affiche2 () const ; // affichage des infos de la seconde lecture
94
95
96 // lecture des infos
97 // dans la premiere lecture il s'agit actuellement des epaisseurs, des sections
98 void Lecture1(UtilLecture & entreePrinc,LesReferences& lesRef);
99
100 // dans la seconde lecture il s'agit des masses additionnelles
101 void Lecture2(UtilLecture & entreePrinc,LesReferences& lesRef);
102
103 // retourne le tableau des epaisseurs
104 const Tableau < BlocDdlLim<BlocScal_ou_fctnD> > & TabEpaiss() const { return tabEpaiss;};
105
106 // retourne le tableau des largeurs
107 const Tableau < BlocDdlLim<BlocScal_ou_fctnD> > & TabLargeurs() const { return tabLargeurs;};
108
109 // retourne le tableau des section
110 const Tableau < BlocDdlLim<BlocScal_ou_fctnD> > & TabSect() const { return tabSection;};
111
112 // retourne le tableau des variations de section
113 const Tableau < BlocDdlLim<BlocScal> > & TabVarSect() const { return tabVarSection;};
114
115 // retourne le tableau des masses volumiques
116 const Tableau < BlocDdlLim<BlocScal_ou_fctnD> > & TabMasseVolu() const { return tabMasseVolu;};
117
118 // retourne le tableau des masses additionnelles
119 const Tableau < BlocDdlLim<BlocScal_ou_fctnD> > & TabMasseAddi() const { return tabMasseAddi;};
120
121 // retourne le tableau des dilatations thermique

```

```

122     const Tableau < BlocDdlLim<BlocScal_ou_fctnD> > & TabCoefDila() const { return tabCoefDila;};
123
124     // retourne le tableau des gestions d'hourglass
125     const Tableau < BlocDdlLim<BlocGen_3_1> > & TabGesHourglass() const { return tabGesHourglass;};
126
127     // retourne le tableau d'intégration de volume d'une grandeur
128     const Tableau < BlocDdlLim<BlocGen_3_0> >& TabIntegVol() const {return tabIntegVol;};
129
130     // retourne le tableau d'intégration de volume et en temps d'une grandeur
131     const Tableau < BlocDdlLim<BlocGen_3_0> >& TtabIntegVol_et_temps() const {return
tabIntegVol_et_temps;};
132
133     // retourne le tableau sur la stabilisation transversale de membrane ou biel
134     const Tableau < BlocDdlLim<BlocGen_4_0> >& TabStabMemBiel() const {return tabStabMembraneBiel;};
135
136     // retourne le tableau sur la définition d'un repère d'anisotropie aux éléments
137     const Tableau < BlocDdlLim<BlocGen_6_0> >& TabRepAnisotrope() const {return tabRepAnisotrope;};
138
139     // lecture de donnée en fonction d'un indicateur : int type
140     // pour l'instant ne fait rien
141     void LectureDonneesExternes(UtilLecture& , LesReferences& , const int , const string& ) {};
142
143     // retourne le tableau sur statistique d'une ref de noeuds pour une grandeur quelconque
144     const Tableau < BlocDdlLim<BlocGen_3_0> >& TabStatistique() const {return tabStatistique;};
145
146     // retourne le tableau sur statistique d'une ref de noeuds cumulée en temps pour une grandeur
147     // queconque
148     const Tableau < BlocDdlLim<BlocGen_3_0> >& TabStatistique_et_temps() const {return
tabStatistique_et_temps;};
149
150     // affichage et definition interactive des commandes
151     void Info_commande_DiversStockage1(UtilLecture & entreePrinc);
152     void Info_commande_DiversStockage2(UtilLecture & entreePrinc);
153
154     //----- lecture écriture dans base info -----
155     // cas donne le niveau de la récupération
156     // = 1 : on récupère tout
157     // = 2 : on récupère uniquement les données variables (supposées comme telles)
158     void Lecture_base_info(ifstream& ent,const int cas);
159     // cas donne le niveau de sauvegarde
160     // = 1 : on sauvegarde tout
161     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
162     void Ecriture_base_info(ofstream& sort,const int cas);
163
164 private :
165     // VARIABLES PROTEGEES :
166
167     // infos sur les epaisseurs
168     Tableau < BlocDdlLim<BlocScal_ou_fctnD> > tabEpais;
169     // infos sur les largeurs
170     Tableau < BlocDdlLim<BlocScal_ou_fctnD> > tabLargeurs;
171
172     // infos sur les sections de biellette
173     Tableau < BlocDdlLim<BlocScal_ou_fctnD> > tabSection;
174
175     // infos sur les variations de sections de biellette
176     Tableau < BlocDdlLim<BlocScal> > tabVarSection;
177
178     // infos sur les masses volumique
179     Tableau < BlocDdlLim<BlocScal_ou_fctnD> > tabMasseVolu;
180
181     // infos sur les masses additionnelles
182     Tableau < BlocDdlLim<BlocScal_ou_fctnD> > tabMasseAddi;
183
184     // infos sur les coefficients de dilatation
185     Tableau < BlocDdlLim<BlocScal_ou_fctnD> > tabCoefDila;
186
187     // infos sur la gestion d'hourglass
188     Tableau < BlocDdlLim<BlocGen_3_1> > tabGesHourglass;
189
190     // infos sur intégration de volume d'une grandeur queconque
191     Tableau < BlocDdlLim<BlocGen_3_0> > tabIntegVol;
192
193     // infos sur intégration de volume et en temps d'une grandeur queconque
194     Tableau < BlocDdlLim<BlocGen_3_0> > tabIntegVol_et_temps;
195
196     // infos sur la stabilisation transversale de membrane ou biel
197     // a priori de type 4,0 mais peut s'agrandir si nécessaire à la lecture !!
198     Tableau < BlocDdlLim<BlocGen_4_0> > tabStabMembraneBiel;
199
200     // infos sur la définition d'un repère d'anisotropie aux éléments
201     Tableau < BlocDdlLim<BlocGen_6_0> > tabRepAnisotrope;
202
203     // infos sur statistique d'une ref de noeuds pour une grandeur quelconque
204     Tableau < BlocDdlLim<BlocGen_3_0> > tabStatistique;
205

```

```

206 // infos sur statistique d'une ref de noeuds cumulée en temps pour une grandeur queconque
207 Tableau < BlocDdlLim<BlocGen_3_0> > tabStatistique_et_temps;
208
209 };
210
211 #endif

```

## 7.349 I\_O\_Condilinaire.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      19/01/2007
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:      gérer les conditions limites linéaires en I/O
39 *             en particulier comme conteneur.
40 *   informations: pour l'instant on considère une conditions
41 *                 linéaires entre les ddl d'une même famille d'un noeud.
42 *   Une instance de I_O_Condilinaire peut générer un grand nombres
43 *   de conditions linéaires particulières, en fait autant que de
44 *   noeuds appartenant à la référence principal. Ces conditions
45 *   particulières ne sont pas stockées dans I_O_Condilinaire !!
46 *   par contre, elles peuvent être générées par I_O_Condilinaire.
47 *
48 *   *****
49 *
50 *   VERIFICATION:
51 *
52 *   ! date ! auteur ! but
53 *   -----
54 *   ! ! ! !
55 *   *****
56 *   MODIFICATIONS:
57 *
58 *   ! date ! auteur ! but
59 *   -----
60 *   $
61 *****/
62 #ifndef I_O_CONDILINAIRE_H
63 #define I_O_CONDILINAIRE_H
64
65 #include "Condilinaire.h"
66 #include "Plan.h"
67 #include "Droite.h"
68
69 /// @addtogroup Les_classes_relatives_aux_conditions_limites
70 /// @{
71
72 class I_O_Condilinaire : public Condilinaire
73 {
74 // surcharge de l'operator de lecture
75 // les informations sont typées
76 friend istream & operator » (istream &, I_O_Condilinaire &);
77 // surcharge de l'operator d'écriture typée

```

```

76     friend ostream & operator << (ostream &, const I_O_Condilinaire &);
77
78     public :
79         // CONSTRUCTEURS :
80
81         // par défaut
82         I_O_Condilinaire();
83         // de copie
84         I_O_Condilinaire(const I_O_Condilinaire& nd);
85         // DESTRUCTEUR :
86         ~I_O_Condilinaire();
87
88
89         // METHODES PUBLIQUES :
90
91         // Retourne la reference attache à la condition linéaire
92         const string& NomRef () const { return refe; };
93         // retourne un pointeur de string donnant le nom du maillage associé
94         // = NULL si aucun maillage
95         const string* NomMaillage() const {return nom_maillage;};
96
97         // retourne le tableau des références associées (peut-être vide !)
98         const Tableau <string> & ReferenceAssociees() const {return refs_associe;};
99         // retourne le tableau des nom de maillage associées aux noms de ref (peut être = 0)
100        // tail = 0 si il n'y a pas de nom de maillage sur la ref principale
101        const Tableau <string > & NomMaillageAssociees() const {return nom_mail_associe;};
102
103        // examine si la condition passée en argument possède les mêmes cibles que this
104        // mais que les data associés sont différents
105        // ramène true si c'est vrai, false sinon
106        bool MemeCibleMaisDataDifferentes(I_O_Condilinaire& d);
107
108        // Affiche les donnees
109        void Affiche () const ;
110        // Realise l'egalite
111        I_O_Condilinaire& operator= (const I_O_Condilinaire& d);
112
113        //Surcharge d'operateur logique: ne concerne que la condition initiale (pas la condition actuelle,
114        // c-a-d les coeff actuelles et les noeuds actuellement en cause etc.)
115        bool operator == ( I_O_Condilinaire& a) const ;
116
117        bool operator != ( I_O_Condilinaire& a) const { return !(*this == a);};
118
119        // lecture de la condition linéaire sur le fichier d'entree
120        void Lecture(UtilLecture & entreePrinc);
121
122        // retourne si la condition est relative ou pas
123        // une condition relative signifie que la condition s'effectue par rapport au pas de temps
124        // précédent i.e. t=t
125        // une condition non relative, donc absolue signifie que la condition s'effectue par rapport à
126        // t = 0
127        bool ConditionRelative() const {return condition_relative;};
128
129        //--- cas des conditions linéaires type plan ou droite -----
130
131        // ramène true si on a effectivement une condi linéaire type plan ou droite
132        bool PlanDroite() const {return def_auto_par_rotation;};
133
134        // indique si le blocage qui existe déjà sur les noeuds à projeter, est cohérent
135        // avec la projection:
136        // typiquement si la direction de blocage est normale au plan de rotation
137        // ramène true si on peut conserver la condition de projection
138        // en entrée: indi qui donne la direction déjà bloquée
139        bool Coherence_condi_PlanDroite(int indi)const;
140
141        // ramène true si on a effectivement une condi linéaire type stricte_egalite
142        bool StricteEgalite() const {return stricte_egalite;};
143
144        // le nom du maillage éventuellement et le numéro du noeud s'il y a un noeud rotation
145        // sinon ramène un pointeur null
146        const String_et_entier* RefNoeudRotation() const {if (type_centre==1){return NULL;}
147                                                else {return &mailEtNumCentreNoeud;}};
148        // definition du noeud: centre rotation
149        void ChangeCentreNoeudRotation( Noeud* noe);
150        // définition de la direction actuelle ou normale pour plan droite
151        void ActualiseDirectionPlanDroite();
152        // projection (normale) du point M sur le plan ou droite, ramène le point M à la place du
153        // projeté, et calcul de la condition linéaire correspondant à la projection
154        Coordonnee& ProjectionSurPlanDroite_et_calValCondilin(Coordonnee& M );
155
156
157
158        // --- fin des condi linéaires types plan droite -----
159
160        // Construction de la condition : en particulier les pointeurs de ddl permettant d'imposer la
161        // condition
162        // et def des coefficients en fonction éventuellement du temps ou de l'incrément du temps selon

```

```

162 // le paramètre condition_relative
163 // pour cela
164 // 1) on doit renseigner les fonctions de charge si elles existent
165 // a) retour du nombre de fonction de charge
166 int ExisteFonctionChargeCoef() const {return tab_co_charge.Taille();};
167 // b) le nom de la courbe
168 const string & NomFctch(int i) const {return tab_co_charge(i);};
169 // b) on renseigne les valeurs des fonctions de charge
170 void Valeur_fonctionChargeCoef(const Vecteur & fctch_, const Vecteur & delta_fctch_)
171 {fctch = fctch_; delta_fctch = delta_fctch_;};
172 // 2)
173 // Construction de la condition : 1) mise en place d'une valeur imposée due à la condition linéaire
174 // def des coefficients en fonction éventuellement des fonctions de charges et du paramètre
condition_relative
175 // NB: pour les conditions correspondant à une projection sur un plan ou une droite, il faut que la
direction du plan
176 // ou de la droite ait déjà été affecté auparavant
177 // !! la condition sur le ddl n'est pas imposée à ce niveau, par contre elle est stockée dans
Condilinaire
178 Condilinaire& ConstructionCondition(Tableau <Noeud *> & t_noe, Condilinaire& condi_actuelle);
179
180 // affichage des commandes particulières à la définition d'une condition linéaire
181 void Info_commande_conditionLineaire(ofstream & sort, bool plusieurs_maillages);
182
183 // récupération de l'énuméré du type de ddl considéré par la condition limite linéaire
184 Enum_ddl TypeEnu() const {return enu;};
185
186 // retourne un booléen qui indique si oui ou non le temps passé en paramètre
187 // est situé entre le temps min et maxi de la condition linéaire
188 bool Temps_actif(const double& temps) const
189 {if ((t_min < temps) && (temps <= t_max)) return true;
190     else return false; };
191 // ramène vrai si le temps est inférieur au temps actif, ou supérieur au temps max
192 // ou si d'une part le temps n'est pas actif et qu'au pas précédent
193 // il n'était pas également actif
194 bool Pas_a_prendre_en_compte(const double& temps) const
195 {if ((temps <= t_min) || ((temps > t_max) && !precedent))
196     return true;
197     else return false;
198 };
199 // ramène vrai si le temps est inférieur ou égale au temps actif, ou supérieur au temps max
200 bool Pas_a_prendre_en_compte_dans_intervalle(const double& temps) const
201 {if ((temps <= t_min) || (temps > t_max))
202     return true;
203     else return false;
204 };
205 // Validation on non de l'activité de la charge
206 void Validation(const double& temps) {precedent = Temps_actif(temps);
207     valeur_precedente_t = beta;};
208 // retour du statut de validation
209 // vrai signifie que l'état enregistré est actif
210 bool Etat_validation() const {return precedent;};
211
212 // initialisation des grandeurs de travail
213 void Initialisation_condil();
214
215 protected :
216 // informations: pour l'instant on considère une conditions linéaires entre les ddl d'une même
217 // famille d'un noeud.
218
219 // données
220 string* nom_maillage; // nom de maillage associé, éventuellement!
221 string refe; // reference principale attachee aux ddl
222 Tableau <string> refs_associe;
223 // dans le cas où il y a une condition linéaire entre plusieurs
224 // noeuds, la méthode retenue est la suivante: à chaque noeud de refe on associe un noeuds
225 // de chaque référence du tableau refs_associe, et on prend les ddl de type enu de chaque ref
226 // et on a une combinaison linéaire entre chacun. Dans ce cas, il faut donc que toutes les
227 // référence aient le même nombre de noeuds
228 Tableau <string > nom_mail_associe; // dans le cas où nom_maillage est non null
229 // il faut que toutes les références associées aient un nom de maillage associé
230 double t_min, t_max; // temps mini et maxi de durée des ddl imposés
231 double echelle; // échelle globale éventuelle
232 Enum_ddl enu; // identificateur du premier ddl de la famille pour laquelle on a une condition
linéaire
233 // les ddl s'applique d'abord à suivre à chaque noeud
234 int nbddl_famille; // nombre de ddl de la famille associée à enu
235 int condition_relative; // indique si oui ou non la condition à t+dt s'effectue
236 // par rapport à t ou par rapport à 0
237 int precedent; // pour la description de l'évolution d'un temps à l'autre
238 double valeur_precedente_t; // valeur à t
239
240 Tableau <string> tab_co_charge; // même dimension que (le nombre de ddl de type enu) * (1+le nombre
241 // de ref associé) s'il y a des courbes de charge
242 // Permet d'avoir des coefficients qui varient. S'il y a une courbe de charge
243 // alors toutes existent sinon aucune
244 Vecteur fctch, delta_fctch; // vecteur de travail qui contient les valeurs des fonctions de charge

```

```

245                                     // et leur accroissement entre t et t+deltat
246
247 //--- variables particulières pour la définition automatique d'une condition linéaire sur plan ou
droite
248 bool def_auto_par_rotation;// signale le cas particulier du condition linéaire automatique
249 int type_centre; // = 1 on a un centre fixe c-a-d: centre_rotation, =2 le centre est centre_noeud
à t=0
250                                     // =3 le centre est centre_noeud à t
251 Coordonnee centre_rotation; // position du centre de rotation fixe
252 String_et_entier mailEtNumCentreNoeud; // maillage et numéro du centre noeud s'il existe
253 Noeud* centre_noeud;
254 Plan pl;
255 Droite dr;
256
257 //--- variables particulières pour la condition particulière d'égalité exacte, pour tous les ddl
258 // du type enu, -> traitement particulier
259 bool stricte_egalite;
260
261 // 1) cas simple: pas de ref associe cela signifie que la condition linéaire concerne uniquement
262 // des ddl de chaque noeud de la ref principal. Pour chaque noeud on applique la même condition (au
263 // mvt solide près). La taille de val
264 // 1) cas où l'on a des références secondaire, celle-ci doivent avoir tous le même nombre de noeud,
265 // identique à la ref principal. Supposons qu'il y a na liste associé. La condition linéaire relie
266 // les ddl de na+1 noeud, chacun étant pris dans une ref associé + 1 sur la ref principal. Pour
chaque
267 // noeud la liste des enu_ddl du même type que enu est parcouru: soit t_enu leur nombre. Donc au
final,
268 // la taille de val (qui contient les coeff de la condition linéaire = (na+1)*t_enu
269 };
270 /// @} // end of group
271
272
273 #endif

```

## 7.350 LesCondLim.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Gestion des differentes conditions limites
39 *
40 *      *****
41 *      VERIFICATION:
42 *
43 *      ! date !   auteur !           but
44 *      -----
45 *      !       !       !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date !   auteur !           but

```



```

50 * ----- *
51 *                                     $ *
52 *****/
53 #ifndef LESCONDLIM_H
54 #define LESCONDLIM_H
55
56 #include "LectBloc_T.h"
57 #include "Tableau_T.h"
58 #include "DdlLim.h"
59 #include "UtilLecture.h"
60 #include "LesReferences.h"
61 #include "MotCle.h"
62 #include "LesMaillages.h"
63 #include "CondLim.h"
64 #include "Condilineaire.h"
65 #include "Nb_assemb.h"
66 #include "LesCourbes1D.h"
67 #include "LesFonctions_nD.h"
68 #include "ReferenceNE.h"
69 #include "ReferenceAF.h"
70 #include "Basiques.h"
71 #include "I_O_Condilineaire.h"
72 #include "Temps_CPU_HZpp.h"
73
74 /** @defgroup Les_classes_relatives_aux_conditions_limites
75 *
76 *     BUT:Gestion des differentes conditions limites
77 *
78 * \author   Gérard Rio
79 * \version  1.0
80 * \date    23/01/97
81 * \brief   Gestion des differentes conditions limites
82 *
83 */
84
85 /// @addtogroup Les_classes_relatives_aux_conditions_limites
86 /// @{
87 ///
88 class LesCondLim
89 {
90 public :
91     // CONSTRUCTEURS :
92     LesCondLim (); // par default
93     // DESTRUCTEUR :
94     ~LesCondLim ();
95
96     // METHODES PUBLIQUES :
97
98     // lecture des conditions limites : ddl bloque
99     void Lecture1(UtilLecture & entreePrinc,LesReferences& lesRef);
100
101     // lecture des conditions limites linéaires
102     void Lecture2(UtilLecture & entreePrinc,LesReferences& lesRef);
103
104     // lecture des conditions limites : initialisation
105     void Lecture3(UtilLecture & entreePrinc,LesReferences& lesRef);
106
107     // affichage des informations concernant les conditions limites
108     void Affiche() const ; // affichage de tous les infos
109     void Affiche1() const ; // affichage des ddl bloques
110     void Affiche2() const ; // affichage des conditions limites linéaires
111     void Affiche3() const ; // affichage des ddl d'initialisation
112
113     // introduction des données et variables pour leurs emplois futures, avec init par défaut
114     void IntroductionDonnees (LesMaillages * lesMail,LesReferences* lesRef
115                             ,LesCourbes1D* lesCourbes1D,LesFonctions_nD* lesFonctionsnD);
116
117     // initialisation des ddl avec le tableau de ddl d'init
118     // verif de l'existence de tous les ddl (initialisation et imposes)
119     // ajout si necessaire, dans ce dernier cas l'initialisation est a zero
120     // choix = false : indique qu'il faut initialiser que les ddl a t
121     // choix = true : indique qu'il faut initialiser les ddl a t et t+dt
122     // vérification de l'existence des courbes de charge adoc si nécessaire
123     // cas : indique le cas d'association de ddl en cours, ces ddl sont gérés globalement, c-a-d
    lorsque
124     // le statut d'un ddl est modifié (de bloqué à non bloqué par exemple) tous les ddl
    associés ont
125     // leur statut modifié de manière équivalente
126     // =0 pas d'association
127     // =1 association X V GAMMA
128     void Initial(LesMaillages * lesMail,LesReferences* lesRef,LesCourbes1D* lesCourbes1D
129                ,LesFonctions_nD* lesFonctionsnD,bool choix,int cas);
130
131 // // mise à jour de l'initialisation, par exemple après un restart
132 // // même paramètres que Initial, par contre ici il n'y a aucune création
133 // void Re_initial(LesMaillages * lesMail,LesReferences* lesRef,LesCourbes1D* lesCourbes1D,bool
    choix,int cas);

```

```

134
135 // initialisation du nombre de cas d'assemblage
136 // si inf au nombre actuel, aucune action
137 void InitNombreCasAssemblage(int nb_cas)
138 { if (nb_cas > condlim.Taille())
139     condlim.Change_taille(nb_cas);
140 };
141
142 // incrementation des coordonnees a t+dt et des ddl en fonctions des ddl imposes
143 // et du chargement
144 // coef: est un facteur multiplicatif des ddl sans courbes de charge,
145 //     est supposé intégrer déjà le multiplicateur général
146 // en fait du chargement impose
147 // mult_gene : multiplicateur général de tous les chargements
148 // deltat : incrément de temps actuel
149 // temps : le temps courant où sont calculées les conditions
150 // ch_statut : indique s'il y a changement ou non de statut des conditions limites (pour les ddl aux
noeuds)
151 //     c-a-d un ddl qui passe de bloqué à non bloqué ou inversement: il s'agit uniquement des
ddl aux noeuds
152 //     c-a-d ceux qui ont une influence sur la mise en place des cl sur la raideur et le
résidu
153 // cas : indique le cas d'association de ddl en cours, ces ddl sont gérés globalement, c-a-d
lorsque
154 //     le statut d'un ddl est modifié (de bloqué à non bloqué par exemple) tous les ddl
associés ont
155 //     leur statut modifié de manière équivalente
156 //     =0 pas d'association
157 //     =1 association X V GAMMA
158 // lorsque en_ddl est égal à NU_DDL, cela signifie que l'on met les conditions limites sur tous les
159 // ddl de noeud actifs
160 // lorsque en_ddl est différent de NU_DDL, il donne le type des seules ddl pour lesquels on met les
161 // conditions de blocage: par exemple X1 -> blocage sur X1,X2,X3 selon la dimension
162 void MiseAJour_tdt
163 (const double& mult_gene,LesMaillages * lesMail,const double& deltat,const LesReferences* lesRef
164 ,const double& temps,LesCourbes1D* lesCourbes1D,LesFonctions_nD* lesFonctionsnD
165 ,const double& coef,bool& ch_statut
166 ,int cas , Enum_ddl en_ddl = NU_DDL);
167
168 // mise en place de la répercussion sur les noeuds des conditions linéaires imposées externes (par
les données d'entrées),
169 // ne prend pas en charge par exemple les CL dues aux contacts(gérés par CL)
170 // pour les paramètres: voir MiseAJour_tdt
171 void MiseAJour_condilinaire_tdt
172 (const double& mult_gene,LesMaillages * lesMail,const double& deltat,LesReferences*
lesRef
173 ,const double& temps,LesCourbes1D* lesCourbes1D,LesFonctions_nD* lesFonctionsnD
174 ,const double& coef,bool& ch_statut
175 ,int cas ,Enum_ddl en_ddl = NU_DDL);
176
177 // validation des conditions de blocages et des conditions linéaire, pour l'incrément.
178 // concernant l'activité des ddlLim, elle est enregistrée en fonction du temps
179 // idem pour les conditions linéaires
180 void Validation_blocage (LesReferences* lesRef,const double& temps);
181
182 // test s'il y a changement de statut pour le temps indiqué par rapport à la situation actuelle ou
pas
183 // n'effectue aucune autre opération
184 // si en_ddl est différent de NU_DDL on test le changement de statut uniquement pour le ddl en_ddl
185 bool Changement_statut(const LesMaillages * lesMail,const LesReferences* lesRef
186 ,LesFonctions_nD* lesFonctionsnD
187 ,const double& temps,const Enum_ddl en_ddl = NU_DDL) ;
188 // récupération des tableaux d'indices généraux des ddl bloqués
189 // cas : donne le type d'association de ddl que l'on veut
190 //     =0 -> pas d'association
191 //     =1 -> association de Xi avec Vi avec Gamma1
192 // t_assemb: donne pour chaque type d'association, le numéro d'assemblage correspondant au différent
ddl
193 //     de l'association
194 //     cas=1 -> numéro d'assemblage de X1 puis V1 puis GAMMA1
195 //     : en sortie une liste de Gene_asso, correspondant à tous les ddl bloqués et les
196 //     les ddl associés
197
198 // définition d'un conteneur pour la routine Tableau_indice
199 class Gene_asso
200 { public :
201     Gene_asso () : ty_prin(NU_DDL),pointe() {};
202     Gene_asso (Enum_ddl ty_prin,const Tableau <int>& pointes) :
203         ty_prin(ty_prin),pointe(pointes) {};
204     Gene_asso (const Gene_asso& a) : ty_prin(a.ty_prin),pointe(a.pointe) {};
205     Gene_asso& operator= (const Gene_asso& a)
206     {ty_prin=a.ty_prin;pointe=a.pointe;return (*this);};
207     bool operator== (const Gene_asso& a) const
208     {if((ty_prin==a.ty_prin)&&(pointe==a.pointe)) return true;else return false;};
209     bool operator!= (const Gene_asso& a) const
210     {if ( (*this)==a ) return true; else return false;};
211     Enum_ddl ty_prin; // le ddl principal bloqué

```

```

212     Tableau <int> pointe; // la position générale des ddl, principal et secondaire
213     // pour le cas 1 : pointe(k) -> la position du ddl Xi , puis Vi puis Gammai
214 };
215
216 list <Gene_asso> Tableau_indice(const LesMaillages * lesMail,const Tableau <Nb_assemb> & t_assemb
217     ,const LesReferences* lesRef,const double& temps,int cas );
218
219 // mise en place des conditions limites sur les matrices et second membres
220 // (autres que les conditions linéaires)
221 // nb_casAssemb : le cas d'assemblage
222 // cas : indique un type d'association de ddl
223 //     = 0 -> pas d'association, seules les ddl fournis par l'utilisateur sont considéré
224 //     = 1 -> association de Xi Vi Gammai, qui sont pris en compte dès lors que l'un
225 //         a été fixé par l'utilisateur
226 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que vecglob
227 //         mais sans sauvegarde (correspond par exemple à une partie de vecglob)
228 void ImposeConLimtdt(LesMaillages * lesMail,LesReferences* lesRef,
229     Mat_abstraite & matglob,Vecteur& vecglob
230     ,const Nb_assemb& nb_casAssemb,int cas,Vecteur* vec2);
231
232 // mise en place des conditions limites sur le second membres
233 // nb_casAssemb : le cas d'assemblage
234 // cas : indique un type d'association de ddl
235 //     = 0 -> pas d'association, seules les ddl fournis par l'utilisateur sont considéré
236 //     = 1 -> association de Xi Vi Gammai, qui sont pris en compte dès lors que l'un
237 //         a été fixé par l'utilisateur
238 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que vecglob
239 //         mais sans sauvegarde (correspond par exemple à une partie de vecglob)
240 void ImposeConLimtdt(LesMaillages * lesMail,LesReferences* lesRef,
241     Vecteur& vecglob,const Nb_assemb& nb_casAssemb,int cas,Vecteur* vec2);
242
243 // mise en place des conditions limites sur deux matrices
244 // utilisé par exemple pour le flambement
245 // la première matrice est initialisée avec des 1 sur la diagonale
246 // la seconde avec des 0 sur la diagonale
247 // nb_casAssemb : le cas d'assemblage
248 // cas : indique un type d'association de ddl
249 //     = 0 -> pas d'association, seules les ddl fournis par l'utilisateur sont considéré
250 //     = 1 -> association de Xi Vi Gammai, qui sont pris en compte dès lors que l'un
251 //         a été fixé par l'utilisateur
252 void ImpConLimtdt2Mat(LesMaillages * lesMail,LesReferences* lesRef,
253     Mat_abstraite & matglob,Mat_abstraite & matgeom
254     ,const Nb_assemb& nb_casAssemb,int cas);
255
256 // retourne la valeur absolu du maxi des efforts extérieurs
257 // et le numero ili du ddl correspondant dans l'assemblage global
258 double MaxEffort(int & ili,const Nb_assemb& nb_casAssemb)
259 { return condlim(nb_casAssemb.n).MaxEffort(ili);};
260
261 // calcul des reactions et stockage des valeurs
262 // ceci dans le cas ou il n'y a pas de conditions lineaires appliquee
263 // on se sert des valeurs sauvegardees lors de la mise en place des CL bloquees
264 // nb_casAssemb : le cas d'assemblage
265 // cas : indique un type d'association de ddl
266 //     = 0 -> pas d'association, seules les ddl fournis par l'utilisateur sont considéré
267 //     = 1 -> association de Xi Vi Gammai, qui sont pris en compte dès lors que l'un
268 //         a été fixé par l'utilisateur
269 void CalculReaction(LesMaillages * lesMail,LesReferences* lesRef
270     ,const Nb_assemb& nb_casAssemb,int cas);
271
272 // récupération des reactions initiales, avant les rotations dues aux conditions linéaires
273 // il s'agit des réactions dues aux ddl bloqués et dues aux conditions linéaires
274 // et calcul des torseurs de réaction
275 void ReacAvantCHrepere(Vecteur& residu,LesMaillages * lesMail,
276     LesReferences* lesRef,const Nb_assemb& nb_casAssemb,int cas);
277
278 // récupération des reactions= residu et stockage des valeurs
279 // NB: après changement de repère, il n'y a plus que des ddl bloqués
280 // cas : indique un type d'association de ddl
281 //     = 0 -> pas d'association, seules les ddl fournis par l'utilisateur sont considéré
282 //     = 1 -> association de Xi Vi Gammai, qui sont pris en compte dès lors que l'un
283 //         a été fixé par l'utilisateur
284 void ReacApresCHrepere(Vecteur& residu,LesMaillages * lesMail,
285     LesReferences* lesRef,const Nb_assemb& nb_casAssemb,int cas);
286
287 // affichage sur la sortie sort des reactions
288 void Affiche_reaction(ofstream& sort,const LesMaillages * lesMail) const ;
289
290 // ----- conditions linéaires externes -----
291
292 // mise en place de condition externe lineaires
293 // expression de la raideur et du second membre dans un nouveau repere
294 // ramène si oui ou non, il y a eu un changement effectué
295 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que vecglob
296 //         mais sans sauvegarde (correspond par exemple à une partie de vecglob)
297 bool CoLinCHrepere_ext(Mat_abstraite & matglob,Vecteur& vecglob
298     ,const Tableau <Condilinaire>& tabCondLine

```

```

299         ,const Nb_assemb& nb_casAssemb,Vecteur* vec2);
300
301 // blocage des seconds membres pour les conditions lineaires
302 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que vecglob
303 //      mais sans sauvegarde (correspond par exemple à une partie de vecglob)
304 inline void CoLinBlocage(Mat_abstraite & matglob,Vecteur& vecglob
305         ,const Nb_assemb& nb_casAssemb,Vecteur* vec2)
306     { condlim(nb_casAssemb.n).CondlineaireImpose ( matglob, vecglob,vec2);};
307
308 // retour des ddl dans le repere general, ceci dans le cas d'utilisation
309 // de conditions lineaires
310 inline void RepInitiaux( Vecteur& sol,const Nb_assemb& nb_casAssemb)
311     { condlim(nb_casAssemb.n).RepInitiaux(sol); };
312
313 // mise en place de condition externe lineaires en une opération
314 // retour de la raideur et du second membre sans changement de repere
315 // ramène si oui ou non, il y a eu un changement effectué
316 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que vecglob
317 //      mais sans sauvegarde (correspond par exemple à une partie de vecglob)
318 bool CoLinUneOpe_ext(Mat_abstraite & matglob,Vecteur& vecglob
319         ,const Tableau <Condilinaire>& tabCondLine
320         ,const Nb_assemb& nb_casAssemb,Vecteur* vec2);
321
322
323 // ----- conditions linéaires internes: imposées par les conditions limites en entrée -----
324
325 // mise en place des conditions linéaires imposées par les données d'entrée
326 // expression de la raideur et du second membre dans un nouveau repere
327 // ramène si oui ou non, il y a eu un changement effectué
328 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que vecglob
329 //      mais sans sauvegarde (correspond par exemple à une partie de vecglob)
330 bool CoLinCHreper_int(Mat_abstraite & matglob,Vecteur& vecglob,const Nb_assemb&
nb_casAssemb,Vecteur* vec2);
331 // effacement du marquage de ddl bloque du au conditions lineaire imposées par les conditions
d'entrée
332 void EffMarque();
333
334 // def de la largeur de bande en fonction des conditions linéaire limite en entrée
335 // casAssemb : donne le cas d'assemblage en cours
336 // les condi linéaires ne donnent pas des largeurs de bande sup et inf égales !!!
337 // demi = la demi largeur de bande ,
338 // total = le maxi = la largeur sup + la largeur inf +1
339 // cumule = la somme des maxis, ce qui donnera la largeur finale, due à des multiples
multiplications: une par conditions linéaires
340 //      dans le cas où on tiens compte des conditions linéaires par rotation (c-a-d sans
multiplicateur ou pénalisation)
341 // en retour, ramène un booleen qui :
342 // = true : si la largeur de bande en noeud est supérieure à 1
343 // = false : si non, ce qui signifie dans ce cas qu'il n'y a pas d'augmentation de la largeur
344 //      en noeud
345 bool Largeur_Bande(int& demi,int& total,const Nb_assemb& casAssemb,LesMaillages *
lesMail,LesReferences* lesRef
346         ,int& cumule);
347
348 // création d'un tableau de condition linéaire, correspondant à toutes les conditions linéaires
d'entrées
349 // qu'elles soient actives ou pas (a priori cette méthode est conçu pour donner des infos
relativement à la largeur
350 // de bandes en noeuds due aux CLL)
351 // chacune des condition ne contient "d'exploitable" que le tableau de noeuds associés à la CLL,
352 Tableau <Tableau <Condilinaire> > ConnectionCLL(const LesMaillages * lesMail,const LesReferences*
lesRef) const;
353
354 // idem ConnectionCLL avec une liste, mais spécifiquement les conditions de type "stricte_egalite"
entre ddl de noeud
355 // correspondant à toutes les conditions linéaires d'entrées actives uniquement
356 // (a priori cette méthode est conçu pour donner des infos pour condenser les pointeurs
d'assemblages:
357 // LesMaillages::MiseAJourPointeurAssemblage() )
358 // chacune des condition ne contient "d'exploitable" que le tableau de noeuds associés à la CLL, et
le type enumere de ddl
359 list <Condilinaire> ConnectionCLL_stricte_egalite
360     (const LesMaillages * lesMail,const LesReferences* lesRef) const;
361
362 // récupération du tableau de conditions linéaires en cours
363 //résultant de l'application des conditions lues (indépendamment du contact par exemple)
364 // mis à jour après la méthode: MiseAJour_condilinaire_tdt
365 const Tableau < Tableau <Condilinaire > >& Tab_CLLinApplique() const {return tab_CLLinApplique;};
366
367 // indique si oui ou non il y a des conditions limites linéaires en entrée
368 bool ExisteCondiLimite() const {return tab_iocondiline.Taille();};
369
370 // ----- fin conditions linéaires -----
371
372 //initialisation, mise a zero des sauvegarde
373 void InitSauve(const Nb_assemb& nb_casAssemb);
374

```

```

375 // lecture de donnée en fonction d'un indicateur : int type
376 // pour l'instant ne fait rien
377 void LectureDonneesExternes(UtilLecture& ,LesReferences& ,const int ,const string&) {};
378
379 // affichage et definition interactive des commandes pour les conditions limites CL
380 void Info_commande_LesCondLim1(UtilLecture & entreePrinc);
381
382 // affichage et definition interactive des commandes pour les conditions limites CLL
383 void Info_commande_LesCondLim2(UtilLecture & entreePrinc);
384
385 // affichage et definition interactive des commandes pour les initialisations
386 void Info_commande_LesCondLim3(UtilLecture & entreePrinc);
387
388 //----- lecture écriture de restart -----
389 // cas donne le niveau de la récupération
390 // = 1 : on récupère tout
391 // = 2 : on récupère uniquement les données variables (supposées comme telles)
392 void Lecture_base_info(ifstream& ent,const int cas,LesReferences& lesRef,LesCourbes1D&
LesCourbes1D
393                                     ,LesFonctions_nD& lesFonctionsnD);
394
395 // cas donne le niveau de sauvegarde
396 // = 1 : on sauvegarde tout
397 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
398 void Ecriture_base_info(ofstream& sort,const int cas);
399
400 //----- temps cpu -----
401 // retourne temps cumulé pour imposer les CL imposées
402 const Temps_CPU_HZpp& Temps_cpu_CL() const {return tempsCL;};
403 // retourne le temps cumulé pour imposer les CLL
404 const Temps_CPU_HZpp& Temps_cpu_CLL() const {return tempsCLL;};
405
406 // --- conteneur pour les réactions -----
407 class ReactStoc
408 { public :
409     // surcharge de l'operator de lecture typée
410     friend istream & operator » (istream &, ReactStoc &);
411     // surcharge de l'operator d'écriture typée
412     friend ostream & operator « (ostream &, const ReactStoc &);
413
414     //Surcharge d'opérateur logique
415     bool operator == ( const ReactStoc& a) const
416     { return ((numMail==a.numMail)&&(numNoeud==a.numNoeud)
417             &&(ddl==a.ddl)&&(casAss==a.casAss));
418     };
419     bool operator != ( const ReactStoc& a) const { return !(*this == a);};
420     // !*!*!* classement uniquement sur le numéro de noeud !*!*!*
421     // ---> non: changement 3/1/2018
422     bool operator < ( const ReactStoc& a) const
423     {if (numMail < a.numMail) return true;
424     else if (numMail == a.numMail) return (numNoeud < a.numNoeud);
425     else // cas où numMail > a.numMail
426     return false;
427     //return (numNoeud < a.numNoeud);
428     };
429     bool operator > ( const ReactStoc& a) const { return !(*this < a);};
430
431     public : int numMail; // numero de maillage
432             int numNoeud ; // numero de noeud
433             Ddl ddl; // le ddl bloque
434             int casAss; // cas d'assemblage
435     };
436
437 class TorseurReac
438 { // surcharge de l'operator de lecture typée
439   friend istream & operator » (istream &, TorseurReac &);
440   // surcharge de l'operator d'écriture typée
441   friend ostream & operator « (ostream &, const TorseurReac &);
442
443   public :
444   // constructeurs
445   TorseurReac():existe_torseur_reac(false),resultante(),moment(),bloque_ou_CLL(true) {};
446   TorseurReac(const TorseurReac& a):
447     existe_torseur_reac(a.existe_torseur_reac),resultante(a.resultante)
448     ,moment(a.moment),bloque_ou_CLL(a.bloque_ou_CLL) {};
449   // méthodes
450   void Activation(int dima) {existe_torseur_reac=true;resultante.Change_dim(dima);
451     moment.Change_dim(dima);};
452   void Zero_init_torseur() {resultante.Zero();moment.Zero();};
453
454   // données
455   bool existe_torseur_reac; // indique si le Torseur doit-être calculé ou non
456   Coordonnee resultante; // la résultante du torseur
457   Coordonnee moment; // le moment du torseur par rapport à l'origine
458   bool bloque_ou_CLL; // true provient d'un ddl bloqué, false provient d'une CLL
459   };
460

```

```

461 // retourne les réactions aux conditions limites à un noeud noe d'un maillage mail
462 // s'il n'y a pas de conditions limites : retour d'une liste vide
463 // const list<const LesCondLim::ReactStoc*>& Reaction_noeud_mail(int mail,int noeud) const ;
464
465 // retourne la liste des types de ddl actuellement imposés
466 // aux noeuds pour chaque maillage, la liste est exhaustive
467 // elle contient tous les types au moins une fois utilisée
468 // cela ne signifie pas que le ddl en question soit présent
469 // pour tous les noeud du maillage considéré
470 // - le tableau de retour est indicé par le numéro de maillage correspondant
471 // - il y a une liste par maillage
472 Tableau <List_io < Ddl_enum_etendu> > Les_type_de_ddl_en_reaction() const;
473
474 // retourne la liste des torseurs de réaction actuellement actif pour les maillages
475 // appelons tab_lili la liste:
476 // - tab_lili(i) concerne le maillage nb i
477 // - les éléments de tab_lili(i) sont les noms des références qui conduisent à un torseur de
réaction
478 // actif, et un indice de gestion utilisé par LesCondLim pour
retrouver
479 // rapidement les informations
480 Tableau < List_io <String_et_entier > > TabListTorseurReaction(const LesMaillages & lesmail) const;
481 // retourne le torseur de réaction correspondant à l'indice i
482 const TorseurReac & Torseur_de_reaction(int i) const {return tab_torseurReac(i);};
483
484 private :
485 // VARIABLES PROTEGEES :
486 Tableau <DdlLim> tabBloq; // tableau des ddl bloqués
487 Tableau <TorseurReac> tab_torseurReac; // tableau des torseurs de réaction
488 // relatif aux ddl bloqués et aux conditions linéaires
489 // de 1 à ttrG.Taille() on a les torseurs relatifs à tabBloq
490 // puis de ttrG.Taille() à la fin c-a-d ttrG.Taille()+ttrGCLL.Taille(),
491 // on a les torseurs relatifs à tab_iocondiline
492
493 // un tableau, qui permet de passer de tabBloq à tab_torseurReac
494 Tableau <int> ttrG; // tabBloq(i) contribue au torseur tab_torseurReac(ttrG(i))
495 // ttorBloq(i) contient tous les numéros de tabBloq qui contribue au torseur i
496 Tableau < List_io <int> > ttorBloq;
497 Tableau <DdlLim> tabInit; // tableau des ddl d'initialisation
498 Tableau <ReactStoc> reaction; // les reactions pour les ddl bloqués
499 Tableau <ReactStoc> reaction_CLin; // les reactions pour les conditions linéaires
500 Tableau <ReactStoc> reactionApresCHreperere; // les reactions après CHreperere
501
502 // tableau des conditions linéaires a imposer par les données d'entrée
503 Tableau <I_O_Condilineaire > tab_iocondiline;
504 Tableau < Tableau <Condilineaire > > tab_CLinApplique; // tableau des conditions linéaires
505 //résultant de l'application de tab_iocondiline
506
507 // un tableau à deux dim, qui permet de passer de tab_CLinApplique à tab_torseurReac
508 Tableau < Tableau <int> > ttrGCLL; // tab_CLinApplique(i)(j) contribue au torseur
509 tab_torseurReac(ttrGCLL(i)(j))
510 // ttorCLL(i) contient tous les numéros de tab_iocondiline qui contribue au torseur i, avec
511 // le premier élément de l'instance DeuxEntiers = le numéros dans tab_iocondiline, le second
512 // élément = le rang de
513 // la référence dans la condition linéaire
514 Tableau < List_io <DeuxEntiers> > ttorCLL;
515
516 //deux liste de noms de ref qui servent pour accéder aux grandeurs globales correspondantes
517 // aux composantes des torseurs de réaction
518 // voir: DimensionneTorseurs() :
519 // ttrG_noms_ref + _Re_ + i -> ième composante de Re a tdt
520 // et à la place de Re -> Mo pour le moment
521 Tableau <string> ttrG_noms_ref; // nom générique associé aux torseurs de ddl bloqué
522 Tableau <string> ttrGCLL_noms_ref; // idem pour les CLL
523
524 // une map pour faire l'association entre numéro de maillage et numéro de noeud d'une part
525 // et le tableau de réaction pour un accès quasi-directe
526 // map < string, list<const ReactStoc*> , std::less <string> > map_reaction;
527 int nb_maillage; // sauvegarde du nombre de maillage
528
529 Tableau < CondLim > condlim; // les fonctions et données qui permettent d'imposer les cl
530 // aux matrices, aux second membres, de remonter aux efforts apres
531 // resolution etc ..., le tableau est indicé par le numéro de cas d'assemblage
532
533 //----- temps cpu -----
534 Temps_CPU_HZpp tempsCL; // temps cumulé pour imposer les CL imposées
535 Temps_CPU_HZpp tempsCLL; // temps cumulé pour imposer les CLL
536
537 // ----fonctions internes pour simplifier
538
539 // cas de l'initialisation: prise en compte de l'appartenance à un groupe
540 // fixe : donne le type d'initialisation que l'on veut
541 // une_variable: indique si le ddl est une variable (=true) ou une donnée (=false)
542 void Init_appart_groupe(bool une_variable,Enum_ddl a,bool fixe,Noeud& noe, int cas,bool choix);
543 //cas courant: mise en place du blocage sur un ddl ou un groupe si besoin est
544 void Mise_cl(int nd,DdlLim& ddl,Enum_ddl a,Noeud& noe, int cas);
545 // cas courant : on retire le blocage sur un ddl ou un groupe si besoin est

```

```

543 void Retire_cl(int nd,DdlLim& ddbloqu,Enum_ddl enu,Noeud& noe, int cas);
544 // vérification qu'il n'y a pas de surcharge de blocage
545 // choix indique si l'on vérifie à t ou à tdt
546 void Verif_surcharge_blocage(const LesMaillages * lesMail,const LesReferences* lesRef,const double&
temps,int cas) ;
547 // fourni la liste des ddl associé de même dim, pour un du groupe et en fonction du cas
d'association
548 // ramène une référence sur tab_travail qui est mis à jour
549 Tableau <Ddl>& Ddl_associe( Ddl& ddl,Noeud& noe,Tableau <Ddl>& tab_travail,int cas);
550 // cas particulier des mouvements solides, dans le second passage pour la méthode MiseAJour_tdt
551 void MiseAJour_tdt_second_passage_mvtSolide(const double& mult_gene,LesMaillages * lesMail
552 ,const double& deltat,int cas
553 ,LesCourbes1D* lesCourbes1D,LesFonctions_nD* lesFonctionsnD,const double&
coef
554 ,DdlLim& tabBloq_i,const ReferenceNE & ref,const double& temps,Enum_ddl
en_ddl);
555 // mise à jour et calcul éventuel du torseur de réaction: uniquement pour les ddl X1, X2, X3
556 // --> pour au final obtenir le torseur des efforts globaux résultant de l'application d'un DdlLim
557 void CalculTorseurReaction(TorseurReac& tr, const Noeud& noe, ReactStoc& reac);
558
559 // récupération de valeurs à un noeud pour les grandeur enu
560 // ici il s'agit de grandeur scalaires
561 Tableau <double> Valeur_multi_interpoler_ou_calculer
562 (const Noeud& noe, Enum_dure temps,const List_io<Ddl_enum_etendu>& enu);
563 // récupération de valeurs à un noeud pour les grandeur enu
564 // ici il s'agit de grandeurs tensorielles
565 void Valeurs_Tensorielles_interpoler_ou_calculer
566 (const Noeud& noe, Enum_dure temps,List_io<TypeQuelconque>& enu);
567 // dimensionnement du tableau des torseurs ainsi que des tableaux tTRG et ttorBloq
568 // en fonction de tabBloq (utilisé plusieurs fois), puis de tTRGCLL en fonction de tab_iocondiline
569 void DimensionneTorseurs();
570
571 // DdlLim à un noeud: mise à jour des temps min et temps max, lorsque ceci dépendent de fonction nD
572 void Mise_a_jour_t_minmax_ddlLim(Noeud& noe,DdlLim& ddlLim,LesFonctions_nD* lesFonctionsnD);
573 // DdlLim général: mise à jour des temps min et temps max, lorsque ceci dépendent de fonction nD
574 // ici les fct nD doivent dépendre uniquement de grandeur générale
575 void Mise_a_jour_t_minmax_ddlLim(DdlLim& ddlLim,LesFonctions_nD* lesFonctionsnD);
576
577 };
578 /// @} // end of group
579
580
581 #endif

```

## 7.351 LesMaillages.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 * DATE: 23/01/97 *
32 * * $ *
33 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
34 * * $ *
35 * PROJET: Herezh++ *
36 * * $ *
37 *****/
38 * BUT: definir le groupe de maillage *
39 * * $ *

```

```

40 *      *
41 *      VERIFICATION: *
42 * * *
43 *      ! date ! auteur ! but ! *
44 *      ----- *
45 *      ! ! ! ! *
46 *      $ *
47 *      *
48 *      MODIFICATIONS: *
49 *      ! date ! auteur ! but ! *
50 *      ----- *
51 *      $ *
52 *****/
53 #ifndef LESMAILLAGES_H
54 #define LESMAILLAGES_H
55
56 #include "Maillage.h"
57 #include "Tableau_T.h"
58 #include "UtilLecture.h"
59 #include "ParaGlob.h"
60 #include "LesReferences.h"
61 // #include "bool.h"
62 #include "DiversStockage.h"
63 #include "LesLoisDeComp.h"
64 #include "Front.h"
65 #include "Enum_dure.h"
66 #include "Nb_assemb.h"
67 #include "Basiques.h"
68 #include "Enum_dure.h"
69 #include "TypeQuelconque.h"
70 #include "Enum_IO_XML.h"
71
72 /** @defgroup Les_Maillages Les_Maillages
73 *
74 *      BUT: definir le groupe de maillage *
75 *
76 *      \author Gérard Rio
77 *      \version 1.0
78 *      \date 23/01/97
79 *      \brief Définition du groupe de maillage
80 *
81 */
82
83 /// @addtogroup Les_Maillages
84 /// @{
85 ///
86
87 //-----
88 //! LesMaillages: l'ensemble des maillages
89 //-----
90 /// \author Gérard Rio
91 /// \version 1.0
92 /// \date 23/01/97
93
94 class LesMaillages
95 {
96 public :
97     // VARIABLES PUBLIQUES :
98
99     // CONSTRUCTEURS : le pointeur sur UtilLecture permet d'avoir acces a la lecture
100     // le second pointe sur les variables globales
101     // le troisieme pointe sur l' instance des references
102     LesMaillages (UtilLecture *, ParaGlob *, LesReferences* lesRef);
103     // constructeur par défaut
104     LesMaillages();
105     // Constructeur de copie,
106     LesMaillages(const LesMaillages& lesmail);
107     // DESTRUCTEUR :
108     ~LesMaillages();
109     // METHODES PUBLIQUES :
110     // lecture des maillages et des references s'y rapportant
111     void LectureLesMaillages();
112     // création et ajout d'un nouveau maillage en fonction d'un nom et d'une liste
113     // d'éléments
114     // *** il n'y a pas de création de nouveaux noeuds et de nouveaux éléments,
115     // ce sont les éléments et noeuds passés en paramètres qui sont ceux du maillage créé
116     // » ramène le numéro du nouveau maillage
117     int Creation_nouveau_maillage
118         (list <Noeud*>& li_noeud, list <Element*>& list_elem, const string& nom_maillage);
119     // suppression d'un maillage existant
120     // par défaut, tous les noeuds et éléments du maillage sont supprimés
121     // si sans_conservation_noeuds_elements est false: les noeuds et les éléments ne sont pas supprimés
122     // mais ils ne sont plus référencés dans ce maillage !
123     void Suppression_maillage( const string& nom_maillage, const bool sans_conservation_noeuds_elements =
124         true);

```



```

125     void Ajout_de_Noeuds(const list <Noeud *> & taN, int numMail)
126     {tabMaillage(numMail)->Ajout_de_Noeuds(taN)};
127     // -- ajout de noeuds, éléments et éventuellement de ref à un maillage
128     // ajout d'une liste d'éléments et de noeud à un maillage
129     // si le numéro de maillage associé à l'élément ou noeud est nul, il est remplacé par celui du
maillage
130     // si le numéro de maillage est déjà existant et est différent ce celui de this, il y a
131     // création d'un nouvel item identique, avec le numéro this
132     // ajout éventuel d'une liste de références associées , si celle-ci est non-nulle
133     // il y a création de nouvelles ref correspondantes au numéro de maillage de this
134     // et ces références sont rajoutées à lesRef
135     // les noeuds qui sont associés aux éléments de taE, doivent faire partie : soit de taN, soit du
maillage this
136     void Ajout_elements_et_noeuds(const list <Noeud *> & taN, const list <Element *> & taE, list <const
Reference*>* lref, LesReferences* lesRef, int numMail )
137     {tabMaillage(numMail)->Ajout_elements_et_noeuds(taN,taE,lref,lesRef)};
138
139     // affichage et definition interactive des commandes
140     void Info_commande_lesMaillages();
141
142     // Affiche les donnees du maillage
143     void Affiche () const ;
144     // Affiche les donnees du maillage dans le fichier de nom nom_fichier
145     // au format du fichier ".her"
146     void Affiche (char* nom_fichier) const ;
147
148     // test si toutes les informations des maillages sont completes
149     // = true -> complet
150     // = false -> incomplet
151     bool Complet();
152
153     // introduction des lois de comportement dans les elements qui le necessite
154     // des sections pour les biellettes etc c-a-d , completer les elements
155     // avec les donnees qui ont ete acquises apres la lecture du maillage
156     // def des tableaux de ddl dans les noeuds
157     // def des pointeurs d'assemblage dans les noeuds
158     void Completer(DiversStockage* divers, LesLoisDeComp* lesLois
, LesFonctions_nD* lesFonctionsnD);
159
160
161     // mise à jour des repères d'anisotropie
162     void Mise_a_jour_repere_anisotropie(DiversStockage* divers, LesFonctions_nD* lesFonctionsnD);
163
164     // ramene le nombre de maillage
165     inline int NbMaillage() const
166     { return nbMaillageTotal};
167
168     // ramene le nombre d'element du maillage i
169     inline int Nombre_element(int i) const
170     { return tabMaillage(i)->Nombre_element()};
171
172     // ramene le nombre de noeud du maillage i
173     inline int Nombre_noeud(int i) const
174     { return tabMaillage(i)->Nombre_noeud()};
175
176     // ramene l'element j du maillage i
177     inline Element& Element_LesMaille(int i, int j) const
178     {return tabMaillage(i)->Element_mail(j)};
179     // idem mais en version constant
180     inline const Element& Element_LesMaille_const(int i, int j) const
181     {return tabMaillage(i)->Element_mail_const(j)};
182
183     // ramene le noeud j du maillage i
184     inline Noeud& Noeud_LesMaille(int i, int j) const
185     {return tabMaillage(i)->Noeud_mail(j)};
186
187     // Retourne le tableau des noeuds du maillage i
188     inline Tableau<Noeud *>& Tab_noeud (int i)
189     { return tabMaillage(i)->Tab_noeud(); };
190
191     // Retourne le nom du maillage i
192     string NomMaillage(int i) const {return tabMaillage(i)->NomDuMaillage()};
193     // retourne le numéro du maillage de nom donné
194     // ou 0 si le nom ne correspond pas à un maillage
195     int NumMaillage(const string& nom) const
196     {map < string, int , std::less <string> >::const_iterator itmap = mapNomMail.find(nom);
197     if (itmap!=mapNomMail.end()) {return (*itmap).second;}
198     else {return 0};
199     };
200
201     // suppression éventuelle des noeuds, non référencés par les éléments et les références
202     // dans tous les maillages
203     void SuppressionNoeudNonReferencer(LesReferences& lesRef)
204     { for (int imail=1;imail<=nbMaillageTotal;imail++)
tabMaillage(imail)->SuppressionNoeudNonReferencer(lesRef)};
205     // Affichage des noeuds, non référencés par les éléments
206     void AffichageNoeudNonReferencer()
207     { for (int imail=1;imail<=nbMaillageTotal;imail++)

```

```

tabMaillage(imail)->AffichageNoeudNonReferencer();};
208
209 // renumérotation des noeuds de tous les maillages en même temps, + prise en compte
210 // des conditions linéaires qui existent entre les noeuds
211 // ramène false si rien n'a changé (à cause d'un pb ou parce que la renumérotation n'est pas
meilleure), vrai sinon
212 // si le pointeur d'assemblage est non nulle, cela veut dire que l'on veut également une mise à jour
213 // globale des pointeurs d'assemblages (ce qui est différent de la méthode :
MiseAJourPointeurAssemblage(
214 // qui agit maillage après maillage)
215 // si le pointeur d'assemblage est non nulle et le drapeau: sans_changement_num_noeud = true
216 // cela signifie que l'on désire uniquement une renumérotation de pointeur sans les noeuds
217 // ramène dans tous les cas les nouvelles largeurs en ddl
218 // nouvelles_largeur_en_ddl.un = la largeur totale résultante
219 // nouvelles_largeur_en_ddl.deux = la demie largeur totale résultante
220 // nouvelles_largeur_en_ddl.trois = la demie largeur maximale pour la partie éléments finis
221 // uniquement (sans les CLL)
222 //
223 bool Renumerotation(LesReferences& lesRef,const Tableau <Tableau <Condilinaire> >& condCLL
224 ,TroisEntiers& nouvelles_largeur_en_ddl,const Nb_assemb* nb_casAssemb = NULL
225 ,bool sans_changement_num_noeud = false);
226
227 // renumérotation des noeuds maillages par maillage,
228 // === sans prise en compte de conditions linéaires ===
229 // en sortie les maillages sont mis à jour si la nouvelle numérotation conduit à une largeur de
bande
230 // plus faible que la largeur initiale: en noeuds
231 // ramène: false si rien n'a changé (à cause d'un pb ou parce que la renumérotation n'est pas
meilleure)
232 // vrai sinon
233 bool Renumerotation(LesReferences& lesRef);
234
235 // création éventuelle d'une référence sur les noeuds, non référencés par les éléments
236 // dans tous les maillages
237 void CreationRefNoeudNonReferencer(LesReferences& lesRef)
238 { for (int imail=1;imail<=nbMaillageTotal;imail++)
tabMaillage(imail)->CreationRefNoeudNonReferencer(lesRef);};
239
240 // ramène le numéro du noeud le plus proche du point donné pour t=0,
241 // ceci pour le maillage i, par défaut i vaut 1
242 inline int Noeud_le_plus_proche_0(const Coordonnee& M,int i = 1)
243 { return tabMaillage(i)->Noeud_le_plus_proche_0(M); };
244 // idem à t
245 inline int Noeud_le_plus_proche_t(const Coordonnee& M,int i = 1)
246 { return tabMaillage(i)->Noeud_le_plus_proche_t(M); };
247 // idem à tdt
248 inline int Noeud_le_plus_proche_tdt(const Coordonnee& M,int i = 1)
249 { return tabMaillage(i)->Noeud_le_plus_proche_tdt(M); };
250
251 // ramène le numéro de l'élément qui contient un point donné et le numéro du point
252 // d'intégration le plus proche pour les ddl de la liste (ddl spécifique à l'élément c'est-à-dire
253 // hors des ddl des noeuds de l'éléments)
254 // ceci pour le maillage i, (par défaut i vaut 1)
255 // si pas de numéro d'élément ramène un numéro d'élément nulle
256 // si les numéros de point d'intégration ne sont pas identique pour l'ensemble
257 // des ddl, pb !!, le numéro du pt integ de retour est alors négatif
258 // si pb ramène un numéro d'élément nulle
259 // enu_temps: dit si les coordonnées du point M sont à 0 ou t ou tdt
260 inline Maillage::NBelemEtptInteg Element_le_plus_proche
261 (Enum_dure enu_temps,const List_io <Ddl_enum_etendu>& list_enu,const Coordonnee& M,int i = 1)
262 { return tabMaillage(i)->Element_le_plus_proche(enu_temps,list_enu,M); };
263
264 // ramène la liste des problèmes physiques gérés par les éléments de tous les maillages
265 inline const list <Enum_ddl >& Ddl_representatifs_des_physiques()const
266 {return ddl_representatifs_des_physiques;};
267
268 // ramene la liste des degrés de liberté inconnus, associés aux pb
269 // physiques gérés par les éléments qui existent dans tous les maillages
270 // Si éléments mécaniques -> ddl Xi voir Vi et gamma_i
271 // Si éléments thermiques -> ddl de température
272 // Si éléments méca + éléments thermiques -> ddl Xi et température
273 // etc. en fonction des éléments qui existent dans les maillages
274 inline const list <EnumElemTypeProblem >& Types_de_problemes() const
275 {return types_de_problemes;};
276
277
278 // ramene le nombre total de ddl actifs du pb
279 int NbTotalDdlActifs() const ;
280 // idem mais pour un type de ddl donné, dans le cas de type
281 // vectoriel on cumule les ddl de l'ensemble de la dimension
282 int NbTotalDdlActifs(Enum_ddl enum_ddl) const;
283
284 // ramene le nombre total de points d'intégration correspondant à un ddl donné
285 int NbTotalPtInteg(Enum_ddl enum_ddl) const ;
286
287 // ramene le nombre total de grandeurs génératrices, calculées aux points d'intégrations,
288 // correspondant à un ddl donné.

```

```

289     int NbTotalGrandeursGeneratrices(Enum_ddl enu) const ;
290
291     //récupération d'une grandeur vectoriel de dimension, la dimension
292     // de l'espace, défini au noeud et transféré dans un vecteur global
293     // qui cumule de manière séquentielle toutes les grandeurs
294     // en entrée : enum_ddl donne le type de la grandeur à récupérer
295     // en fait de la première composante
296     // duree : indique à quel temps doit s'effectuer le transfert, t=0 ou t ou tdt
297     // enum_actif : le transfert s'effectue que si le ddl enum_actif est actif
298     // ce qui permet la différentiation entre les différents ddl
299     // vect : est le vecteur global de stockage qui normalement a été
300     // au préalable dimensionné avec NbTotalDdlActifs(Enum_ddl enum_ddl)
301     // en retour : une référence sur vect
302     Vecteur & Vect_loc_vers_glob(Enum_duree duree,Enum_ddl enum_actif
303     ,Vecteur& vect,Enum_ddl enum_ddl);
304
305     // fonction inverse de Vect_loc_vers_glob, il s'agit ici de passer
306     // de la grandeur globale aux grandeurs locale
307     void Vect_glob_vers_local(Enum_duree duree,Enum_ddl enum_actif
308     ,const Vecteur& vect,Enum_ddl enum_ddl) ;
309
310
311     //récupération d'une grandeur vectoriel de dimension, la dimension
312     // de l'espace, défini au noeud et transféré dans un vecteur global
313     // qui cumule de manière séquentielle toutes les grandeurs
314     // en entrée : tab_enum_ddl donne le tableau des type de la grandeur à récupérer
315     // en fait de la première composante
316     // duree : indique à quel temps doit s'effectuer le transfert, t=0 ou t ou tdt
317     // tab_enum_actif : pour chaque élément tab_enum_ddl(i), le transfert s'effectue que si
318     // le ddl tab_enum_actif(i) est actif
319     // ce qui permet la différentiation entre les différents ddl
320     // vect : est le vecteur global de stockage qui normalement a été
321     // au préalable dimensionné avec somme des NbTotalDdlActifs(Enum_ddl enum_ddl), avec enum_ddl
322     // qui balaie l'ensemble des éléments de tab_enum_ddl
323     // Important: pour chaque famille de ddl, les ddl sont classés de manière croissante, ce qui
324     // signifie
325     // que c'est l'ordre des pointeurs d'assemblage si et seulement si, ces ddl ont été rangés dans les
326     // noeuds
327     // au préalable
328     // en retour : une référence sur vect
329     Vecteur & Vect_loc_vers_glob(Enum_duree duree,const Tableau <Enum_ddl>& tab_enum_actif
330     ,Vecteur& vect,const Tableau <Enum_ddl>& tab_enum_ddl);
331
332     // fonction inverse de Vect_loc_vers_glob, il s'agit ici de passer
333     // de la grandeur globale aux grandeurs locale
334     void Vect_glob_vers_local(Enum_duree duree,const Tableau <Enum_ddl>& tab_enum_actif
335     ,const Vecteur& vect,const Tableau <Enum_ddl>& tab_enum_ddl) ;
336
337     //récupération d'une grandeur scalaire
338     // défini au noeud et transféré dans un vecteur global
339     // qui cumule de manière séquentielle toutes les grandeurs
340     // en entrée : enum_ddl donne le type de la grandeur à récupérer
341     // duree : indique à quel temps doit s'effectuer le transfert, t=0 ou t ou tdt
342     // enum_actif : le transfert s'effectue que si le ddl enum_actif est actif
343     // ce qui permet la différentiation entre les différents ddl
344     // vect : est le vecteur global de stockage qui normalement a été
345     // au préalable dimensionné avec NbTotalDdlActifs(Enum_ddl enum_ddl)
346     // en retour : une référence sur vect
347     Vecteur & Scalaire_loc_vers_glob(Enum_duree duree,Enum_ddl enum_actif
348     ,Vecteur& vect,Enum_ddl enum_ddl);
349
350     // fonction inverse de Scalaire_loc_vers_glob, il s'agit ici de passer
351     // de la grandeur globale à la grandeur locale
352     void Scalaire_glob_vers_local(Enum_duree duree,Enum_ddl enum_actif
353     ,Vecteur& vect,Enum_ddl enum_ddl);
354
355     // deux fonctions idem que pour les ddl : mais pour un Ddl_etendu
356     Vecteur & Scalaire_loc_vers_glob(Enum_ddl enum_actif,Vecteur& vect,const Ddl_enum_etendu&
357     enum_ddl_etendu);
358     void Scalaire_glob_vers_local(Enum_ddl enum_actif,Vecteur& vect,const Ddl_enum_etendu&
359     enum_ddl_etendu);
360
361     // deux fonctions idem que pour les ddl_etendu : mais pour une grandeur quelconque
362     Vecteur & Quelconque_loc_vers_glob(Enum_ddl enum_actif,Vecteur& vect,const TypeQuelconque&
363     type_generique);
364     void Quelconque_glob_vers_local(Enum_ddl enum_actif,Vecteur& vect,const TypeQuelconque&
365     type_generique);
366
367     // retourne la liste des types de ddl principaux actuellement utilisé
368     // aux noeuds pour chaque maillage, la liste est exhaustive
369     // elle contient tous les types au moins une fois utilisée
370     // cela ne signifie pas que le ddl en question soit présent
371     // pour tous les noeud du maillage considéré
372     // - le tableau de retour est indicé par le numéro de maillage correspondant
373     // - il y a une liste par maillage
374     Tableau <List_io <Ddl_enum_etendu> > Les_type_de_ddl_par_noeud(bool absolue);
375
376

```

```

370 // retourne la liste des types de ddl étendu actuellement utilisé
371 // aux noeuds pour chaque maillage, la liste est exhaustive
372 // elle contient tous les types au moins une fois utilisée
373 // cela ne signifie pas que le ddl en question soit présent
374 // pour tous les noeud du maillage considéré
375 // - le tableau de retour est indicé par le numéro de maillage correspondant
376 // - il y a une liste par maillage
377 Tableau <List_io <Ddl_enum_etendu> > Les_type_de_ddl_etendu_par_noeud(bool absolue);
378
379 // retourne la liste des types quelconque actuellement utilisé
380 // aux noeuds pour chaque maillage, la liste est exhaustive
381 // elle contient tous les types au moins une fois utilisée
382 // cela ne signifie pas que le ddl en question soit présent
383 // pour tous les noeud du maillage considéré
384 // - le tableau de retour est indicé par le numéro de maillage correspondant
385 // - il y a une liste par maillage
386 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
387 Tableau <List_io <TypeQuelconque> > Les_type_de_TypeQuelconque_par_noeud(bool absolue);
388
389 // initialisation par défaut de tous les conteneurs aux noeuds
390 // de tous les maillages
391 // contenant li_restreinte_TQ
392 // ces conteneurs sont supposés déjà existés
393 // typiquement si le conteneurs est un scalaire, on met 0
394 void Init_par_defaut_conteneurs(List_io < TypeQuelconque >& li_restreinte_TQ);
395 // idem pour une seule grandeur
396 void Init_par_defaut_conteneurs(TypeQuelconque_enum_etendu enuTypeQuelconque);
397
398 // intro de certains conteneurs internes en relation par exemple avec les demandes de visualisation
399 // ou autre
400 // exemple: si VECT_REAC_N qui est un type quelconque, a été choisit
401 // il faut qu'il soit présent aux noeuds, alors qu'il est alimenté par les ddl pur ...
402 // on introduit donc le type quelconque associé
403 // NB: le conteneur passé en paramètre ne sert que pour localiser les grandeurs
404 void Intro_Conteneurs_internes_noeud_relier_auto_autres_grandeur
405     ( const List_io < TypeQuelconque > & glob_noeud_evol_retenu);
406
407 // idem mais ciblé en fonction d'un tableau indicé sur les maillages
408 void Intro_Conteneurs_internes_noeud_relier_auto_autres_grandeur
409     ( const Tableau < List_io < TypeQuelconque > > & tab_noeud_evol_retenu);
410
411 // retourne la liste des types de ddl actuellement utilisé
412 // aux éléments pour chaque maillage, la liste est exhaustive
413 // elle contient tous les types au moins une fois utilisée
414 // cela ne signifie pas que le ddl en question soit présent
415 // pour tous les éléments du maillage considéré
416 // - le tableau de retour est indicé par le numéro de maillage correspondant
417 // - il y a une liste par maillage
418 Tableau <List_io <Ddl_enum_etendu> > Les_type_de_ddl_par_element(bool absolue);
419 // idem pour les grandeurs évoluées interne actuellement utilisés
420 // par les éléments, c'est-à-dire comme les ddl mais directement sous forme de vecteur, tenseurs ...
421 Tableau <List_io <TypeQuelconque> > Les_type_de_donnees_evolues_internes_par_element(bool
absolue);
422 // idem pour les grandeurs particulières
423 Tableau <List_io <TypeQuelconque> > Les_type_de_donnees_particulieres_par_element(bool absolue);
424
425 // retourne la liste des types de grandeur quelconque actuellement utilisé
426 // aux faces élément pour chaque maillage, la liste est exhaustive
427 // elle contient tous les types au moins une fois utilisée
428 // cela ne signifie pas que la grandeur en question soit présent
429 // pour tous les éléments du maillage considéré
430 // - le tableau de retour est indicé par le numéro de maillage correspondant
431 // - il y a une liste par maillage
432 Tableau <List_io <TypeQuelconque> > Les_type_de_donnees_evolues_internes_par_face_element(bool
absolue);
433 // idem pour les arêtes d'élément
434 Tableau <List_io <TypeQuelconque> > Les_type_de_donnees_evolues_internes_par_arete_element(bool
absolue);
435
436 // Au niveau des noeuds: transfert des coordonnées de grandeurs vectorielles à des grandeurs
évoluées
437 // stockée sous forme de grandeurs TypeQuelconque
438 // exemple: les réactions qui sont naturellement stockée en composantes
439 // le principe est que ce passage s'effectue si les conteneurs existent au niveau des noeuds
440 void PassageInterneDansNoeud_composantes_vers_vectorielles();
441
442 // ..... transfert de grandeurs des points d'intégration aux noeuds .....
443 // 1- en entrée les type évoluées et les types particuliers que l'on veut transférer
444 // 2- en entrée: cas qui indique la méthode de transfert à utiliser
445 // =1 : les valeurs aux noeuds sont obtenue par moyennage des valeurs des pts
d'integ les plus près
446 // des éléments qui entourent le noeud
447 // on décompose le processus en 4 étapes pour éviter d'initialiser plusieurs fois lorsque l'on
refait à chaque fois
448 // le même transfert
449 // les méthodes: AjoutConteneurAuNoeud, InitUpdateAuNoeud, sont générales, peuvent être utilisés
pour autres choses

```

```

450
451
452 // A) première étape def des conteneurs et c'est tout, la méthode peut donc être utilisée
453 // pour autre chose. tabQ: permet d'avoir plusieurs listes de TypeQuelconque
454 // en entrée: tabQ doit-être de dimension 2, donc pointe sur 2 listes, si un des pointeur
455 // est nulle on n'en tient pas compte
456 void AjoutConteneurAuNoeud(int num_maillage,const List_io < Ddl_enum_etendu >& lienu
457                          ,const Tableau <List_io < TypeQuelconque > * >& tabQ)
458 { tabMaillage(num_maillage)->AjoutConteneurAuNoeud(lienu,tabQ)};
459 // fonctions utilitaires du même genre pour tous les maillages
460 // ajout sur tous les maillages d'un ou plusieurs ddl_enum_etendu comme conteneur
461 // en entrée: tabQ doit-être de dimension 2, donc pointée sur 2 listes, si un des pointeur
462 // est nulle on n'en tient pas compte
463 void AjoutConteneurAuNoeud(const List_io < Ddl_enum_etendu >& lienu
464                          ,const Tableau <List_io < TypeQuelconque > * >& tabQ)
465 { for (int imail=1;imail<=nbMaillageTotal;imail++)
466   tabMaillage(imail)->AjoutConteneurAuNoeud(lienu,tabQ)};
467
468 // B) initialisation des updates sur les noeuds
469 // lorsque l'on a des listes différentes pour chaque maillage on peut directement utiliser la
routine de Maillage
470 void InitUpdateAuNoeud(const List_io < Ddl_enum_etendu >& lienu
471                      ,const Tableau <List_io < TypeQuelconque > * >& tabQ,int cas)
472 { for (int imail=1;imail<=nbMaillageTotal;imail++)
tabMaillage(imail)->InitUpdateAuNoeud(lienu,tabQ,cas)};
473 void InitUpdateAuNoeud(int numMail,const List_io < Ddl_enum_etendu >& lienu
474                      ,const Tableau <List_io < TypeQuelconque > * >& tabQ,int cas)
475 { tabMaillage(numMail)->InitUpdateAuNoeud(lienu,tabQ,cas)};
476 // C) exécution du transfert
477 // transfert de ddl des points d'intégrations (de tous) aux noeuds d'un éléments (on ajoute
aux noeuds, on ne remplace pas)
478 // les ddl doivent déjà exister aux noeuds sinon erreur
479 // il doit s'agir du même type de répartition de pt d'integ pour toutes les grandeurs
480 void TransfertPtIntegAuNoeud(int numMail,Element& ele,const List_io < Ddl_enum_etendu >& lietendu
481                             ,const Tableau <double> > & tab_val,int cas)
482 { tabMaillage(numMail)->TransfertPtIntegAuNoeud(ele,lietendu,tab_val,cas)};
483 // idem pour des grandeurs quelconques
484 // les informations sont ici contenues dans les types quelconques
485 // liQ_travail: est une liste de travail qui sera utilisée dans le transfert
486
487 void TransfertPtIntegAuNoeud(int numMail,Element& ele,const Tableau <List_io < TypeQuelconque >
488 >& tab_liQ
489                             ,List_io < TypeQuelconque > & liQ_travail,int cas)
490 { tabMaillage(numMail)->TransfertPtIntegAuNoeud(ele,tab_liQ,liQ_travail,cas)};
491 // D) dernière étape: (par exemple calcul des moyennes en chaque noeud)
492 void FinTransfertPtIntegAuNoeud(const List_io < Ddl_enum_etendu >& lienu
493                               ,const Tableau <List_io < TypeQuelconque > * >& tabQ,int cas)
494 { for (int i=1;i<=nbMaillageTotal;i++)
tabMaillage(i)->FinTransfertPtIntegAuNoeud(lienu,tabQ,cas)};
495 void FinTransfertPtIntegAuNoeud(int numMail,const List_io < Ddl_enum_etendu >& lienu
496                               ,const Tableau <List_io < TypeQuelconque > * >& tabQ,int cas)
497 { tabMaillage(numMail)->FinTransfertPtIntegAuNoeud(lienu,tabQ,cas)};
498 // ..... cumul et moyenne de grandeurs venant des éléments vers les noeuds (exemple la pression
appliquée) .....
499 // on décompose le processus en 4 étapes pour éviter d'initialiser plusieurs fois lorsque l'on
refait à chaque fois
500 // la même opération (typiquement à chaque incrément)
501 // on peut utiliser:
502 // A) AjoutConteneurAuNoeud : pour ajouter des conteneurs ad hoc aux noeuds
503 // B) InitUpdateElementAuNoeud: avant le cumul, initialise les conteneurs
504 // C) Accumul_aux_noeuds : balaie les éléments avec cumul aux noeuds, uniquement des grandeurs
gérées par l'élément
505 // D) MoyenneCompteurAuNoeud : effectue les moyennes aux noeuds
506
507 // accumulation aux noeuds de grandeurs venant de tous les éléments vers ses noeuds (exemple la
pression appliquée)
508 // autres que celles aux pti classiques, mais directement disponibles
509 // le contenu du conteneur stockées dans liQ est utilisé en variable intermédiaire
510 void Accumul_aux_noeuds(int numMail,const List_io < Ddl_enum_etendu >& lietendu
511                       ,List_io < TypeQuelconque > & liQ,int cas)
512 { tabMaillage(numMail)->Accumul_aux_noeuds(lietendu,liQ,cas)};
513
514 // fonctions utilitaires du même genre pour tous les maillages
515
516 // ajout sur tous les maillages d'un conteneur particulier quelconque
517 void AjoutConteneurAuNoeud(TypeQuelconque& tQ)
518 { for (int imail=1;imail<=nbMaillageTotal;imail++) tabMaillage(imail)->AjoutConteneurAuNoeud(tQ)};
519
520 // ajout sur tous les maillages d'un ou plusieurs ddl_enum_etendu comme conteneur
521 void AjoutConteneurAuNoeud(const List_io < Ddl_enum_etendu >& lienu)
522 { for (int imail=1;imail<=nbMaillageTotal;imail++)
tabMaillage(imail)->AjoutConteneurAuNoeud(lienu)};
523 // initialisation des updates de ddl_etendu uniquement sur les noeuds: on met à 0 les ddl_etendu
correspondant,
524 // les compteurs, comptant le nombre de fois où les noeuds sont modifiés, sont mis à 0

```

```

525 void InitUpdateAuNoeud(const List_io < Ddl_enum_etendu >& lienu)
526 { for (int imail=1;imail<=nbMaillageTotal;imail++)
tabMaillage(imail)->InitUpdateAuNoeud(lienu);};
527 // idem pour un seul ddl_etendu
528 void InitUpdateAuNoeud(const Ddl_enum_etendu & enu)
529 { for (int imail=1;imail<=nbMaillageTotal;imail++)
tabMaillage(imail)->InitUpdateAuNoeud(enu);};
530 // moyenne des valeurs aux noeuds (en fonction du nombre ou le noeud a été modifié)
531 void MoyenneCompteurAuNoeud(const Ddl_enum_etendu & enu)
532 { for (int imail=1;imail<=nbMaillageTotal;imail++)
tabMaillage(imail)->MoyenneCompteurAuNoeud(enu);};
533
534 // initialisation d'un ou de plusieurs nouveau cas d'assemblage
535 // ramène le numéro du premier nouveau cas d'assemblage
536 Nb_assemb InitNouveauCasAssemblage(int nb_cas);
537
538 // ramène le nombre total de cas d'assemblage actuellement pris en compte dans les
539 // maillages
540 int Nb_total_en_cours_de_cas_Assemblage() const {return tab_nb_assemb;};
541
542
543 // met a jour les pointeurs d'assemblage dans les noeuds pour un cas d'assemblage
544 // a effectuer si l'on a changer de nb de noeuds, de nb de ddl, de nb de maillage
545 // casAssemb : donne le cas d'assemblage qui est a considérer
546 void MiseAJourPointeurAssemblage(const Nb_assemb& nb_casAssemb);
547
548 // mise a zero de tous les ddl actifs autres que les déplacements
549 // si indic = false; pas de creation des tableaux a t+dt
550 // si indic = true; creation des tableaux a t+dt
551 // cas = true; les coordonnées à t et éventuellement à t+dt sont initialisées
552 // aux valeurs de t=0 (cas par défaut),
553 // sinon on ne les modifie pas.
554 void ZeroDdl(bool indic,bool cas = true);
555
556 // force la mise à une valeur d'un ddl (ou de la liste de ddl fonction de la dimension)
particulier, quelques soit son activité
557 // si fonction_de_la_dimension = true : c'est toute les ddl fct de la dimension qui sont mis à la
valeur
558 void Force_Ddl_aux_noeuds_a_une_valeur(Enum_ddl enu, const double& val,Enum_dure temps, bool
fonction_de_la_dimension)
559 { for (int imail=1;imail<=nbMaillageTotal;imail++)
560
tabMaillage(imail)->Force_Ddl_aux_noeuds_a_une_valeur(enu,val,temps,fonction_de_la_dimension);};
561 // mise à zéro de dd_enum_etendu aux noeuds : force la mise à une valeur à 0
562 void Force_Ddl_etendu_aux_noeuds_a_zero(const Tableau<Ddl_enum_etendu>& tab_enu)
563 { for (int imail=1;imail<=nbMaillageTotal;imail++)
564 tabMaillage(imail)->Force_Ddl_etendu_aux_noeuds_a_zero(tab_enu);};
565
566 // ramene la demi largeur de bande en ddl et la largeur de bande
567 // casAssemb : donne le cas d'assemblage qui est a considérer
568 void Largeur_Bande(int& demi, int& total,const Nb_assemb& nb_casAssemb);
569
570 // méthode permettant le calcul des matrices de connexion pour chaque
571 // élément ceci par rapport à la numérotation absolu des ddl
572 // casAssemb : donne le cas d'assemblage qui est a considérer
573 void Table_connexion
574 (Tableau < Tableau <int> >& petites_matricespetites_matrices
575 ,const Nb_assemb& nb_casAssemb) const;
576
577 // actualisation des ddl et des grandeurs actives de t+dt vers t
578 void TdtversT();
579 // actualisation des ddl et des grandeurs actives de t vers tdt
580 void TversTdt();
581
582 // actualisation des ddl a t+dt, a partir du resultat de la resolution
583 // casAssemb : donne le cas d'assemblage qui est a considérer
584 void PlusDelta_tdt(Vecteur& sol,const Nb_assemb& nb_casAssemb);
585 // actualisation des ddl actifs a t, a partir du resultat de la resolution
586 // casAssemb : donne le cas d'assemblage qui est a considérer
587 void PlusDelta_t(Vecteur& sol,const Nb_assemb& nb_casAssemb);
588 // récupération du vecteur correspondant à l'incrément de ddl entre t et tdt
589 // en paramètre le vecteur vide et en sortie le vecteur rempli
590 // casAssemb : donne le cas d'assemblage qui est a considérer
591 Vecteur& RecupDepde_tatdt(Vecteur& sol,const Nb_assemb& nb_casAssemb);
592 // changement des ddl à tdt par ceux correspondant au vecteur passé en paramètre
593 // casAssemb : donne le cas d'assemblage qui est a considérer
594 void ChangeDdla_tdt(Vecteur& sol,const Nb_assemb& nb_casAssemb);
595
596 // retrouver le ddl correspondant a un pointeur de position
597 // d'assemblage, le nb du noeud et du maillage
598 // insol = le pointeur d'assemblage;
599 // ddl = le ddl en sortie; a t+dt si elle il existe
600 // sinon la valeur a t
601 // casAssemb : donne le cas d'assemblage qui est a considérer
602 Ddl NoeudIndexe(int inSol,int& nbNoeud, int& nbMaillage
603 ,const Nb_assemb& nb_casAssemb);
604 // idem en ramenant en plus la valeur du ddl a 0

```

```

605 // casAssemb : donne le cas d'assemblage qui est a considérer
606 Ddl NoeudIndice(int inSol,int& nbNoeud, int& nbMaillage, double& val0
607                ,const Nb_assemb& nb_casAssemb);
608
609 // calcule des normales aux noeuds: dans le cas d'éléments 1D ou 2D uniquement
610 // a priori le calcul s'effectue par une moyenne des normales des éléments qui
611 // entourent le noeud.
612 // init -> calcul des normales à t=0
613 // et ajout conteneur aux noeuds des normales à t = 0 et t
614 void InitNormaleAuxNoeuds()
615     { for (int imail=1;imail<=nbMaillageTotal;imail++)
616         tabMaillage(imail)->InitNormaleAuxNoeuds();};
617 // mise à jour -> mise à jour des normales à t
618 void MiseAJourNormaleAuxNoeuds()
619     { for (int imail=1;imail<=nbMaillageTotal;imail++)
620         tabMaillage(imail)->MiseAJourNormaleAuxNoeuds();};
621 // mise à jour -> mise à jour des normales à t
622 // mais ici, on calcule les normales à tdt, et on transfère à t
623 // cette méthode est utile si on veut utiliser des normales à t pour une valeur
624 // particulière (transitoire) de la géométrie à tdt
625 // cf: l'algo non dyna par exemple
626 void MiseAJourNormaleAuxNoeuds_de_tdt_vers_T()
627     { for (int imail=1;imail<=nbMaillageTotal;imail++)
628         tabMaillage(imail)->MiseAJourNormaleAuxNoeuds_de_tdt_vers_T();};
629
630 // ----- particularité aux contacts -----
631 // creation des elements frontiere: cela n'a lieu qu'une seule fois
632 // si les frontieres existent déjà, --> aucune action, --> ramène 0
633 // sinon il y a réellement création, et --> ramène 1
634 int CreeElemFront();
635 // ramene le nombre de maillage esclave
636 inline int NbEsclave() { return domEsclave;};
637 // ramene le tableau des list des elements frontiere
638 inline Tableau <LaLIST <Front>*>& ListFrontiere() { return listFrontiere;};
639 // ramene un tableau des noeuds des frontieres des maillages esclaves
640 Tableau < Tableau <Noeud*> *> Tab_noeud_frontiere_esclave();
641 // ramene le tableau des noeuds des frontieres des maillages
642 const Tableau <Tableau <Noeud*> *>& Tab_noeud_frontiere() {return tt_noeud_front;};
643 // calcul et ramene le tableau de tous les noeuds des maillages esclaves
644 // noeuds de la frontiere et noeuds internes
645 // s'ils existent, sinon le tableau est vide
646 // le tableau n'est pas sauvegarde
647 const Tableau<Noeud *> Esclave();
648 // mise à jour des boites d'encombrements des éléments, qui contiennent des éléments frontières
649 // et des éléments frontières eux-même
650 void Mise_a_jour_boite_encombrement_elem_front(Enum_dure temps);
651 // crée et ramene pour tous les maillages, la liste des éléments qui contiennent chaque noeud
652 // mis à jour lorsque lors de la création des frontières
653 const Tableau < const Tableau <List_io < Element* > > *>& Indice() ;
654
655 // inactive tous les ddl et les données
656 void Inactive_ddl_et_donnees();
657 // inactive les ddl mais pas les données
658 void Inactive_ddl();
659
660 // inactive les ddl primaires
661 void Inactive_ddl_primaire();
662 // active les ddl primaires
663 void Active_ddl_primaire();
664 // introduction des ddl de contraintes si cela veut dire quelques chose
665 // pour l'élément
666 void Plus_ddl_Sigma();
667 // inactivation des ddls de contraintes si cela veut dire quelques chose
668 // pour l'élément
669 void Inactive_ddl_Sigma();
670 // activation des ddls de contraintes si cela veut dire quelques chose
671 // pour l'élément
672 void Active_ddl_Sigma();
673 // activation du premier ddl de contraintes si cela veut dire quelques chose
674 // pour l'élément
675 void Active_premier_ddl_Sigma();
676 // introduction des ddl d'erreur si cela veut dire quelques chose
677 // pour l'élément
678 void Plus_ddl_Erreur();
679 // inactivation des ddls d'erreur si cela veut dire quelques chose
680 // pour l'élément
681 void Inactive_ddl_Erreur();
682 // activation des ddls d'erreur si cela veut dire quelques chose
683 // pour l'élément
684 void Active_ddl_Erreur();
685
686 // d'une manière plus générique une fonction pour activer une série de ddl
687 // donnée par un identificateur, si c'est une grandeur vectoriel c'est l'ensemble
688 // des ddl du vecteur qui sont inactivés.
689 void Active_un_type_ddl_particulier(Enum_ddl en);
690 void Active_un_type_ddl_particulier(Tableau<Enum_ddl>& tab_en); // idem mais pour un tableau
691 void Active_un_type_ddl_particulier(const list<Enum_ddl>& list_en); // idem pour une liste

```



```

692 // idem la fonction Active_ddl_noeud mais ici pour l'inactivation
693 void Inactive_un_type_ddl_particulier(Enum_ddl en);
694 void Inactive_un_type_ddl_particulier(Tableau<Enum_ddl>& tab_en); // idem mais pour un tableau
695 void Inactive_un_type_ddl_particulier(const list <Enum_ddl>& list_en); // idem pour une liste
696 // -- encore plus générique
697 // changement de toutes les conditions données (service, variable, fixage ..)
698 // selon le tableau de ddl passé en paramètre
699 // par contre les valeurs de ta ne sont pas utilisé donc les valeurs actuelles restent inchangé
700 void ChangeToutesLesConditions(const Tableau<Ddl>& ta);
701 // changement de statu des ddl d'une combinaison, en fonction du statut
702 // de enuta dans chaque noeud, les ddl de la combinaison, prennent le même statut que celui
703 // de enuta dans chaque noeud.
704 // cas est la combinaison,
705 void ChangeStatut(int cas,Enum_ddl enuta);
706 // changement de statu des ddl d'une combinaison dans chaque noeud, en fonction
707 // de enubold, les ddl de la combinaison, prennent le même statut que enubold
708 // cas est la combinaison,
709 void ChangeStatut(int cas,Enum_boolddl enubold);
710
711 //change le statut de tous les ddl liés à la physique en cours
712 //par exemple: met à libre ou bloque les ddl liés à la physique en cours
713 void Libere_Ddl_representatifs_des_physiques(Enum_boolddl enubold);
714
715 // Calcul de l'erreur sur l'ensemble des éléments
716 // type indique le type d'erreur retenue
717 // type = 1 : cas d'un calcul aux moindres carrés
718 // et retour un tableau de tableau de grandeurs sur les maillages en cours
719 // ret(i) : concerne le maillage i
720 // ret(i)(1) : somme des erreurs sur l'ensemble des éléments: est homogène à
721 // un |delta contrainte| * domaine
722 // ret(i)(2) : somme de la grandeur de ref du calcul d'erreur sur l'ensemble des
723 // éléments: est homogène à une |contrainte| * domaine
724 // ret(i)(3) : le maxi pour les tous les éléments de |delta contrainte| * domaine
725 // ret(i)(4) : le maxi pour les tous les éléments de |contrainte| * domaine
726 Tableau <Tableau <double > > ErreurSurChaqueElement(int type);
727
728 // lecture de donnée en fonction d'un indicateur : type
729 // type = 1 , on lit les tenseurs de contraintes
730 void LectureDonneesExternes(const int type ,const string& nomMaillage);
731
732 // mise en place du travail à t sur les maillages
733 // indique que l'on va utiliser les ddl en 0, t
734 // si les grandeurs en tdt existaient, elles sont supprimées
735 void Travail_t();
736 // idem pour un maillage donné de numéro num
737 void Travail_t(int num);
738
739 // définition des coordonnées à t identiques à ceux à t=0,
740 void Insert_coordl();
741
742 // mise en place du travail à t tdt sur les maillages
743 // indique que l'on va utiliser les ddl en 0, t, tdt
744 void Travail_tdt();
745
746 // initialisation des coordonnées à t et tdt aux mêmes valeurs qu'à 0
747 // utile quand on veut utiliser les métriques pour un pb non couplés
748 void Init_Xi_t_et_tdt_de_0() { for (int imail=1;imail<=nbMaillageTotal;imail++)
749     tabMaillage(imail)->Init_Xi_t_et_tdt_de_0();};
750
751 // ramène le maximum de variation de coordonnée entre t et tdt de tous les noeuds du maillage
752 double Max_var_dep_t_a_tdt() const;
753
754 // ramène le minimum de la distance entre deux noeuds de l'ensemble des éléments pour tous les
755 // maillages
756 double Min_dist2Noeud_des_elements(Enum_dure temps) const ;
757
758 // indique aux éléments un niveau de précision de calcul désiré pour les prochains calculs
759 // precision = 0 : aucune précision demandée, precision >=0 : précision maximale demandée
760 void Drapeau_preparation_calcul_precis(int precision);
761
762 // ----- calcul dynamique -----
763 // ajout des ddl de vitesse pour tous les maillages
764 // val_fixe indique si l'on veut des ddl libres ou pas
765 void Plus_Les_ddl_Vitesse(Enum_boolddl val_fixe);
766 // ajout des ddl d'accélération pour tous les maillages
767 // val_fixe indique si l'on veut des ddl libres ou pas
768 void Plus_Les_ddl_Acceleration(Enum_boolddl val_fixe);
769 // calcul de la longueur d'arrête d'élément minimal
770 // divisé par la célérité dans le matériau
771 double Longueur_arrete_mini_sur_c(Enum_dure temps);
772 // initialisation éventuelle du bulk viscosity
773 // choix peut-être égale à 0, 1 ou 2
774 void Init_bulk_viscosity(int choix,const DeuxDoubles & coef);
775
776 //----- lecture écriture de restart -----
777 // cas donne le niveau de la récupération
778 // = 1 : on récupère tout

```



```

778 // = 2 : on récupère uniquement les données variables (supposées comme telles)
779 void Lecture_base_info(ifstream& ent,const int cas);
780 // cas donne le niveau de sauvegarde
781 // = 1 : on sauvegarde tout
782 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
783 void Ecriture_base_info(ofstream& sort,const int cas);
784 // sortie du schemaXML: en fonction de enu
785 void SchemaXML_LesMaillages(ofstream& sort,const Enum_IO_XML enu) const ;
786
787
788 // ----- informations utiles par exemples pour la visualisation
789 // retourne les dimensions minis et maxi suivant les axes du repère
790 // absolu du maillage numéro nbmail (en faite le calcul est fondé
791 // uniquement sur la position des noeuds du maillage
792 // le premier vecteur contient les minimums
793 // le deuxième vecteur contient les maximums
794 Tableau <Vecteur> Taille_boite(int nbmail);
795 // dans le cas ou aucun numéro de maillage n'est fournis
796 // c'est l'encombrement de tous les maillages qui est fourni
797 Tableau <Vecteur> Taille_boite();
798
799
800 // ----- utilitaires de manipulation de maillage
801 // création de maillage quadratiques incomplets à partir de maillages linéaires.
802 // En fait il y création de maillages identiques aux maillages déjà existants, les éléments qui sont
de types
803 // linéaires sont remplacés par des éléments quadratiques incomplets correspondants.
804 // Il y a création de références correspondantes
805 void CreeMaillagesQuadratiques_a_partir_des_lineaires(LesReferences* lesRef);
806 // création de maillage quadratiques complets. En fait il y création de maillages identiques aux
maillages
807 // déjà existants, les éléments qui sont de types quadratiques incomplets sont remplacés par des
éléments
808 // quadratiques complets correspondants.
809 // Il y a création de références correspondantes
810 void CreeMaillagesQuadratiquesComplets_a_partir_des_incomplets(LesReferences* lesRef);
811 // création de maillages par extrusion
812 // Il y a création de références correspondantes
813 void CreeMaillageExtrusion2D3D(LesReferences* lesRef);
814 // définition interactive de listes de références
815 void CreationInteractiveListesRef(LesReferences* lesRef);
816 // modification de l'orientation d'éléments
817 void Modif_orientation_element(int cas_orientation,LesReferences* lesRef);
818 // collapse de noeuds très proche: appartenant à des éléments différents
819 // rayon : donne la distance maxi entre les noeuds qui doivent être collapsés
820 void Collapse_noeuds_proches(double rayon, LesReferences* lesRef);
821 // Collapse des éléments superposés, c-a-d identiques, dans le cas où il en existe
822 void Collapse_element_supperpose(LesReferences* lesRef);
823 // création d'un nouveau maillage issue de la fusion de maillages existants
824 // nom_mails_a_fusionner : la liste des maillages à fusionner
825 // new_mail : le nom du nouveau maillage à construire
826 // NB: si new_mail correspond à un maillage déjà existant, il y a fusion de ce maillage
827 // avec les autres, sans création d'un nouveau maillage
828 void Fusion_maillages(List_io < string >& nom_mails_a_fusionner,const string& new_mail
829 ,LesReferences* lesRef);
830 // création d'un nouveau maillage issue d'un maillages existants et d'une ref d'éléments
831 // le nouveau maillage = les éléments de la ref
832 void Cree_sous_maillage(int num_mail,LesReferences* lesRef, string nom_ref,const string& new_mail);
833 // création d'éléments SFE en fonction d'éléments classiques
834 // il y a création d'un nouveau maillage
835 void CreationMaillageSFE();
836
837 // Affiche les donnees des maillages dans des fichiers dont le nom est construit à partir du nom de
838 // chaque maillage au format ".her" et ".lis"
839 // le paramètre optionnel indique le numéro du maillage à afficher, s'il vaut -1, on affiche tous
les maillages
840 void Affiche_maillage_dans_her_lis(Enum_dure temps,LesReferences &lesRef,int imail=-1);
841
842 // relocalisation des points milieux des arrêtes des éléments quadratiques
843 void RelocPtMilieuMailleQuadra();
844
845 // --- utilitaires pour calculs particuliers-----
846 // calcul des diverses intégrations: volume et volume + temps,
847 // alimentation des grandeurs globales associées
848 void Integration();
849
850 // calcul des diverses statistiques sur des ref de noeuds et avec éventuellement
851 // cumul sur le temps
852 // alimentation des grandeurs globales associées
853 void CalStatistique();
854
855 private :
856 // VARIABLES PROTEGEES :
857 Tableau<Maillage *> tabMaillage ; // tableau de maillages
858 // liste des noms de maillages associée à un numéro sous forme d'un arbre pour faciliter la
recherche
859 // cette liste est modifiée que par chaque maillage

```

```

860 map < string, int , std::less <string> > mapNomMail;
861 // const int nbEnreg; // nb de maillage initiaux
862 int nbEnreg; // nb de maillage initiaux
863 int nbMaillageTotal; // nb de maillage effectivement en cours
864 int nbPortion ; // nbPortion = le nombre maxi de maillage enregistrable
865 // avant une nouvelle allocation dynamique de nbEnreg places supplementaires
866 UtilLecture * entreePrinc; // acces a la lecture du fichier principal
867 ParaGlob * paraGlob ; // parametres globaux
868 LesReferences* lesRef; // references
869 int domEsclave; // nombre de domaine esclave, lorsqu'il est different de zero
870 // il indique le nombre de maillage a partir de 1, qui sont esclave
871
872 // ---- stockage des integrales de volumes sur des references d'elements ----
873 // --- cas des integrales volumiques: definition du conteneur, il peut egalement s'agir d'une
integration temporelle en +
874 // d'ou la grandeur courante et celle a t
875 // 1) integration de volume uniquement
876 Tableau <TypeQuelconque> integ_vol_typeQuel, integ_vol_typeQuel_t;
877 Tableau <const Reference*> ref_integ_vol; // les references associees
878 // si la reference est nulle, cela signifie que l'integrale est figee: sa valeur ne change pas
879
880 // 2) integration de volume et en temps: donc on commule le delta
881 Tableau <TypeQuelconque> integ_vol_t_typeQuel, integ_vol_t_typeQuel_t;
882 Tableau <const Reference*> ref_integ_vol_t; // les references associees
883 // si la reference est nulle, cela signifie que l'integrale est figee: sa valeur ne change pas
884
885 // ---- stockage des statistiques sur des references de noeuds ----
886 // --- cas des statistique: definition du conteneur, il peut egalement s'agir d'un cumul temporelle
en +
887 // d'ou la grandeur courante et celle a t
888 // 1) statistique de ref de noeuds uniquement
889 Tableau <TypeQuelconque> statistique_typeQuel, statistique_typeQuel_t;
890 Tableau <const Reference*> ref_statistique; // les references associees
891 // si la reference est nulle, cela signifie que la statistique
892 // est figee: sa valeur ne change pas
893
894 //pour_statistique_de_ddl a la meme dimension que ref_statistique
895 // 1) Dans le cas ou ref_statistique(i) est une statistique de Ddl_enum_etendu
896 // pour_statistique_de_ddl(i) == le Ddl_enum_etendu
897 // 2) sinon, pour_statistique_de_ddl(i) == NU_DDL
898 Tableau < Ddl_enum_etendu > pour_statistique_de_ddl;
899
900 // 2) statistique avec cumul en temps: donc on commule le delta
901 Tableau <TypeQuelconque> statistique_t_typeQuel, statistique_t_typeQuel_t;
902 Tableau <const Reference*> ref_statistique_t; // les references associees
903 // si la reference est nulle, cela signifie que la statistique est figee: sa valeur ne change pas
904
905 //pour_statistique_t_de_ddl a la meme dimension que ref_statistique_t
906 // 1) Dans le cas ou ref_statistique_t(i) est une statistique de Ddl_enum_etendu
907 // pour_statistique_t_de_ddl(i) == le Ddl_enum_etendu
908 // 2) sinon, pour_statistique_de_ddl(i) == NU_DDL
909 Tableau < Ddl_enum_etendu > pour_statistique_t_de_ddl;
910
911 // cumule les liste des types de degres de liberte inconnus,
912 // qui vont etre calcules par la resolution des problemes
913 // physiques geres par les elements qui existent dans les maillages
914 // Si elements mecaniques -> ddl Xi voir Vi et gamma_i
915 // Si elements thermiques -> ddl de temperature
916 // Si elements meca + elements thermiques -> ddl Xi et temperature
917 // etc. en fonction des elements qui existent dans les maillages
918 // (genere a la lecture du maillage, ce qui permet d'optimiser la consultation par la suite)
919 list <Enum_ddl > ddl_representatifs_des_physiques;
920 // idem au niveau des types de problemes geres par les elements
921 list <EnumElemTypeProblem > types_de_problemes;
922
923 // stockage des pointeur de listes d'element frontiere
924 Tableau <LaLIST <Front*>> listFrontiere;
925 // idem pour les noeuds des frontieres
926 Tableau <Tableau <Noeud*> *> tt_noeud_front;
927 // cree et ramene pour tous les maillages, la liste des elements qui contiennent chaque noeud
928 // mis a jour lorsque lors de la creation des frontieres
929 Tableau < const Tableau <List_io < Element* > > *> tous_indices;
930 // tableau d'indexage des ddl gere par les noeuds
931 // t_i_n (i)(j) -> donne les infos pour retrouver le ddl numero j du cas de charge i
932 Tableau < Tableau <Posi_ddl_noeud> > t_i_n;
933 // nombre actuellement de cas d'assemblage initialise
934 int tab_nb_assemb;
935
936 // METHODES PROTEGEES :
937 // on s'occupe de mettre a jour les types de pb et les ddl types associes
938 void Mise_a_jour_type_pb_type_associe_ddl();
939
940 // met a jour les pointeurs d'assemblage dans les noeuds pour un cas d'assemblage
941 // a effectuer si l'on a changer de nb de noeuds, de nb de ddl, de nb de maillage
942 // casAssemb : donne le cas d'assemblage qui est a considerer
943 // ici, l'assemblage suit l'ordre du tableau de noeud passe en parametre
944 // le tableau de noeuds rassemble tous les noeuds des maillages mais avec une numerotation propre

```

```

945 void MiseAJourPointeurAssemblage_interne(const Nb_assemb& nb_casAssemb,Tableau <Noeud* >&
tab_N_final);
946
947 // remise à jour des tableaux de pointeurs t_i_n uniquement, due à un changement de numéro de noeud
948 // en fonction d'un changement de num de noeud (mais pas de changement de pointeur d'assemblage
949 // pour chaque noeud, tab_N_final(i) correspond au noeud qui avait le numéro i ancien
950 // et qui a maintenant le numéro tab_N_final(i)->Num_noeud()
951 void MiseAJourTableau_t_i_n(const Nb_assemb& nb_casAssemb,Tableau <Noeud* >& tab_N_final);
952
953
954 };
955 /// @} // end of group
956
957 #endif

```

## 7.352 Maillage.h

```

1 // FICHER : Maillage.h
2 // CLASSE : Maillage
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          23/01/97
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *
40 *      BUT:           def de la classe Maillage,
41 *                   Une instance de la classe Maillage est identifiée a partir
42 *                   de la dimension, des tableaux des noeuds et des elements.
43 *
44 *      *****
45 *
46 *      VERIFICATION:
47 *
48 *      ! date !   auteur !           but
49 *      !-----!-----!-----!
50 *
51 *      *****
52 *      MODIFICATIONS:
53 *
54 *      ! date !   auteur !           but
55 *      !-----!-----!-----!
56 *
57 *      *****/
58
59 #ifndef MAILLAGE_H
60 #define MAILLAGE_H
61
62
63 #include <iostream>
64 #include <map>
65
66
67 #include "Noeud.h"

```

```

68 #include "Element.h"
69 #include "Tableau_T.h"
70 #include "UtilLecture.h"
71 #include "Enum_geom.h"
72 #include "Enum_interpol.h"
73 #include "LesReferences.h"
74 #include "Front.h"
75 #include "LaList.h"
76 #include "Nb_assemb.h"
77 #include "Ddl_enum_etendu.h"
78 #include "Droite.h"
79 #include "Plan.h"
80 #include "Sphere.h"
81 #include "Cylindre.h"
82 #include "Cercle.h"
83 #include "Condilineaire.h"
84 #include "DiversStockage.h"
85
86
87 /// @addtogroup Les_Maillages
88 /// @{
89 ///
90
91 //-----
92 //!      Maillage: un maillage particulier
93 //-----
94 /// \author      Gérard Rio
95 /// \version    1.0
96 /// \date      23/01/97
97
98 class Maillage
99 {
100     public :
101     friend class LesMaillages;
102
103     // CONSTRUCTEURS :
104     // pour tous les constructeurs: map < string, int , std::less <string> >* lisNomMail,
105     // est un tableau associatif nom <=> numéro de maillage, qui est utilisé par
106     // les maillages, mais mis à jour par chaque maillage.
107     // nom_maillage : est facultatif, s'il est différent de ".", il est pris en compte
108
109     // Constructeur par défaut
110     Maillage (map < string, int , std::less <string> > & lisNomMail,int nmail=1,int dim=3
111             ,const string& nom_maillage = ".");
112
113     // Constructeur fonction d'une dimension, du nombre de noeuds
114     // du nombre d'elements, et d'un numero d'identification (le nb de maillage)
115     Maillage (map < string, int , std::less <string> > & lisNomMail
116             ,int dim,int n_noeud,int n_elt,int nmail
117             ,const string& nom_maillage = ".");
118
119     // Constructeur fonction de la plupart des informations (qui peuvent être vide
120     // mais doivent être cohérentes)
121     // *** il n'y a pas de création de nouveaux noeuds et de nouveaux éléments,
122     // ce sont les éléments et noeuds passés en paramètres qui sont ceux du maillage créé
123     Maillage (map < string, int , std::less <string> > & lisNomMail
124             ,int dim,list <Noeud* >& li_noeud, list <Element* > li_element
125             ,int nmail
126             ,const string& nom_maillage = ".");
127
128     // Constructeur de copie, cependant ici il n'y a pas de création de noeud ni d'élément
129     // c'est seulement une création de nouveaux conteneurs de pointeurs
130     // cependant le numéro de maillage et le nom de maillage n'est pas valide, il faut
131     // ensuite les définir
132     Maillage (const Maillage& mail);
133
134     // Constructeur de copie, avec création de nouveaux noeuds et éléments identiques à ceux passées
135     // en argument, nmail: donne le numéro du nouveau maillage créée, qui est donc a priori
différent
136     // de celui de mail, idem pour le nom du maillage
137     // les frontières ne sont pas transmises ni calculées !
138     Maillage (map < string, int , std::less <string> > & lisNomMail
139             ,int nmail, const string& nomDuMaillage, const Maillage& mail);
140
141     // DESTRUCTEUR :
142     ~Maillage ();
143
144
145     // METHODES :
146
147     // lecture de maillages au travers des outils de la classe
148     // UtilLecture et def des references s'y rapportant
149     void LectureMaillage(UtilLecture * entreePrinc,LesReferences& lesRef);
150     // lecture et application des opérations d'affinages sur le maillage: ex:déplacement solide
151     void LectureEtApplicationAffinage(UtilLecture * entreePrinc,LesReferences& lesRef);
152
153     // ajout d'une liste de noeud à un maillage

```

```

154 // si le numéro de maillage associé au noeud est nul, il est remplacé par celui du maillage
155 // si le numéro de maillage est déjà existant et est différent ce celui de this, il y a
156 // création d'un nouveau noeud identique, avec le numéro this
157 // ajout éventuel d'une liste de référence de noeuds, si celle-ci est non-nulle
158 // il y a création de nouvelles ref correspondantes au numéro de maillage de this
159 // et ces références sont rajoutées à lesRef
160 void Ajout_de_Noeuds(const list <Noeud *> & taN, list <const Reference*>* lref=NULL, LesReferences*
    lesRef=NULL );
161 // ajout d'une liste d'éléments et de noeud à un maillage
162 // si le numéro de maillage associé à l'élément ou noeud est nul, il est remplacé par celui du
    maillage
163 // si le numéro de maillage est déjà existant et est différent ce celui de this, il y a
164 // création d'un nouvel item identique, avec le numéro this
165 // ajout éventuel d'une liste de références associées , si celle-ci est non-nulle
166 // il y a création de nouvelles ref correspondantes au numéro de maillage de this
167 // et ces références sont rajoutées à lesRef
168 // les noeuds qui sont associés aux éléments de taE, doivent faire partie : soit de taN, soit du
    maillage this
169 void Ajout_elements_et_noeuds(const list <Noeud *> & taN, const list <Element *> & taE, list <const
    Reference*>* lref, LesReferences* lesRef );
170
171 // affichage et definition interactive des commandes
172 // cas = 1: interactif complet
173 // cas = 2: entrée uniquement de noms de fichier
174 void Info_commande_Maillages(UtilLecture * entreePrinc, LesReferences& lesRef, int cas);
175
176 // Affiche les donnees du maillage
177 void Affiche () const ;
178
179 // Affiche les donnees du maillage dans un fichier
180 // dont le nom est construit à partir du nom du maillage
181 // au format ".her" et ".lis"
182 void Affiche_dans_her_lis(LesReferences &lesRef, Enum_dure temps);
183
184 //modification du maillage pour le restreindre aux seuls éléments de la référence passée en paramètre
185 // toutes les infos relatives à des éléments supprimés, sont également supprimés
186 void Restreint_sous_maillage(LesReferences* lesRef, string nom_ref);
187
188 // Surcharge de l'operateur = : realise l'egalite de deux maillages
189 // cependant le numéro de maillage et le nom de maillage n'est pas valide, il faut
190 // ensuite les définir
191 Maillage& operator= (Maillage& mail);
192
193 inline int Dimension () const
194 // Retourne la dimension
195 { return dimension; };
196
197 // ramène la liste des problèmes physiques gérés par les éléments du maillage
198 inline const Tableau <Enum_ddl >& Ddl_representatifs_des_physiques() const
199 {return ddl_representatifs_des_physiques;};
200
201 // ramene la liste des degrés de liberté inconnus, associés aux pb
202 // physiques gérés par les éléments qui existent dans le maillage
203 // Si éléments mécaniques -> ddl Xi voir Vi et gamma_i
204 // Si éléments thermiques -> ddl de température
205 // Si éléments méca + éléments thermiques -> ddl Xi et température
206 // etc. en fonction des éléments qui existent dans le maillage
207 inline const Tableau <EnumElemTypeProblem >& Types_de_problemes() const
208 {return types_de_problemes;};
209
210 inline int Nombre_noeud_elt(int i) const
211 // Retourne le nombre de noeuds lies au ieme element
212 { return tab_element(i)->Nombre_noeud(); };
213
214 inline int Nombre_noeud() const
215 // Retourne le nombre de noeuds du maillage
216 { return tab_noeud.Taille(); };
217
218 inline int Nombre_element() const
219 // Retourne le nombre d'elements du maillage
220 { return tab_element.Taille(); };
221
222 inline Tableau<Noeud *>& Tab_noeud()
223 // Retourne le tableau des noeuds
224 { return tab_noeud; };
225
226 inline Tableau<Element *>& Tab_element()
227 // Retourne le tableau des elements
228 { return tab_element; };
229
230 inline Noeud& Noeud_mail(int i)
231 // Retourne le ieme noeud Noeud du tableau tab_noeud
232 { return *tab_noeud(i); };
233
234 inline Element& Element_mail(int i)
235 // Retourne le ieme element Element du tableau tab_element
236 { return *tab_element(i); };

```

```

237         // idem mais en version constant
238         inline const Element& Element_mail_const(int i)
239         { return *tab_element(i); };
240
241         // test si toutes les informations des maillages sont completes
242         // = true -> complet
243         // = false -> incomplet
244         bool Complet();
245
246         // ramene la demi largeur de bande en ddl et la largeur de bande
247         void Largeur_Bande(int& demi, int& total, const Nb_assemb& nb_casAssemb);
248
249         // test pour savoir si tous les coordonnées des noeuds d'un maillage sont imposé
250         // ramène 1 si tout est fixé, 0 sinon
251         int Tous_Xi_fixes(const Nb_assemb& casAss) const;
252
253         // calcule des normales aux noeuds: dans le cas d'éléments 1D ou 2D uniquement
254         // a priori le calcul s'effectue par une moyenne des normales des éléments qui
255         // entourent le noeud.
256         // init -> calcul des normales à t=0
257         // et ajout conteneur aux noeuds des normales à t = 0 et t
258         void InitNormaleAuxNoeuds();
259         // mise à jour -> mise à jour des normales à t
260         void MiseAJourNormaleAuxNoeuds();
261         // mise à jour -> mise à jour des normales à t
262         // mais ici, on calcule les normales à tdt, et on transfert à t
263         // cette méthode est utile si on veut utiliser des normales à t pour une valeur
264         // particulière (transitoire) de la géométrie à tdt
265         // cf: l'algo non dyna par exempel
266         void MiseAJourNormaleAuxNoeuds_de_tdt_vers_T();
267
268         // creation des elements frontiere
269         void CreeElemFront();
270
271         // ramene un pointeur sur la liste des elements frontieres
272         inline LaLIST <Front>* ListFront() { return &listFrontiere;};
273         // ramene le tableau des noeuds de la frontiere
274         Tableau<Noeud *>& Tab_noeud_front() {return tab_noeud_front; };
275
276         // ramene le nom du maillage
277         string NomDuMaillage() {return nomDuMaillage;};
278
279         // change le nom et le numéro du maillage
280         void ChangeNomNumeroMaillage(const string & nom, int num);
281
282         // ramène le numéro du noeud le plus proche du point donné à t=0,
283         int Noeud_le_plus_proche_0(const Coordonnee& M);
284         // idem à t
285         int Noeud_le_plus_proche_t(const Coordonnee& M);
286         // idem à tdt
287         int Noeud_le_plus_proche_tdt(const Coordonnee& M);
288
289         // ramène le maximum de variation de coordonnée entre t et tdt de tous les noeuds du maillage
290         double Max_var_dep_t_a_tdt() const;
291
292         // ramène le minimum de la distance entre deux noeuds de l'ensemble des éléments
293         double Min_dist2Noeud_des_elements(Enum_dure temps) const;
294
295         // transfert de grandeurs des points d'intégration aux noeuds
296         // 1- en entrée les type de ddl internes que l'on veut transférer
297         // idem pour les type évoluées et les types particuliers
298         // 2- en entrée: cas qui indique la méthode de transfert à utiliser
299         // =1 : les valeurs aux noeuds sont obtenue par moyennage des valeurs des pts d'integ
300         // les plus près
301         // des éléments qui entourent le noeud
302         // on décompose le processus en trois étapes pour éviter d'initialiser plusieurs fois lorsque l'on
303         // refait à chaque fois
304         // le même transfert
305
306         // A) première étape def des conteneurs et c'est tout, la méthode peut donc être utilisée
307         // pour autre chose. tabQ: permet d'avoir plusieurs listes de TypeQuelconque
308         // en entrée: tabQ doit-être de dimension 2, donc pointe sur 2 listes, si un des pointeur
309         // est nulle on n'en tient pas compte
310         void AjoutConteneurAuNoeud(const List_io < Ddl_enum_etendu >& lienu
311         ,const Tableau <List_io < TypeQuelconque > * >& tabQ);
312
313         // B) initialisation des updates sur les noeuds
314         void InitUpdateAuNoeud(const List_io < Ddl_enum_etendu >& lienu
315         ,const Tableau <List_io < TypeQuelconque > * >& tabQ, int cas);
316
317         // C) exécution du transfert
318         // transfert incrémental (pour un élément et tous les pt d'integ):
319         // transfert de ddl de tous les pt d'integ aux noeuds d'un éléments (on ajoute aux noeuds, on ne
320         remplace pas)
321         // les ddl doivent déjà exister aux noeuds sinon erreur
322         // il doit s'agir du même type de répartition de pt d'integ pour toutes les grandeurs
323         void TransfertPtIntegAuNoeud(Element& ele, const List_io < Ddl_enum_etendu >& lietendu
324         ,const Tableau <Tableau <double> > & tab_val, int cas)

```

```

321     {ele.TransfertAjoutAuNoeuds(lietendu,tab_val,cas)};};
322 // idem pour des grandeurs quelconques, transfert de "tous" les points d'intégration en même temps
323 // ceci pour optimiser, les informations sont ici contenues dans les types quelconques

324 // liQ_travail: est une liste de travail qui sera utilisée dans le transfert

325 // - transfert de type quelconque des points d'intégrations (de tous) aux noeuds d'un éléments (on
326 // - on ne remplace pas). Les types quelconques doivent déjà exister.
327 // - un tableau tab_liQ correspondent aux grandeurs quelconque pour tous les pt integ. tab_liQ(i) est
328 // - Toutes les listes sont identiques au niveau des descripteurs (types...) ce sont uniquement les
329 // - valeurs numériques
330 // - c-a-dire les valeurs associées à TypeQuelconque::Grandeur qui sont différentes (elles sont
331 // - associées à chaque pt d'integ)
332 // - liQ_travail: est une liste de travail qui sera utilisée dans le transfert
333 void TransfertPtIntegAuNoeud(Element& ele,const Tableau <List_io < TypeQuelconque > >& tab_liQ
334     ,List_io < TypeQuelconque > & liQ_travail,int cas)
335 {ele.TransfertAjoutAuNoeuds(tab_liQ,liQ_travail,cas)};};
336 // D) dernière étape: (par exemple calcul des moyennes en chaque noeuds)
337 // les résultats sont stockés aux noeuds
338 void FinTransfertPtIntegAuNoeud(const List_io < Ddl_enum_etendu >& lienu
339     ,const Tableau <List_io < TypeQuelconque > * >& tabQ,int cas);
340
341 // ..... cumul et moyenne de grandeurs venant des éléments vers les noeuds (exemple la pression
342 // appliquée) .....
343 // on décompose le processus en 4 étapes pour éviter d'initialiser plusieurs fois lorsque l'on
344 // refait à chaque fois
345 // la même opération (typiquement à chaque incrément)
346 // on peut utiliser:
347 // A) AjoutConteneurAuNoeud : pour ajouter des conteneurs ad hoc aux noeuds
348 // B) InitUpdateAuNoeud: avant le cumul, initialise les conteneurs
349 // C) Accumul_aux_noeuds : balaie les éléments avec cumul aux noeuds
350 // D) MoyenneCompteurAuNoeud : effectue les moyennes aux noeuds
351
352 // accumulation aux noeuds de grandeurs venant des éléments vers leurs noeuds (exemple la pression
353 // appliquée)
354 // autres que celles aux pti classiques, mais directement disponibles
355 // le contenu du conteneur stockées dans liQ est utilisé en variable intermédiaire
356 void Accumul_aux_noeuds(const List_io < Ddl_enum_etendu >& lietendu
357     ,List_io < TypeQuelconque > & liQ,int cas)
358 {int NBE=tab_element.Taille();
359 for (int i=1;i<=NBE;i++)
360     tab_element(i)->Accumul_aux_noeuds(lietendu,liQ,cas);
361 };
362
363 // fonctions utilitaires du même genre
364
365 // ajout sur le maillage d'un conteneur particulier quelconque
366 void AjoutConteneurAuNoeud(TypeQuelconque& tQ);
367 // ajout sur le maillage d'un ou plusieurs ddl_enum_etendu comme conteneur
368 void AjoutConteneurAuNoeud(const List_io < Ddl_enum_etendu >& lienu);
369 // initialisation des updates de ddl_etendu uniquement sur les noeuds: on met à 0 les ddl_etendu
370 // correspondant,
371 // les compteurs, comptant le nombre de fois où les noeuds sont modifiés, sont mis à 0
372 void InitUpdateAuNoeud(const List_io < Ddl_enum_etendu >& lienu);
373 // idem pour un seul ddl_etendu
374 void InitUpdateAuNoeud(const Ddl_enum_etendu & enu);
375 // moyenne des valeurs aux noeuds (en fonction du nombre ou le noeud a été modifié)
376 void MoyenneCompteurAuNoeud(const Ddl_enum_etendu & enu);
377
378 // initialisation des coordonnées à t et tdt aux mêmes valeurs qu'à 0
379 // utile quand on veut utiliser les métriques pour un pb non couplés
380 void Init_Xi_t_et_tdt_de_0();
381
382 // def d'un conteneur pour deux numéros: elem et pti
383 class NBelemEtptInteg
384 {public: int nbElem; int nbPtInteg;
385 // surcharge de l'operator de lecture
386 friend istream & operator » (istream & ent, NBelemEtptInteg & de)
387 { ent » de.nbElem » de.nbPtInteg; return ent;};
388 // surcharge de l'operator d'écriture
389 friend ostream & operator « (ostream & sort , const NBelemEtptInteg & de)
390 { sort « de.nbElem « " " « de.nbPtInteg « " "; return sort;};
391 bool operator < (const NBelemEtptInteg& c) const; bool operator >
392 (const NBelemEtptInteg& c) const;
393 bool operator == (const NBelemEtptInteg& c) const {return
394 ((nbElem==c.nbElem)&&(nbPtInteg==c.nbPtInteg));};
395 bool operator != (const NBelemEtptInteg& c) const {return
396 !((nbElem==c.nbElem)&&(nbPtInteg==c.nbPtInteg));};
397 };
398
399 // def d'un conteneur pour 3 numéros: elem, num face ou arete, et pti
400 class NBelemFAetptInteg
401 {public: int nbElem; int nbFA; int nbPtInteg;
402 // surcharge de l'operator de lecture
403 friend istream & operator » (istream & ent, NBelemFAetptInteg & de)

```

```

395     { ent > de.nbElem > de.nbFA > de.nbPtInteg; return ent;};
396     // surcharge de l'operator d'écriture
397     friend ostream & operator << (ostream & sort , const NBelemFAEtptInteg & de)
398     { sort << de.nbElem << " " << de.nbFA << " " << de.nbPtInteg << " "; return sort;};
399     bool operator < (const NBelemFAEtptInteg& c) const; bool operator > (const NBelemFAEtptInteg&
c) const;
400     bool operator == (const NBelemFAEtptInteg& c) const {return
((nbElem==c.nbElem)&&(nbFA==c.nbFA)&&(nbPtInteg==c.nbPtInteg));};
401     bool operator != (const NBelemFAEtptInteg& c) const {return
!((nbElem==c.nbElem)&&(nbFA==c.nbFA)&&(nbPtInteg==c.nbPtInteg));};
402     };
403
404     // ramène le numéro de l'élément qui contient un point donné et le numéro du point
405     // d'intégration le plus proche pour les ddl de la liste, (ddl spécifique à l'élément c'est-à-dire
406     // hors des ddl des noeuds de l'éléments)
407     // si pas de numéro d'élément ramène un numéro d'élément nulle
408     // si les numéros de point d'intégration ne sont pas identique pour l'ensemble
409     // des ddl, pb !!, le numéro du pt integ de retour est alors négatif
410     // enu_temps: dit si les coordonnées du point M sont à 0 ou t ou tdt
411     NBelemEtptInteg Element_le_plus_proche
412     (Enum_dure enu_temps,const List_io <Ddl_enum_etendu>& list_enu,const Coordonnee& M);
413
414     // ramène le numéro de l'élément dont le centre de gravité à t = enu_temps est le plus proche d'un
point donné
415     // Le point peut être n'importe où, en particulier à l'extérieur de la matière
416     // si pb retour de null
417     const Element* Centre_de_Gravite_Element_le_plus_proche(Enum_dure enu_temps,const Coordonnee& M);
418
419     // ramène pour chaque noeud, la liste des éléments qui contiennent le noeud
420     // si le tableau n'existe pas, il est construit, sinon uniquement un retour
421     // et la miss à jour est uniquement faite lors de la création des frontières
422     const Tableau <List_io < Element* > >& Indice()
423     {if (indice.Taille() == 0) Calcul_indice();
424     return indice; };
425
426     //----- lecture écriture dans base info -----
427     // cas donne le niveau de la récupération
428     // = 1 : on récupère tout
429     // = 2 : on récupère uniquement les données variables (supposées comme telles)
430     void Lecture_base_info(ifstream& ent,const int cas);
431     // cas donne le niveau de sauvegarde
432     // = 1 : on sauvegarde tout
433     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
434     void Ecrire_base_info(ofstream& sort,const int cas);
435     // sortie du schemaXML: en fonction de enu
436     static void SchemaXML_Maillages(ofstream& sort,const Enum_IO_XML enu) ;
437
438     // ----- informations utiles par exemples pour la visualisation
439     // retourne les dimensions minis et maxi suivant les axes du repère
440     // absolu du maillage (en faite le calcul est fondé
441     // uniquement sur la position des noeuds du maillage
442     // le premier vecteur contient les minimums
443     // le deuxième vecteur contient les maximums
444     Tableau <Vecteur> Taille_boiteMail();
445
446     // ----- utilitaires de manipulation de maillage
447     // test pour savoir si le maillage contient des éléments à interpolation linéaire
448     bool Contient_lineaire();
449     // test pour savoir si le maillage contient des éléments à interpolation quadratique incomplète
450     bool Contient_quadratique_incomplet();
451     // transformation des éléments linéaires du maillage en quadratiques.
452     // les éléments linéaires sont supprimés,
453     // Important: dans la procédure de renumérotation, la routine modifie la numérotation initiale des
noeuds !!
454     // il y a également création de référence adaptée au nouveau maillage en cohérence avec l'ancien
maillage
455     void Transfo_lin_quadraIncomp(LesReferences &lesRef);
456     // transformation des éléments quadratiques incomplet du maillage en quadratiques complets.
457     // les éléments incomplets sont supprimée,
458     // Important: dans la procédure de renumérotation, la routine modifie la numérotation initiale des
noeuds !!
459     // il y a également création de référence adaptée au nouveau maillage en cohérence avec l'ancien
maillage
460     void Transfo_quadraIncomp_quadraComp(LesReferences &lesRef);
461     // relocalisation des points milieux des arêtes des éléments quadratiques
462     void RelocPtMilieuMailleQuadra();
463     // définition interactive de listes de références
464     void CreationInteractiveListesRef(LesReferences* lesRef);
465     // modification de l'orientation d'éléments
466     void Modif_orientation_element(int cas_orientation,LesReferences* lesRef);
467     // Lecture et Collapse des éléments superposés, c-a-d identiques, dans le cas où il en existe
468     void LectureEtCollapse_element_superpose(UtilLecture * entreePrinc,LesReferences* lesRef);
469     // Collapse des éléments superposés, c-a-d identiques, dans le cas où il en existe
470     void Collapse_element_superpose(LesReferences* lesRef);
471     // Lecture et collapse de noeuds très proche: appartenant à des éléments différents
472     void LectureEtCollapse_noeuds_proches(UtilLecture * entreePrinc, LesReferences* lesRef);
473     // collapse de noeuds très proche: appartenant à des éléments différents

```



```

474 // rayon : donne la distance maxi entre les noeuds qui doivent être collapsé
475 void Collapse_noeuds_proches(double rayon, LesReferences* lesRef);
476 // Lecture et suppression d'elements à 2 noeuds, de distances très proches
477 void LectureEtSup_Elem_noeudsConfondu(UtilLecture * entreePrinc, LesReferences* lesRef);
478 // suppression d'elements à 2 noeuds, de distances très proches
479 // rayon : donne la distance maxi entre les noeuds
480 void Sup_Elem_noeudsConfondu(double rayon, LesReferences* lesRef);
481 // création d'éléments SFE en fonction d'éléments classiques
482 void CreationMaillageSFE();
483 // test pour savoir si le maillage est ok pour être transformée en sfe
484 bool OKPourTransSfe();
485 // lecture et suppression éventuelle des noeuds, non référencés par les éléments et les références
486 void LectureEtSuppressionNoeudNonReferencer(UtilLecture * entreePrinc, LesReferences& lesRef);
487 // uniquement suppression éventuelle des noeuds, non référencés par les éléments et les références
488 void SuppressionNoeudNonReferencer(LesReferences& lesRef);
489 // Affichage des noeuds, non référencés par les éléments
490 void AffichageNoeudNonReferencer();
491
492 // lecture et création éventuelle d'une ref sur les noeuds, non référencés par les éléments
493 void LectureEtCreationRefNoeudNonReferencer(UtilLecture * entreePrinc, LesReferences& lesRef);
494 // création éventuelle d'une référence sur les noeuds, non référencés par les éléments
495 void CreationRefNoeudNonReferencer(LesReferences& lesRef);
496 // vérification que toutes les références de noeuds, d'éléments, d'arêtes et de faces sont
    valides
497 // c-a-d se réfèrent à des éléments existants
498 void VerifReference(LesReferences& lesRef);
499 // lecture et renumérotation éventuelle des noeuds
500 void LectureEtRenumerotation(UtilLecture * entreePrinc, LesReferences& lesRef);
501 // renumérotation des noeuds du maillage, en fonction de conditions linéaires éventuelles
502 // ramène false si rien n'a changé (à cause d'un pb ou parce que la renumérotation n'est pas
    meilleure), vrai sinon
503 bool Renumerotation(LesReferences& lesRef, const Tableau <Tableau <Condilinaire> >& condCLL);
504 // création automatique des références globales de frontière si demandé dans le .info
505 void CreationRefFrontiere(UtilLecture * entreePrinc, LesReferences& lesRef);
506 // demande de création automatique des références globales de frontière
507 void CreationRefFrontiere(LesReferences& lesRef);
508 // force la mise à une valeur d'un ddl (ou de la liste de ddl fonction de la dimension) particulier,
    quelques soit son activité
509 // si fonction_de_la_dimension = true : c'est toute les ddl fct de la dimension qui sont mis à la
    valeur
510 void Force_Ddl_aux_noeuds_a_une_valeur(Enum_ddl enu, const double& val, Enum_dure temps, bool
    fonction_de_la_dimension);
511 // mise à zéro de dd_enum_etendu aux noeuds : force la mise à une valeur à 0
512 void Force_Ddl_etendu_aux_noeuds_a_zero(const Tableau<Ddl_enum_etendu>& tab_enu);
513
514
515 protected :
516     int idmail ; // numero de maillage
517     string nomDuMaillage;
518     // liste communes de tous les noms de maillages associée à un numéro
519     // sous forme d'un arbre pour faciliter la recherche
520     // cette liste n'est modifiée que par chaque maillage
521     map < string, int , std::less <string> >& listeNomMail;
522     int dimension; // dimension du maillage
523     Tableau<Noeud * > tab_noeud; // tableau des noeuds du maillage
524     Tableau<Element * > tab_element; // tableau des elements du maillage
525     // list des elements frontieres du maillage, c-a-d des frontières uniques
526     LaLIST <Front> listFrontiere;
527     // tableau des noeuds des éléments frontières
528     Tableau <Noeud * > tab_noeud_front;
529     // tableau utilitaire:
530     // indice(i) contient la liste des éléments qui contiennent le noeud i
531     Tableau <List_io < Element* > > indice;
532     // pour chaque élément "i" , mitoyen_de_chaque_element(i)(j) contient l'élément Front
533     // qui décrit la frontière "j" de l'élément et dedans, les éléments mitoyens de cette frontière
534     // appartenant à d'autres éléments (ce tableau est construit par la méthode : Mitoyen())
535     Tableau < Tableau <Front> > mitoyen_de_chaque_element;
536
537     // définie la liste des types de degrés de liberté inconnus, qui vont être calculés par la
    résolution des problèmes
538     // physiques gérés par les éléments qui existent dans le maillages
539     // Si éléments mécaniques -> ddl Xi voir Vi et gamma_i
540     // Si éléments thermiques -> ddl de température
541     // Si éléments méca + éléments thermiques -> ddl Xi et température
542     // etc. en fonction des éléments qui existent dans les maillages
543     // (généralisé à la lecture du maillage, ce qui permet d'optimiser la consultation par la suite)
544     Tableau <Enum_ddl > ddl_representatifs_des_physiques;
545     // idem au niveau des types de problèmes gérés par les éléments
546     Tableau <EnumElemTypeProblem > types_de_problemes;
547
548     //indicateur uniquement utilisé pour la destruction, par défaut est toujours vrai
549     // via la méthode Preparation_destruction_avec_conservation_noeuds_elements(), on peut le modifier
550     bool detruire_les_noeuds_et_elements;
551
552 // -- variables internes utilisées par Orientation_elements_mitoyens_recurusif
553 // tab_sens_element(i) : = 1 au début, puis vaut -1 si le sens de l'élément i à été changé
554     Tableau <double> tab_sens_element; // pendant les différents appels de

```

```

Orientation_elements_mitoyens_recurusif
555 // - on définit un tableau d'indicateur, permettant de savoir si un élément a été traité ou pas
556 // une fois qu'un élément a été traité: a) on ne permet plus son changement de sens
557 // b) de plus dans le cas d'une suite d'appels récursif, il n'est plus retenue pour les nouvelles
    listes
558     Tableau <bool> ind_elem; // def et init à 0, c-à-d non traité
559
560 // -- fin variables internes utilisées par Orientation_elements_mitoyens_recurusif
561
562 // METHODES PROTEGEES :
563
564 // cas particulier de destruction, sans suppression des noeuds et éléments
565 // l'utilisation de la méthode suivante, permet ensuite de supprimer le maillage
566 // tout en évitant la destruction des noeuds internes et éléments internes
567 // à utiliser avec précaution, intéressant si l'on veut créer un nouveau maillage
568 // avec les noeuds et éléments de ce maillage
569 void Preparation_destruction_avec_conservation_noeuds_elements()
570 { detruire_les_noeuds_et_elements = false; };
571
572 // change le numéro de maillage
573 void Change_numero_maillage(int new_num);
574
575 // definition des elements mitoyens aux elements de frontiere
576 // à la fin du programme tous les éléments mitoyens sont stocké dant les éléments
577 // Front et non les éléments frontières qui eux sont supprimés
578 void MitoyenFront();
579 // création pour chaque noeud de la liste des éléments qui contiennent le noeud
580 void Calcul_indice();
581
582 // definition d'un stockage contenant tous les Front associés à toutes les frontières de tous les
    éléments
583 // puis définition des elements mitoyens à ces Front : deux éléments Front sont mitoyens
584 // s'ils correspondent à des Element différents, et si les frontières associées possèdent les
    mêmes noeuds
585 // toutes ces informations sont stockées dans : mitoyen_de_chaque_element (cf. sa description)
586 void Calcul_tous_les_front_et_leurs_mitoyens();
587
588 // dans le cas où les éléments frontières sont des lignes, on les ordonne
589 // de manière à former une ligne continue
590 void OrdonancementDesLigne();
591 // lectures des infos pour le choix d'un élément
592 void Lecture_info_lelement(UtilLecture * entreePrinc,int& num_elt,Enum_geom& id_geom
    ,Enum_interpol& id_interpol,EnumElemTypeProblem& id_typeProb
    ,string& discriminant);
593 // lecture des mouvements solides si nécessaire
594 void Lecture_des_mouvements_solides(UtilLecture * entreePrinc);
595
596 // une classe de travail qui sert pour pouvoir classer les noeuds par leur position géométrique
    initiale
597 // deux élément sont identiques si leur position initiale sont identique
598 // a >= b si : soit a.x >= b.x, ou a.x==b.x et a.y >= b.y, ou a.x==b.x et a.y == b.y et a.z >= b.z
599 class PosiEtNoeud
600 { public:
601     Noeud * noe; // le noeud
602     Element * el; // l'élément auquel il est rattaché
603     // ----- constructeur -----
604     PosiEtNoeud() : noe(NULL),el(NULL) {}; // constructeur par défaut
605     PosiEtNoeud(Noeud * no,Element * e) : noe(no),el(e) {}; // constructeur normal
606     PosiEtNoeud(const PosiEtNoeud& po) : noe(po.noe),el(po.el) {}; // constructeur de copie
607     //----- surcharges qui ne travaillent que sur la position -----
608     PosiEtNoeud operator= (const PosiEtNoeud& po); // affectation
609     bool operator == (const PosiEtNoeud& po) const; // test d'égalité
610     bool operator != (const PosiEtNoeud& po) const; // test d'inégalité
611     bool operator < (const PosiEtNoeud& po) const; // relation d'ordre
612     bool operator <= (const PosiEtNoeud& po) const; // relation d'ordre
613     bool operator > (const PosiEtNoeud& po) const; // relation d'ordre
614     bool operator >= (const PosiEtNoeud& po) const; // relation d'ordre
615 };
616
617 // fonctions simples ayant pour but de bien repérer des opérations dangereuses
618
619 // affectation d'un noeud au maillage c-a-d au tableau de pointeur de noeud,
620 // a condition que la place ne soit pas déjà occupée sinon on change le numéro
621 // de noeud et on augmente le tableau
622 void Affectation_noeud (Noeud& noeud);
623
624 // affectation d'un element au maillage c-a-d au tableau de pointeur d'element,
625 // a condition que la place ne soit pas déjà occupée sinon on change le numéro
626 // de l'élément et on augmente le tableau
627 void Affectation_element (Element& element);
628
629 // Modifie le nombre de noeuds du maillage (N.B.: Fait appel
630 // a la methode Change_taille de la classe Tableau<Noeud>)
631 // les éléments supplémentaires ont un pointeur mis à Null
632 void Change_nb_noeud (int nouveau_nb);
633
634 // Modifie le nombre d'elements du maillage (N.B.: Fait appel

```

```

637 // a la methode Change_taille de la classe Tableau<Element>
638 // les éléments supplémentaires ont un pointeur mis à Null
639 void Change_nb_element (int nouveau_nb);
640
641 // ** pour l'instant ne fonctionne que pour les éléments surfaces
642 // orientation automatique des éléments mitoyens, à l'élément num_elem, et ensuite
643 // récursivement à tous les éléments mitoyens des mitoyens jusqu'à ce que la chaîne s'arrête
644 // L'ensemble des éléments est alors groupé dans une référence qui est construit à partir du numéro
num_elem
645 // et qui est ajouté aux refs déjà existantes
646 // ensuite, le programme passe en revue les éléments restants, et regarde s'ils font parti
647 // d'un ensemble homogène orienté, si non, ramène la liste des éléments hors ensemble orienté
648 // *** ne concerne que les éléments surfaces, les autres sont ignorés
649 // ind_elem(i) : (tableau interne) indique si l'élément i a été traité (=true) ou non (=false)
650 // ind_elem: est pris en compte puis mis à jour par le programme, c-a-d que les nouveaux éléments
orienté
651 // passe de false à true, par contre tous les éléments déjà à true, ne sont pas pris en
compte dans le traitement
652 // angle_maxi : angle maximum entre deux éléments, au dessus duquel on considère qu'il y a une rupture
de la mitoyenneté
653 // nom_ref : s'il est différent de "_", donne le nom de base voulu à la série de référence qui va être
construite
654 // inverse : indique si l'on souhaite partir d'une orientation inverse de celle existante avant
application de l'algo
655 // ceci pour l'élément de départ: au premier appel, l'opération est toujours possible,
ensuite cela dépend
656 // si l'élément trouvé a déjà été traité ou pas
657 // recursiv : indique si l'on se situe dans une suite récursive d'appel de la méthode
658 // si oui, seule les éléments non déjà pris en compte dans les appels précédents, sont
examiné
659 // c'est ind_elem qui permet de s'en assurer
660 // si non: cas par exemple d'un angle_maxi qui change, on réexamine tous les éléments,
cependant
661 // la ré-orientation éventuelle n'est faite qu'une seule fois (toujours via ind_elem)
662 list <int> Orientation_elements_mitoyens_recurusif(bool recursif,string& nom_ref,int num_elem
, LesReferences& lesRef,double& angle_maxi,bool
inverse);
664 // méthode pour initialiser les différents tableaux utilisés par
Orientation_elements_mitoyens_recurusif
665 void Init_Orientation_elements_mitoyens_recurusif();
666
667 // méthode pour orienter des éléments en fonction d'un rayon: AG, A étant un point donné, G étant le
centre de
668 // gravité d'une facette
669 // A : coordonnée d'un point, si indic_G(i) est true, alors A(i) est remplacé par la coordonnée G(i)
du centre
670 // de gravité de la facette
671 // zone_a_traiter: le nom de la référence des éléments a traiter
672 // inverse : si true, l'orientation des normales des facettes est identique à celles de AG, sinon
c'est l'inverse
673
674 void Orientation_via_rayon(const Tableau <bool> & indic_G,const Coordonnee & A
, const string& zone_a_traiter, LesReferences& lesRef, bool inverse);
675
676 // ----- pour la définition interactive de liste de ref -----
677 // def d'un conteneur pour deux numéros: nb element nb face
678 class NBelemEtFace {public: int nbElem; int nbFace;
679 bool operator < (const NBelemEtFace& c) const; bool operator >
680 (const NBelemEtFace& c) const;
681 bool operator == (const NBelemEtFace& c) const {return
((nbElem==c.nbElem)&&(nbFace==c.nbFace));};
682 bool operator != (const NBelemEtFace& c) const {return
!((nbElem==c.nbElem)&&(nbFace==c.nbFace));};
683 };
684 // def d'un conteneur pour deux numéros: nb element nb arête
685 class NBelemEtArete {public: int nbElem; int nbArete;
686 bool operator < (const NBelemEtArete& c) const; bool operator > (const NBelemEtArete& c) const;
687 bool operator == (const NBelemEtArete& c) const {return
((nbElem==c.nbElem)&&(nbArete==c.nbArete));};
688 bool operator != (const NBelemEtArete& c) const {return
!((nbElem==c.nbElem)&&(nbArete==c.nbArete));};
689 };
690
691 // calcul des listes de références en fonction de la demande:
692 // list_nomReference: contient les types de refs que l'on veut
693 // list_methode: contient la ou les méthodes que l'on veut utiliser
694 // cas 1D
695 void CalculListRef_1D(list<string>& list_nomReference,LesReferences* lesRef,list<string>&
list_methode
696 ,const Enum_ddl & enu_ddl);
697 // cas 2D
698 void CalculListRef_2D(list<string>& list_nomReference,LesReferences* lesRef,list<string>&
list_methode
699 ,const Enum_ddl & enu_ddl);
700 // cas 3D
701 void CalculListRef_3D(list<string>& list_nomReference,LesReferences* lesRef,list<string>&
list_methode

```

```

702         ,const Enum_ddl & enu_ddl);
703 // les méthodes de bases
704 // fonction générique pour des condition PresDe
705 void PresDe(const list<string>& list_nomReference, list <Noeud *>& list_noeud_restant
706            ,list <Element *>& list_element_restant,const Enum_ddl & enu_ddl
707            ,list <NBelemEtptInteg>& list_elemPtin_restant,bool& premLpti
708            ,list <NBelemEtFace>& list_elemFace_restant,bool& premLface
709            ,list <NBelemEtArete>& list_elemArrete_restant,bool& premLarrete);
710 // fonction générique pour des condition "tout dedans"
711 void ToutDedans(const list<string>& list_nomReference, list <Noeud *>& list_noeud_restant
712               ,list <Element *>& list_element_restant,const Enum_ddl & enu_ddl
713               ,list <NBelemEtptInteg>& list_elemPtin_restant,bool& premLpti
714               ,list <NBelemEtFace>& list_elemFace_restant,bool& premLface
715               ,list <NBelemEtArete>& list_elemArrete_restant,bool& premLarrete);
716 // constitution des références
717 void EnregRef(const list<string>& list_nomReference,list <Noeud *>& list_noeud_restant
718             ,list <NBelemEtptInteg>& list_elemPtin_restant,list <Element *>& list_element_restant
719             ,list <NBelemEtFace>& list_elemFace_restant,list <NBelemEtArete>&
720             list_elemArrete_restant
721             ,LesReferences* lesRef);
722 // définition des fonctions conditions
723 // def de data internes qui servent pour les fonctions init et exe: dimensionnées dans init, utilisé
724 // dans exe
725 // donc: init et exe doivent être utilisé absolument à suivre
726 Tableau <Coordonnee> t_poi;
727 Tableau <Droite> t_droit;
728 Tableau <double> t_para;
729 Tableau <Plan> t_plan;
730 Tableau <Sphere> t_sphere;
731 Tableau <Cylindre> t_cylindre;
732 Tableau <Cercle> t_cercle;
733 // cas de croisement avec des références existantes: def de variables de passage pour InRef et
734 // OutRef
735 // les pointeurs qui suivent ne servent pas à créer des refs, mais uniquement a récupérer les
736 // adresses
737 list <const Reference*> list_refIn;
738 list <const Reference*> list_refOut;
739 LesReferences* lesRefIn; // variable utilisée uniquement pour le passage d'info à
740 // ,InitInRef(), ExeInRef(), InitOutRef(), ExeOutRef()
741 // tout d'abord les pointeurs de fonctions en cours
742 // - pour les conditions Presde
743 // acquisition interactive des paramètres de la condition
744 void (Maillage::*initConditionPresDe) (double& dist);
745 // exécution de la condition : ramène true si ok, false sinon
746 bool (Maillage::*ExeConditionPresDe) (const double& dist,const Coordonnee& M)const;
747 // - pour les condition ToutDedans
748 // acquisition interactive des paramètres de la condition
749 void (Maillage::*initConditionToutDedans) ();
750 // exécution de la condition : ramène true si ok, false sinon
751 bool (Maillage::*ExeConditionToutDedans) (const Tableau <Coordonnee> & tab_M)const;
752 // recherche près d'un point
753 // def interactive du point et de la distance
754 void InitPresPoint(double& dist);
755 // exécution de la condition
756 bool ExePresPoint(const double& dist,const Coordonnee& M) const;
757 // recherche d'un coté d'un point (en 1D)
758 void InitCotePoint();
759 // ramène true si tous les points sont du même coté que t_poi(1)
760 bool ExeCotePoint(const Tableau <Coordonnee> & t_M)const;
761 // recherche entre deux points (1D)
762 void InitEntrePoint();
763 // ramène true si tous les points sont entre t_poi(1) et t_poi(2)
764 bool ExeEntrePoint(const Tableau <Coordonnee> & t_M)const;
765 // recherche entre deux points (1D) _avec_distance
766 void InitEntrePoint_avec_distance();
767 // ramène true si tous les points sont entre t_poi(1) et t_poi(2)
768 bool ExeEntrePoint_avec_distance(const Tableau <Coordonnee> & t_M)const;
769 // recherche près d'une ligne
770 void InitPresLigne(double& dist);
771 bool ExePresLigne(const double& dist,const Coordonnee& M)const;
772 // dans le cas particulier d'une ligne, on peut ordonner certaines listes résultantes / à la lignes
773 // il s'agit des noeuds et des pt d'integ: on les ordonne par rapport à leurs projections sur la
774 // lignes
775 void InitOrdonneLigne();
776 void OrdonneLigne(list <Noeud *>& list_noeud_restant,list <NBelemEtptInteg>& list_elemPtin_restant
777                ,const Enum_ddl & enu_ddl);
778 // recherche près d'un cercle
779 void InitPresCercle(double& dist);

```

```
784 bool ExePresCercle(const double& dist,const Coordonnee& M)const;
785
786 // recherche près d'un plan
787 void InitPresPlan(double& dist);
788 bool ExePresPlan(const double& dist,const Coordonnee& M)const;
789
790 // recherche près d'un cylindre
791 void InitPresCylindre(double& dist);
792 bool ExePresCylindre(const double& dist,const Coordonnee& M)const;
793
794 // recherche près d'une sphere
795 void InitPresSphere(double& dist);
796 bool ExePresSphere(const double& dist,const Coordonnee& M)const;
797
798 // recherche d'un coté d'un plan
799 void InitCotePlan();
800 // ramène true si tous les points sont du même coté que t_poi(1) du plan
801 bool ExeCotePlan(const Tableau <Coordonnee> & t_M)const;
802
803 // condition entre plans //
804 void InitEntrePlan();
805 // ramène true si tous les points sont entre les deux plans
806 bool ExeEntrePlan(const Tableau <Coordonnee> & t_M)const;
807
808 // condition entre plans _avec_distance //
809 void InitEntrePlan_avec_distance();
810 // ramène true si tous les points sont entre les deux plans
811 bool ExeEntrePlan_avec_distance(const Tableau <Coordonnee> & t_M)const;
812
813 // condition dans ou dehors un cylindre
814 void InitDansCylindre();
815 // ramène true si tous les points sont dans le cylindre
816 bool ExeDansCylindre(const Tableau <Coordonnee> & t_M)const;
817 // ramène true si tous les points sont à l'extérieur du cylindre
818 bool ExeOutCylindre(const Tableau <Coordonnee> & t_M)const;
819
820 // condition entre deux cylindre
821 void InitEntreCylindre();
822 // ramène true si tous les points sont entre les deux cylindres
823 bool ExeEntreCylindre(const Tableau <Coordonnee> & t_M)const;
824
825 // condition dans ou dehors d'une sphere
826 void InitDansSphere();
827 // ramène true si tous les points sont dans la sphere
828 bool ExeDansSphere(const Tableau <Coordonnee> & t_M)const;
829 // ramène true si tous les points sont à l'extérieur de la sphere
830 bool ExeOutSpheres(const Tableau <Coordonnee> & t_M)const;
831
832 // condition entre deux spheres concentriques
833 void InitEntreSpheres();
834 // ramène true si tous les points sont entre les deux spheres
835 bool ExeEntreSpheres(const Tableau <Coordonnee> & t_M)const;
836
837 // condition du même coté d'une droite (en 2D seulement)
838 void InitCoteDroite();
839 // ramène true si tous les points sont du même coté que t_poi(1) de la droite (en 2D uniquement)
840 bool ExeCoteDroite(const Tableau <Coordonnee> & t_M)const;
841
842 // condition entre 2 droites (en 2D seulement)
843 void InitEntreDroite();
844 // ramène true si tous les points sont entre les 2 droites (en 2D uniquement)
845 bool ExeEntreDroite(const Tableau <Coordonnee> & t_M)const;
846
847 // condition entre 2 droites (en 2D seulement) _avec_distance
848 void InitEntreDroite_avec_distance();
849 // ramène true si tous les points sont entre les 2 droites (en 2D uniquement)
850 bool ExeEntreDroite_avec_distance(const Tableau <Coordonnee> & t_M)const;
851
852 // condition dans ou dehors un cercle (en 2D seulement)
853 void InitDansCercle();
854 // ramène true si tous les points sont dans le cercle
855 bool ExeDansCercle(const Tableau <Coordonnee> & t_M)const;
856 // ramène true si tous les points sont à l'extérieur du cercle
857 bool ExeOutCercle(const Tableau <Coordonnee> & t_M)const;
858
859 // condition entre cercles concentriques (en 2D seulement)
860 void InitEntreCercles();
861 // ramène true si tous les points sont entre les deux cercles concentriques
862 bool ExeEntreCercles (const Tableau <Coordonnee> & t_M)const;
863
864 // acquisition d'un plan
865 Plan Acquisition_interactive_plan();
866
867 // acquisition d'un point
868 Coordonnee Acquisition_interactive_point();
869
870 // acquisition d'une droite
```

```

871 Droite Acquisition_interactive_droite();
872
873 // acquisition d'un vecteur=direction
874 Coordonnee Acquisition_interactive_vecteur();
875
876 // condition d'appartenance à une référence existante
877 void InitInRef();
878 // condition d'exclusion à une référence existante
879 void InitOutRef();
880 // ---- mise en place de la condition d'appartenance ou d'exclusion à des références existantes
881 // contrairement aux autres conditions, cette condition est prise en compte au moment de la création
    finale
882 // de la référence dans la méthode : EnregRef, via les méthodes du 2)
883 // 1) une fonction qui ne fait rien, mais est là pour que le tout fonctionne normalement
884 bool Exe_In_out_avecRefExistantes(const Tableau <Coordonnee> & t_M) const; // ne fait rien
885 // 2) fonctions qui réellement font quelque chose: utilisées dans la méthode EnregRef
886 void Exe_In_out_avecRefExistantes_N(list <Noeud *> & list_noeud_restant);
887 void Exe_In_out_avecRefExistantes_E(list <Element *> & list_element_restant);
888 void Exe_In_out_avecRefExistantes_G(list <NBelemEtptInteg> & list_elemPtin_restant);
889 void Exe_In_out_avecRefExistantes_F(list <NBelemEtFace> & list_elemFace_restant);
890 void Exe_In_out_avecRefExistantes_A(list <NBelemEtArete> & list_elemArrete_restant);
891 // void InterOuDiff_avecRefExistantes(const Tableau <Coordonnee> & t_M) const;
892
893 //----- méthodes particulières pour l'optimisation de largeur de bande
    -----
894
895 // une classe intermédiaire pour pouvoir classer les noeuds en fonction du degré
896 class Noeud_degre
897 { public: Noeud* noe; int degre;
898   Noeud_degre (): noe(NULL), degre(0) {};
899   Noeud_degre (Noeud* no, int deg): noe(no), degre(deg) {};
900   Noeud_degre (const Noeud_degre& no ): noe(no.noe), degre(no.degre) {};
901   ~Noeud_degre() {};
902   bool operator > (const Noeud_degre& a) const { return (this->degre > a.degre);};
903   bool operator >= (const Noeud_degre& a) const { return (this->degre >= a.degre);};
904   bool operator < (const Noeud_degre& a) const { return (this->degre < a.degre);};
905   bool operator <= (const Noeud_degre& a) const { return (this->degre <= a.degre);};
906   Noeud_degre& operator= (const Noeud_degre& de) { degre = de.degre; noe = de.noe; return
    (*this);};
907   };
908
909 // méthode static: c-a-d indépendante des données du maillage (elle est générale)
910 // calcul du point de départ, en fonction de conditions linéaires éventuelles
911 // il y a également calcul des voisins et de la descendance du noeud de retour
912 // s'il y a un pb, calcul_ok en retour est false
913 static Noeud* Point_de_depart(const Tableau<Element *> & tab_elem
914                               ,const Tableau<Noeud *> & tab_noe
915                               ,const Tableau <Tableau <Noeud*> *> tt_noeud_front
916                               ,Tableau < LaLIST_io <Noeud*> > & t_voisin
917                               , list < list < Noeud_degre > > & lis_descent
918                               ,const Tableau <Tableau <Condilinaire> > & condCLL, bool& calcul_ok);
919
920 // méthode static: c-a-d indépendante des données du maillage (elle est générale)
921 // calcul des voisins de tous les noeuds du maillage en fonction des éléments et de conditions
922 // linéaires éventuelles,
923 // en entrée/sortie t_voisin
924 // restriction: il faut que tab_noe contienne tous les noeuds référencés dans tab_elem et tt_t_condCLL
925 // s'il y a un pb, calcul_ok en retour est false
926 static Tableau < LaLIST_io <Noeud*> > & Voisins(const Tableau<Element *> & tab_elem
927                                                  ,const Tableau<Noeud *> & tab_noe
928                                                  ,Tableau < LaLIST_io <Noeud*> > & t_voisin
929                                                  ,const Tableau <Tableau <Condilinaire> > & t_t_condCLL, bool&
    calcul_ok);
930
931 // méthode static: c-a-d indépendante des données du maillage (elle est générale)
932 // calcul de la descendance d'un noeud noe,
933 // on considère que les voisins sont déjà correctement définis
934 // en entrée la liste qui est modifiée en interne et retournée en sortie
935 // le premier élément de la liste c'est la dernière descendance
936 // s'il y a un pb, calcul_ok en retour est false
937 static list < list < Noeud_degre > > & Descendance (const int& taille_tabnoeud
938                                                    ,Noeud * noe, Tableau < LaLIST_io <Noeud*> > & t_voisin
939                                                    , list < list < Noeud_degre > > & lient, bool& calcul_ok);
940 // méthode static: c-a-d indépendante des données du maillage (elle est générale)
941 // algorithme de Cuthill Mac Kee directe
942 // noe : le noeud de départ
943 // lis_descent : les descendants de noe, qui va être modifié dans le processus
944 // ramène une nouvelle numérotation (mais les numéros internes des noeuds ne sont pas changé)
945 static Tableau <Noeud*> Cuthill_Mac_Kee(const int& taille_tabnoeud, Noeud * noe
946                                       ,Tableau < LaLIST_io <Noeud*> > & t_voisin
947                                       , list < list < Noeud_degre > > & lis_descent);
948 // méthode static: c-a-d indépendante des données du maillage (elle est générale)
949 // calcul de la largeur de bande en noeuds
950 static int LargeurBandeEnNoeuds(const Tableau<Element *> & tab_elem);
951
952 // on s'occupe de mettre à jour les types de pb et les ddl types associés
953 void Mise_a_jour_type_pb_type_associe_ddl();

```

```

954
955
956 };
957 /// @} // end of group
958
959 #endif

```

## 7.353 Noeud.h

```

1 // FICHER : Noeud.h
2 // CLASSE : Noeud
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          23/01/97
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *****/
40 * Une instance de la classe Noeud est definie a partir d'un numero d'identification
41 * et du tableau de connexite des degres de liberte lies au noeud. Trois instances
42 * de la classe Coordonnee sont aussi associes a la classe Noeud (coordonnees initiales,
43 * aux instants t et t+dt).
44 *
45 *      *****
46 *      VERIFICATION:
47 *
48 *      ! date !   auteur   !           but
49 *      -----
50 *      !           !           !
51 *      *****
52 *      MODIFICATIONS:
53 *
54 *      ! date !   auteur   !           but
55 *      -----
56 *
57 *****/
58
59 // Une instance de la classe Noeud est definie a partir d'un numero d'identification
60 // et du tableau de connexite des degres de liberte lies au noeud. Trois instances
61 // de la classe Coordonnee sont aussi associes a la classe Noeud (coordonnees initiales,
62 // aux instants t et t+dt).
63
64
65 #ifndef NOEUD_H
66 #define NOEUD_H
67
68
69 #include <iostream>
70
71
72 #include "Coordonnee.h"
73 #include "Ddl.h"
74 #include "Tableau_T.h"
75 #include "ParaGlob.h"
76 #include "Vecteur.h"

```

```

77 #include "Tenseur.h"
78 #include "List_io.h"
79 #include "EnumTypeGrandeur.h"
80 #include "MathUtil.h"
81 #include "TypeQuelconque.h"
82 #include "Ddl_etendu.h"
83 #include "Enum_liaison_noeud.h"
84
85
86 /// @addtogroup Les_Maillages
87 /// @{
88 ///
89
90 //-----
91 //!      Posi_ddl_noeud: un conteneur qui associe numéro de noeud, de maillage, et enu ddl
92 //-----
93 /// \author      Gérard Rio
94 /// \version    1.0
95 /// \date      23/01/97
96
97
98 // ----- class de stockage -----
99 class Posi_ddl_noeud
100 { // surcharge lecture écriture
101     friend istream & operator » (istream &, Posi_ddl_noeud &);
102     friend ostream & operator « (ostream &, const Posi_ddl_noeud &);
103     public :
104     // constructeurs
105     Posi_ddl_noeud () : nb_maillage(-1),nb_noeud(-1),enu(NU_DDL) {};
106     Posi_ddl_noeud (int aa, Enum_ddl enuu,int bb) :
107         nb_maillage(aa),nb_noeud(bb),enu(enuu) {};
108     Posi_ddl_noeud (const Posi_ddl_noeud & a):
109         nb_maillage(a.nb_maillage),nb_noeud(a.nb_noeud),enu(a.enu) {};
110     // opérateur
111     Posi_ddl_noeud& operator = ( const Posi_ddl_noeud & a)
112     {nb_maillage=a.nb_maillage;nb_noeud=a.nb_noeud;enu=a.enu; return *this;};
113     bool operator == ( const Posi_ddl_noeud & a)
114     { if((nb_maillage==a.nb_maillage) && (nb_noeud==a.nb_noeud)
115         && (enu==a.enu)) return true; else return false; };
116     bool operator != ( const Posi_ddl_noeud & a)
117     { if((*this) == a) return false; else return true; };
118     // Affiche les donnees liees au noeud
119     void Affiche (ofstream& sort) const;
120     // récup des données en lecture écriture
121     int& Nb_maillage() {return nb_maillage;};
122     int& Nb_noeud() {return nb_noeud;};
123     Enum_ddl& Enu() {return enu;};
124     // récup des données en lecture uniquement
125     const int& Const_Nb_maillage() const {return nb_maillage;};
126     const int& Const_Nb_noeud() const {return nb_noeud;};
127     const Enum_ddl& Const_Enu() const {return enu;};
128
129     protected:
130     // données
131     int nb_maillage; // le numéro du maillage
132     int nb_noeud; // le numéro du noeud
133     Enum_ddl enu; // l'identifieur du ddl
134     };
135 /// @} // end of group
136
137
138 /// @addtogroup Les_Maillages
139 /// @{
140 ///
141
142 //-----
143 //!      Noeud: un noeud
144 //-----
145 /// \author      Gérard Rio
146 /// \version    1.0
147 /// \date      23/01/97
148
149
150 class Noeud
151 {
152     // surcharge de l'operator de lecture, avec typage
153     friend istream & operator » (istream &, Noeud &);
154     // surcharge de l'operator d'écriture avec typage
155     friend ostream & operator « (ostream &, const Noeud &);
156
157     public :
158
159
160     // CONSTRUCTEURS :
161
162     // IMPORTANT!!! : l'ordre de succession des ddl doit etre conforme
163     // aux infos de l'enumeration (cf enum_ddl)

```



```

164
165 // Constructeur par défaut
166 Noeud (int num_id=-3,int num_maill=0);
167
168 // Constructeur fonction d'un numero d'identification et de la
169 // dimension de l'espace du probleme et du nb de maillage
170 Noeud (int num_id,int dimension,int num_maill);
171
172 // Constructeur fonction d'un numero d'identification et des coordonnees
173 // initiales et du nb de maillage
174 Noeud (int num_id,const Coordonnee& c0,int num_Mail);
175
176 // Constructeur fonction d'un numero d'identification, des coordonnees
177 // initiales et du tableau des degres de liberte, et du nb de maillage
178 // les ddl t=0 et t sont initialises avec les valeurs du tableaux
179 // par défaut les tableaux de ddl a 0 et t sont initialises mais pas
180 // a t+dt,
181 Noeud (int num_id,const Coordonnee& c0,const Tableau<Ddl>& tab,int num_Mail);
182
183 // Constructeur fonction d'un numero d'identification, des coordonnees
184 // initiales, a un instant t, et du tableau des degres de liberte, et du nb de maillage
185 // les ddl t=0 sont initialises avec les valeurs du tableaux
186 // par défaut les tableaux de ddl a 0 et t sont initialises mais pas
187 // a t+dt
188 // les ddl correspondant a Xi sont ajoute s'ils n'existaient pas
189 Noeud (int num_id,const Coordonnee& c0,const Coordonnee& c1,const Tableau<Ddl>& tab,int
num_Mail);
190
191 // Constructeur fonction d'un numero d'identification, de trois instances
192 // de Coordonnee et du tableau des degres de liberte, et du nb de maillage
193 Noeud (int num_id,const Coordonnee& c0,const Coordonnee& c1,const Coordonnee& c2,
194        const Tableau<Ddl>& tab,int num_Mail);
195
196 // Constructeur de copie
197 Noeud (const Noeud& nd);
198
199
200 // DESTRUCTEUR :
201 ~Noeud ();
202
203
204 // METHODES :
205
206 // Affiche à l'écran les donnees liees au noeud
207 // le niveau différent de 0 permet d'accéder à plus d'info
208 void Affiche(int niveau=0) const;
209 // Affiche dans sort les donnees liees au noeud
210 void Affiche(ofstream& sort) const;
211
212 inline void Change_num_noeud(int nouveau_num)
213 // Modifie le numero d'identification du noeud
214 // ATTENTION : Les modifications liees au changement de numero du noeud
215 // sont a la charge de l'utilisateur.
216 { num_noeud=nouveau_num; };
217
218 inline void Change_num_Mail(int nouveau_num)
219 // Modifie le numero de maillage du noeud
220 { num_Mail=nouveau_num; };
221
222 // récup du type de liaison entre noeud éventuel
223 // pour le cas d'assemblage nb_assemb
224 inline Enum_liaison_noeud Enu_liaison(int nb_assemb) const
225 {if (tab_enu_liaison != NULL) return (*tab_enu_liaison)(nb_assemb);
226 else return PAS_LIER;};
227
228 // récup d'un pointeur de tableau des ddl affectés par une liaison entre noeud
229 // pour le cas d'assemblage nb_assemb, NULL s'il n'y en a pas
230 inline const Tableau <Posi_ddl_noeud >* Ddl_Noeud_liier(int nb_assemb) const
231 {if (tab_ddl_liier != NULL) return &((*tab_ddl_liier)(nb_assemb));
232 else return NULL; };
233 // modification du type de liaison entre noeud éventuel avec init des tableaux de liaison
234 // pour le cas d'assemblage nb_assemb
235 void Change_Enu_liason(int nb_assemb, Enum_liaison_noeud enu_liai
236 ,const Tableau <Posi_ddl_noeud>& tab_liier);
237 // suppression de tous les liaisons noeuds
238 void Suppression_tous_liaisons_noeuds();
239
240 // Modifie les valeurs des coordonnees coord0
241 inline void Change_coord0(const Coordonnee& nouveau_coord0)
242 { *coord0=nouveau_coord0;};
243
244 // insert des coordonnees coord1 si elles n'existent pas encore
245 // permet de définir des coordonnées à t, dans ce cas des ddl XI sont inclus
246 // et sont mis inactif par défaut
247 void Insert_coord1(const Coordonnee& nouveau_coord1);
248 // idem mais sans paramètre indique que l'on initialise aux coordonnées à t=0
249 // s'ils n'existent pas sinon on ne fait rien

```

```

250 void Insert_coord1();
251
252 // Modifie les valeurs des coordonnees coord1
253 void Change_coord1(const Coordonnee& nouveau_coord1);
254
255
256 // Modifie les valeurs des coordonnees coord2
257 inline void Change_coord2(const Coordonnee& nouveau_coord2)
258 { *coord2=nouveau_coord2; };
259
260
261 // Modifie les valeurs des coordonnees coord2 par l'ajout d'un delta
262 inline void Ajout_coord2(const Coordonnee& delta_coord2)
263 { *coord2=*coord2+delta_coord2; };
264
265
266 inline const Coordonnee& Coord0() const
267 // Retourne les coordonnees initiales du noeud
268 { return *coord0; };
269
270 // test si les coordonnées à t=0 existent
271 inline bool ExisteCoord0() const
272 { if (coord0 == NULL) return false; else return true;};
273
274 // Retourne les coordonnees coord1 du noeud
275 // uniquement pour la lecture
276 inline Coordonnee Coord1() const
277 { int nbcoo = coord1.Taille();
278 Coordonnee coo(nbcoo);
279 for (int i=1;i<=nbcoo;i++)
280 coo(i) = *(coord1(i));
281 return coo;
282 };
283
284 // test si les coordonnées à t existent
285 inline bool ExisteCoord1() const
286 { if (coord1.Taille() == 0) return false; else return true;};
287
288 // Retourne les coordonnees coord2 du noeud
289 inline const Coordonnee& Coord2() const
290 { return *coord2; };
291
292 // retourne la valeur absolu du maximum de variation de coordonnée entre t et tdt
293 inline double Max_var_coor_t_a_tdt() const
294 { double ret=0.;
295 if (ExisteCoord2() && ExisteCoord1())
296 {int nbcoo=coord2->Dimension();
297 for (int i=1;i<=nbcoo;i++) ret = Max(Dabs((*coord2)(i) - *(coord1(i))),ret);
298 };
299 // sinon cela veut dire que le maillage n'a pas bougé (cas d'un solide fixe)
300 return ret;
301 };
302
303 // test si les coordonnées à tdt existent
304 inline bool ExisteCoord2() const
305 { if (coord2 == NULL) return false; else return true;};
306
307 // Retourne le nombre de tous les degres de liberte du noeud
308 // qu'ils soient actifs ou non, fixes ou non
309 inline int Nombre_ddl() const
310 { return tab_ddl.Taille(); };
311
312 // Retourne le nombre de variables degres de liberte du noeud
313 // qui sont actuellement actives
314 inline int Nombre_var_ddl_actives() const
315 { return tab_var_actives.Taille();};
316
317 // Retourne le nombre de variables degres de liberte du noeud
318 // qui sont actives pour un cas d'assemblage donné.
319 // (ces ddl peuvent ne pas être actuellement actif !!)
320 inline int NB_ddl_actif_casAssemb (int nb_assemb) const
321 { Tableau <int >& tib = (*t_enum_s(nb_assemb));
322 return (tib(tib.Taille()));};
323
324 // ramene le nombre de variables ddl actives pour un type de ddl donné,
325 // c'est-à-dire 1 ou la dimension, suivant que le type dépend
326 // ou pas de la dimension
327 int Nombre_var_ddl_actives (Enum_ddl en) const ;
328
329 // retourne la liste de tous les types de ddl actuellement utilisé
330 // par le noeud (actif ou non)
331 // absolue: indique si oui ou non on sort les tenseurs dans la base absolue ou une base particulière
332 List_io <Enum_ddl> Les_type_de_ddl(bool absolue);
333 // retourne la liste de tous les Ddl_enum_etendu disponibles
334 List_io <Ddl_enum_etendu> Les_type_de_ddl_etendu(bool absolue);
335 // retourne la liste de tous les TypeQuelconque disponibles
336 List_io <TypeQuelconque> Les_TypeQuelconque(bool absolue);

```

```

337
338 // récupération d'une liste d'info
339 // le tableau de retour à la taille de li_enu_scal et contient
340 // les valeurs correspondantes aux Ddl_enum_etendu stockées au noeud
341 // en fait ici on cumule les ddl pur "et" les ddl_étendue,
342 // li_quelc : est modifié par les valeurs contenues au noeud
343 Tableau <double> Valeur_multi_et_Tensorielle
344 (const List_io <Ddl_enum_etendu>& li_enu_scal, List_io <TypeQuelconque >& li_quelc) const;
345
346 // Retourne le numero d'identification du noeud
347 inline int Num_noeud() const
348 { return num_noeud; };
349
350 // Retourne le numero de maillage auquel appartient le noeud
351 inline int Num_Mail() const
352 { return num_Mail; };
353
354 // Retourne la dimension des coordonnées
355 inline int Dimension() const
356 { return coord0->Dimension(); };
357
358 // surcharge de l'affectation entre deux noeuds
359 // attention, le pointeur d'assemblage de this est identique a celui de n
360 // il faut donc l'updater pour pouvoir l'utiliser!!
361 Noeud& operator= (const Noeud& n);
362
363 // acces en lecture au pointeur de position pour l'assemblage
364 // i : donne le numéro du cas d'assemblage associé
365 inline int PosiAssemb(int i) const
366 { return posiAssemb(i);};
367
368 // initialisation d'un ou de plusieurs nouveau cas d'assemblage
369 void InitNouveauCasAssemb(int nb_cas);
370
371 // enregistrement de l'ordre des variables ddl actives, ceci pour un cas d'assemblage
372 // donné, cette fonction permet ensuite d'utiliser la fonction
373 // Position_ddl qui donne la position d'un ddl pour un cas d'assemblage
374 // donné
375 void Enreg_ordre_variable_ddl_actives(int nb_assemb);
376
377 // position d'un ddl par rapport à un cas d'assemblage donné (de 1 à ..)
378 // il n'y a pas de vérification que le ddl en question est actif ou pas
379 // cette fonction ne peut être utilisé, que après l'utilisation
380 // de la fonction Enreg_ordre_variable_ddl_actives
381 // si l'énuméré n'existe pas dans la liste des ddl actif du cas d'assemblage
382 // on retourne -1
383 inline int Position_ddl(Enum_ddl enu, int nb_assemb) const
384 { int pointe = (*(t_enum_s(nb_assemb)))(enu);
385   if (pointe == 0) return -1; else return pointe; }
386
387 // modification du pointeur de position du noeud pour l'assemblage
388 // i : donne le numéro du cas d'assemblage associé
389 inline void ChangePosiAssemb(int a, int i)
390 { posiAssemb(i) = a;};
391
392 // ramène le pointeur d'assemblage pour un ddl donné, et un cas d'assemblage
393 // le résultat correspond à la position globale d'assemblage du noeud +
394 // la position du ddl dans la liste des ddl actifs du cas d'assemblage
395 // si l'énuméré ne correspond pas à un ddl actif du cas d'assemblage
396 // on retourne -1, dans le cas ok, c'est un nombre de 1 à ...
397 inline int Pointeur_assemblage(Enum_ddl enu, int nb_assemb) const
398 { int pointe = (*(t_enum_s(nb_assemb)))(enu);
399   if (pointe == 0) return -1;
400   return pointe + posiAssemb(nb_assemb); };
401
402 // le noeud peut contenir éventuellement une base rattachée
403 // récupération de la base, si le pointeur en retour est null
404 // cela signifie que la base n'existe pas
405 // a) en constant, la base actuelle
406 const BaseB* Const_BaseB_Noeud() const {return baseB;};
407 // b) en constant, la base à t
408 const BaseB* Const_BaseB_Noeud_t() const {return baseB_t;};
409 // c) en constant, la base initiale
410 const BaseB* Const_BaseB_Noeud_0() const {return baseB_0;};
411
412 // d) en lecture écriture, la base actuelle
413 BaseB* BaseB_Noeud() const {return baseB;};
414 // e) en lecture écriture, la base à t
415 BaseB* BaseB_Noeud_t() const {return baseB_t;};
416 // f) en lecture écriture, la base à 0
417 BaseB* BaseB_Noeud_0() const {return baseB_0;};
418
419 // test si le noeud est complet
420 // = 1 tout est ok, =0 element incomplet
421 int TestCompleet() const;
422
423 // ajout de ddl

```

```

424 // 1) s'il n'y a pas de ddl il y a initialisation
425 // 2) si les ddl existent deja, on l'egalise au ddl passé en para
426 // 3) lorsque le ddl appartient à une famille, tous les ddl de la famille sont
427 // ajoutées, les ddl supplémentaires ont une valeurs par défaut nulle
428 void PlusDdl(Ddl& a);
429
430 // ajout d'un tableau de ddl
431 // 1) s'il n'y a pas de ddl il y a initialisation
432 // 2) si les ddl existent deja, on egalise avec les ddl passés en par
433 // 3) lorsque un nouveau ddl appartient à une famille, et que tous les ddl de la famille
434 // ne sont pas présents dans les ddl passés en paramètre, tous les ddl manquant de la famille
sont
435 // ajoutées, avec une valeurs par défaut nulle
436 void PlusTabDdl(Tableau<Ddl>& ta);
437
438 // ajout du tableau de ddl d'un autre noeud
439 void PlusTabDdl(const Noeud & noe);
440
441 // modification du blocage ou non d'un ddl
442 inline void Change_fixe(Enum_ddl en, bool val)
443 { tab_ddl((*pos_enum)(en)).Change_fixe(val); };
444 inline void Change_fixe(int nb, bool val)
445 { tab_ddl(nb).Change_fixe(val); };
446 // test pour savoir si le ddl est fixé ou pas
447 inline bool Ddl_fixe(Enum_ddl en) const
448 { return tab_ddl((*pos_enum)(en)).Fixe(); };
449 // test pour savoir si le ddl est SURFIXE ou SOUSLIBRE
450 // (c-a-d plusieurs fois fixé ou libre à la suite)
451 inline bool Ddl_sous_ou_sur_fixe(Enum_ddl en) const
452 { return tab_ddl((*pos_enum)(en)).SousSurFixe(); };
453 // remise à normal si le ddl est en souslibre ou surfixe
454 // s'il est en souslibre il passe en libre et s'il est en surfixe il passe en fixe
455 inline void Retour_a_la_normal_sur_ou_sous_fixe(Enum_ddl en)
456 { tab_ddl((*pos_enum)(en)).Retour_normal_sur_ou_sous_fixe(); };
457
458 // modification du statut d'un ddl
459 // si est inconnu, aucune action
460 void Met_hors_service(Enum_ddl en);
461 void Met_en_service(Enum_ddl en);
462 // modification du statut d'un tableau de ddl
463 // si un des elements du tableau est inconnu, aucune action
464 void Met_hors_service(const Tableau<Enum_ddl>& taben);
465 void Met_en_service(const Tableau<Enum_ddl>& taben);
466 // modification du statut de tous les ddl actuellement présent
467 // y compris les données
468 // si un des elements du tableau est inconnu, aucune action
469 void Met_hors_service();
470 void Met_en_service();
471 // idem mais restreint uniquement aux ddl (pas les données)
472 void Met_hors_service_ddl();
473 void Met_en_service_ddl();
474
475 // test si un ddl est en service ou pas
476 bool En_service (Enum_ddl en) const { return tab_ddl((*pos_enum)(en)).Service(); };
477 // test si un ddl est une variable ou pas
478 bool UneVariable(Enum_ddl en) const { return tab_ddl((*pos_enum)(en)).UneVariable(); };
479 // changement du statut de variable à donnée d'un ddl
480 void ChangeVariable_a_Donnee(Enum_ddl en)
481 { tab_ddl((*pos_enum)(en)).ChangeVariable_a_Donnee(); MiseAJourActif(); };
482 // changement du statut de donnée à variable d'un ddl
483 void ChangeDonnee_a_Variable(Enum_ddl en)
484 { tab_ddl((*pos_enum)(en)).ChangeDonnee_a_Variable(); MiseAJourActif(); };
485 // changement du statut de variable à donnée pour un tableau d'enum de ddl
486 void ChangeVariable_a_Donnee(const Tableau<Enum_ddl>& taben);
487 // changement du statut de donnée à variable pour un tableau d'enum de ddl
488 void ChangeDonnee_a_Variable(const Tableau<Enum_ddl>& taben);
489 // changement de toutes les conditions données (service, variable, fixage ..)
490 // selon le tableau de ddl passé en paramètre
491 // par contre la valeur n'est pas utilisé donc la valeur actuelle reste inchangé
492 void ChangeToutesLesConditions(const Tableau<Ddl>& ta);
493 // changement de statu des ddl d'une combinaison, (cf. Enum_ddl)
494 // les ddl de la combinaison, prennent le même statut que celui
495 // actuellement stocké, correspondant à enuta
496 // cas est la combinaison,
497 void ChangeStatut(int cas, Enum_ddl enuta);
498 // changement de statu des ddl d'une combinaison, (cf. Enum_ddl)
499 // les ddl de la combinaison, prennent le même statut que celui de enuta
500 // cas est la combinaison,
501 void ChangeStatut(int cas, Enum_boolddl enubold);
502
503 // ----- manipulation de ddl identifié par un énuméré -----
504 // *****!!!!***** attention il n'y a pas vérification de l'existence
505 // de ce type énuméré dans des ddl du noeud
506
507 // change la valeur du ddl identifie par en ,
508 inline void Change_val_0(Enum_ddl en, double val) // t=0
509 { tab_0((*pos_enum)(en)) = val; };

```

```

510     inline void Change_val_t(Enum_ddl en,double val) // t=t
511         { tab_ddl((*pos_enum)(en)).Valeur() = val;};
512     inline void Change_val_tdt(Enum_ddl en,double val) // t=tdt
513         { tab_tdt((*pos_enum)(en)) = val;};
514     // incremente la valeur du ddl identifie par en ,
515     inline void Ajout_val_0(Enum_ddl en,double val) // t=0
516         { tab_0((*pos_enum)(en)) += val; };
517     inline void Ajout_val_t(Enum_ddl en,double val) // t=t
518         { tab_ddl((*pos_enum)(en)).Valeur() += val;};
519     inline void Ajout_val_tdt(Enum_ddl en,double val) // t=tdt
520         { tab_tdt((*pos_enum)(en)) += val;};
521     // retourne la valeur du ddl identifie par en
522     inline double Valeur_0(Enum_ddl en) const // t=0
523         { return tab_0((*pos_enum)(en)); };
524     inline double Valeur_t(Enum_ddl en) const // t=t
525         { return tab_ddl((*pos_enum)(en)).Valeur();};
526     inline double Valeur_tdt(Enum_ddl en) const // t=tdt
527         { return tab_tdt((*pos_enum)(en));};
528     // Retourne le ddl à t=0 identifie par enu, en lecture
529     inline Ddl Ddl_noeud_0(Enum_ddl enu) const
530         {return Ddl(enu,tab_0((*pos_enum)(enu)),tab_ddl((*pos_enum)(enu)).Retour_Fixe());};
531     // Retourne le ddl à t=t identifie par en, en lecture
532     inline Ddl Ddl_noeud_t(Enum_ddl enu) const
533         {return Ddl(enu,tab_ddl((*pos_enum)(enu)).Valeur(),tab_ddl((*pos_enum)(enu)).Retour_Fixe());};
534     // Retourne le ddl à t=tdt identifie par en, en lecture
535     inline Ddl Ddl_noeud_tdt(Enum_ddl enu) const
536         {return Ddl(enu,tab_tdt((*pos_enum)(enu)),tab_ddl((*pos_enum)(enu)).Retour_Fixe());};
537
538     // indique que l'on va utiliser les ddl en 0, t et tdt
539     // les grandeurs en tdt sont cree et initialisees par default
540     void Travail_tdt();
541
542     // indique que l'on va utiliser les ddl en 0, t
543     // si les grandeurs en tdt existaient, elles sont supprimées
544     void Travail_t();
545
546     // teste si un ddl existe
547     inline bool Existe_ici(const Enum_ddl en) const
548         { if (Existe(en) == 0)
549             return false;
550           else
551             return true;
552         };
553
554     // mise a zero de tous les variables ddl actives
555     // sauf dans le cas de coordonnees entrainees. dans ce dernier
556     // cas les coordonnees correspondants a Xi pour t et t+dt
557     // sont mis a la valeur de celles de t=0
558     // a moins que le paramètre booléen est mis à false,
559     // dans ce cas les coordonnées à t sont inchangées
560     void ZeroVariablesDdl(bool cas);
561
562     // actualisation des ddl actifs (variables et donnees) de t+dt vers t
563     // seuls sont actualisée, les ddl actifs
564     void TdtversT();
565     // actualisation des ddl actifs (variables et donnees) de t vers t+dt
566     // seuls sont actualisée, les ddl actifs
567     void TversTdt();
568
569     // retourne true si le tableau a t+dt existe , false sinon
570     inline bool Tdt()
571         {if (tab_tdt.Taille() != 0) return true; else return false; };
572
573     // retourne dans le vecteur d'entrée les réels représentant les contraintes
574     // si les degrés de liberté contrainte existent
575     // sinon retourne false, en ne modifiant pas l'entrée
576     bool Contrainte(Vecteur& Sig);
577
578     // retourne dans le tenseur d'entrée les réels représentant les contraintes
579     // si les degrés de liberté contrainte existent
580     // sinon retourne l'entrée sans modification
581     // le tenseur est choisit mixte pour représenter tous les cas
582     // !!! important : le tenseur Sig doit avoir la bonne dimension
583     // en entrée, car il n'est pas redimensionné dans la méthode
584     TenseurHB& Contrainte(TenseurHB& Sig);
585
586     // lecture du noeud dans la mémoire
587     void Lecture (UtilLecture *entreePrinc);
588     // sortie du schemaXML: en fonction de enu
589     static void SchemaXML_Noeud(ofstream& sort,const Enum_IO_XML enu);
590
591     // affichage et definition interactive des commandes
592     // coor: des coordonnees fixée pour un affichage different pour chaque noeud
593     void Info_commande_Noeud(UtilLecture * entreePrinc,Coordonnee& coor,int nb_du_noeud);
594
595     // lecture sur le flot d'entée de déplacement
596     // et mise à jour des positions à t et t+dt correspondantes

```

```

597 void LectureDeplacements(UtilLecture *entreePrinc);
598
599 //----- lecture écriture dans base info -----
600 // cas donne le niveau de la récupération
601 // = 1 : on récupère tout
602 // = 2 : on récupère uniquement les données variables (supposées comme telles)
603 // ( ici on ne redimensionne pas,
604 // cela signifie que le nombre de ddl augmente mais ne diminue pas !!)
605 void Lecture_base_info(ifstream& ent,int cas);
606 // cas donne le niveau de sauvegarde
607 // = 1 : on sauvegarde tout
608 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
609 void Ecriture_base_info(ofstream& sort,int cas);
610
611 // ----- manipulation de types quelconques stockées aux noeuds -----
612 // les opérations dans les types quelconques ne sont pas faites par le noeud
613 // le noeud gère les grandeurs, mais ne les modifie pas directement
614
615 // ajout d'un type quelconque au noeud (il s'agit ici de définir le conteneur)
616 // dans le cas où les types existent déjà, on ne fait rien
617 // cas d'un tableau
618 void AjoutTabTypeQuelconque(const Tableau <TypeQuelconque > & tab_t_quel);
619 // cas d'une seule grandeur quelconque
620 void AjoutUnTypeQuelconque(const TypeQuelconque & t_quel);
621 // supprime une grandeur quelconque (si elle est présente)
622 // opération aussi longue que l'ajout !!
623 void SupprimeUnTypeQuelconque(const TypeQuelconque & t_quel);
624 // supprime un tableau de grandeurs quelconques (si elles sont présentes)
625 // opération aussi longue que l'ajout !!
626 void SupprimeTabTypeQuelconque(const Tableau <TypeQuelconque > & tab_t_quel);
627 // ramène un booléen indiquant si la grandeur quelconque existe ou pas
628 inline bool Existe_ici(TypeQuelconque_enum_etendu en) const
629 { if (Existe(en) == 0) return false; else return true; };
630 // récupération d'une grandeur quelconque pour modification
631 // après l'appel de cette méthode, la grandeur quelconque est réputée updaté
632 TypeQuelconque& ModifGrandeur_quelconque(TypeQuelconque_enum_etendu a);
633 // récupération d'une grandeur quelconque pour lecture uniquement
634 inline const TypeQuelconque& Grandeur_quelconque(TypeQuelconque_enum_etendu enu) const
635 { return (tab_type_quel((*pos_Quelconque)(enu.Position()))); };
636 // ramène le nombre de fois où la grandeur quelconque a été updaté
637 inline int Grandeur_quelconque_update(TypeQuelconque_enum_etendu enu) const
638 { return (update_type_quel((*pos_Quelconque)(enu.Position()))); };
639 // signale que toutes les grandeurs quelconques sont non-updatées
640 // en particulier les grandeurs en question ne seront pas sauvegarder dans base_info
641 void Mise_non_update_grandeurs_quelconques();
642 // signale qu'une grandeur quelconque est non-updatées
643 // en particulier la grandeur en question ne sera pas sauvegarder dans base_info
644 void Mise_non_update_grandeurs_quelconques(TypeQuelconque_enum_etendu enu);
645
646 // ----- manipulation de types Ddl_etendu stockées aux noeuds -----
647 // les opérations dans les types Ddl_enum_etendu ne sont pas faites par le noeud
648 // le noeud gère les grandeurs, mais ne les modifie pas directement
649
650 // ajout d'un type Ddl_enum_etendu au noeud (il s'agit ici de définir le conteneur)
651 // dans le cas où les types existent déjà, on ne fait rien
652 // cas d'un tableau
653 void AjoutTabDdl_etendu(const Tableau <Ddl_enum_etendu > & tab_ddletendu);
654 // cas d'un seul Ddl_enum_etendu
655 void AjoutUnDdl_etendu(const Ddl_enum_etendu & ddletendu);
656 // supprime un Ddl_enum_etendu (si elle est présente)
657 // opération aussi longue que l'ajout !!
658 void SupprimeUnDdl_etendu(const Ddl_enum_etendu & ddletendu);
659 // supprime un tableau de Ddl_enum_etendu (si ils sont présents)
660 // opération aussi longue que l'ajout !!
661 void SupprimeTabDdl_etendu(const Tableau <Ddl_enum_etendu > & tab_ddletendu);
662 // ramène un booléen indiquant si le Ddl_enum_etendu existe ou pas
663 inline bool Existe_ici_ddlEtendu(const Ddl_enum_etendu& ddletendu) const
664 { if (Existe_ddlEtendu(ddletendu) == 0) return false; else return true; };
665 // récupération d'un Ddl_etendu pour modification
666 // après l'appel de cette méthode, le Ddl_enum_etendu est réputée updaté
667 Ddl_etendu& ModifDdl_etendu(const Ddl_enum_etendu& addletendu);
668 // récupération d'un Ddl_etendu pour lecture uniquement
669 inline const Ddl_etendu& DdlEtendue(const Ddl_enum_etendu& ddlenumetendu) const
670 { return (tab_ddletendu((*pos_ddletendu)(ddlenumetendu.Position()))); };
671 // ramène le nombre de fois que le ddl a été updaté
672 inline int DdlEtendue_update(const Ddl_enum_etendu& ddlenumetendu)
673 { return (update_ddletendu((*pos_ddletendu)(ddlenumetendu.Position()))); };
674 // signale que tous les Ddl_etendu sont non-updatées ( mise à zéro du nombre d'update)
675 // en particulier les grandeurs en question ne seront pas sauvegarder dans base_info
676 void Mise_non_update_Ddl_etendu();
677 // signale qu'un Ddl_etendu est non-updatées
678 // en particulier le Ddl_etendu en question ne sera pas sauvegarder dans base_info
679 void Mise_non_update_Ddl_etendu(Ddl_enum_etendu ddlenumetendu);
680
681 //=====
682 protected :
683 //=====

```

```

684
685 // donnees de base
686
687     int num_noeud; // numero d'identification dans le maillage
688     int num_Mail; // numero de maillage auquel appartient le noeud
689     Tableau <Ddl > tab_ddl; // tableau des degres de liberte
690         // les valeurs sont celles a l'instant t
691     Coordonnee* coord0; // coordonnee initiale
692
693 // liaison entre noeuds : il s'agit d'un stockage uniquement: l'utilisation et la manipulation
694 // est à la charge des classes appelantes
695 Tableau <Enum_liaison_noeud>* tab_enu_liaison; // tab_enu_liaison(i) définit un type éventuel
696 // de liaison pour le cas d'assemblage i
697 Tableau < Tableau <Posi_ddl_noeud> >* tab_ddl_liier; // tab_ddl_liier(i) donne la liste des ddl liés
698 // avec les noeuds associés
699
700 // donnees evoluant avec le calcul
701 // les differents tableaux si-dessous peuvent ne pas etre affecte
702 // cela depend du pb, certaines valeurs ont plusieurs meme nom
703 // comme coord0 coord1 et coord2, mais un seul endroit de stockage
704 Tableau<double> tab_0; // valeurs a l'instant 0 des ddl
705 Tableau<double> tab_tdt; // valeurs a l'instant t+dt des ddl
706 Tableau <double*> coord1; // coordonnee a un instant t
707 // pour l'exterieur le tableau est transforme en coordonnees mais
708 // en interne comme les infos peuvent etre en double avec le tableau de ddl
709 // et non contigues, on utilise un adressage indirecte
710 // Sa taille est différente de 0 si les coordonnées à t varie ou
711 // sont susceptibles de varier.
712 // a priori si coord1 est affecté c'est uniquement en pointant sur tab_ddl
713 // donc il ne crée pas avec new
714 Coordonnee* coord2; // coordonnee a un instant t+dt
715 // tableau pour une gestion rapide des ddl en service
716 // et que l'on appelle par leur identification
717 Tableau<short int> tab_var_actives; // les numéros des variables ddl actives
718 Tableau<short int> tab_actif; // les numéros des donnees et variables ddl actives
719
720 Tableau <int> posiAssemb; // tableau de pointeurs de position pour l'assemblage
721 // defini l'endroit ou est mis le premier ddl ds les seconds
722 //membres et raideurs globaux
723
724 // le noeud peut contenir éventuellement une base rattachée
725 BaseB* baseB_0; // actuelle
726 BaseB* baseB_t; // et à t
727 BaseB* baseB; // actuelle
728
729     Tableau <int> tab_nb_Elem; // tableau éventuel des numéros d'éléments qui contiennent le noeud
730
731 // --- cas des types quelconques
732 Tableau <TypeQuelconque > tab_type_quel; // grandeur quelconques stockées aux noeuds
733 Tableau <int> update_type_quel; // indique si la grandeur quelconque est update ou pas
734 static short int posi_type_quel; // utilisée par la méthode Existe_quelconque(..
735 // pour memoriser la dernière position accédée
736 int nbTypeQ_update; // indique le nombre de type quelconque updaté actuel (pour optimiser
vitesse)
737
738 // --- cas des ddl étendus
739 Tableau <Ddl_etendu > tab_ddletendu; // ddl étendu stockées aux noeuds
740 Tableau <int> update_ddletendu; // indique si le ddl étendu est update ou pas
741 static short int posi_ddletendu; // utilisée par la méthode Existe_ddletendu(..
742 // pour memoriser la dernière position accédée
743 int nbddletendu_update; // indique le nombre de ddl étendu updaté actuel (pour optimiser
vitesse)
744
745 // --- pour un acces rapide à un ddl donné repéré par un type énuméré on met en place ---
746 // une liste de type de succession d'enum ddl
747 // en ce servant de l'équivalence enum int
748 // il y a une liste pour tous les noeuds
749 static list <Tableau <int > > list_tab_enu;
750 // et un type de tableau pour chaque noeud
751 list <Tableau <int > >::iterator pos_enu ;
752 // (* posi_enu)(enum) = l'indice du ddl dans les
753 // tableau de ddl : tab_ddl, tab_0 etc..
754 // la liste n'est jamais diminuée, par contre comme les noeuds
755 // sont construit d'une manière automatique et systématique on
756 // suppose qu'il y aura en fait très peu d'élément dans la liste
757
758 // --- pour déterminer rapidement la position dans la liste
759 // des ddl actifs, d'un ddl donné repéré par un type énuméré
760 // même principe que pour list_tab_enu
761 static List_io <Tableau <int > > list_tab_posi_actif;
762 // et un tableau de pointeur d'élément de la liste,
763 // chaque tableau étant relatif à un type de combinaison de ddl actif
764 Tableau < List_io <Tableau <int > >::iterator > t_enu_s ;
765
766 // ---- idem pour les types quelconques repéré par un type énuméré
767 // même principe que pour list_tab_enu
768 static List_io <Tableau <int > > list_tab_typeQuelconque;

```



```

768 // et un type de tableau pour chaque noeud
769 list <Tableau <int > >::iterator pos_Quelconque ;
770
771 // ---- idem pour les types Ddl_etendu repéré par un type énuméré
772 // même principe que pour list_tab_enum
773 static List_io <Tableau <int > > list_tab_ddletendu;
774 // et un type de tableau pour chaque noeud
775 list <Tableau <int > >::iterator pos_ddletendu ;
776
777 // fonctions internes
778
779 // liaison des ddl et des deplacement a t et 0
780 // nb = le nb du début des deplacement dans les ddl
781 void Liaison_t(int nb);
782
783 // liaison des ddl et des deplacement a t,tdt et 0
784 // nb = le nb du début des deplacement dans les ddl
785 void Liaison_tdt(int nb);
786
787 // liaison des ddl et des deplacement a tdt seulement
788 // nb = le nb du début des deplacement dans les ddl
789 void LiaiSeulettdt(int nb);
790
791 // mise a jour des liaisons ddl/deplacement
792 void MiseAJour();
793
794 // mise a jour de l'adressage via l'identificateur d'énumération
795 // c'est-à-dire la gestion via list_tab_enum et pos_enum
796 // ne doit être appelé que si l'on a introduit un nouveau ddl !!
797 // ou changé l'ordre etc..
798 void MiseAJourEnum();
799
800 // mise a jour du tableau de ddl des variables actives
801 void MiseAJourActif();
802
803 // retourne le numero du ddl recherche identifie par en,
804 // s'il existe sinon 0
805 int Existe(const Enum_ddl en) const { return (*pos_enum)(en);};
806 // --- cas des grandeurs quelconques ----
807 // lecture des grandeurs quelconques sur flot
808 void Lecture_grandeurs_quelconque(istream & ent);
809 // écriture des grandeurs quelconques sur flot
810 void Ecriture_grandeurs_quelconque(ostream & sort) const;
811 // récupe de l'indice d'une grandeur quelconque
812 // si la grandeur n'existe pas -> ramène 0
813 int Indice_grandeur_quelconque(TypeQuelconque_enum_etendu a);
814 // mise a jour de l'adressage via l'identificateur d'énumération
815 // c'est-à-dire la gestion via list_tab_typeQuelconque et pos_Quelconque
816 // ne doit être appelé que si l'on a introduit un nouveau type quelconque !!
817 // ou changé l'ordre etc..
818 void MiseAJourTypeQuelconque();
819 // retourne le numero du type quelconque recherche identifie par en,
820 // s'il existe sinon 0
821 int Existe(TypeQuelconque_enum_etendu en)
822     const { return ( pos_Quelconque == list_tab_typeQuelconque.end()) ? 0 :
(*pos_Quelconque)(en.Position());};
823 // --- cas des Ddl_etendu ----
824 // lecture des Ddl_etendu sur flot
825 void Lecture_Ddl_etendu(istream & ent);
826 // écriture des Ddl_etendu sur flot
827 void Ecriture_Ddl_etendu(ostream & sort) const;
828 // récupe de l'indice d'un Ddl_etendu
829 // si la grandeur n'existe pas -> ramène 0
830 int Indice_Ddl_etendu(const Ddl_enum_etendu& a);
831 // mise a jour de l'adressage via l'identificateur d'énumération
832 // c'est-à-dire la gestion via list_tab_ddletendu et pos_ddletendu
833 // ne doit être appelé que si l'on a introduit un nouveau Ddl_etendu !!
834 // ou changé l'ordre etc..
835 void MiseAJourDdl_etendu();
836 // retourne le numero du Ddl_enum_etendu recherche identifie par en,
837 // s'il existe sinon 0
838 int Existe_ddlEtendu(const Ddl_enum_etendu& en)
839     const { return ( pos_ddletendu == list_tab_ddletendu.end()) ? 0 :
(*pos_ddletendu)(en.Position());};
840
841 // // ordonne et on met a suivre les ddl d'une même famille suivant un ordre croissant
842 // // ceci pour des ddl que l'on vient d'ajouter, QUE L'ON CONSIDÈRE ÊTRE À LA FIN DE TAB_DDL !
843 // // en paramètre le ou les ddl
844 // void OrdonneDdlMemeFamille(const Ddl& a);
845 // void OrdonneDdlMemeFamille(const Tableau<Ddl>& ta);
846 // // permutation de deux ddl (sert pour OrdonneDdlMemeFamille)
847 // void Permutation_2ddl(const int& ipol,cont int& ipo2);
848 // // ordonner un tableau de ddl suivant un ordre croissant et avec des ddl à suivre
849 void OrdonnerTableauDdl(Tableau<Ddl>& ta);
850 // liste des ddl (en famille) à ajouter
851 // recopie des ddl existant déjà
852 // change_donn_var indique si les ddl existant on été modifiée

```



```

853 void ListeDdlAjouter(list<Ddl>& liDdl,Tableau<Ddl>& ta,bool& change_donn_var);
854
855 };
856 /// @} // end of group
857
858
859 #endif
860

```

## 7.354 VariablesExporter.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      06/03/2023
32 *
33 *      AUTEUR:      G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:      Herezh++
36 *
37 *
38 *      BUT:      def d'une classe relative a l'exportation en variables
39 *                globales de grandeurs définies dans les maillages.
40 *                Cela concerne les constantes utilisateur et les variables
41 *                calculées par Herezh.
42 *
43 *      *****
44 *      VERIFICATION:
45 *
46 *      ! date !   auteur !           but
47 *      -----
48 *      !       !           !           !
49 *
50 *      *****
51 *      MODIFICATIONS:
52 *      ! date !   auteur !           but
53 *      -----
54 *
55 *****/
56
57
58 #ifndef VARIABLES_EXPORTER_H
59 #define VARIABLES_EXPORTER_H
60
61
62 #include <iostream>
63 #include <map>
64 #ifndef ENLINUX_STREAM
65 #include <sstream> // pour le flot en memoire centrale
66 #else
67 #include <sstream> // pour le flot en memoire centrale
68 #endif
69 #include "UtilLecture.h"
70 #include "Enum_IO_XML.h"
71
72
73 #include "LesMaillages.h"
74 #include "Basiques.h"

```

```

75
76
77 /** @defgroup Les_classes_exportation_en_variables
78 *
79 *   BUT: def de classes relatives a l'exportation en variables
80 *   globales de grandeurs définies dans les maillages.
81 *   Cela concerne les constantes utilisateur et les variables
82 *   calculées par Herezh.
83 *
84 *   \author   Gérard Rio
85 *   \version  1.0
86 *   \date    23/01/97
87 *   \brief   Définition de classes relatives a l'exportation en variables
88 *
89 */
90
91 /// @addtogroup Les_classes_exportation_en_variables
92 /// @{
93 ///
94
95 class VariablesExporter
96 {
97     public :
98     friend class LesMaillages;
99
100    //=====%%%%%%%% classes de conteneurs %%%%%%%%%=====
101
102    // classe conteneur de base pour noeud, élément etc.
103    class A_un_NE
104    { // surcharge de l'operator de lecture avec le type
105      friend istream & operator » (istream &, A_un_NE &);
106      // surcharge de l'operator d'écriture
107      friend ostream & operator « (ostream &, const A_un_NE &);
108
109      public :
110      A_un_NE(); // constructeur par défaut
111      // constructeur fonction de toutes les grandeurs
112      A_un_NE(string ref
113              ,string nom_mail_,string nom_var_);
114      // constructeur de copie
115      A_un_NE (const A_un_NE& a);
116      // DESTRUCTEUR :
117      virtual ~A_un_NE ();
118
119      // opérateurs
120      A_un_NE& operator= (const A_un_NE& a);
121      bool operator== (const A_un_NE& a) const;
122      bool operator!= (const A_un_NE& a) const
123      {if ( (*this)==a ) return true; else return false;};
124      bool operator < (const A_un_NE& a) const;
125      bool operator <= (const A_un_NE& a) const;
126      bool operator > (const A_un_NE& a) const;
127      bool operator >= (const A_un_NE& a) const;
128      void Affiche();
129
130      // retour des grandeurs
131      const string& Ref_const() const {return ref;}; // la ref
132      const string& Nom_mail_const() const {return nom_mail;}; // le nom du maillage du noeud
133      const string& Nom_var_const() const {return nom_var;}; // nom de la variable associée
134
135      string& Ref_NE() {return ref;}; // la ref
136      string* Pointeur_Ref_NE() {return &ref;}; // la ref
137      string& Nom_mail() {return nom_mail;}; // le nom du maillage du noeud
138      string* Pointeur_Nom_mail() {return &nom_mail;}; // le nom du maillage du noeud
139      string& Nom_var() {return nom_var;}; // nom de la variable associée
140
141      // récupération du pointeur de la classe
142      const A_un_NE* PointeurClass_const() const {return this;};
143      A_un_NE* PointeurClass() {return this;};
144
145
146
147      protected :
148      string ref; // la ref associée
149      string nom_mail; // le nom du maillage du noeud ou élément
150      string nom_var; // nom de la variable associée
151    };
152
153    // classe conteneur pour un ddl à un noeud
154    class Ddl_a_un_noeud
155    { // surcharge de l'operator de lecture avec le type
156      friend istream & operator » (istream &, Ddl_a_un_noeud &);
157      // surcharge de l'operator d'écriture
158      friend ostream & operator « (ostream &, const Ddl_a_un_noeud &);
159
160      public :
161      Ddl_a_un_noeud (); // constructeur par défaut

```

```

162         // constructeur fonction de toutes les grandeurs
163         Ddl_a_un_noeud (Ddl_enum_etendu e, string ref_noeud
164             , string nom_mail_, string nom_var_, Enum_dure tps);
165         // constructeur de copie
166         Ddl_a_un_noeud (const Ddl_a_un_noeud& a);
167         // DESTRUCTEUR :
168         virtual ~Ddl_a_un_noeud ();
169
170         // opérateurs
171         Ddl_a_un_noeud& operator= (const Ddl_a_un_noeud& a);
172         bool operator== (const Ddl_a_un_noeud& a) const;
173         bool operator!= (const Ddl_a_un_noeud& a) const
174             {if ( (*this)==a ) return true; else return false;};
175         bool operator < (const Ddl_a_un_noeud& a) const;
176         bool operator <= (const Ddl_a_un_noeud& a) const;
177         bool operator > (const Ddl_a_un_noeud& a) const;
178         bool operator >= (const Ddl_a_un_noeud& a) const;
179         void Affiche();
180
181         // retour des grandeurs
182         const Ddl_enum_etendu& Enu_const() const {return enu;}; // le ddl
183         const string& Ref_Num_NE_const() const {return ref_num_NE;}; // la ref de num du noeud
184         const string& Nom_mail_const() const {return nom_mail;}; // le nom du maillage du noeud
185         const string& Nom_var_const() const {return nom_var;}; // nom de la variable associée
186         const Enum_dure& Temps_const() const {return temps;}; // temps de validation du ddl
187         string* Pointeur_Nom_mail() {return &nom_mail;}; // le nom du maillage du noeud
188
189         Ddl_enum_etendu& Enu() {return enu;}; // le ddl
190         string& Ref_Num_NE() {return ref_num_NE;}; // la ref de num du noeud
191         string& Nom_mail() {return nom_mail;}; // le nom du maillage du noeud
192         string& Nom_var() {return nom_var;}; // nom de la variable associée
193         Enum_dure& Temps() {return temps;}; // temps de validation du ddl
194
195     protected :
196         Ddl_enum_etendu enu; // le ddl
197         string ref_num_NE; // le num du noeud
198         string nom_mail; // le nom du maillage du noeud
199         string nom_var; // nom de la variable associée
200         Enum_dure temps; // temps de validation du ddl
201     };
202
203     // classe conteneur pour un ddl étendu à un noeud
204     class Ddl_etendu_a_un_noeud : public A_un_NE
205     { // surcharge de l'opérateur de lecture avec le type
206         friend istream & operator » (istream &, Ddl_etendu_a_un_noeud &);
207         // surcharge de l'opérateur d'écriture
208         friend ostream & operator « (ostream &, const Ddl_etendu_a_un_noeud &);
209
210     public :
211         Ddl_etendu_a_un_noeud (); // constructeur par défaut
212         // constructeur fonction de toutes les grandeurs
213         Ddl_etendu_a_un_noeud (Ddl_enum_etendu e, string ref_noeud
214             , string nom_mail_, string nom_var_);
215         // constructeur de copie
216         Ddl_etendu_a_un_noeud (const Ddl_etendu_a_un_noeud& a);
217         // DESTRUCTEUR :
218         virtual ~Ddl_etendu_a_un_noeud ();
219
220         // opérateurs
221         Ddl_etendu_a_un_noeud& operator= (const Ddl_etendu_a_un_noeud& a);
222         bool operator== (const Ddl_etendu_a_un_noeud& a) const;
223         bool operator!= (const Ddl_etendu_a_un_noeud& a) const
224             {if ( (*this)==a ) return true; else return false;};
225         bool operator < (const Ddl_etendu_a_un_noeud& a) const;
226         bool operator <= (const Ddl_etendu_a_un_noeud& a) const;
227         bool operator > (const Ddl_etendu_a_un_noeud& a) const;
228         bool operator >= (const Ddl_etendu_a_un_noeud& a) const;
229         void Affiche();
230
231         // retour des grandeurs
232         Ddl_enum_etendu Enu_const() const {return enu;}; // le ddl
233         Ddl_enum_etendu& Enu() {return enu;}; // le ddl
234
235     protected :
236         Ddl_enum_etendu enu; // le ddl
237     };
238
239     // classe conteneur pour une grandeur quelconque à un noeud
240     class Quelconque_a_un_noeud : public A_un_NE
241     { // surcharge de l'opérateur de lecture avec le type
242         friend istream & operator » (istream &, Quelconque_a_un_noeud &);
243         // surcharge de l'opérateur d'écriture
244         friend ostream & operator « (ostream &, const Quelconque_a_un_noeud &);
245
246     public :
247         Quelconque_a_un_noeud (); // constructeur par défaut
248         // constructeur fonction de toutes les grandeurs

```

```

249     Quelconque_a_un_noeud (TypeQuelconque_enum_etendu e, string ref_noeud
250         , string nom_mail_, string nom_var_, int num_ord);
251     // constructeur de copie
252     Quelconque_a_un_noeud (const Quelconque_a_un_noeud& a);
253     // DESTRUCTEUR :
254     virtual ~Quelconque_a_un_noeud ();
255
256     // opérateurs
257     Quelconque_a_un_noeud& operator= (const Quelconque_a_un_noeud& a);
258     bool operator== (const Quelconque_a_un_noeud& a) const;
259     bool operator!= (const Quelconque_a_un_noeud& a) const
260         {if ( (*this)==a ) return true; else return false;};
261     bool operator < (const Quelconque_a_un_noeud& a) const;
262     bool operator <= (const Quelconque_a_un_noeud& a) const;
263     bool operator > (const Quelconque_a_un_noeud& a) const;
264     bool operator >= (const Quelconque_a_un_noeud& a) const;
265     void Affiche();
266
267     // retour des grandeurs
268     const TypeQuelconque_enum_etendu& Quelc_const() const {return quelc;} ; // la grandeur constante
269     TypeQuelconque_enum_etendu& Quelc() {return quelc;} ; // la grandeur en i/o
270
271     const int& Num_ordre_const() const {return num_ordre;};
272     int& Num_ordre() {return num_ordre;};
273
274 protected :
275     TypeQuelconque_enum_etendu quelc; // la grandeur
276     int num_ordre; // le numéro de la composante
277 };
278
279 // classe conteneur pour un pti d'élément
280 class A_un_E : public A_un_NE
281 { // surcharge de l'operator de lecture avec le type
282     friend istream & operator » (istream &, A_un_E &);
283     // surcharge de l'operator d'écriture
284     friend ostream & operator « (ostream &, const A_un_E &);
285
286 public :
287     A_un_E (); // constructeur par défaut
288     // constructeur fonction de toutes les grandeurs
289     A_un_E (int absolu_, string ref_
290         , string nom_mail_, string nom_var_, int nbpti);
291     // constructeur de copie
292     A_un_E (const A_un_E& a);
293     // DESTRUCTEUR :
294     virtual ~A_un_E ();
295
296     // opérateurs
297     A_un_E& operator= (const A_un_E& a);
298     bool operator== (const A_un_E& a) const;
299     bool operator!= (const A_un_E& a) const
300         {if ( (*this)==a ) return true; else return false;};
301     bool operator < (const A_un_E& a) const;
302     bool operator <= (const A_un_E& a) const;
303     bool operator > (const A_un_E& a) const;
304     bool operator >= (const A_un_E& a) const;
305     void Affiche();
306
307     // récupération du pointeur de la classe
308     const A_un_E* PointeurClass_E_const() const {return this;};
309     A_un_E* PointeurClass_E() {return this;};
310
311     // retour des grandeurs
312
313     const int& NBpti_const() const {return num_pti;}; // num du pti en const
314     int& NBpti() {return num_pti;}; // idem direct
315     const int& Absolue_const() const {return absolu;}; // absolu ou pas
316     int& Absolue() {return absolu;};
317
318 protected :
319     int num_pti; // numéro du point d'intégration
320     int absolu; // indique si la grandeur est dans un repère absolu ou local
321 };
322
323 // classe conteneur pour un ddl à un élément
324 class Ddl_a_un_element : public A_un_E
325 { // surcharge de l'operator de lecture avec le type
326     friend istream & operator » (istream &, Ddl_a_un_element &);
327     // surcharge de l'operator d'écriture
328     friend ostream & operator « (ostream &, const Ddl_a_un_element &);
329
330 public :
331     Ddl_a_un_element (); // constructeur par défaut
332     // constructeur fonction de toutes les grandeurs
333     Ddl_a_un_element (int absolu_, Ddl_enum_etendu e, string ref_noeud
334         , string nom_mail_, Enum_dure tps, string nom_var_, int nbpti);
335     // constructeur de copie

```

```

336     Ddl_a_un_element (const Ddl_a_un_element& a);
337     // DESTRUCTEUR :
338     virtual ~Ddl_a_un_element ();
339
340     // opérateurs
341     Ddl_a_un_element& operator= (const Ddl_a_un_element& a);
342     bool operator== (const Ddl_a_un_element& a) const;
343     bool operator!= (const Ddl_a_un_element& a) const
344         {if ( (*this)==a ) return true; else return false;};
345     bool operator < (const Ddl_a_un_element& a) const;
346     bool operator <= (const Ddl_a_un_element& a) const;
347     bool operator > (const Ddl_a_un_element& a) const;
348     bool operator >= (const Ddl_a_un_element& a) const;
349     void Affiche();
350
351     // retour des grandeurs
352     const Ddl_enum_etendu& Enu_const() const {return enu;}; // le ddl en const
353     Ddl_enum_etendu& Enu() {return enu;}; // le ddl direct
354     const Enum_dure& Temps_const() const {return temps;}; // temps de validation du ddl
355     Enum_dure& Temps() {return temps;}; // temps de validation du ddl
356
357     protected :
358         Ddl_enum_etendu enu; // le ddl
359         Enum_dure temps; // temps de validation du ddl
360 };
361
362 // classe conteneur pour un TypeQuelconque à un élément
363 class TypeQuelconque_a_un_element : public A_un_E
364 { // surcharge de l'operator de lecture avec le type
365     friend istream & operator » (istream &, TypeQuelconque_a_un_element &);
366     // surcharge de l'operator d'écriture
367     friend ostream & operator « (ostream &, const TypeQuelconque_a_un_element &);
368
369     public :
370         TypeQuelconque_a_un_element (); // constructeur par défaut
371         // constructeur fonction de toutes les grandeurs
372         TypeQuelconque_a_un_element (int absolu_, TypeQuelconque_enum_etendu e, string ref_NE
373             , string nom_mail_, Enum_dure tps, string nom_var_, int nbpti, int num_ord);
374         // constructeur de copie
375         TypeQuelconque_a_un_element (const TypeQuelconque_a_un_element& a);
376         // DESTRUCTEUR :
377         virtual ~TypeQuelconque_a_un_element ();
378
379         // opérateurs
380         TypeQuelconque_a_un_element& operator= (const TypeQuelconque_a_un_element& a);
381         bool operator== (const TypeQuelconque_a_un_element& a) const;
382         bool operator!= (const TypeQuelconque_a_un_element& a) const
383             {if ( (*this)==a ) return true; else return false;};
384         bool operator < (const TypeQuelconque_a_un_element& a) const;
385         bool operator <= (const TypeQuelconque_a_un_element& a) const;
386         bool operator > (const TypeQuelconque_a_un_element& a) const;
387         bool operator >= (const TypeQuelconque_a_un_element& a) const;
388         void Affiche();
389
390         // récupération du pointeur de la classe
391         const TypeQuelconque_a_un_element* PointeurClass_Quelc_const() const {return this;};
392         TypeQuelconque_a_un_element* PointeurClass_Quelc() {return this;};
393
394         // retour des grandeurs
395         const TypeQuelconque_enum_etendu& Quelc_const() const {return quelc;}; // la grandeur constante
396         TypeQuelconque_enum_etendu& Quelc() {return quelc;}; // la grandeur en i/o
397         const int& Num_ordre_const() const {return num_ordre;};
398         int& Num_ordre() {return num_ordre;};
399
400     protected :
401         TypeQuelconque_enum_etendu quelc; // la grandeur
402         int num_ordre; // le numéro de la composante
403 };
404
405 // classe conteneur pour les grandeurs particulières à un élément
406 class TypeParticulier_a_un_element : public TypeQuelconque_a_un_element
407 { // surcharge de l'operator de lecture avec le type
408     friend istream & operator » (istream &, TypeParticulier_a_un_element &);
409     // surcharge de l'operator d'écriture
410     friend ostream & operator « (ostream &, const TypeParticulier_a_un_element &);
411
412     public :
413         TypeParticulier_a_un_element(): TypeQuelconque_a_un_element() {}; // constructeur par défaut
414         // constructeur fonction de toutes les grandeurs
415         TypeParticulier_a_un_element (int absolu_, TypeQuelconque_enum_etendu e, string ref_noeud
416             , string nom_mail_, Enum_dure tps, string nom_var_, int nbpti, int num_ord):
417             TypeQuelconque_a_un_element (absolu_, e, ref_noeud, nom_mail_, tps, nom_var_, nbpti, num_ord) {};
418         // constructeur de copie
419         TypeParticulier_a_un_element (const TypeParticulier_a_un_element& a):
420             TypeQuelconque_a_un_element (a) {};
421         // DESTRUCTEUR :
422         virtual ~TypeParticulier_a_un_element () {};

```

```

423
424 // opérateurs
425 TypeParticulier_a_un_element& operator= (const TypeParticulier_a_un_element& a)
426 {(*PointeurClass_Quelc())= (*a.PointeurClass_Quelc_const());};
427 bool operator== (const TypeParticulier_a_un_element& a) const
428 {return ((*PointeurClass_Quelc_const()) == (*a.PointeurClass_Quelc_const()));};
429 bool operator!= (const TypeParticulier_a_un_element& a) const
430 {if ( (*this)==a ) return true; else return false;};
431 bool operator < (const TypeParticulier_a_un_element& a) const
432 {return ((*PointeurClass_Quelc_const()) < (*a.PointeurClass_Quelc_const()));};
433 bool operator <= (const TypeParticulier_a_un_element& a) const
434 {return ((*PointeurClass_Quelc_const()) <= (*a.PointeurClass_Quelc_const()));};
435 bool operator > (const TypeParticulier_a_un_element& a) const
436 {return ((*PointeurClass_Quelc_const()) > (*a.PointeurClass_Quelc_const()));};
437 bool operator >= (const TypeParticulier_a_un_element& a) const
438 {return ((*PointeurClass_Quelc_const()) == (*a.PointeurClass_Quelc_const()));};
439 void Affiche();
440 };
441
442 // classe conteneur pour les grandeurs évoluées à un élément
443 class TypeEvoluee_a_un_element : public TypeQuelconque_a_un_element
444 { // surcharge de l'opérateur de lecture avec le type
445 friend istream & operator » (istream &, TypeEvoluee_a_un_element &);
446 // surcharge de l'opérateur d'écriture
447 friend ostream & operator « (ostream &, const TypeEvoluee_a_un_element &);
448
449 public :
450 TypeEvoluee_a_un_element(): TypeQuelconque_a_un_element() {}; // constructeur par défaut
451 // constructeur fonction de toutes les grandeurs
452 TypeEvoluee_a_un_element (int absolu_,TypeQuelconque_enum_etendu e,string ref_noeud
453 ,string nom_mail_,Enum_dure tps,string nom_var_,int nbpti,int num_ord):
454 TypeQuelconque_a_un_element (absolu_,e,ref_noeud,nom_mail_,tps,nom_var_,nbpti,num_ord) {} ;
455 // constructeur de copie
456 TypeEvoluee_a_un_element(const TypeEvoluee_a_un_element& a):
457 TypeQuelconque_a_un_element(a) {};
458 // DESTRUCTEUR :
459 virtual ~TypeEvoluee_a_un_element() {};
460
461 // opérateurs
462 TypeEvoluee_a_un_element& operator= (const TypeEvoluee_a_un_element& a)
463 {(*PointeurClass_Quelc())= (*a.PointeurClass_Quelc_const());};
464 bool operator== (const TypeEvoluee_a_un_element& a) const
465 {return ((*PointeurClass_Quelc_const()) == (*a.PointeurClass_Quelc_const()));};
466 bool operator!= (const TypeEvoluee_a_un_element& a) const
467 {if ( (*this)==a ) return true; else return false;};
468 bool operator < (const TypeEvoluee_a_un_element& a) const
469 {return ((*PointeurClass_Quelc_const()) < (*a.PointeurClass_Quelc_const()));};
470 bool operator <= (const TypeEvoluee_a_un_element& a) const
471 {return ((*PointeurClass_Quelc_const()) <= (*a.PointeurClass_Quelc_const()));};
472 bool operator > (const TypeEvoluee_a_un_element& a) const
473 {return ((*PointeurClass_Quelc_const()) > (*a.PointeurClass_Quelc_const()));};
474 bool operator >= (const TypeEvoluee_a_un_element& a) const
475 {return ((*PointeurClass_Quelc_const()) == (*a.PointeurClass_Quelc_const()));};
476 void Affiche();
477 };
478
479 // classe conteneur pour un TypeQuelconque à une face ou arête
480 class TypeQuelconque_a_Face_arete : public TypeQuelconque_a_un_element
481 { public :
482 TypeQuelconque_a_Face_arete (); // constructeur par défaut
483 // constructeur fonction de toutes les grandeurs
484 TypeQuelconque_a_Face_arete (int absolu_,TypeQuelconque_enum_etendu e,string ref_element
485 ,string nom_mail_,int nbFA, Enum_dure tps,string nom_var_,int nbpti,int
486 num_ord);
487 // constructeur de copie
488 TypeQuelconque_a_Face_arete (const TypeQuelconque_a_Face_arete& a);
489 // DESTRUCTEUR :
490 virtual ~TypeQuelconque_a_Face_arete ();
491
492 // opérateurs
493 TypeQuelconque_a_Face_arete& operator= (const TypeQuelconque_a_Face_arete& a);
494 bool operator== (const TypeQuelconque_a_Face_arete& a) const;
495 bool operator!= (const TypeQuelconque_a_Face_arete& a) const
496 {if ( (*this)==a ) return true; else return false;};
497 bool operator < (const TypeQuelconque_a_Face_arete& a) const;
498 bool operator <= (const TypeQuelconque_a_Face_arete& a) const;
499 bool operator > (const TypeQuelconque_a_Face_arete& a) const;
500 bool operator >= (const TypeQuelconque_a_Face_arete& a) const;
501
502 // récupération du pointeur de la classe
503 const TypeQuelconque_a_Face_arete* PointClass_QuelcFA_const() const {return this;};
504 TypeQuelconque_a_Face_arete* PointClass_QuelcFA() {return this;};
505
506 // retour des grandeurs
507 const int& Num_FA_const() const {return num_FA;}; // la grandeur constante
508 int& Num_FA() {return num_FA;}; // la grandeur en i/O

```

```

509     protected :
510         int num_FA; // ref de numéro de la face ou de l'arête
511     };
512
513     // classe conteneur pour les grandeurs quelconque à une face d'élément
514     class TypeQuelc_face_a_un_element : public TypeQuelconque_a_Face_arete
515     { // surcharge de l'operator de lecture avec le type
516         friend istream & operator » (istream &, TypeQuelc_face_a_un_element &);
517         // surcharge de l'operator d'écriture
518         friend ostream & operator « (ostream &, const TypeQuelc_face_a_un_element &);
519
520     public :
521         TypeQuelc_face_a_un_element(): TypeQuelconque_a_Face_arete() {}; // constructeur par défaut
522         // constructeur fonction de toutes les grandeurs
523         TypeQuelc_face_a_un_element (int absolu_, TypeQuelconque_enum_etendu e, string ref_element
524             , string nom_mail_, int nbFA, Enum_dure tps, string nom_var_, int nbpti, int num_ord):
525             TypeQuelconque_a_Face_arete(absolu_, e, ref_element, nom_mail, nbFA, tps, nom_var_, nbpti, num_ord)
526         { };
527         // constructeur de copie
528         TypeQuelc_face_a_un_element(const TypeQuelc_face_a_un_element& a):
529             TypeQuelconque_a_Face_arete(a) {};
530         // DESTRUCTEUR :
531         virtual ~TypeQuelc_face_a_un_element() {};
532
533         // opérateurs
534         TypeQuelc_face_a_un_element& operator= (const TypeQuelc_face_a_un_element& a)
535         { (*PointClass_QuelcFA())= (*a.PointClass_QuelcFA_const()); };
536         bool operator== (const TypeQuelc_face_a_un_element& a) const
537         { return ((*PointClass_QuelcFA_const()) == (*a.PointClass_QuelcFA_const())); };
538         bool operator!= (const TypeQuelc_face_a_un_element& a) const
539         { if ( (*this)==a ) return true; else return false; };
540         bool operator < (const TypeQuelc_face_a_un_element& a) const
541         { return ((*PointClass_QuelcFA_const()) < (*a.PointClass_QuelcFA_const())); };
542         bool operator <= (const TypeQuelc_face_a_un_element& a) const
543         { return ((*PointClass_QuelcFA_const()) <= (*a.PointClass_QuelcFA_const())); };
544         bool operator > (const TypeQuelc_face_a_un_element& a) const
545         { return ((*PointClass_QuelcFA_const()) > (*a.PointClass_QuelcFA_const())); };
546         bool operator >= (const TypeQuelc_face_a_un_element& a) const
547         { return ((*PointClass_QuelcFA_const()) == (*a.PointClass_QuelcFA_const())); };
548         void Affiche();
549     };
550
551     // classe conteneur pour les grandeurs quelconque à une arête d'élément
552     class TypeQuelc_arete_a_un_element : public TypeQuelconque_a_Face_arete
553     { // surcharge de l'operator de lecture avec le type
554         friend istream & operator » (istream &, TypeQuelc_arete_a_un_element &);
555         // surcharge de l'operator d'écriture
556         friend ostream & operator « (ostream &, const TypeQuelc_arete_a_un_element &);
557
558     public :
559         TypeQuelc_arete_a_un_element(): TypeQuelconque_a_Face_arete() {}; // constructeur par défaut
560         // constructeur fonction de toutes les grandeurs
561         TypeQuelc_arete_a_un_element (int absolu_, TypeQuelconque_enum_etendu e, string ref_element
562             , string nom_mail_, int nbFA, Enum_dure tps, string nom_var_, int nbpti, int num_ord):
563             TypeQuelconque_a_Face_arete(absolu_, e, ref_element, nom_mail, nbFA, tps, nom_var_, nbpti, num_ord)
564         { };
565         // constructeur de copie
566         TypeQuelc_arete_a_un_element(const TypeQuelc_arete_a_un_element& a):
567             TypeQuelconque_a_Face_arete(a) {};
568         // DESTRUCTEUR :
569         virtual ~TypeQuelc_arete_a_un_element() {};
570
571         // opérateurs
572         TypeQuelc_arete_a_un_element& operator= (const TypeQuelc_arete_a_un_element& a)
573         { (*PointClass_QuelcFA())= (*a.PointClass_QuelcFA_const()); };
574         bool operator== (const TypeQuelc_arete_a_un_element& a) const
575         { return ((*PointClass_QuelcFA_const()) == (*a.PointClass_QuelcFA_const())); };
576         bool operator!= (const TypeQuelc_arete_a_un_element& a) const
577         { if ( (*this)==a ) return true; else return false; };
578         bool operator < (const TypeQuelc_arete_a_un_element& a) const
579         { return ((*PointClass_QuelcFA_const()) < (*a.PointClass_QuelcFA_const())); };
580         bool operator <= (const TypeQuelc_arete_a_un_element& a) const
581         { return ((*PointClass_QuelcFA_const()) <= (*a.PointClass_QuelcFA_const())); };
582         bool operator > (const TypeQuelc_arete_a_un_element& a) const
583         { return ((*PointClass_QuelcFA_const()) > (*a.PointClass_QuelcFA_const())); };
584         bool operator >= (const TypeQuelc_arete_a_un_element& a) const
585         { return ((*PointClass_QuelcFA_const()) == (*a.PointClass_QuelcFA_const())); };
586         void Affiche();
587     };
588
589
590     // classe conteneur pour une composante d'une grandeur globale
591     class TypeQuelc_Une_composante_Grandeur_globale
592     { // surcharge de l'operator de lecture avec le type
593         friend istream & operator » (istream &, TypeQuelc_Une_composante_Grandeur_globale &);
594         // surcharge de l'operator d'écriture
595         friend ostream & operator « (ostream &, const TypeQuelc_Une_composante_Grandeur_globale &);

```

```

596
597 public :
598     TypeQuelc_Une_composante_Grandeur_globale(): var() {}; // constructeur par défaut
599     // constructeur fonction de toutes les grandeurs
600     TypeQuelc_Une_composante_Grandeur_globale (string nom_var, string nom_glob, int num_ord):
601         var(nom_var,nom_glob,num_ord)
602     {
603         // constructeur de copie
604         TypeQuelc_Une_composante_Grandeur_globale(const TypeQuelc_Une_composante_Grandeur_globale& a) :
605             var(a.var) {};
606         // DESTRUCTEUR :
607         virtual ~TypeQuelc_Une_composante_Grandeur_globale() {};
608
609         // opérateurs
610         TypeQuelc_Une_composante_Grandeur_globale& operator= (const
TypeQuelc_Une_composante_Grandeur_globale& a)
611         {var = a.var;};
612         bool operator== (const TypeQuelc_Une_composante_Grandeur_globale& a) const
613         {return (var == a.var);};
614         bool operator!= (const TypeQuelc_Une_composante_Grandeur_globale& a) const
615         {if ( (*this)==a ) return true; else return false;};
616         bool operator < (const TypeQuelc_Une_composante_Grandeur_globale& a) const
617         {return (var < a.var);};
618         bool operator <= (const TypeQuelc_Une_composante_Grandeur_globale& a) const
619         {return (var <= a.var);};
620         bool operator > (const TypeQuelc_Une_composante_Grandeur_globale& a) const
621         {return (var > a.var);};
622         bool operator >= (const TypeQuelc_Une_composante_Grandeur_globale& a) const
623         {return (var >= a.var);};
624         void Affiche();
625
626         // retour des grandeurs
627         const string& Nom_var_const() const {return var.nom1;}; // nom de la variable associée
628         string& Nom_var() {return var.nom1;}; // nom de la variable associée
629         const string& Nom_grandeur_globale_const() const {return var.nom2;}; // nom de la grandeur
globale associé
630         string& Nom_grandeur_globale() {return var.nom2;}; // nom de la grandeur globale associé
631         const int& Indice_const() const {return var.n;};
632         int& Indice() {return var.n;};
633
634         Deux_String_un_entier Deux_String_un_entier_const() const {return var;};
635
636         Deux_String_un_entier& Deux_String_un_entier_() {return var;};
637
638     protected :
639         Deux_String_un_entier var ;
640         // var.nom1 -> nom de la variable
641         // var.nom2 -> nom de la grandeur globale
642         // var.n -> le numéro de la composante
643     };
644
645
646
647 // METHODES :
648
649 //=====la classe principale =====
650
651 // CONSTRUCTEURS :
652
653 // Constructeur par défaut
654 VariablesExporter ();
655
656 // Constructeur de copie, cependant ici il n'y a pas de création de noeud ni d'élément
657 // c'est seulement une création de nouveaux conteneurs de pointeurs
658 // cependant le numéro de maillage et le nom de maillage n'est pas valide, il faut
659 // ensuite les définir
660 VariablesExporter (const VariablesExporter& var);
661
662 // DESTRUCTEUR :
663 ~VariablesExporter ();
664     // lecture
665     void LectureVariablesExporter(UtilLecture * entreePrinc);
666
667 // insertion des constantes et variables utilisateurs vers globales
668 void InsertConstVarUtilisateur_dans_globale();
669
670 // initialisation des conteneurs quelconques
671 // création des conteneurs aux noeuds s'il s'agit d'une variable
672 // associée à un vecteur global (au sens des algorithmes globaux)
673 void InitialisationConteneursQuelconques(LesMaillages& lesMail,const List_io < TypeQuelconque >&
listeVecGlob,const LesReferences& lesRef);
674
675 // ramène les liste des infos qui seront consultés pour les noeuds
676 const List_io < Ddl_a_un_noeud >& List_noeud_type_ddl() const {return list_noeud_type_ddl;}; // ddl
pur aux noeuds
677 const List_io < Ddl_etendu_a_un_noeud >& List_noeud_type_ddlEtendu() const {return
list_noeud_type_ddlEtendu;}; // idem pour les ddl étendu

```



```

678  const List_io <Quelconque_a_un_noeud>& List_noeud_type_quelconque() const {return
        list_noeud_type_quelconque;}; // idem en quelconque
679
680      // affichage et definition interactive des commandes
681      void Info_commande_VariablesExporters(UtilLecture * entreePrinc);
682
683      // Affiche
684      void Affiche () const ;
685
686      // Surcharge de l'operateur = : realise l'egalite de deux instances
687      VariablesExporter& operator= (VariablesExporter& var);
688
689      // test si toutes les informations sont completes
690      // = true -> complet
691      // = false -> incomplet
692      bool Complet_VariablesExporter(const LesReferences& lesRef);
693
694      // mises à jour de constantes globales définies par l'utilisateur
695      // permet de changer les valeurs, lors d'une suite .info par exemple
696      void MiseAJourConstantesUtilisateur(UtilLecture& lec);
697
698      // renseigne les variables définies par l'utilisateur
699      // via les valeurs calculées par Herezh
700      void RenseigneVarUtilisateur(LesMaillages& lesMail,const LesReferences& lesRef);
701
702      //----- lecture écriture dans base info -----
703      // cas donne le niveau de la récupération
704      // = 1 : on récupère tout
705      // = 2 : on récupère uniquement les données variables (supposées comme telles)
706      void Lecture_base_info(ifstream& ent,const int cas);
707      // cas donne le niveau de sauvegarde
708      // = 1 : on sauvegarde tout
709      // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
710      void Ecriture_base_info(ofstream& sort,const int cas);
711      // sortie du schemaXML: en fonction de enu
712      static void SchemaXML_VariablesExporters(UtilLecture * entreePrinc,const Enum_IO_XML enu) ;
713
714      protected :
715
716      // données:
717      // ---- les constantes utilisateur
718      list<TypeQuelconque> li_Q; // les constantes
719      list<string> li_nom; // le nom des constantes
720
721      // ----- les variables
722
723      // -> noeuds
724      List_io < Ddl_a_un_noeud > list_noeud_type_ddl; // ddl pur aux noeuds
725      List_io < Ddl_etendu_a_un_noeud > list_noeud_type_ddlEtendu; // idem pour les ddl étendu
726      List_io <Quelconque_a_un_noeud> list_noeud_type_quelconque; // idem en quelconque
727
728      // -> éléments
729      List_io < Ddl_a_un_element > list_element_type_ddl; // ddl pur aux pti d'élément
730      // idem pour les grandeurs particulières et évoluées
731      List_io < TypeParticulier_a_un_element > list_element_type_particulier;
732      List_io < TypeQuelconque > list_quelc_element_type_particulier; // les conteneurs associés
733
734      List_io < TypeEvoluee_a_un_element > list_element_type_evoluee;
735      List_io < TypeQuelconque > list_quelc_element_type_evoluee; // les conteneurs associés
736
737      // -> face d'éléments
738      List_io <TypeQuelc_face_a_un_element > list_face_element_type_quelc;
739      List_io < TypeQuelconque > list_quelc_face_element_type_quelc; // les conteneurs associés
740
741      // -> arête d'éléments
742      List_io <TypeQuelc_arete_a_un_element > list_arete_element_type_quelc;
743      List_io < TypeQuelconque > list_quelc_arete_element_type_quelc; // les conteneurs associés
744
745      bool initiaConteneurQuelconque; // pour gérer une seule initialisation
746
747      // -> une composante d'une variable globale de type multidimensionnel
748      List_io <TypeQuelc_Une_composante_Grandeur_globale > list_var_glob_sur_grandeur_globale;
749
750
751      // méthodes
752
753      //---- constantes utilisateurs
754      // lecture éventuelle de constantes globales définies par l'utilisateur
755      void LecConstantesUtilisateur(UtilLecture& lec);
756      // insertion des constantes utilisateur dans les variables globales
757      void InsertConstUtilisateur_dans_globale();
758
759      // def de constantes utilisateur pour une sortie sur un fichier de commande
760      void Info_commande_ConstantesUtilisateur(UtilLecture * lec);
761
762      // ---- variables utilisateur
763      // lecture éventuelle des variables globales définies par l'utilisateur

```

```

764 void LecVariablesUtilisateur(UtilLecture& lec);
765
766 // def de variables utilisateur pour une sortie sur un fichier de commande
767 void Info_commande_VariablesUtilisateur(UtilLecture * lec);
768
769 // insertion et stockage des variables au niveau de ParaGlob
770 void Insert_VarUtilisateur_dans_globale();
771
772 // vérification que 2 nom de variables ne soient pas identique
773 // si avec_sortie = true -> arrêt
774 // si avec_sortie = false -> retour du test: true si c'est ok
775 //                               false si pb
776 bool VerifNomVariable(const string& nom,bool avec_sortie) const;
777
778 // lecture interactive du nom de la variable: hors variable relais
779 string Lect_interactive_nom_var();
780
781 // lecture interactive du nom d'une variable relais
782 string Lect_interactive_nom_var_relais();
783
784 // insertion d'une composante d'une grandeur quelconque en globale
785 // sous forme d'un double
786 void InsertCompGrandeurQuelc(const string& non_var,const TypeQuelconque_enum_etendu& enu);
787
788 };
789 /// @} // end of group
790
791
792
793 #endif

```

## 7.355 Banniere.h

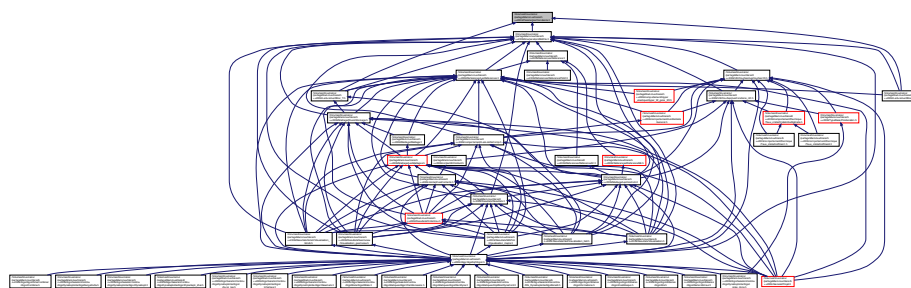
```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      07/01/2003
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:  Bannière du programme
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date !  auteur !      but
44 *      -----
45 *      !           !           !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date !  auteur !      but
50 *      -----
51 *

```



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Variables

- `Tableau`< char \* > `tab_Zero_chaine`
- `Tableau`< string > `tab_Zero_string`

### 7.356.1 Description détaillée

def de tableaux d'initialisation.

def de grandeurs statiques globales.

## 7.357 Constante.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file Constante.h
2     \brief def de tableaux d'initialisation.
3  */
4
5  // This file is part of the Herezh++ application.
6  //
7  // The finite element software Herezh++ is dedicated to the field
8  // of mechanics for large transformations of solid structures.
9  // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 // constantes de tout poil utilisee dans le programme
34
35
36 #ifndef CONSTANCE_H
37 #define CONSTANCE_H
38
39 #include "Tableau_T.h"
40 #include "string"
41
42
43 extern Tableau<char*> tab_Zero_chaine; // tableau de 0 chaine
44 extern Tableau<string> tab_Zero_string; // tableau de 0 string
45 // string rien_string; // string vide
46

```

```
47 #endif
```

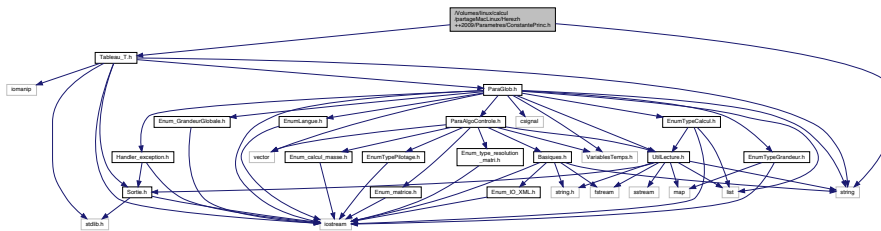
## 7.358 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Parametres/ConstantePrinc.h

def de tableaux d'initialisation.

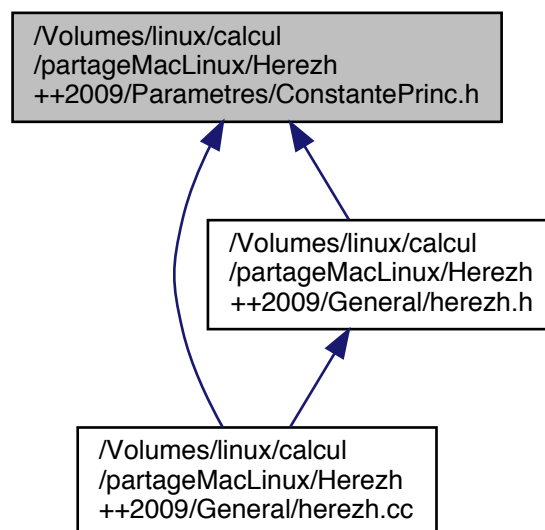
```
#include "Tableau_T.h"
```

```
#include "string"
```

Graphe des dépendances par inclusion de ConstantePrinc.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Variables

- `Tableau`< char \* > `tab_Zero_chaine`
- `Tableau`< string > `tab_Zero_string`

### 7.358.1 Description détaillée

def de tableaux d'initialisation.

## 7.359 ConstantePrinc.h

Aller à la documentation de ce fichier.

```

1  /*! \file ConstantePrinc.h
2     \brief def de tableaux d'initialisation.
3  */
4
5  // This file is part of the Herezh++ application.
6  //
7  // The finite element software Herezh++ is dedicated to the field
8  // of mechanics for large transformations of solid structures.
9  // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 // constantes de tout poil utilisee dans le programme
34
35
36 #ifndef CONSTANCE_H
37 #define CONSTANCE_H
38
39 #include "Tableau_T.h"
40 #include "string"
41
42
43 Tableau<char*> tab_Zero_chaine; // tableau de 0 chaine
44 Tableau<string> tab_Zero_string; // tableau de 0 string
45 // string rien_string; // string vide
46
47 #endif

```

## 7.360 ConstMath.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      30/09/2002      *

```

```

32 *                                     $ *
33 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)           *
34 *                                     $ *
35 *   PROJET:     Herezh++                                       *
36 *                                     $ *
37 * *****
38 *   BUT:        Constantes mathématiques utilisées dans le programme *
39 *                                     $ *
40 *   ////////////////////////////////////////////////// *
41 *   *
42 *   VERIFICATION:                                             *
43 *   ! date !   auteur !           but                       ! *
44 *   -----
45 *   !           !           !                               ! *
46 *   ////////////////////////////////////////////////// $ *
47 *   *
48 *   MODIFICATIONS:                                           *
49 *   ! date !   auteur !           but                       ! *
50 *   -----
51 *   ////////////////////////////////////////////////// $ *
52 * *****/
53
54 #ifndef CONSTMATHPRINC_H
55 #define CONSTMATHPRINC_H
56
57 /// @addtogroup Les_parametres_generaux
58 /// @{
59 ///
60
61
62 class ConstMath
63 { public :
64     static double trespetit;
65     static double pasmalpetit;
66     static double petit;
67     static double unpeupetit;
68     static double unpeugrand;
69     static double grand;
70     static double pasmalgrand;
71     static double tresgrand;
72     static double Pi;
73     static double eps_machine;
74 private:
75     // méthodes
76     // calcul du nombre le plus petit détectable par la machine
77     static double d_epsilon ();
78
79 };
80 /// @} // end of group
81
82 #endif

```

## 7.361 ConstPhysico.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****

```

```

31 *      DATE:          28/01/2008
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *****
38 *      BUT:           Constantes physico chimiques utilisees dans le programme
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date !   auteur !           but
44 *      -----
45 *      !           !           !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date !   auteur !           but
50 *      -----
51 *
52 *****/
53
54 #ifndef CONSTPHYSICO_H
55 #define CONSTPHYSICO_H
56
57 /// @addtogroup Les_parametres_generaux
58 /// @{
59 ///
60
61
62 class ConstPhysico
63 { public :
64     static double R ; // constante des gazs parfaits en J Š K-1 Š mol-1
65     private:
66         // méthodes
67
68 };
69 /// @} // end of group
70
71 #endif

```

## 7.362 EnteteParaGlob.h

```

1 /*! \file Constante.h
2   \brief def de grandeurs statiques globales.
3 */
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 #include "ParaGlob.h"
34
35 //=====
36 // entete pour les parametres globaux : declaration des membres statics
37 //=====
38
39 ParaGlob * ParaGlob::param = NULL; // pointeur sur le seul membre ouvert
40 int ParaGlob::dimensionPb = 1; // dimension du probleme = 1 unidimensionnel par default

```



```

41     EnumLangue ParaGlob::langueHZ = FRANCAIS; // langue utilisée pour les entrées sorties
42     int ParaGlob::nbComposantesTenseur = 1; // nombre de composantes par défaut a 1
43     int ParaGlob::nivImpression = 2; // niveau d'impression
44     string ParaGlob::nbVersion = "6.991" ; // numéro de version du logiciel
45     string ParaGlob::NbVersionsurfichier = ""; // numéro de version lue en entrée fichier
46     int ParaGlob::nb_diggit_double_calcul= 17; // nombre de chiffre significatifs utilisé pour
47         // l'affichage des double précision pour l'archivage
48     int ParaGlob::nb_diggit_double_graphique= 12; // nombre de chiffre significatifs utilisé pour
49         // l'affichage des double précision pour l'affichage du graphique
50     int ParaGlob::nb_diggit_double_ecran= 8; // nombre de chiffre significatifs utilisé pour
51         // l'affichage des double précision sur l'écran
52     VariablesTemps* ParaGlob::tempo=NULL; // pour les grandeurs liés au temps
53     bool ParaGlob::geometrie_axisymetrique = false; // par défaut espace non axisymétrique
54     // un tableau de grandeurs globales quelconques pour être accessibles de partout
55     std::vector <double> ParaGlob::grandeurs_globales(taille_Enum_GrandeurGlobale);

```

## 7.363 ParaAlgoControle.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          23/01/97
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *      *****
38 *      BUT:  Stockage des parametres de controle.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date !   auteur !           but
44 *      -----
45 *      !           !           !
46 *      *****
47 *
48 *      MODIFICATIONS:
49 *      ! date !   auteur !           but
50 *      -----
51 *
52 *      *****/
53 #ifndef PARA_ALGOCONTROLE_H
54 #define PARA_ALGOCONTROLE_H
55
56 #include "UtilLecture.h"
57 #include "Enum_matrice.h"
58 #include "Enum_type_resolution_matri.h"
59 #include "Enum_calcul_masse.h"
60 #include "VariablesTemps.h"
61 #include "Basiques.h"
62 #include "EnumTypePilotage.h"
63 #include <vector>
64
65 /// @addtogroup Les_parametres_generaux
66 /// @{

```

```

67 ///
68
69
70 class ParaAlgoControle
71 { public :
72     // CONSTRUCTEUR
73     // par défaut
74     ParaAlgoControle();
75     // de copie
76     ParaAlgoControle(const ParaAlgoControle& p);
77     //destructeur
78     ~ ParaAlgoControle() {};
79     // Méthodes publiques
80
81     // Surcharge de l'affectation
82     ParaAlgoControle& operator= (const ParaAlgoControle& p);
83     // lecture sur le flot d'entrée
84     void Lecture_paraAlgoControle(UtilLecture & entreePrinc);
85     // affichage des paramètres de controle
86     void Affiche() const ;
87     // définition interactive exhaustive des paramètres de contrôle indépendamment de l'algorithme
88     // écriture du fichier de commande
89     void Info_commande_ParaAlgoControle(UtilLecture& lec);
90
91     // sauvegarde sur base info
92     // cas donne le niveau de sauvegarde
93     // = 0 : initialisation de la sauvegarde -> c'est-à-dire de la sortie base info
94     // = 1 : on sauvegarde tout
95     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
96     // incre : numero d'incrément auquel on sauvegarde
97     // éventuellement est défini de manière spécifique pour chaque algorithme
98     // dans les classes filles
99     void Ecriture_base_info_Para
100         (ofstream& sort,const int cas) const;
101     // cas de la lecture spécifique à l'algorithme dans base_info
102     void Lecture_base_info_Para(ifstream& ent,const int cas);
103
104     // --- fonctions d'accès en lecture des paramètres
105     // 1) ==== parametres de controle généraux pour la résolution de l'équilibre
106     // générale
107     // nombre servant à gérer la frequence de sauvegarde
108     // il faut utiliser Cas_de_sauvegarde() pour gérer correctement le nombre ramené en paramètre
109     bool Sauvegarde() const { if (sauvegarde != 0.) return true; else return false;};
110     // indique si la sauvegarde est autorisée, en fonction des paramètres données par l'utilisateur
111     // et les variables
112     // passée en argument
113     // dernier_calcul : indique si c'est le dernier calcul ou pas
114     bool SauvegardeAutorisee(int incre,const double& temps_derniere_sauvegarde,bool dernier_calcul)
115     const;
116
117     // maxi d'iteration pour converger
118     int Iterations() const { return iterations;};
119     // precision de la convergence
120     double Precision() const { return precision;};
121     // precision sur le temps final
122     double Prectemps() const { return tempo_specifique_algo.prectemps;};
123     // type de norme de convergence
124     Deux_String Norme() const { return norme;};
125     // cinematique sur l'increment ?
126     bool Cinematique() const { return cinematique;};
127     // convergence forcee ?
128     bool Conv_forcee() const { return conv_forcee;};
129     // multiplicateur de la charge
130     double Multiplicateur() const { return multiplicateur;};
131     // increment de temps
132     double Deltat() const { return tempo_specifique_algo.deltat;};
133     // increment de temps maxi
134     double Deltatmaxi () const {return tempo_specifique_algo.deltatmaxi;};
135     // increment de temps mini
136     double Deltatmini () const {return tempo_specifique_algo.deltatmini;};
137     // temps de fin de calcul
138     double Tempsfin() const { return tempo_specifique_algo.tempsfin;};
139     // test pour savoir si les incréments de temps courant ou maxi dépendent d'un temps critique
140     bool DeltatOuDeltatmaxDependantTempsCritique() const
141     {if ((coef_pas_critique_deltat!= coef_defaut_pa_critique)
142     // ((coef_pas_critique_deltat!=0.)
143     ||(coef_pas_critique_deltatmaxi!=0.)) return true; else return false;};
144     double Coefficient_pas_critique_deltat() const {return coef_pas_critique_deltat;};
145     static double Coef_defaut_pa_critique() {return coef_defaut_pa_critique;} // valeur par défaut
146     de coef_pas_critique_deltat
147
148     double Coefficient_pas_critique_deltatmaxi() const {return coef_pas_critique_deltatmaxi;};
149     // maximum d'increment de temps
150     int Maxincre () const { return maxincre;};
151     // maximum de tentative d'increment de temps permis
152     int Max_essai_incre () const { return max_essai_incre;};
153     // increment de redemarrage de calcul

```

```

151     int Restart() const { return restart;};
152     // maximum de puissance tolérée
153     double Max_puissance() const { return max_puissance;};
154     // indique si oui ou non on utilise le line_search
155     bool Line_search() const { return line_search;};
156     // indique si oui ou non le chargement externe participe à la raideur
157     bool Var_charge_externe() const {return var_charge_externe;};
158     // indique si oui ou non la variation du jacobine participe à la raideur
159     bool Var_jacobien() const {return var_jacobien;};
160     // indique si oui ou non la variation de la vitesse de déformation participe à la raideur
161     bool Var_D() const {return var_D;};
162     //--- globalisation des paramètres liés au temps
163     static const VariablesTemps& Variables_de_temps() {return tempo;};
164     // idem pour les grandeurs spécifique
165     const VariablesTemps& Variables_de_temps_specifiques_algo() {return tempo_specifique_algo;};
166
167     // 2) ==== paramètres liés au système d'équations linéaires
168     // type de stockage matriciel
169     Enum_matrice Type_Matrice() const { return type_matrice;};
170     // assemblage symétrique ou non
171     bool Symetrie_matrice () const {return symetrie_matrice;};
172     // type de résolution
173     Enum_type_resolution_matri Type_resolution() const { return type_resolution;};
174     // type de préconditionnement
175     Enum_preconditionnement Type_preconditionnement() const { return type_preconditionnement;};
176     // nombre d'itération dans le cas d'une méthode itérative
177     int Nb_iter_nondirecte() const { return nb_iter_nondirecte;};
178     // tolérance de convergence dans le cas d'une méthode itérative
179     double Tolerance() const { return tolerance;};
180     // nombre de vecteur sauvegardé dans le cas d'une méthode
181     // itérative avec redépart
182     int Nb_vect_restart() const { return nb_vect_restart;};
183     // même chose pour les matrices secondaires éventuelles
184     // a priori les listes sont parcourues selon l'orde de type_matrice_secondaire, si les listes
185     // ont des // tailles différentes, les algos s'en débrouillent (à voir dans la doc et les algos)
186     // *** ce n'est pas la peine d'essayer de remplacer les listes par des tableaux, c'est
187     // impossible // car Tableau_T intègre ParaGlob qui intègre ParaAgloControle et donc c'est un cercle infernal
188     // donc si l'on veut vraiment des tableaux il faut passer par Tableau<T>::Init_from_list
189     // et en dehors de la classe ParaAlgoControle !!
190     const list < Enum_matrice >& Type_matrice_secondaire() const {return type_matrice_secondaire;};
191     const list < Enum_type_resolution_matri >& Type_resolution_secondaire() const {return
192     type_resolution_secondaire;};
193     const list < Enum_preconditionnement >& Type_preconditionnement_secondaire() const {return
194     type_preconditionnement_secondaire;};
195     const list < int >& Nb_iter_nondirecte_secondaire() const {return
196     nb_iter_nondirecte_secondaire;};
197     const list < double >& Tolerance_secondaire() const {return tolerance_secondaire;};
198     const list < int >& NB_vect_restart_secondaire() const {return nb_vect_restart_secondaire;};
199
200     // indique si l'on veut ou non une optimisation des pointeurs d'assemblage
201     // notamment en fonction des CLL
202     int Optimisation_pointeur_assemblage() const {return opti_pointeur_assemblage;};
203
204     // 3) ==== paramètres liés au pilotage de l'équilibre global
205     // indication du type de pilotage retenu
206     EnumTypePilotage TypeDePilotage() const {return type_de_pilotage;};
207     // facteur de diminution de l'incrément de chargement
208     double Facteur_diminution() const {return facteur_diminution;};
209     // facteur d'augmentation de l'incrément de chargement
210     double Facteur_augmentation() const {return facteur_augmentation;};
211     // facteurs de diminution après une mauvaise convergence
212     double Fact_dim_en_mauvaiseConv() const {return fact_dim_en_mauvaiseConv;};
213     // nb de bonne convergence -> augmentation du deltatt
214     int Nb_bonne_convergence() const {return nb_bonne_convergence;};
215     // nb d'iter pour statuer sur la bonne convergence
216     int Nb_iter_pour_bonne_convergence() const {return nb_iter_pour_bonne_convergence;};
217     // nb d'iter pour statuer sur une mauvaise convergence
218     int Nb_iter_pour_mauvaise_convergence() const {return nb_iter_pour_mauvaise_convergence;};
219     // nombre d'itération avec un comportement tangent simple
220     // type élasticité par exemple, pour stabiliser les itérations avec des
221     // comportements plus complexes.
222     int Init_comp_tangent_simple() const {return init_comp_tangent_simple;};
223     // récup du facteur de sur ou sous relaxation
224     double SurSousRelaxation() const {return sur_sous_relaxation;};
225     // récup de la limitation maxi de l'incrément de ddl
226     double NormeMax_increment() const {return norme_incre_max;};
227     // récup de la limitation maxi de l'incrément de ddl sur les déplacements
228     double NormeMax_X_increment() const {return norme_incre_X_max;};
229     // récup de la limitation maxi de l'incrément de ddl sur les vitesses
230     double NormeMax_V_increment() const {return norme_incre_V_max;};
231     // récup de la limite supérieure permise pour la variation des ddl, pour le test convergence
232     double VarMaxiDdl() const {return varMaxiDdl;};
233     // récup de la limite inférieure permise pour la variation des ddl, pour le test convergence
234     double VarMiniDdl() const {return varMiniDdl;};
235     // récup du nombre de cycle pour le controle du max de résidu

```

```

233     int NbCycleControleResidu() const {return nbCycleControleResidu;};
234     // récup de combien de fois à suivre il faut que le controle de variation de résidu soit faux
235     // pour conclure à la non convergence
236     int PlageControleResidu() const {return plageControleResidu;};
237     // initialisation de l'incrément avec l'incrément du pas précédent : oui ou non
238     double IniIncreAvecDeltaDdlPrec() const { return initIncreAvecDeltaDdlPrec; };
239     // ramène l'indicateur de cas : pour le traitement du cas d'un jacobien négatif
240     int JacobienNegatif() const { return jacobien_negatif; };
241     // ramène la variation maxi autorisé pour le jacobien
242     // si <=1 veut dire que le paramètre n'est pas activé (il ne faut pas en tenir compte)
243     double MaxiVarJacobien() const { return var_maxi_jacobien; };
244
245     // ramène l'indicateur de cas : pour le traitement ou une fonction nD
246     // pour le chargement, signale un pb
247     int Cas_fctnD_charge() const { return cas_fctnD_charge; };
248
249     // 4) paramètres liés à la dynamique
250     // type de calcul de la masse
251     Enum_calcul_masse Type_calcul_masse() const {return type_calcul_masse;};
252     // limitation du temps maxi stable : active ou pas
253     bool Limit_temps_stable() const { return limitation_temps_maxi_stable;};
254     // indique si l'on inclut automatiquement un amortissement visqueux artificiel
255     // =0 pas d'amortissement, = 1 amortissement de tyme rayleigh
256     // = 2 amortissement calculé à partir de l'amortissement critique approché (première méthode)
257     // = 3 idem 2, mais autre méthode de calcul
258     // = 4 amortissement critique utilisé avec les algo de relaxation dynamique
259     int Amort_visco_artificielle() const {return amort_visco_arti;};
260     // change le paramètre indiquant l'amortissement visqueux artificiel
261     void Change_amort_visco_artificielle(int val_amort) { amort_visco_arti = val_amort;};
262     // valeur de la viscosité dynamique
263     double Visco_artificielle() const {return visco_arti; };
264     // la matrice d'amortissement dynamique [C] est construite à partir des coeffs de rayleigh
265     // sur la masse et sur la raideur
266     // coefficient de rayleigh sur la masse, pour la construction de la matrice de viscosité
267     double CoefRayleighMasse( )const {return coef_rayleigh_masse;};
268     // coefficient de rayleigh sur la raideur, pour la construction de la matrice de viscosité
269     double CoefRayleighRaideur() const {return coef_rayleigh_raideur;};
270     // cas du calcul de la viscosité critique : coef de bornage de la viscosité critique
271     double BorneMaxiSurCfonctionDeLaMasse() const {return maxi_C_en_fonction_M;};
272     // booleen indiquant si l'on veut l'intervention du bulk viscosity ou non
273     int BulkViscosity() const { return bulk_viscosity;};
274     // les deux coefficients de la formule du bulk viscosity: q=ro l (coef1 l trace(D) - coef2 a
    trace(D)^2)
275     DeuxDoubles CoefsBulk() const { return DeuxDoubles(c_Tracebulk,c_Trace2bulk);};
276
277     // 5) paramètres liés à l'affichage des résultats
278     // fréquence sur les increments de l'affichage des résultats
279     int Freq_affich_incre() const {return frequence_affichage_increment;};
280     // fréquence sur les itération de l'affichage des résultats
281     int freq_affich_iter() const {return frequence_affichage_iteration;};
282
283     // nombre servant à gérer la fréquence de sauvegarde au fil du calcul
284     // il faut utiliser SauvegardeFilCalculAutorisee() pour gérer correctement le nombre ramené en
    paramètre
285     bool Sortie_fil_calcul() const { if (sortie_fil_calcul > 0.) return true; else return false;};
286     // indique si la sauvegarde au fil du calcul est autorisée, en fonction des paramètres données
    par l'utilisateur et les variables
287     // passée en argument
288     bool SauvegardeFilCalculAutorisee(int incre,const double& temps_derniere_sauvegarde,bool
    dernier_calcul) const;
289
290     // ramène vraie si le numéro d'increment est nulle modulo la fréquence d'affichage d'incrément
291     // si ce dernier est > 0, sinon =0 modulo la fréquence de sauvegarde si celle-ci est > 0 sinon
    false
292     // dans le cas où les sorties s'effectue en fonction du temps, passage du dernier temps de
    sauvegarde
293     bool Vrai_commande_sortie(int icharge,const double& temps_derniere_sauvegarde) const;
294     // nombre de diggit utilisé pour afficher des réels en double précision pour l'archivage calcul
    (restart
295     // on sortie des résultats) (en fait utilisé avec ParaGlob avec une fonction statique)
296     int Nb_diggit_double_calcul() const { return nb_diggit_double_calcul;};
297     // nombre de diggit utilisé pour afficher des réels en double précision pour la visualisation
    graphique
298     // (en fait utilisé avec ParaGlob avec une fonction statique)
299     int Nb_diggit_double_graphique() const { return nb_diggit_double_graphique;};
300     // nombre de diggit utilisé pour afficher des réels en double précision sur l'écran
301     // (en fait utilisé avec ParaGlob avec une fonction statique)
302     int Nb_diggit_double_ecran() const { return nb_diggit_double_ecran;};
303
304     // 6) paramètres liés au contact
305     // ramène la précision voulu sur le test de la pénétration d'un point initial avant le début du
    calcul
306     double Precision_point_interne_debut() const { return prec_pt_int_deb; };
307     // ramène un facteur multiplicatif du déplacement maxi entre t et tdt, donnant la
308     // distance maxi admissible entre le point et le point projeté
309     double FacPourRayonAccostage() const {return factPourRayonAccostage;};
310     // ramène la distance maxi adminssible entre le point et le point projeté

```

```

311         double DistanceMaxiAuPtProjete() const {return distanceMaxiAuPtProjete;};
312 // ramène le pourcentage de plus que l'on prend pour les boites de prélocalisation d'élément
313 double Extra_boite_prelocalisation() const { return extra_boite_prelocalisation; };
314 // ramène un paramètre "r" qui multiplié par la taille maxi d'un élément donne l'épaisseur
315 // que l'on met autour de la boite de prélocalisation pour éviter certaine dimension nulle à
cette boite
316 double Rapport_Extra_boite_mini_prelocalisation() const { return
mini_extra_boite_prelocalisation; };
317 // ajout_extra_boite_prelocalisation est un paramètre qui est ajouté dans tous les directions
aux boites de préloca
318 double Ajout_extra_boite_prelocalisation() const {return ajout_extra_boite_prelocalisation;};
319
320 // ramène le type de contact et son existence
321 // ==0 : pas de contact, !=0 -> le type de contact
322 // ==1 : contact cinématique, sans multiplicateur ni pénalisation (contact original)
323 // ==2 : contact avec pénalisation
324 int ContactType() const {return contact_type;};
325 // ramène un string décrivant une fonction nD de pilotage pour le type de contact 4
326 // = "_" si aucune fonction n'est définit
327 string Fct_nD_bascul_contact_type_4() const {return fct_nD_bascul_contact_type_4;};
328
329 // ramène le facteur brut de pénalisation pour la pénétration
330 double PenalisationPenetrationContact() const {return penalisationPenetration;};
331 // ramène un string décrivant une fonction nD de pilotage équivalente
332 // = "_" si aucune fonction n'est définit
333 string Fct_nD_penalisationPenetration() const {return fct_nD_penalisationPenetration;};
334
335 // ramène le type de contrôle du facteur de pénalisation pour la pénétration
336 int TypeCalculPenalisationPenetration() const {return typePenalisationPenetration;};
337 // ramène une borne maxi de pénétration (dont l'utilisation dépend de l'algo)
338 double Penetration_contact_maxi() const {return penetration_contact_maxi;};
339 // ramène un string décrivant une fonction nD de pilotage équivalente
340 // = "_" si aucune fonction n'est définit
341 string Fct_nD_penetration_contact_maxi() const {return fct_nD_penetration_contact_maxi;};
342
343 // ramène une borne de régularisation sur la penetration (dont l'utilisation dépend de l'algo)
344 double Penetration_borne_regularisation() const {return penetration_borne_regularisation;};
345 // ramène un string décrivant une fonction nD de pilotage équivalente
346 // = "_" si aucune fonction n'est définit
347 string Fct_nD_penetration_borne_regularisation() const {return
fct_nD_penetration_borne_regularisation;};
348
349 // ramène une borne maxi pour la force de réaction du noeud (dont l'utilisation dépend de
l'algo)
350 double Force_contact_noeud_maxi() const {return force_contact_noeud_maxi;};
351 // ramène un string décrivant une fonction nD de pilotage équivalente
352 // = "_" si aucune fonction n'est définit
353 string Fct_nD_force_contact_noeud_maxi() const {return fct_nD_force_contact_noeud_maxi;};
354
355 // ramène le facteur de pénalisation pour le déplacement tangentiel
356 double PenalisationTangentielleContact() const {return penalisationTangentielle;};
357 // ramène un string décrivant une fonction nD de pilotage équivalente
358 // = "_" si aucune fonction n'est définit
359 string Fct_nD_penalisationTangentielle() const {return fct_nD_penalisationTangentielle;};
360
361 // ramène le type de contrôle du facteur de pénalisation pour le dep tangentiel
362 int TypeCalculPenalisationTangentielle() const {return typePenalisationTangentielle;};
363 // ramène une borne maxi de dep tangentiel (dont l'utilisation dépend de l'algo)
364 double Tangentielle_contact_maxi() const {return tangentielle_contact_maxi;};
365 // ramène un string décrivant une fonction nD de pilotage équivalente
366 // = "_" si aucune fonction n'est définit
367 string Fct_nD_tangentielle_contact_maxi() const {return fct_nD_tangentielle_contact_maxi;};
368
369 // ramène une borne de régularisation sur le dep tangentiel (dont l'utilisation dépend de
l'algo)
370 double Tangentielle_borne_regularisation() const {return tangentielle_borne_regularisation;};
371 // ramène un string décrivant une fonction nD de pilotage équivalente
372 // = "_" si aucune fonction n'est définit
373 string Fct_nD_tangentielle_borne_regularisation() const {return
fct_nD_tangentielle_borne_regularisation;};
374
375 // ramène une borne maxi pour la force tangentiel sur le noeud (dont l'utilisation dépend de
l'algo)
376 double Force_tangentielle_noeud_maxi() const {return force_tangentielle_noeud_maxi;};
377 // ramène un string décrivant une fonction nD de pilotage équivalente
378 // = "_" si aucune fonction n'est définit
379 string Fct_nD_force_tangentielle_noeud_maxi() const {return
fct_nD_force_tangentielle_noeud_maxi;};
380
381 // précision de positionnement sur les X^ar d'un pt en contact sur une frontière
382 double Precision_pt_sur_front() const {return prec_pt_sur_frontiere;};
383 // nb de boucle maxi pour la recherche du positionnement du pt en contact
384 int Nb_boucle_newton_position_sur_frontiere()const {return
nb_boucle_newton_position_frontiere;};
385 // nb de fois un noeud décolle pour n'être plus considéré en contact
386 int NbDecolAutorise()const {return nbDecolAutorise;};
387 // retourne le type de méthode utilisée pour gérer le décollement

```

```

388     int TypeDeDecolement() const {return typeDeDecolement;};
389     // retourne le nombre de positions successives, utilisé pour faire une moyenne glissante
de ces positions
390     int Nb_moy_glissant() const {return nb_glissant;};
391     // retourne le niveau de commentaire entre 0 et 10, pour les algo de contact
392     // le niveau utilisé est a priori le max du niveau général et de celui-ci
393     int Niveau_commentaire_contact() const {return niveau_commentaire_contact;};
394
395     // retour d'un indication concernant une optimisation éventuelle
396     //de la numerotation en tenant compte des éléments de contact
397     int Optimisation_numerotation() const {return optimisation_numerotation;};
398
399     // 7) paramètres liés aux calculs des énergies
400     // ramène l'incrément mini à partir duquel on calcul les énergies cumulées
401     int NbIncrCalEnergie() const {return nb_incr_cal_ener;};
402     // indique si oui ou non le calcul et l'affichage des énergies sur l'incrément est effectué
403     bool AfficheIncrEnergie() const {return affiche_incr_energie;};
404
405     // 8) paramètres liés aux calculs géométriques sur les éléments
406     // -- paramètres relatifs à la recherche d'un point interne à un élément (dans ElemMeca)
407     // coordonnée thetai ok si delta thetai < à la prec
408     double PointInterneDeltaThetaiMaxi() const {return point_interne_delta_thetai_maxi;};
409     // precision sur thetai sur le test du point interne
410     double PointInternePrecThetaiInterne() const {return point_interne_prec_thetai_interne;};
411     // maxi boucle de Newton sur la recherche des thetai
412     int PointInterneNbBoucleSurDeltaThetai() const {return
point_interne_nb_boucle_sur_delta_thetai;};
413     //nb max de test positif "externe" pour statuer un point externe
414     int PointInterneNbExterne() const {return point_interne_nb_externe;};
415     // indique si oui ou non on calcul les volumes entre la surface et les 3 plans de ref
(valable uniquement en 3D pour des surfaces)
416     bool CalVolTotalEntreSurfaceEtPlansRef() const {return
cal_vol_total_entre_surface_et_plans_ref;};
417     // indique le rapport maxi autorisé entre le jacobien de la facette centrale
418     // et le jacobien au pti pour les éléments coques
419     double Ratio_maxi_jacoMembrane_jacoPti() const {return ratio_maxi_jacoMembrane_jacoPti;};
420     // indique quelle méthode numérique est utilisée pour effectuer
421     // l'inversion des tenseurs métriques: exe : utilisation de la méthode de Cramer (méthode
historique)
422     // ou encore : utilisation d'une méthode LU avec équilibrage des ligne (second méthode
implantée)
423     Enum_type_resolution_matri Type_calnum_inversion_metrrique() const {return
type_calnum_inversion_metrrique;};
424
425     // --- fonctions d'accès en modification des paramètres -----
426     // 1) ==== parametres de controle généraux pour la résolution de l'équilibre
427     // générale
428     // modification de l'increment de temps,
429     // si c'est >= au deltat mini et <= deltat maxi, => retour 0
430     // si c'est < au deltat mini et que l'on est déjà au temps mini ==> retour -1 : pas de
modification du deltat
431     // si c'est > au deltat maxi et que l'on est déjà au temps maxi ==> retour 1 : pas de
modification du deltat
432     int Modif_Deltat(double nouveau_temps);
433     // modification de l'incrément dans les bornes
434     // si nouveau_temps < deltat mini ==> nouveau_temps = deltat mini
435     // si nouveau_temps > deltat maxi ==> nouveau_temps = deltat maxi
436     // sinon on laisse le delta_t à la valeur qu'il a actuellement
437     // ramène oui ou non si ça a été modifié
438     bool Modif_Deltat_dans_borne(double& nouveau_temps);
439     // modification directe du temps
440     void Modif_Temps(double nouveau_temps) { tempo_specifique_algo.temps = nouveau_temps =
nouveau_temps;};
441     // mise à jour éventuel du pas de temps et du pas de temps maxi et mini dans le cas où ils sont
définit avec des coeff
442     // d'un pas critique, ramène true s'il y a eu modification du pas de temps courant, false sinon
443     // temps_critique : temps critique de l'algorithme
444     // temps_critiqueDFC : temps critique de l'algo DFC = condition de courant
445     bool Modif_Deltat_DeltatMaxi(double temps_critique, double temps_critiqueDFC);
446     // modif arbitraire (sans aucune vérification) des paramètres de temps courant
447     // ** a utiliser à bon essient
448     void Modif_temps(const double& tem,const double& deltatem)
449     {tempo_specifique_algo.temps = tempo.temps = tem;
450     tempo_specifique_algo.deltat = tempo.deltat = deltatem;};
451     // modification de var_D
452     void Modif_Var_D(bool va_D) {var_D = va_D;};
453     // modification de var_jacobien
454     void Modif_Var_jacobien(bool va_jacobien) {var_jacobien = va_jacobien;};
455
456     //----- gestion de certaines interruptions systèmes -----
457     // ces variables ne sont pas sauvegardées, elles sont accessibles de partout pour un projet
donné
458     // on se sert de ParaAlgoControle pour le stockage, l'accès et la modification
459     //---
460     void ChangeSortieEquilibreGlobal(bool val) {sortieEquilibreGlobal=val;};
461     bool EtatSortieEquilibreGlobal() const {return sortieEquilibreGlobal;};
462     //---

```

```

463     void ChangeSortieEtatActuelDansBI(bool val) {sortieEtatActuelDansBI=val;};
464     bool EtatSortieEtatActuelDansBI() const {return sortieEtatActuelDansBI;};
465     //---
466     void ChangeSortieEtatActuelDansCVisu(bool val) {sortieEtatActuelDansCVisu=val;};
467     bool EtatSortieEtatActuelDansCVisu() const {return sortieEtatActuelDansCVisu;};
468     // on met à jour les infos globales
469     std::vector<double>& Mise_a_jour_Grandeurs_globales();
470
471 protected :
472     // VARIABLES PROTEGEES :
473     // 1) parametres de controle généraux pour la résolution de l'équilibre
474     // générale
475     double sauvegarde; // un entier ou double servant a contrôler la frequence de sauvegarde
476     // dans le cas le plus simple : = le nombre d'incrément séparant deux pas de sauvegarde
477     int cas_de_sauvegarde; // associé au paramètre sauvegarde: indique comment le contrôle de la
fréquence de
478     // sauvegarde s'effectue: =1 : cas le plus simple (cf si-dessus)
479     //                                     =2 : cas où sauvegarde indique le deltat qui existe entre 2
sauvegardes
480     //                                     =3 : cas où on sauvegarde l'incrément 0 et le dernier incrément
481     //                                     =4 : cas 1 + le dernier incrément
482     //                                     =5 : cas 2 + le dernier incrément
483     int iterations; // maxi d'iteration pour converger
484     double precision; // precision de la convergence
485     Deux_String norme; // type de norme de convergence: le premier string = la norme sauf si
486     // le premier= fonction_nD: dans ce cas le second = le nom de la fonction
nD
487     bool cinematique; // cinematique sur l'incrément ?
488     bool conv_forcee; // convergence forcee ?
489     double multiplicateur; // multiplicateur de la charge
490     // ---- le temps,
491     // contrairement aux autres variables, il s'agit d'une variable static pour qu'il n'y ait qu'un
seul
492     // temps dans le calcul. C'est aussi une nécessité pour la liaison avec la classe ParaGlob,
493     // sinon cela pose des pbs dans le cas de plusieurs variable ParaAlgoControle
494     static VariablesTemps tempo; // contiens les grandeurs courantes:
495     // temps courant, l'incrément de temps, l'incrément de temps maxi
496     // le temps fin, la précision sur les tests sur les temps
497     // les grandeurs spécifiques de l'algo:
498     // la seule grandeurs qui est commune c'est le temps courant: celui stocké dans "tempo"
499     VariablesTemps tempo_specifique_algo;
500     bool force_deltat; // indique si oui ou non, lors d'un restart on force l'utilisation de
l'incrément de temps
501     // indiqué dans le .info, ou au contraire on reprend celui du pas précédent
502     // ne sert qu'une fois, ensuite elle est mise à false !!
503     double coef_pas_critique_deltat; // coeff du pas critique pour deltat s'il existe
504     static double coef_defaut_pa_critique; // valeur par défaut de coef_pas_critique_deltat
505     double coef_pas_critique_deltatmaxi; // coeff du pas critique pour deltatmaxi s'il existe
506     double coef_pas_critique_deltatmini; // coeff du pas critique pour deltatmini s'il existe
507     bool typeDFC_pas_critique_deltat; // = true: il s'agit du pas critique de DFC, false : pas
critique spécifique
508     bool typeDFC_pas_critique_deltatmaxi; // idem
509     bool typeDFC_pas_critique_deltatmini; // idem
510     // --- fin pour le tempq
511     int maxincree ; // maximum d'incrément de temps
512     int max_essai_incre ; // maximum de tentative d'incrément de temps permis
513     int restart; // increment de redemarrage de calcul
514     double max_puissance; // maximum de puissance tolérée
515     bool line_search; // indique si oui ou non on utilise le line_search
516     bool var_charge_externe; // indique si oui ou non le chargement externe
517     // participe à la raideur
518     bool var_jacobien; // indique si l'on tiend compte de la variation du jacobien
519     // dans le calcul de la raideur
520     bool var_D; // indique si l'on tiend compte de la variation de D* dans la raideur
521
522     // 2) paramètres liés au système d'équations linéaires
523
524     bool symetrie_matrice; // indique si le stockage est symétrique ou non
525     Enum_matrice type_matrice; // type de stockage matriciel
526     Enum_type_resolution_matri type_resolution; // type de résolution
527     Enum_preconditionnement type_preconditionnement; // type de preconditionnement
528     int nb_iter_nondirecte; // nombre d'itération dans le cas d'une méthode itérative
529     double tolerance; // tolerance de convergence dans le cas d'une méthode itérative
530     int nb_vect_restart; // nombre de vecteur sauvegardé dans le cas d'une méthode
531     // itérative avec redépart
532     // même chose pour les matrices secondaires éventuelles
533     list< Enum_matrice > type_matrice_secondaire; // type de stockage matriciel
534     list< Enum_type_resolution_matri > type_resolution_secondaire; // type de résolution
535     list< Enum_preconditionnement > type_preconditionnement_secondaire; // type de
preconditionnement
536     list< int > nb_iter_nondirecte_secondaire; // nombre d'itération dans le cas d'une méthode
itérative
537     list< double > tolerance_secondaire; // tolerance de convergence dans le cas d'une méthode
itérative
538     list< int > nb_vect_restart_secondaire; // nombre de vecteur sauvegardé dans le cas d'une
méthode

```



```

539                                     // itérative avec redépart
540 // indique si l'on veut ou non une optimisation des pointeurs d'assemblage
541 // notamment en fonction des CLL
542 int opti_pointeur_assemblage; // = 0 : pas d'optimisation,
543
544 // 3) paramètres liés au pilotage de l'équilibre global
545 EnumTypePilotage type_de_pilotage; // définit le type de pilotage retenue
546 double facteur_diminution; // facteurs d'augmentation, ou de diminution
547 double facteur_augmentation; // de l'incrément de chargement
548 double fact_dim_en_mauvaiseConv; // facteurs de diminution après une mauvaise convergence
549 int nb_bonne_convergence; // nb de bonne convergence -> augmentation du deltat
550 int nb_iter_pour_bonne_convergence; // nb d'iter pour statuer sur la bonne convergence
551 int nb_iter_pour_mauvaise_convergence; // nb d'iter pour statuer sur une mauvaise convergence
552 double varMiniDdl, varMaxiDdl; // limites mini et maxi des ddl
553 int nbCycleControleResidu, plageControleResidu; // nb cycle et plage de controle du résidu
554
555 int init_comp_tangent_simple; // nombre d'itération avec un comportement tangent simple
556 // type élasticité par exemple, pour stabiliser les itérations avec des
557 // comportements plus complexes.
558 double sur_sous_relaxation; // facteur de sur ou sous relaxation
559 double norme_incre_max; // limitation de l'incrément maxi des ddl
560 double norme_incre_X_max; // limitation de l'incrément maxi des ddl de déplacement
561 double norme_incre_V_max; // limitation de l'incrément maxi des ddl de vitesse
562 double initIncreAvecDeltaDdlPrec; // initialisation de l'incrément avec l'incrément du pas
563 précédent
564 int jacobien_negatif; // traitement des jacobiens négatifs
565 double var_maxi_jacobien; // variation maxi autorisée du jacobien, si <= 0 pas pris en compte
566 int cas_fctnD_charge; // traitement du cas ou une fonction nD pour le chargement, signale un pb
567
568 // 4) paramètres liés à la dynamique
569 Enum_calcul_masse type_calcul_masse; // type de calcul de la masse
570 // le type de stockage de la matrice masse est celui de type_matrice
571 // utilisé que si la matrice masse n'est pas diagonale
572 bool limitation_temps_maxi_stable; // booléen qui indique si la limitation supérieur
573 // du temps pour satisfaire la stabilité, est active ou pas
574 int amort_visco_arti; // indique si l'on inclut automatiquement un amortissement visqueux
575 artificiel
576 // =0 pas d'amortissement, = 1 amortissement de tyme rayleigh
577 // = 2 amortissement calculé à partir de l'amortissement critique approché
578
579 double visco_arti; // valeur de la viscosité dynamique ou de la proportion d'amortissement
580 critique
581 double maxi_C_en_fonction_M; // bornage de C: maxi C = maxi_C_en_fonction_M * M (cas du calcul
582 de C critique)
583 // la matrice d'amortissement dynamique [C] est construite à partir des coeffs de rayleigh
584 // sur la masse et sur la raideur
585 double coef_rayleigh_masse, coef_rayleigh_raideur;
586 // int indiquant si l'on veut l'intervention du bulk viscosity ou non
587 // =0 pas d'intervention, =1 intervention si I_D est négatif, =2 intervention quelque soit
588 I_D
589 int bulk_viscosity;
590 // les deux coefficients de la formule du bulk viscosity: q=ro 1 (coef1 1 trace(D) - coef2 a
591 trace(D)^2)
592 double c_Tracebulk, c_Trace2bulk;
593
594 // 5) paramètres liés à l'affichage des résultats
595 // ---- a l'écran
596 int frequence_affichage_increment; // fréquence sur les increments de l'affichage des résultats
597 int frequence_affichage_iteration; // fréquence sur les itération de l'affichage des résultats
598 // --- en sortie au fil du calcul
599 double sortie_fil_calcul; // fréquence sur les incréments, pour la sortie au fil du calcul
600 int cas_de_sortie_fil_calcul; // associé au paramètre sortie_fil_calcul:
601 // indique comment le contrôle de la fréquence de sortie au fil du calcul s'effectue
602 // =1 : cas le plus simple (cf si-dessus)
603 // =2 : cas où sauvegarde indique le deltat qui existe entre 2 sauvegardes
604 // nombre de chiffre significatifs utilisé pour l'affichage des double précision: le premier
605 pour
606 // l'archivage du calcul le secon pour la visualisation graphique
607 // le troisième pour l'affichage à l'écran
608 int nb_diggit_double_calcul;
609 int nb_diggit_double_graphique;
610 int nb_diggit_double_ecran;
611
612 // 6) paramètres liés au contact
613 double prec_pt_int_deb; // précision du test: point à l'intérieur d'un élément
614 double factPourRayonAccostage; // facteur multiplicatif du déplacement maxi entre t et
615 tdt, donnant
616 // distance maxi admissible entre le point et le point
617 projeté
618 double distanceMaxiAuPtProjeté; // distance maxi admissible entre le point et le point
619 projeté
620 double extra_boite_prelocalisation; // donne la proportion de plus que l'on prend pour les
621 boites
622 // mini_extra_boite_prelocalisation est un paramètre qui multiplié par la taille maxi d'un
623 élément donne
624 // l'épaisseur que l'on met autour de la boite de prélocalisation pour éviter certaine dimension
625 nulle à cette boite

```



```
611     double mini_extra_boite_prelocalisation;
612     // ajout_extra_boite_prelocalisation est un paramètre qui est ajouté dans tous les directions
aux boites de préloca
613     double ajout_extra_boite_prelocalisation;
614     int contact_type; // indique si oui (!= 0) ou non (==0) il y a contact, et le nombre indique le
type de contact
615     string fct_nD_bascul_contact_type_4; // fonction nD qui permet de piloter le type 4
616
617     double penalisationPenetration; // facteur de pénalisation pour la pénétration
618     string fct_nD_penalisationPenetration; // fct nD dans le cas d'une valeur pilotée
619
620     int typePenalisationPenetration; // indique le type de calcul pour la pénalisation en
pénétration
621     double penetration_contact_maxi; // indique une borne maxi de pénétration (dont
l'utilisation dépend de l'algo)
622     string fct_nD_penetration_contact_maxi; // fct nD dans le cas d'une valeur pilotée
623
624     double penetration_borne_regularisation; // borne de régularisation sur la penetration
(dont l'utilisation dépend de l'algo)
625     string fct_nD_penetration_borne_regularisation; // fct nD dans le cas d'une valeur pilotée
626
627     double force_contact_noeud_maxi; // indique une borne maxi pour la force de réaction du
noeud (dont l'utilisation dépend de l'algo)
628     string fct_nD_force_contact_noeud_maxi; // fct nD dans le cas d'une valeur pilotée
629
630     double penalisationTangentielle; // facteur de pénalisation pour le déplacement tangentiel
631     string fct_nD_penalisationTangentielle;
632     int typePenalisationTangentielle; // indique le type de calcul pour la pénalisation en dep
tangentiel
633
634     double tangentielle_contact_maxi; // indique une borne maxi de dep tangentiel (dont
l'utilisation dépend de l'algo)
635     string fct_nD_tangentielle_contact_maxi; // fct nD dans le cas d'une valeur pilotée
636
637     double tangentielle_borne_regularisation; // borne de régularisation sur le dep tangentiel
638     string fct_nD_tangentielle_borne_regularisation; // fct nD dans le cas d'une valeur pilotée
639
640     double force_tangentielle_noeud_maxi; // indique une borne maxi pour la force tangentielle sur
le noeud
641     string fct_nD_force_tangentielle_noeud_maxi; // fct nD dans le cas d'une valeur pilotée
642
643     double prec_pt_sur_frontiere; // précision de positionnement sur les X*ar d'un pt en contact sur
une frontière
644     int nb_boucle_newton_position_frontiere; // nb de boucle maxi pour la recherche du
positionnement du pt en contact
645     int nbDecolAutorise; // nb de fois un noeud décolle pour n'être plus considéré en contact
646     int typeDeDecolement; // indique le type de méthode utilisée pour gérer le décollement
647     int nb_glissant; // le nombre de positions successives du noeud esclave, utilisé pour faire une
moyenne glissante
648     int niveau_commentaire_contact; // de 0 à 10: niveau des commentaires dans les algo de contact
649
650     int optimisation_numerotation; // indique éventuellement une optimisation de la numerotation en
tenant compte
651
652     // des éléments de contact
653
654     // 7) paramètres liés aux calculs des énergies et puissances
655     int nb_incr_cal_ener; // donne l'incrément mini, à partir duquel on calcul les énergie cumulée
656     bool affiche_incr_energie; // affichage oui ou non de l'énergie et du bilan sur l'incrément
657
658     // 8) paramètres liés aux calculs géométriques sur les éléments
659     double point_interne_delta_thetai_maxi; // coordonnée theta_i ok si delta theta_i < à la prec
660     double point_interne_prec_thetai_interne; // precision sur theta_i sur le test du point interne
661     int point_interne_nb_boucle_sur_delta_thetai; // maxi boucle de Newton sur la recherche des
theta_i
662     int point_interne_nb_externe; // nb max de test positif "externe" pour statuer un point externe
663     bool cal_vol_total_entre_surface_et_plans_ref; // indique si oui ou non on calcul ces volumes
(entre la surface et les 3 plans)
664     double ratio_maxi_jacoMembrane_jacoPti; // rapport maxi autorisé entre le jacobien de la facette
centrale
665     // et le jacobien au pti pour les éléments coques
666
667     Enum_type_resolution_matri type_calnum_inversion_métrique; // indique quelle méthode numérique
est utilisée pour effectuer
668     // l'inversion des tenseurs métriques: ex: utilisation de la méthode de Cramer (méthode
historique)
669     // ou utilisation d'une méthode LU avec équilibrage des ligne (second méthode
implantée)
670
671     //----- gestion de certaines interruptions systèmes -----
672     // ces variables ne sont pas sauvegardées, elles sont accessibles de partout pour un projet
donné
673     // on se sert de ParaAlgoControle pour le stockage, l'accès et la modification
674     bool sortieEquilibreGlobal; // par défaut non
675     bool sortieEtatActuelDansBI; // par défaut non
676     bool sortieEtatActuelDansCVisu; // par défaut non
677 };
```

```

678 /// @} // end of group
679
680 #endif

```

## 7.364 ParaGlob.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 #ifndef PARAGLOB_H
31 #define PARAGLOB_H
32 /*****
33 *   DATE:      19/01/2001
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *
40 *BUT:Ensemble de variables et de parametres globaux a tout le programme*
41 *   les methodes permettent de changer certaines valeurs ou
42 *   de leurs donner une seule fois une valeur differente
43 *   de celle par default.
44 *
45 *   *****
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !           but
50 *   -----
51 *   !           !           !
52 *   *****
53 *
54 *   MODIFICATIONS:
55 *
56 *   ! date !   auteur !           but
57 *   -----
58 *
59 *   *****/
60 #include <iostream>
61 using namespace std;
62 #include <string>
63 #include <list>
64 #include <vector>
65 #include <csignal> // pour les interruptions systèmes
66 #include "UtilLecture.h"
67 #include "EnumTypeCalcul.h"
68 #include "VariablesTemps.h"
69 #include "ParaAlgoControle.h"
70 #include "EnumLangue.h"
71 #include "Handler_exception.h"
72 #include "Enum_GrandeurGlobale.h"
73 #include "EnumTypeGrandeur.h"
74
75
76 /** @defgroup Les_parametres_generaux
77 *
78 *   BUT:   groupe relatif à la gestion des paramètres généraux

```

```

79 *
80 *
81 * \author   Gérard Rio
82 * \version  1.0
83 * \date     19/01/2001
84 * \brief    groupe relatif à la gestion des paramètres généraux
85 *
86 */
87
88 /// @addtogroup Les_parametres_generaux
89 /// @
90 ///
91
92
93 class ParaGlob
94 { public :
95     // tout d'abord une méthode friend, qui a pour objectif de modifier éventuellement
96     // l'instance ParaAlgoControle courante
97     friend void Handler_signal(int theSignal);
98     friend class ParaAlgoControle;
99     friend class AlgoriCombine;
100
101     // déclaration de classe
102     ParaGlob (); // constructeur par défaut
103     ParaGlob (int dim); // constructeur en fonction de la dimension du pb
104     ~ParaGlob (); // destructeur
105     // Constructeur de copie
106     ParaGlob (const ParaGlob & nd);
107
108     // les methodes publiques
109
110     static const int Dimension () // retourne la dimension du pb
111     { return dimensionPb; };
112     const int NombreMaillage() // retourne la taille initiale du tableau de maillage
113     { return NombreMaxiDeMaillage; };
114
115     // la dimension du pb ne peut etre change qu'une seule fois
116     void ChangeDimension(int ); //
117
118     // retourne le nombre de composantes des tenseurs en absolu
119     static const int NbCompTens ()
120     { return nbComposantesTenseur; };
121
122
123     static ParaGlob * param; // pointeur sur le seul membre ouvert
124
125     // Ramène une référence sur l'ensemble des paramètres de contrôles qui sont actuellement "actifs"
126     // ne sera utile que pour de la consultation
127     const ParaAlgoControle & ParaAlgoControleActifs() const {return *paraAlgoControle;};
128     // change l'ensemble des paramètres de contrôles qui sont actuellement "actifs"
129     void ChangeParaAlgoControleActifs( ParaAlgoControle * para) {paraAlgoControle = para;};
130
131     // lecture du type de calcul et d'une liste de sous_type éventuel
132     void LecTypeCalcul(UtilLecture& lec);
133
134     // définition interactive du type de calcul et d'une liste de sous_type éventuel
135     // écriture du fichier de commande
136     void Info_commande_TypeCalcul(UtilLecture& lec);
137
138     // retourne le type de calcul principal: en fait le type maître
139     // qui est unique et dure pendant toute l'exécution
140     // il peut-être différent de celui de l'algo en cours, du coup il permet de savoir
141     // à l'algo en cours s'il est maître ou pas
142     EnumTypeCalcul TypeCalcul_maître() {return typeDeCalcul_maître;};
143     // retourne vraie si le type de calcul demandé est celui en cours
144     // et s'il est actif
145     // dans le cas où le paramètre optionnel est true, il n'y a pas de test
146     // si le type de calcul demandé est actif ou pas
147     bool TypeCalcul_principal(EnumTypeCalcul type,bool actif=false);
148     // détermine si le sous type de calcul existe et s'il est actif
149     bool SousTypeCalcul(EnumSousTypeCalcul soustype);
150
151     // détermine si le type de calcul est activé ou pas
152     bool Avec_typeDeCalcul() const { return avec_typeDeCalcul;};
153     // retourne la liste de sous_types en cours
154     list <EnumSousTypeCalcul> const & LesSousTypesDeCalcul() const { return soustypeDeCalcul;};
155     // retourne une liste de booléen cohérente avec la liste des sous_types
156     // chaque booléen indique si le sous_type correspondant est actif ou pas
157     list <bool> const& Avec_soustypeDeCalcul() const{ return avec_soustypeDeCalcul;};
158
159     // affichage du type de calcul et des sous_types éventuelles
160     void AfficheTypeEtSousTypes();
161
162     // retourne le niveau d'impression
163     static int NiveauImpression() {return nivImpression;};
164     // modifie le niveau d'impression
165     void Change_niveau_impimpression(int n );

```

```

166 // ramène l'identificateur de la version d'herezh++
167 // sur 5 caractères
168 static string NbVersion() { return nbVersion;};
169 // affichage de la version sur la sortie passée en argument
170 static void Sortie_Version(ofstream & sort)
171 { sort << "\n version " << nbVersion ;};
172 // affichage de la version sur cout
173 static void Sortie_Version()
174 { cout << "\n version " << nbVersion ;};
175 // lecture de la version sur l'entrée passée en argument
176 static string Lecture_Version(istream& entr)
177 { string toto; entr >toto > NbVersionsurfichier; return NbVersionsurfichier;};
178 // lecture de la version sur l'entrée passée en argument
179 static string Lecture_Version(istream& entr)
180 { string toto; entr >toto > NbVersionsurfichier; return NbVersionsurfichier;};
181 // retour de la version lue sur fichier
182 static string NbVersion_de_fichier(){return NbVersionsurfichier;};
183
184
185 // lecture de l'indicateur de langue sur l'entrée passée en argument
186 static EnumLangue Lecture_Langue(istream& entr)
187 { string toto; entr >toto > langueHZ; return langueHZ;};
188 // lecture de l'indicateur de langue sur l'entrée passée en argument
189 static EnumLangue Lecture_Langue(istream& entr)
190 { string toto; entr >toto > langueHZ; return langueHZ;};
191 // retour de la langue
192 static EnumLangue Langage(){return langueHZ;};
193 static bool Anglais() {return (langueHZ == ENGLISH);};
194 static bool Francais() {return (langueHZ == FRANCAIS);};
195 // affichage de la langue sur la sortie passée en argument
196 static void Sortie_Langue(ofstream & sort) { sort << "\n langue " << langueHZ ;};
197
198 // ramène le nombre de diggit utilisé pour afficher des réels en double précision pour l'archivage
199 // de calcul
200 static int NbdigdoCA() { return nb_diggit_double_calcul;};
201 // ramène le nombre de diggit utilisé pour afficher des réels en double précision pour l'affichage
202 // du graphique
203 static int NbdigdoGR() { return nb_diggit_double_graphique;};
204 // ramène le nombre de diggit utilisé pour afficher des réels en double précision pour l'affichage
205 // sur l'écran
206 static int NbdigdoEC() { return nb_diggit_double_ecran;};
207
208 // change le nombre de diggit utilisé pour l'affichage des réels en double précision
209 // int cas =1 pour le calcul, = 2 pour le graphique
210 // =3 pour l'écran
211 void Change_Nb_diggit_double(int cas, int nevez_nb_diggit);
212
213 // indique si oui ou non, l'espace est axisymétrique
214 static bool Axisymetrie() {return geometrie_axisymetrique;};
215 // changement d'espace non axisymétrique en un espace axisymétrique
216 // (c'est le seule changement accepter actuellement)
217 static void Change_en_Axisymetrie() {geometrie_axisymetrique=true;};
218
219 // une méthode permettant de lire les grandeurs globales
220 static std::vector <double>& Grandeurs_globales() {return grandeurs_globales;};
221
222 //----- lecture écriture dans base info -----
223 // cas donne le niveau de la récupération
224 // = 1 : on récupère tout
225 // = 2 : on récupère uniquement les données variables (supposées comme telles)
226 void Lecture_base_info(istream& ent,const int cas);
227 // cas donne le niveau de sauvegarde
228 // = 1 : on sauvegarde tout
229 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
230 void Ecriture_base_info(ofstream& sort,const int cas);
231
232 //----- gestion lecture dans le .info -----
233 // --- variable de lecture du .info, utilisées par des classes externes
234 // en particulier: Projet et les Algorithmes
235 // 1) récupération de l'état de la lecture dans le fichier .info
236 // =-1 : non initialisée
237 // =0 après la première lecture "globale" du .info
238 // =n après la nième lecture "secondaire" dans le .info
239 int EtatDeLaLecturePointInfo() const {return etatDeLaLecturePointInfo;};
240 void ChangeEtatDeLaLecturePointInfo(int etat) { etatDeLaLecturePointInfo=etat;};
241 // 1) demande d'une lecture secondaire dans le point info
242 // 0 pas de demande: valeur par défaut, = 1 demande d'une lecture supplémentaire
243 void ChangeDemandeLectureSecondaireInPointInfo(int val) {demandeLectureSecondaireInPointInfo=val;};
244 int EtatDemandeLectureSecondaireInPointInfo() const {return demandeLectureSecondaireInPointInfo;};
245
246
247 // ===== gestion du temps =====
248 static void Init_temps(VariablesTemps& temps)
249 { tempo = &temps;};
250 // récup des paramètres liés au temps
251 static const VariablesTemps& Variables_de_temps() {return *tempo;};
252

```

```

253 // ---- aller-retour en lecture sur des grandeurs globalisées
254 // la classe ParaGlob ne sert que de relais, elle stocke uniquement la grandeur
255 // mais n'est pas responsable de leur utilisation
256 // Il faudra utiliser un try, lors de l'utilisation des grandeurs pour tracer
257 // les erreurs éventuelles de déréférencement, car il faut ensuite faire un cast
258 // pour utiliser la grandeur
259
260
261 // retourne un pointeur de la grandeur: qui est tjs de type : TypeQuelconque
262 // si jamais le nom de la grandeur n'existe pas, le retour pointe sur NULL
263 const void * GrandeurGlobal(Enum_GrandeurGlobale enu ) const
264 {if (listegrandeurs.find(enu) == listegrandeurs.end())
265     return NULL; else return listegrandeurs.at(enu);
266 };
267 // idem mais pour une grandeur typée par une chaîne de caractère,
268 const void * GrandeurGlobal(const string& nom_ref ) const
269 {if (!(nom_ref[0]=='V') && (nom_ref[1]=='R') && (nom_ref[2] == '_'))
270     // cas normal, générique
271     {std::map< string, const void * , std::less <string> >::const_iterator
272     il_ref =listegrandeurs_par_string.find(nom_ref);
273     if (il_ref == listegrandeurs_par_string.end())
274         {return NULL;}
275     else
276         { return il_ref->second;};
277     }
278     else
279         // là il s'agit d'une variable composante
280         {return this->GrandeurGlobal_relaie(nom_ref) };
281     };
282
283 // ajout d'une nouvelle grandeur qui restera ensuite disponible
284 void Ajout_grandeur_consultable(void* grandeur, Enum_GrandeurGlobale enu);
285 // idem mais pour une grandeur typée par une chaîne de caractère,
286 void Ajout_grandeur_consultable(void* grandeur, string nom_ref);
287
288 // récupération d'une grandeur quelconque pour modification
289 // après l'appel de cette méthode, la grandeur quelconque est réputée updaté
290 void * Mise_a_jour_grandeur_consultable(Enum_GrandeurGlobale enu);
291 // idem mais pour une grandeur typée par une chaîne de caractère,
292 void * Mise_a_jour_grandeur_consultable(const string& nom_ref);
293 // raccourci pour mettre à jour une grandeur consultable de type: Grandeur_scalaire_double
294 void Mise_a_jour_grandeur_consultable_Scalaire_double(Enum_GrandeurGlobale enu,const double&
valeur);
295 void Mise_a_jour_grandeur_consultable_Scalaire_double(const string& nom_ref,const double& valeur);
296
297 // modification interactive d'une constante utilisateur
298 void Modif_interactive_constante_utilisateur()
299 {UtilLecture::Modif_interactive_constante_utilisateur()};
300
301 // suppression d'une grandeur
302 void Suppression_grandeur_consultable(Enum_GrandeurGlobale enu);
303 // idem mais pour une grandeur typée par une chaîne de caractère,
304 void Suppression_grandeur_consultable(const string& nom_ref);
305
306 // affichage de la liste des noms de grandeurs actuelles accessibles globalement
307 void Affiche_GrandeurGlobal(ostream & sort) const;
308
309 // récupération de la liste des noms de grandeurs actuelles accessibles globalement
310 // la liste passée en paramètre est vidée puis remplie par les grandeurs actuelles
311 void Recup_list_GrandeurGlobal(list <string >& list_grandeurs_globals) const;
312
313 // cas particulier du temps courant
314 // mise à jour du temps : permet de mettre à jour la variable globale TEMPS_COURANT
315 // doit-être appelé à chaque modification du temps courant
316 void Mise_a_jour_TEMPS_COURANT() {*pt_temps_courant = (tempo->TempsCourant());};
317
318
319 // // --- gestion des typeQuelconque_enum_etendu
320 // // un drapeau qui permet de savoir
321 // // globalement, que la liste des typeQuelconque_enum_etendu a été modifié
322 // bool T_typeQuelconque_enum_etendu_modifier()
323 //     const {return tab_typeQuelconque_enum_etendu_modifier;};
324 // // modification du drapeau
325 // void Change_T_typeQuelconque_enum_etendu_modifier(bool new_statut)
326 //     {tab_typeQuelconque_enum_etendu_modifier=new_statut;};
327
328
329 //----- uniquement protégé -----
330 protected:
331 // Ramène une référence sur l'ensemble des paramètres de contrôles qui sont actuellement "actifs"
332 // n'est accessible que des fonctions friends, permet la modification de l'instance
333 ParaAlgoControle & ParaAlgoContActifs() {return *paraAlgoControle;};
334 // changement de paramètres pointés
335 void Change_ParaAlgoControle(ParaAlgoControle& pa)
336     {paraAlgoControle = &pa;
337     tempo->Mise_a_jour_bornes(pa.Variables_de_temps_specifiques_algo());
338     };

```

```

339
340 // un tableau de grandeurs globales quelconques pour être accessibles de partout
341 // ici il s'agit d'un conteneur intermédiaire qui est mis à jour par ParaAlgoControle
342 static std::vector <double> grandeurs_globales;
343 // une méthode permettant de mettre à jour les éléments du tableau
344 // uniquement accessible par la classe friend ParaAlgoControle
345 static std::vector <double>& Mise_a_jour_Grandeurs_globales() {return grandeurs_globales;};
346
347 // liste des grandeurs consultables de partout sous forme d'un arbre pour faciliter la recherche
348 std::map < Enum_GrandeurGlobale, const void * , std::less <Enum_GrandeurGlobale> > listegrandeurs;
349 // idem pour une liste des grandeurs consultables de partout typée par une chaîne de caractères
350 std::map < string, const void * , std::less <string> > listegrandeurs_par_string;
351 double * pt_temps_courant; // pointeur interne pour lier TEMPS_COURANT avec tempo
352
353 private :
354     static EnumLangue langueHZ; // langue utilisée pour les entrées sorties
355     static int dimensionPb; // dimension du probleme = 1 unidimensionnel par défaut
356     int chdimensionPb ; // blocage de la dimension
357     // nombre de composantes des tenseurs en absolu
358     static int nbComposantesTenseur;
359     static int nivImpression; // niveau d'impression
360     static string nbVersion; // numéro de version du logiciel
361     static string NbVersionsurfichier; // numéro de version lue sur l'entrée
362     // info pour l'affichage:
363     // nombre de chiffre significatifs utilisé pour l'affichage des double précision
364     // pour l'archivage, pour le graphique et pour l'écran
365     static int nb_diggit_double_calcul;
366     static int nb_diggit_double_graphique;
367     static int nb_diggit_double_ecran;
368     //----- info sur le temps -----
369     // c'est paraAlgoControle qui gère le temps, cependant paraglob sert de relais pour
370     // ramené les infos sur le temps
371     static VariablesTemps* tempo; // contient les différentes grandeurs liées au temps
372
373     int NombreMaxiDeMaillage; // taille initiale du tableau des maillages
374     EnumTypeCalcul typeDeCalcul_maitre; // identificateur du type de Calcul principal ou maître
375     static bool geometrie_axisymetrique; // indique si oui ou non, l'espace est axisymétrique
376     // éventuellement par la suite on aura plusieurs type de calcul à enchaîner
377     // d'où il faudra le type en cours et une liste de tous les types à pourvoir
378     list <EnumSousTypeCalcul> soustypeDeCalcul; // identificateurs secondaire optionnel
379     //du type de Calcul renseigne sur le fait de calculer l'erreur,
380     // une relocation, un raffinement, éventuel.
381     bool avec_typeDeCalcul; // def si le calcul a lieu effectivement
382     list <bool> avec_soustypeDeCalcul; // def si les sous-types existent seulement ou
383     // si il faut les prendre en compte effectivement.
384     // bool tab_typeQuelconque_enum_etendu_modifier; // un drapeau qui permet de savoir
385     // globalement, que la liste des typeQuelconque_enum_etendu a été modifié
386
387     // pointeur sur l'ensemble des paramètres de contrôles qui sont actuellement "actifs"
388     // la valeur pointée peut évoluer pendant le calcul (il peut y avoir plusieurs paquets de
paramètres
389     // de contrôle présent dans le programme)
390     ParaAlgoControle * paraAlgoControle;
391
392     // un tableau de grandeurs globales quelconques pour être accessibles de partout
393     static std::vector <double> grandeurs_globales;
394     // une méthode permettant de mettre à jour les éléments du tableau de grandeurs globales
395     // via uniquement la classe friend paraAlgoControle
396     void Mise_a_jour_Grandeurs_globales(std::vector <double>& grandeurs)
397     {grandeurs_globales=grandeurs;};
398
399     // --- variables qui sont stockées ici, mais sont utilisées par des classes externes
400     // en particulier: Projet et les Algorithmes
401     int etatDeLaLecturePointInfo; // =0 après la première lecture "globale" du .info
402     // =n après la nième lecture "secondaire" dans le .info
403     int demandeLectureSecondaireInPointInfo; // 0 pas de demande: valeur par défaut
404     // 1 demande d'une lecture supplémentaire
405
406     // ---définition d'une structure permettant de gérer les interruptions systèmes type: controle-c
407     struct sigaction prepaSignal;
408
409     // récup d'une composante d'une grandeur quelconque d'une grandeur globale
410     const void * GrandeurGlobal_relaie(const string& nom_ref ) const;
411
412 };
413 /// @} // end of group
414
415
416 #endif

```

## 7.365 Type\_Calcul.h

```

1 /*****
2 *   UNIVERSITE DE BRETAGNE SUD (UBS)   --- I.U.P/I.U.T. DE LORIENT   *
3 *****/

```

```

4 *           LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)           *
5 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex           *
6 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr   *
7 *****
8 *   DATE:           23/01/97                                           *
9 *                                                         $           *
10 *   AUTEUR:        G RIO (mailto:gerard.rio@univ-ubs.fr)               *
11 *                 Tel 0297874571 fax : 02.97.87.45.72                 *
12 *                                                         $           *
13 *   PROJET:        Herezh++                                           *
14 *                                                         $           *
15 *****
16 *   BUT:           Définir tous les paramètres qui gèrent les types de *
17 *                 de calcul.                                           *
18 *                                                         $           *
19 *   *****
20 *   VERIFICATION:
21 *
22 *   ! date ! auteur ! but ! ! !
23 *   -----
24 *   ! ! ! ! !
25 *   $
26 *   *****
27 *   MODIFICATIONS:
28 *   ! date ! auteur ! but ! ! !
29 *   -----
30 *   $
31 *****/
32 #ifndef TYPE_CALCUL_H
33 #define TYPE_CALCUL_H
34
35 #include <iostream>
36 #include "UtilLecture.h"
37
38 class Type_Calcul
39 { // surcharge de l'operator de lecture
40   friend istream & operator > (istream & entree, Type_Calcul& a);
41   // surcharge de l'operator d'écriture
42   friend ostream & operator < (ostream & sort, const Type_Calcul& a);
43
44 public :
45   // CONSTRUCTEURS :
46   Type_Calcul();
47
48   // DESTRUCTEUR :
49   ~Type_Calcul();
50   // METHODES PUBLIQUES :
51
52   // Retourne le nom d'un type de calcul a partir de son identificateur de
53   // type enumere id_TypeCalcul correspondant
54   //char* Nom_TypeCalcul (const EnumTypeCalcul id_TypeCalcul);
55   // Retourne le nom du sous type de calcul a partir de son identificateur de
56   // type enumere id_SousTypeCalcul correspondant, si il existe sinon retourne : aucun_soustypedecalcul
57   //char* Nom_SousTypeCalcul (const EnumSousTypeCalcul id_SousTypeCalcul);
58
59   // Retourne l'identificateur de type enumere associe au nom du type de calcul
60   //EnumTypeCalcul Id_nom_TypeCalcul (const char* nom_TypeCalcul);
61   // Retourne l'identificateur de type enumere associe au nom du sous type de calcul
62   //EnumSousTypeCalcul Id_nom_SousTypeCalcul (const char* nom_SousTypeCalcul);
63
64   //lecture du type de calcul principal et des sous-types eventuel
65   void LectureTypeCalcul(UtilLecture& lec);
66   // indique s'il y a un sous-type ou pas -> retour vrai s'il y a un sous type
67   bool SousType();
68   // indique s'il y a remonte ou pas dans le sous-type
69   bool Remonte_in();
70   // indique s'il y a "avec" ou pas dans le sous-type
71   bool Avec_in(const EnumSousTypeCalcul sousTypeCalcul);
72   // indique s'il y a un sous-type de type "erreur"
73   bool Erreur_in();
74
75 private :
76   // VARIABLES PROTEGEES :
77   string typePrincipal; // type principal de calcul
78   list <string> sousType; // liste des sous type de calcul
79
80   // METHODES PROTEGEES :
81
82 };
83
84
85 #endif

```

## 7.366 TypeCalcul.h

```
1 /*****
2  *      UNIVERSITE DE BRETAGNE SUD (UBS)   --- I.U.P/I.U.T. DE LORIENT   *
3  *****/
4 *      LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)           *
5 *      Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex   *
6 *      tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
7 *****/
8 *      DATE:           23/01/97                                       *
9 *                                                              $     *
10 *      AUTEUR:         G RIO (mailto:gerard.rio@univ-ubs.fr)          *
11 *                   Tel 0297874571   fax : 02.97.87.45.72           *
12 *                                                              $     *
13 *      PROJET:         Herezh++                                        *
14 *                                                              $     *
15 *****/
16 *      BUT:           Définir tous les paramètres qui gèrent les types de
17 *                   de calcul.                                       *
18 *                                                              $     *
19 *      *****/
20 *      VERIFICATION:
21 *
22 *      ! date !      auteur !              but              !         *
23 *      -----
24 *      !           !              !              !              !         *
25 *      -----
26 *      *****/
27 *      MODIFICATIONS:
28 *      ! date !      auteur !              but              !         *
29 *      -----
30 *      -----
31 *****/
32 #ifndef TYPE_CALCUL_H
33 #define TYPE_CALCUL_H
34
35 #include <iostream>
36 #include "UtilLecture.h"
37
38 class Type_Calcul
39 {
40 public :
41     // CONSTRUCTEURS :
42
43     // DESTRUCTEUR :
44
45     // METHODES PUBLIQUES :
46
47 private :
48     // VARIABLES PROTEGEES :
49
50     // CONSTRUCTEURS :
51
52     // DESTRUCTEUR :
53
54     // METHODES PROTEGEES :
55
56 };
57
58 #endif
```

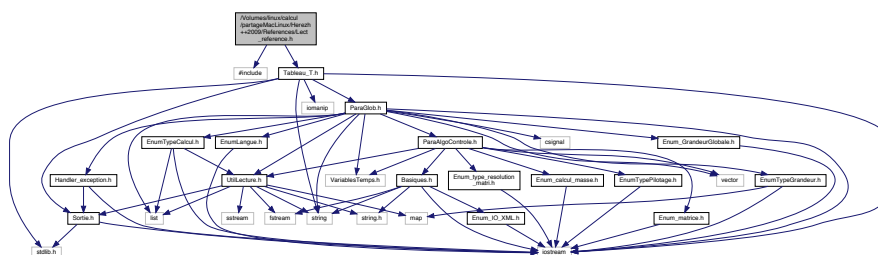
## 7.367 Référence du fichier /Volumes/linux/calcul/partageMacLinux/← Herezh++2009/References/Lect\_reference.h

Lecture des references definies dans le fichier au format ".lis" de nom : nom\_fichier.

```
#include "#include"
#include "Tableau_T.h"
```



Graphe des dépendances par inclusion de Lect\_reference.h:



## Fonctions

- `Tableau< Reference > Lect_reference (char *nom_fichier)`  
Lecture des references definies dans le fichier au format ".lis" de nom : nom\_fichier.

### 7.367.1 Description détaillée

Lecture des references definies dans le fichier au format ".lis" de nom : nom\_fichier.

## 7.368 Lect\_reference.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Lect_reference.h
2    \brief Lecture des references definies dans le fichier au format ".lis" de nom : nom_fichier
3 */
4 // FICHER : Lect_reference.h
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35
36 #ifndef LECT_REFERENCE_H
37 #define LECT_REFERENCE_H
38
39
40 #include "Liste_T.cc"
41 #include "Reference.h"
42 #include "Tableau_T.h"
43
44
45 /// @addtogroup Les_classes_Reference
46 /// @{
47 ///
48

```

```

49 /// Lecture des references definies dans le fichier au format ".lis" de nom : nom_fichier
50 Tableau<Reference> Lect_reference (char* nom_fichier);
51 /// @} // end of group
52
53
54 #endif

```

## 7.369 LesReferences.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           23/01/97
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *
38 *   BUT:            Gestion des listes de references.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date ! auteur ! but
44 *   -----
45 *   ! ! ! ! !
46 *   $
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date ! auteur ! but
50 *   -----
51 *   $
52 *   *****/
53 #ifndef LESREFERENCES_H
54 #define LESREFERENCES_H
55
56 #include "Reference.h"
57 #include <list>
58 #include <map>
59 #include "UtilLecture.h"
60 #include "MotCle.h"
61
62 /// @addtogroup Les_classes_Reference
63 /// @{
64 ///
65
66 /// Gestion des listes de references.
67 ///
68 /// \author Gérard Rio
69 /// \version 1.0
70 /// \date 23/01/97
71
72 //----- une méthode permettant d'utiliser des tableaux de map de références -----
73 // surcharge de l'operator d'écriture
74 ostream & operator << (ostream & sort, const map < string, Reference*, std::less <string> > & );
75

```

```

76 //----- fin de la méthode permettant d'utiliser des tableaux de map de références -----
77 /// @} // end of group
78
79 /// @addtogroup Les_classes_Reference
80 /// @{
81 ///
82
83 /// Gestion des listes de references.
84 ///
85 /// \author Gérard Rio
86 /// \version 1.0
87 /// \date 23/01/97
88
89 class LesReferences
90 {
91 public :
92 // CONSTRUCTEURS :
93 LesReferences (); // par défaut
94 // DESTRUCTEUR :
95 ~LesReferences ();
96 // METHODES PUBLIQUES :
97
98 // def du numero de maillage courant, utilise avant la lecture
99 void NbMaille(int nb);
100 // def du type de reference a lire, utilise avant la lecture
101 // type peut-être : noeud,element,surface,arete
102 void Indic(string type);
103
104 // lecture des references
105 void Lecture(UtilLecture & entreePrinc);
106
107 // ajout d'une référence déjà construite par ailleurs
108 void Ajout_reference(Reference * refi);
109
110 // suppression d'une référence
111 void SupprimeReference(const string & stl,int num_mail);
112 // suppression de tous les référence d'un maillage donné
113 // si avec_diminution_num_maillage = true:
114 // tous les numéros de maillage, > num_mail, associé aux maillages qui restent
115 // sont diminués de un, pour tenir compte de la disparition du maillage et donc
116 // d'une nouvelle numérotation des maillages
117 // si avec_diminution_num_maillage = false, le conteneur des ref de maillage num_mail
118 // est vidé, mais il continu à exister: on considère qu'il y a toujours un maillage num_mail
119 // potentiel
120 void Supprime_tour_lesRef_un_maillage(int num_mail, bool avec_diminution_num_maillage);
121
122 // affichage des informations contenu dans les references
123 // par défaut affiche toutes les infos
124 // si niveau = 1 : affiche que les noms des ref
125 void Affiche(int niveau=0) const ;
126
127 // affichage des informations contenu dans les references d'un certain type
128 // défini par indic :
129 // indic = 1 -> noeud, =2 -> element
130 // =3 -> surface associée à un élément , =4 -> arete associée à un élément
131 // =5 -> noeud associée à un élément
132 // =6 -> point d'intégration associée à un élément
133 // =7 -> de points d'intégrations relatifs à des surfaces d'éléments
134 // =8 -> de points d'intégrations relatifs à des arete d'éléments
135 // =0 -> rien_actuellement
136 // par défaut affiche toutes les infos
137 // si niveau = 1 : affiche que les noms des ref
138 void Affiche(int indic, int niveau) const ;
139
140 // Affiche les donnees des références pour le maillage imail dans un fichier
141 // dont le nom est construit à partir du nom du maillage au format ".lis"
142 void Affiche_dans_lis(const string& nom_maillage,int imail);
143
144 // affichage et definition interactive des commandes
145 // nbMaxiNoeud: nombre maxi de noeud pour les exemples
146 // nbMaxiElem : nombre maxi d'éléments pour les exemples
147 // cas : =1 premier passage, il s'agit de références de noeuds uniquement
148 // cas : =2 second passage, il s'agit de l'ensemble des possibilités de références
149 void Info_commande_lesRef(int nbMaxiNoeud,int nbMaxiElem,UtilLecture * entreePrinc,int cas);
150
151 //--- utilisation d'une boucle pour balayer l'ensemble des références existantes:
152 // ***** très important: ce balayage ne fonctionne que si on ne modifie pas le stockage
153 // ***** par exemple en supprimant une référence existante ou en ajoutant une nouvelle référence
154 // ***** ou en lisant des références (donc ajout !!)
155 // ***** par contre on peut changer le contenu des références existantes
156 // initialise le questionnement de la récupération de référence et
157 // retourne la première référence si elle existe sinon un pointeur nul
158 const Reference* Init_et_Premiere();
159 // retourne la référence suivante ou, s'il n'y en n'a plus, retourne
160 // un pointeur nul
161 const Reference* Reference_suivante();
162 //--- fin méthode pour faire une boucle pour balayer l'ensemble des références existantes:

```

```

162
163 // test si la reference existe reellement
164 // retourne false si n'existe pas , true sinon
165 bool Existe(const string & stl,int num_mail) const ;
166 bool Existe(const string & stl,const string* nom_mail) const;
167 // retourne la reference correspondant a une cle
168 const Reference& Trouve(const string & stl,int num_mail) const ;
169 const Reference& Trouve(const string & stl,const string* nom_mail) const ;
170 // mise à jour de la map qui fait la liaison nom de maillage <=> numéro de maillage
171 void MiseAJourMap( const map < string, int , std::less <string> >& listNomMail)
172     { listeNomMail = & listNomMail;};
173
174 // mise à jour des références de noeud, dans le cas où les numéros de noeuds ont changés
175 //1) cas où l'on supprime éventuellement des noeuds de la référence, qui ne sont plus référencé
176 // nv_tab est tel que : nv_tab(i) est le nouveau numéro qui avait auparavant le numéro "i"
177 // non_referencer(i) : = true signifie qu'il ne faut plus tenir compte de ce noeud
178 // = false indique qu'il continue d'être actif
179 void Mise_a_jour_ref_noeud(Tableau <int >& nv_tab,int num_mail,Tableau <bool>& non_referencer);
180 //2) cas où on considère tous les noeuds
181 // nv_tab est tel que : nv_tab(i) est le nouveau numéro qui avait auparavant le numéro "i"
182 void Mise_a_jour_ref_noeud(Tableau <int >& nv_tab,int num_mail);
183
184 // mise à jour des références d'élément, dans le cas où les numéros d'élément ont changés
185 //1) cas où l'on supprime éventuellement des éléments de la référence, qui ne sont plus référencé
186 // nv_tab est tel que : nv_tab(i) est le nouveau numéro qui avait auparavant le numéro "i"
187 // non_referencer(i) : = true signifie qu'il ne faut plus tenir compte de cet élément
188 // = false indique qu'il continue d'être actif
189 void Mise_a_jour_ref_element(Tableau <int >& nv_tab,int num_mail,Tableau <bool>& non_referencer);
190
191 // mise à jour du numéro de maillage d'une référence
192 void Mise_a_jour_num_maillage_ref(const string nom_ref, int old_num_maill, int new_num_maill)
193     { Trouve_interne(nom_ref,old_num_maill).Change_Nbmaille(new_num_maill);};
194
195 //----- lecture écriture dans base info -----
196 // cas donne le niveau de la récupération
197 // = 1 : on récupère tout
198 // = 2 : on récupère uniquement les données variables (supposées comme telles)
199 void Lecture_base_info(ifstream& ent,const int cas);
200 // cas donne le niveau de sauvegarde
201 // = 1 : on sauvegarde tout
202 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
203 void Ecriture_base_info(ofstream& sort,const int cas);
204
205 protected :
206 // variable transitoires utilise pendant la lecture
207 int nbMaille; // nb courant de maillage
208 int indic ; // type courant de reference : = 1 -> noeud, =2 -> element
209 // =3 -> surface associée à un élément , =4 -> arete associée à un élément
210 // =5 -> noeud associée à un élément
211 // =6 -> point d'intégration associée à un élément
212 // =7 -> de points d'intégrations relatifs à des surfaces d'éléments
213 // =8 -> de points d'intégrations relatifs à des arete d'éléments
214 // =0 -> rien_actuellement
215 Reference * ref; // une variable de travail
216
217 // VARIABLES PROTEGEES :
218 // list <Reference*> listeDeRef; // liste des references
219
220 // on utilise une map pour récupérer plus rapidement une référence à partir d'un nom
221 // t_mapDeRef (i) est la map relative au maillage i
222 Tableau < map < string, Reference*, std::less <string> > > t_mapDeRef;
223
224
225
226 // liste des noms de maillages associée à un numéro sous forme d'un arbre pour faciliter la
recherche
227 const map < string, int , std::less <string> >* listeNomMail;
228 // a utiliser avec Existe et Trouve, mais pas avant la lecture, seulement après
229
230 static MotCle motCle; // liste des mots clés
231
232 //list <Reference*>::const_iterator iref;
233 // deux variables intermédiaires pour les méthodes Init_et_Premiere et Reference_suivante
234 map < string, Reference*, std::less <string> >::const_iterator iref;
235 int num_mail_presuivant;
236
237
238 // METHODES PROTEGEES :
239 // lecture d'une reference
240 bool LectureReference(UtilLecture & entreePrinc);
241 // retourne la reference correspondant a une cle, en non constant
242 Reference& Trouve_interne(const string & stl,int num_mail) const ;
243 Reference& Trouve_interne(const string & stl,const string* nom_mail) const ;
244
245
246 };
247 /// @} // end of group

```

```
248
249 #endif
```

## 7.370 Reference.h

```
1 // FICHER : Reference.h
2 // CLASSE : Reference
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:           06/02/00
34 *
35 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:         Herezh++
38 *
39 *
40 *   BUT:  classe général virtuel des références. Les classes dérivées*
41 *   permettent de définir précisément les différentes références : *
42 *   de noeuds, d'élément, d'arêtes, de faces etc..
43 *   Une instance de la classe s'identifie a partir d'un nom
44 *
45 *   *****
46 *
47 *   VERIFICATION:
48 *   ! date ! auteur ! but
49 *   -----
50 *   ! ! !
51 *
52 *   *****
53 *
54 *   MODIFICATIONS:
55 *   ! date ! auteur ! but
56 *   -----
57 *
58 *   *****/
59 /** @defgroup Les_classes_Reference
60 *
61 *   BUT:  Définir les différentes références
62 *   de noeuds, d'élément, d'arêtes, de faces, de pti etc..
63 *
64 *
65 * \author Gérard Rio
66 * \version 1.0
67 * \date 06/02/00
68 * \brief Définir les différentes références de noeuds, d'élément, d'arêtes, de faces, de pti etc..
69 *
70 */
71
72 #ifndef REFERENCE_H
73 #define REFERENCE_H
74
75
76 #include <iostream>
77 #include <string>
78 // #include "Debug.h"
79 #include "Tableau_T.h"
```

```

80 #include "UtilLecture.h"
81 #include <list>
82 #include "MotCle.h"
83
84 /// @addtogroup Les_classes_Reference
85 /// @{
86 ///
87
88 /// class Reference: classe général virtuel des références. Les classes dérivées
89 /// permettent de définir précisément les différentes références :
90 /// de noeuds, d'élément, d'arêtes, de faces etc..
91 /// Une instance de la classe s'identifie à partir d'un nom
92 ///
93 /// \author Gérard Rio
94 /// \version 1.0
95 /// \date 06/02/00
96
97 class Reference
98 {
99     public :
100
101         // CONSTRUCTEURS :
102
103         // Constructeur par défaut
104         Reference ( string nom = "rien_actuellement");
105
106         // indic = 1 : il s'agit de reference de noeud
107         // indic = 2 : il s'agit de reference d'element
108         // =3 -> surface relatives à des éléments,
109         // =4 -> arrete relatives à des éléments,
110         // =5 -> des noeuds relatifs à des éléments
111         // =6 -> de points d'intégrations relatifs à des éléments
112         // =7 -> de points d'intégrations relatifs à des surfaces d'éléments
113         // =8 -> de points d'intégrations relatifs à des arrete d'éléments
114
115         // Constructeur fonction du nb de maillage et du type de ref
116         Reference (int nbmaille , int indic);
117
118         // Constructeur fonction d'un nom de reference
119         // du nb de maillage, du type de ref
120         Reference (string nom,int nbmaille , int indic);
121
122         // Constructeur de copie
123         Reference (const Reference& ref);
124
125
126         // DESTRUCTEUR :
127
128         virtual ~Reference ();
129
130
131         // METHODES :
132
133         // création d'une référence du même type contenant les mêmes info
134         // que la référence donnée en argument: utilisation du constructeur de copie
135         virtual Reference* Nevez_Ref_copie() const = 0;
136
137         // Surcharge de l'operateur = : realise l'egalite entre deux references
138         virtual Reference& operator= (const Reference& ref);
139
140         // Retourne le nom associe a la reference
141         string Nom () const;
142
143         // Retourne le type de reference
144         // = 1 : il s'agit de reference de noeud
145         // = 2 : il s'agit de reference d'element
146         // =3 -> surface relatives à des éléments,
147         // =4 -> arrete relatives à des éléments,
148         // =5 -> des noeuds relatifs à des éléments
149         // =6 -> de points d'intégrations relatifs à des éléments
150         // =7 -> de points d'intégrations relatifs à des surfaces d'éléments
151         // =8 -> de points d'intégrations relatifs à des arrete d'éléments
152         int Indic () const;
153
154         // Retourne le numero du maillage auquel la reference se rattache
155         int Nbmaille() const;
156         // change le numéro du maillage associé
157         void Change_Nbmaille(int nv_nm) {nbmaille= nv_nm;};
158
159         // Remplace l'ancien nom de la reference par nouveau_nom
160         void Change_nom (string nouveau_nom);
161
162         // supprime les doublons internes éventuels dans la référence
163         virtual void Supprime_doublons_internes() = 0;
164
165         // Affiche les donnees liees a la reference
166         virtual void Affiche () const;

```

```

167
168 // Affiche les donnees des références dans le flux passé en paramètre
169 virtual void Affiche_dans_lis(ofstream& sort) const =0 ;
170
171 // lecture d'une liste de reference
172 virtual bool LectureReference(UtilLecture & entreePrinc) =0 ;
173
174 // affichage et definition interactive des commandes
175 // nbMaxi: nombre maxi de noeud ou d'éléments pour les exemples
176 // cas : =1 premier passage, il s'agit de références de noeuds uniquement
177 // cas : =2 second passage, il s'agit de l'ensemble des possibilités de références
178 virtual void Info_commande_Ref(int nbMaxi,UtilLecture * entreePrinc,int cas) =0;
179
180 //----- lecture écriture dans base info -----
181 // cas donne le niveau de la récupération
182 // = 1 : on récupère tout
183 // = 2 : on récupère uniquement les données variables (supposées comme telles)
184 virtual void Lecture_base_info(ifstream& ent,const int cas) =0;
185 // cas donne le niveau de sauvegarde
186 // = 1 : on sauvegarde tout
187 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
188 virtual void Ecriture_base_info(ofstream& sort,const int cas) =0;
189
190 protected :
191
192     string nom_ref; // nom de la reference
193     int nbmaille; // numero du maillage auquel la reference se rapporte
194     int indic; // indic = 1 : il s'agit de reference de noeud
195                 // indic = 2 : il s'agit de reference d'element
196                 // =3 -> surface relatives à des éléments,
197                 // =4 -> arrete relatives à des éléments,
198                 // =5 -> des noeuds relatifs à des éléments
199                 // =6 -> de points d'intégrations relatifs à des éléments
200                 // =7 -> de points d'intégrations relatifs à des surfaces d'éléments
201                 // =8 -> de points d'intégrations relatifs à des arrete d'éléments
202     static MotCle motCle; // liste des mots clés
203
204     // methodes appellées par les classes dérivées
205     // cas donne le niveau de la récupération
206     // = 1 : on récupère tout
207     // = 2 : on récupère uniquement les données variables (supposées comme telles)
208     void Lect_int_base_info(ifstream& ent);
209     // cas donne le niveau de sauvegarde
210     // = 1 : on sauvegarde tout
211     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
212     void Ecrit_int_base_info(ofstream& sort);
213
214 };
215 /// @} // end of group
216
217 #ifndef MISE_AU_POINT
218 #include "Reference.cc"
219 #define REFERENCE_H_deja_inclus
220 #endif
221
222 #endif
223
224

```

## 7.371 ReferenceAF.h

```

1 // FICHER : ReferenceAF.h
2 // CLASSE : ReferenceAF
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

```

```

25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *   DATE:      23/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *****/
41 *   BUT:  Def, stockage et manipulation des références pour les
42 *         faces et les arêtes .
43 *
44 *   *****
45 *
46 *   VERIFICATION:
47 *   ! date !   auteur !           but
48 *   -----
49 *   !           !           !
50 *
51 *   *****
52 *   MODIFICATIONS:
53 *   ! date !   auteur !           but
54 *   -----
55 *
56 *****/
57
58
59 #ifndef REFERENCEAF_H
60 #define REFERENCEAF_H
61
62 #include "Reference.h"
63
64
65 /// @addtogroup Les_classes_Reference
66 /// @{
67 ///
68
69 /// Def, stockage et manipulation des références pour les faces et les arêtes
70 ///
71 /// \author   Gérard Rio
72 /// \version  1.0
73 /// \date    23/01/97
74
75 class ReferenceAF : public Reference
76 {
77     public :
78
79         // CONSTRUCTEURS :
80
81         // Constructeur par défaut
82         ReferenceAF ( string nom = "rien_actuellement");
83
84         // indic = 1 : il s'agit de reference de noeud
85         // indic = 2 : il s'agit de reference d'element
86         // =3 -> surface relatives à des éléments,
87         // =4 -> arête relatives à des éléments,
88         // =5 -> des noeuds relatifs à des éléments
89         // =6 -> de points d'intégrations relatifs à des éléments
90         // =7 -> de points d'intégrations relatifs à des surfaces d'éléments
91         // =8 -> de points d'intégrations relatifs à des arrete d'éléments
92
93         // Constructeur fonction du nb de maillage et du type de ref
94         ReferenceAF (int nbmaille , int indic);
95
96         // Constructeur fonction de deux tableaux de numeros, le premier pour les numeros
97         // d'éléments le second pour les numeros de faces, ou d'aretes, ou de noeud d'element,
98         // ou de pt d'integ, du nb de maillage, du type de ref
99         ReferenceAF (const Tableau<int>& tabelle,const Tableau<int>& tab
100                    ,int nbmaille , int indic, string nom = "rien_actuellement");
101         // idem mais avec des listes d'entiers plutôt que des tableaux
102         ReferenceAF (const list <int>& list_elem,const list <int>& list_num
103                    ,int nbmaille , int indic, string nom = "rien_actuellement");
104
105         // Constructeur fonction d'un nom de reference
106         // du nb de maillage, du type de ref
107         ReferenceAF (string nom,int nbmaille , int indic);
108
109         // Constructeur de copie
110         ReferenceAF (const ReferenceAF& ref);

```



```

111
112
113     // DESTRUCTEUR :
114
115     ~ReferenceAF ();
116
117
118     // METHODES :
119
120     // création d'une référence du même type contenant les mêmes info
121     // que la référence donnée en argument: utilisation du constructeur de copie
122     Reference* Nevez_Ref_copie() const
123     {ReferenceAF* ref_ret=new ReferenceAF(*this); return ref_ret;} ;
124
125     Reference& operator= (const Reference& ref);
126
127     // Retourne le tableau des numeros d'élément
128     const Tableau<int>& Tab_Elem() const
129     { return tab_Elem; };
130
131     // Retourne le tableau des numeros de faces, ou d'arêtes, ou de noeud d'element, ou de points
132     // d'integ
133     const Tableau<int>& Tab_FA() const
134     { return tab_FA; };
135
136     // Retourne le ieme element du tableau d'éléments
137     // de la reference (acces lecture uniquement)
138     int NumeroElem(int i) const
139     { return tab_Elem(i); };
140
141     // Retourne le ieme element du tableau de faces, ou d'arêtes, ou de noeud d'element, ou de
142     // points d'integ
143     int NumeroFA(int i) const
144     { return tab_FA(i); };
145
146     // Retourne le nombre de numeros de la reference
147     int Taille () const
148     { return tab_Elem.Taille(); };
149
150     // change un numéro i de référence
151     // nbe: le numéro d'élément, nbAF: le numéro de de faces, ou d'arêtes, ou de noeud d'element,
152     // ou de points d'integ
153     void Change_num_AF_dans_ref(int i,int nbe, int nbAF);
154
155     // change les 2 tableaux de la ref
156     void Change_tab_num(const Tableau<int>& tabelle,const Tableau<int>& tab);
157
158     // indique si le numéro d'élément et le numéro de faces, ou d'arêtes, ou de noeud d'element, ou
159     // de points
160     // d'integ, passé en argument, fait partie de la référence
161     // nbe: le numéro d'élément, nbAF: le numéro de de faces, ou d'arêtes, ou de noeud d'element,
162     // ou de points d'integ
163     bool Contient_AF(int nbe, int nbAF) const;
164
165     // supprime les doublons internes éventuels dans la référence
166     virtual void Supprime_doublons_internes();
167
168     // Affiche les donnees liees a la reference
169     void Affiche () const;
170
171     //----- méthodes découlant de virtuelles -----
172
173     // Affiche les donnees des références dans le flux passé en paramètre
174     void Affiche_dans_lis(ofstream& sort) const ;
175
176     // lecture d'une liste de référence
177     bool LectureReference(UtilLecture & entreePrinc);
178
179     // affichage et definition interactive des commandes
180     // nbMaxi: nombre maxi d'éléments pour les exemples
181     // cas : =1 premier passage, il s'agit de références de noeuds uniquement,
182     // donc ici cela génère une erreur
183     // cas : =2 second passage, il s'agit de l'ensemble des possibilités de références
184     void Info_commande_Ref(int nbMaxi,UtilLecture * entreePrinc,int cas);
185
186     //----- lecture écriture dans base info -----
187     // cas donne le niveau de la récupération
188     // = 1 : on récupère tout
189     // = 2 : on récupère uniquement les données variables (supposées comme telles)
190     void Lecture_base_info(ifstream& ent,const int cas);
191     // cas donne le niveau de sauvegarde
192     // = 1 : on sauvegarde tout
193     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
194     void Ecriture_base_info(ofstream& sort,const int cas);

```

```

195     protected :
196
197         Tableau<int> tab_Elem; // tableau des numeros des éléments de la reference
198         Tableau<int> tab_FA; // tableau des numeros des faces ou arêtes de la reference
199                               // ou noeud d'element, ou de points d'intégrations d'élément
200
201     };
202     /// @} // end of group
203
204     #ifndef MISE_AU_POINT
205     #include "ReferenceAF.cc"
206     #define REFERENCEAF_H_deja_inclus
207     #endif
208
209 #endif
210
211

```

## 7.372 ReferenceFA.h

```

1 // FICHER : ReferenceNE.h
2 // CLASSE : ReferenceNE
3
4 /*****
5 *   UNIVERSITE DE BRETAGNE SUD (UBS) --- I.U.P/I.U.T. DE LORIENT *
6 *****/
7 *   LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M) *
8 *   Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
9 *   tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
10 *****/
11 *   DATE:           23/01/97 *
12 * * * * * $ *
13 *   AUTEUR:         G RIO (mailto:gerard.rio@univ-ubs.fr) *
14 *                   Tel 0297874571 fax : 02.97.87.45.72 *
15 * * * * * $ *
16 *   PROJET:         Herezh++ *
17 * * * * * $ *
18 *****/
19 *   BUT: Def, stockage et manipulation des références pour les *
20 *         noeuds et les éléments. *
21 * * * * * $ *
22 *   ***** *
23 *   VERIFICATION: *
24 * * * * * *
25 *   ! date ! auteur ! but ! *
26 *   ----- *
27 *   ! ! ! ! *
28 * * * * * $ *
29 *   ***** *
30 *   MODIFICATIONS: *
31 *   ! date ! auteur ! but ! *
32 *   ----- *
33 * * * * * $ *
34 *****/
35
36
37 #ifndef REFERENCEAF_H
38 #define REFERENCEAF_H
39
40
41
42
43 class ReferenceNE : public Reference
44 {
45     public :
46
47         // CONSTRUCTEURS :
48
49         // Constructeur par défaut
50         ReferenceNE ( string nom = "rien_actuellement");
51
52         // Constructeur fonction d'un tableau des numeros de la reference
53         // du nb de maillage, du type de ref
54         ReferenceNE (const Tableau<int>& tab,int nbmaille , int indic, string nom = "rien_actuellement");
55
56         // Constructeur fonction d'un nom de reference
57         // du nb de maillage, du type de ref
58         ReferenceNE (string nom,int nbmaille , int indic);
59
60         // Constructeur de copie
61         ReferenceNE (const ReferenceNE& ref);
62
63
64         // DESTRUCTEUR :

```

```

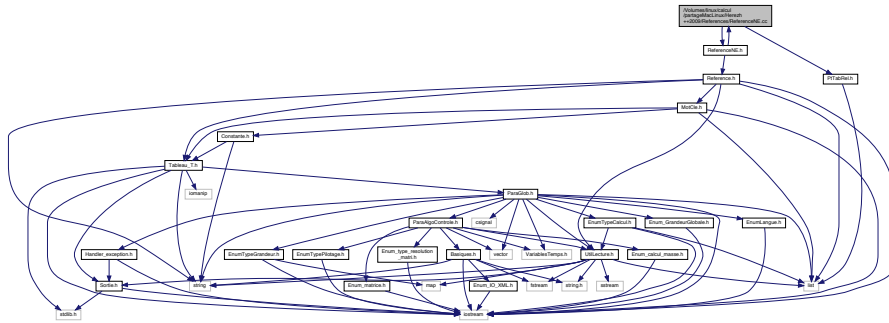
65
66     ~ReferenceNE ();
67
68
69     // METHODES :
70
71     inline Reference& operator= (const Reference& ref)
72     // Surchage de l'operateur = : realise l'egalite entre deux references
73     {
74         indic = ref.indic;
75         nbmaille = ref.nbmaille;
76         nom_ref = ref.nom_ref;
77         tab_num = ref.tab_num;
78         return (*this);
79     };
80
81     inline Tableau<int>& Tab_num ()
82     // Retourne le tableau des numeros de la reference
83     {
84         return tab_num; };
85
86     inline string Nom () const
87     // Retourne le nom associe a la reference
88     {
89         return nom_ref; };
90
91     inline int Indic () const
92     // Retourne le type de reference
93     {
94         return indic; };
95
96     inline int Nbmaille() const
97     // Retourne le numero du maillage auquel la reference se rattache
98     {
99         return nbmaille; };
100
101     inline int& Numero(int i)
102     // Retourne le ieme element du tableau des numeros
103     // de la reference (acces lecture et ecriture)
104     {
105         return tab_num(i); };
106
107     inline int Numero(int i) const
108     // Retourne le ieme element du tableau des numeros
109     // de la reference (acces lecture uniquement)
110     {
111         return tab_num(i); };
112
113     inline int Taille () const
114     // Retourne le nombre de numeros de la reference
115     {
116         return tab_num.Taille(); };
117
118     inline void Change_nom (string nouveau_nom)
119     // Remplace l'ancien nom de la reference par nouveau_nom
120     {
121         nom_ref = nouveau_nom; };
122
123     inline void Affiche () const
124     // Affiche les donnees liees a la reference
125     {
126         cout << "\nNom de la reference : " << nom_ref
127             << ", maillage nb = " << nbmaille << ", de type = " << indic << '\n';
128         cout << "Taille du tableau : " << tab_num.Taille() << "\n";
129         cout << "Composante(s) :\n\t[ ";
130         for (int i=1;i<=tab_num.Taille();i++)
131             cout << tab_num(i) << " ";
132         cout << "]\n\n";
133     };
134
135 protected :
136
137     string nom_ref; // nom de la reference
138     int nbmaille; // numero du maillage auquel la reference se rapporte
139     int indic; // indic = 1 : il s'agit de reference de noeud
140                // indic = 2 : il s'agit de reference d'element
141                // =3 -> surface, =4 -> arrete
142     Tableau<int> tab_num; // tableau des numeros de la reference
143 };
144 #endif
145

```

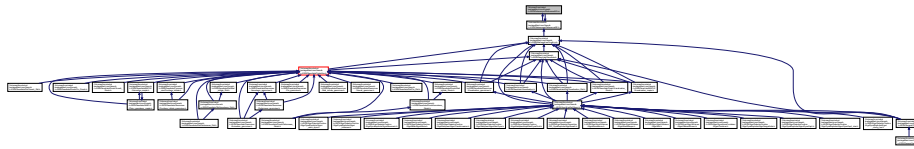
## 7.373 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/References/ReferenceNE.cc

def d'une variable static motCle

```
#include "ReferenceNE.h"
#include "PtTabRel.h"
Graphe des dépendances par inclusion de ReferenceNE.cc:
```



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### 7.373.1 Description détaillée

def d'une variable static motCle

### 7.374 ReferenceNE.h

```
1 // FICHER : ReferenceNE.h
2 // CLASSE : ReferenceNE
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *   DATE:      23/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *****/
```

```

39 *                                                                 $ *
40 *****
41 * BUT: Def, stockage et manipulation des références pour les *
42 * noeuds et les éléments. *
43 *                                                                 $ *
44 * //***** *
45 * VERIFICATION: *
46 * *
47 * ! date ! auteur ! but ! *
48 * ----- *
49 * ! ! ! ! *
50 *                                                                 $ *
51 * //***** *
52 * MODIFICATIONS: *
53 * ! date ! auteur ! but ! *
54 * ----- *
55 *                                                                 $ *
56 *****/
57
58
59 #ifndef REPERENCENE_H
60 #define REPERENCENE_H
61
62 #include "Reference.h"
63
64
65 /// @addtogroup Les_classes_Reference
66 /// @{
67 ///
68
69 /// Def, stockage et manipulation des références pour les noeuds et les éléments
70 ///
71 /// \author Gérard Rio
72 /// \version 1.0
73 /// \date 06/02/00
74
75 class ReferenceNE : public Reference
76 {
77     public :
78
79         // CONSTRUCTEURS :
80
81         // Constructeur par défaut
82         ReferenceNE( string nom = "rien_actuellement");
83
84         // indic = 1 : il s'agit de reference de noeud
85         // indic = 2 : il s'agit de reference d'element
86         // =3 -> surface relatives à des éléments,
87         // =4 -> arete relatives à des éléments,
88         // =5 -> des noeuds relatifs à des éléments
89         // =6 -> de points d'intégrations relatifs à des éléments
90         // =7 -> de points d'intégrations relatifs à des surfaces d'éléments
91         // =8 -> de points d'intégrations relatifs à des arrete d'éléments
92
93         // Constructeur fonction du nb de maillage et du type de ref
94         ReferenceNE(int nbmaille , int indic);
95
96         // Constructeur fonction d'un tableau des numeros de la reference
97         // du nb de maillage, du type de ref
98         ReferenceNE(const Tableau<int>& tab,int nbmaille , int indic, string nom = "rien_actuellement");
99         // idem mais avec une liste d'entiers plutôt qu'un tableau
100        ReferenceNE(const list <int>& list_entiers,int nbmaille , int indic, string nom =
"rien_actuellement");
101
102        // Constructeur fonction d'un nom de reference
103        // du nb de maillage, du type de ref
104        ReferenceNE(string nom,int nbmaille , int indic);
105
106        // Constructeur de copie
107        ReferenceNE(const ReferenceNE& ref);
108
109
110        // DESTRUCTEUR :
111
112        ~ReferenceNE ();
113
114        // METHODES :
115
116        // création d'une référence du même type contenant les mêmes info
117        // que la référence donnée en argument: utilisation du constructeur de copie
118        Reference* Nevez_Ref_copie() const
119        {ReferenceNE* ref_ret=new ReferenceNE(*this); return ref_ret;} ;
120
121        Reference& operator= (const Reference& ref);
122
123        // Retourne le tableau des numeros de la reference en lecture uniquement

```

```

124     const Tableau<int>& Tab_num() const
125     { return tab_num; };
126
127 /*     inline int& Numero(int i)
128     // Retourne le ieme element du tableau des numeros
129     // de la reference (acces lecture et ecriture)
130     { return tab_num(i); };*/
131
132     // Retourne le ieme element du tableau des numeros
133     // de la reference (acces lecture uniquement)
134     int Numero(int i) const
135     { return tab_num(i); };
136
137     // Retourne le nombre de numeros de la reference
138     int Taille() const
139     { return tab_num.Taille(); };
140     // change un numéro de référence
141     void Change_num_dans_ref(int i,int nv_num)
142     { tab_num(i)=nv_num;};
143
144     // change le tableau de numéro de référence
145     void Change_tab_num(const Tableau<int>& tab)
146     {tab_num = tab;};
147
148     // supprime les doublons internes éventuels dans la référence
149     virtual void Supprime_doublons_interne();
150
151     // Affiche les donnees liees a la reference
152     void Affiche() const;
153
154 //----- méthodes découlant de virtuelles -----
155
156     // Affiche les donnees des références dans le flux passé en paramètre
157     void Affiche_dans_lis(ofstream& sort) const ;
158
159     // lecture d'une liste de reference de noeuds ou d'éléments
160     bool LectureReference(UtilLecture & entreePrinc);
161
162     // affichage et definition interactive des commandes
163     // nbMaxi: nombre maxi de noeud ou d'éléments pour les exemples
164     // cas : =1 premier passage, il s'agit de références de noeuds uniquement
165     // cas : =2 second passage, il s'agit de l'ensemble des possibilités de références
166     void Info_commande_Ref(int nbMaxi,UtilLecture * entreePrinc,int cas);
167
168     //----- lecture écriture dans base info -----
169     // cas donne le niveau de la récupération
170     // = 1 : on récupère tout
171     // = 2 : on récupère uniquement les données variables (supposées comme telles)
172     void Lecture_base_info(ifstream& ent,const int cas);
173     // cas donne le niveau de sauvegarde
174     // = 1 : on sauvegarde tout
175     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
176     void Ecriture_base_info(ofstream& sort,const int cas);
177
178     protected :
179
180     Tableau<int> tab_num; // tableau des numeros de noeuds ou d'élément de la reference
181
182 };
183 /// @} // end of group
184
185 #ifndef MISE_AU_POINT
186     #include "ReferenceNE.cc"
187     #define REFERENCENE_H_deja_inclus
188 #endif
189
190 #endif
191
192

```

## 7.375 ReferencePtiAF.h

```

1 // FICHER : ReferencePtiAF.h
2 // CLASSE : ReferencePtiAF
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)

```

```

14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *   DATE:      22/04/2020
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   *****
41 *   BUT:  Def, stockage et manipulation des références pour les
42 *         pti de faces et d'arêtes .
43 *
44 *   *****
45 *
46 *   VERIFICATION:
47 *
48 *   ! date ! auteur ! but
49 *   -----
50 *   !     !     !
51 *   *****
52 *   MODIFICATIONS:
53 *
54 *   ! date ! auteur ! but
55 *   -----
56 *   $
57 *   *****/
58
59 #ifndef REFERENCEPTIAF_H
60 #define REFERENCEPTIAF_H
61
62 #include "Reference.h"
63
64
65 /// @addtogroup Les_classes_Reference
66 /// @{
67 ///
68
69 /// Def, stockage et manipulation des références pour les pti de faces et d'arêtes
70 ///
71 /// \author Gérard Rio
72 /// \version 1.0
73 /// \date 22/04/2020
74
75 class ReferencePtiAF : public Reference
76 {
77     public :
78
79         // CONSTRUCTEURS :
80
81         // Constructeur par défaut
82         ReferencePtiAF ( string nom = "rien_actuellement");
83
84         // indic = 1 : il s'agit de reference de noeud
85         // indic = 2 : il s'agit de reference d'element
86         // =3 -> surface relatives à des éléments,
87         // =4 -> arête relatives à des éléments,
88         // =5 -> des noeuds relatifs à des éléments
89         // =6 -> de points d'intégrations relatifs à des éléments
90         // =7 -> de points d'intégrations relatifs à des surfaces d'éléments
91         // =8 -> de points d'intégrations relatifs à des arrete d'éléments
92
93         // Constructeur fonction du nb de maillage et du type de ref
94         ReferencePtiAF (int nbmaille , int indic);
95
96         // Constructeur fonction de trois tableaux de numeros, le premier pour les numéros
97         // d'éléments le deuxième pour les numéros de faces, ou d'aretes,
98         // le troisième pour le nb du pt d'integ,
99         // puis le nb de maillage, et enfin le type de ref

```

```

100     ReferencePtiAF (const Tableau<int>& tabelle, const Tableau<int>& tabAF, const Tableau<int>& tabPti
101                   , int nbmaille , int indic, string nom = "rien_actuellement");
102     // idem mais avec des listes d'entiers plutôt que des tableaux
103     ReferencePtiAF (const list <int>& list_elem, const list <int>& list_numAF
104                   , const list <int>& list_num_pti
105                   , int nbmaille , int indic, string nom = "rien_actuellement");
106
107     // Constructeur fonction d'un nom de reference
108     // du nb de maillage, du type de ref
109     ReferencePtiAF(string nom, int nbmaille , int indic);
110
111     // Constructeur de copie
112     ReferencePtiAF(const ReferencePtiAF& ref);
113
114
115     // DESTRUCTEUR :
116
117     ~ReferencePtiAF();
118
119
120     // METHODES :
121
122     // création d'une référence du même type contenant les mêmes info
123     // que la référence donnée en argument: utilisation du constructeur de copie
124     Reference* Nevez_Ref_copie() const
125     { ReferencePtiAF* ref_ret = new ReferencePtiAF(*this); return ref_ret; };
126
127     Reference& operator= (const Reference& ref);
128
129     // Retourne le tableau des numeros d'élément
130     const Tableau<int>& Tab_Elem() const
131     { return tab_Elem; };
132
133     // Retourne le tableau des numeros de faces, ou d'arêtes, ou de noeud d'element, ou de points
134     // d'integ
135     const Tableau<int>& Tab_FA() const
136     { return tab_FA; };
137
138     // Retourne le tableau des numeros des points d'integ
139     const Tableau<int>& Tab_Pti() const
140     { return tab_pti; };
141
142     // Retourne le ieme element du tableau d'éléments
143     // de la reference (acces lecture uniquement)
144     int NumeroElem(int i) const
145     { return tab_Elem(i); };
146
147     // Retourne le ieme element du tableau de faces, ou d'arêtes, ou de noeud d'element, ou de
148     // points d'integ
149     // de la reference (acces lecture uniquement)
150     int NumeroFA(int i) const
151     { return tab_FA(i); };
152
153     // Retourne le ieme element du tableau de points d'integ
154     // de la reference (acces lecture uniquement)
155     int NumeroPti(int i) const
156     { return tab_pti(i); };
157
158     // Retourne le nombre de numeros de la reference
159     int Taille () const
160     { return tab_Elem.Taille(); };
161
162     // change un numéro i de référence
163     // nbe: le numéro d'élément, nbAF: le numéro de de faces, ou d'arêtes, nbPti
164     // le num de points d'integ
165     void Change_num_PtiAF_dans_ref(int i, int nbe, int nbAF, int nbPti);
166
167     // change les 3 tableaux de la ref
168     void Change_tab_num(const Tableau<int>& tabelle, const Tableau<int>& tabAF
169                       , const Tableau<int>& tabPti);
170
171     // indique si le numéro d'élément, le numéro de faces ou d'arêtes, et le num de points
172     // d'integ, passé en argument, fait partie de la référence
173     // nbe: le numéro d'élément, nbAF: le numéro de de faces, ou d'arêtes,
174     // nbPti le num de points d'integ
175     bool Contient_PtiAF(int nbe, int nbAF, int nbPti) const;
176
177     // supprime les doublons internes éventuels dans la référence
178     virtual void Supprime_doublons_internes();
179
180     // Affiche les donnees liees a la reference
181     void Affiche () const;
182
183     //----- méthodes découlant de virtuelles -----
184     // Affiche les donnees des références dans le flux passé en paramètre

```



```

185     void Affiche_dans_lis(ofstream& sort) const ;
186
187     // lecture d'une liste de reference
188     bool LectureReference(UtilLecture & entreePrinc);
189
190     // affichage et definition interactive des commandes
191     // nbMaxi: nombre maxi d'éléments pour les exemples
192     // cas : =1 premier passage, il s'agit de références de noeuds uniquement,
193     //       donc ici cela génère une erreur
194     // cas : =2 second passage, il s'agit de l'ensemble des possibilités de références
195     void Info_commande_Ref(int nbMaxi,UtilLecture * entreePrinc,int cas);
196
197     //----- lecture écriture dans base info -----
198     // cas donne le niveau de la récupération
199     // = 1 : on récupère tout
200     // = 2 : on récupère uniquement les données variables (supposées comme telles)
201     void Lecture_base_info(ifstream& ent,const int cas);
202     // cas donne le niveau de sauvegarde
203     // = 1 : on sauvegarde tout
204     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
205     void Ecriture_base_info(ofstream& sort,const int cas);
206
207     protected :
208
209         Tableau<int> tab_Elem; // tableau des numeros des éléments de la reference
210         Tableau<int> tab_FA; // tableau des numeros des faces ou arêtes de la reference
211         Tableau<int> tab_pti; // tableau des points d'intégrations
212
213 };
214 /// @} // end of group
215
216 #ifndef MISE_AU_POINT
217     #include "ReferencePtiAF.cc"
218     #define REFERENCEPTIAF_H_deja_inclus
219 #endif
220
221 #endif
222
223

```

## 7.376 ExceptionsMatrices.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           16/03/2006
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *   ****
38 *   BUT:   Définir des classes d'exception pour la gestion d'erreur
39 *           concernant les matrices.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```

```

43 *
44 * ! date ! auteur ! but ! *
45 * ----- *
46 * ! ! ! ! *
47 * $ *
48 * //***** *
49 * MODIFICATIONS: *
50 * ! date ! auteur ! but ! *
51 * ----- *
52 * $ *
53 *-----*/
54 #ifndef EXCEPTIONSMATRICES_H
55 #define EXCEPTIONSMATRICES_H
56
57 // cas d'une erreur survenue à cause de l'utilisation des matrices
58 // dans le cadre de la résolution d'un système linéaire
59
60 /// @addtogroup Les_classes_Matrices
61 /// @{
62 ///
63
64
65 class ErrResolve_system_lineaire
66 // =0 cas courant, pas d'information particulière
67 // =1 cas où l'erreur est sévère et ne pourra pas être corrigé en refaisant un calcul avec un
68 // pas de temps plus petit. Il faut refaire le calcul en se positionnant plusieurs pas de temps
69 // auparavant (utilisé par l'hystérésis par exemple)
70 { public :
71 int cas;
72 ErrResolve_system_lineaire () : cas(0) {} ; // par défaut
73 ErrResolve_system_lineaire (int ca) : cas(ca) {} ; // pb
74 };
75 /// @} // end of group
76
77
78 #endif

```

## 7.377 Mat\_abstraite.h

```

1 // FICHER : Mat_abstraite.h
2 // CLASSE : Mat_abstraite
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 * DATE: 23/01/97 *
35 * $ *
36 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
37 * $ *
38 * PROJET: Herezh++ *
39 * $ *
40 *****/
41 * La classe Mat_abstraite est une classe abstraite qui permet de cacher les
42 * implémentations spécifiques à un type particulier de matrices.
43 * Cette classe fournit un ensemble de méthodes qui seront implémentées dans
44 * les classes dérivées caractérisant un type particulier de matrice.
45 * $ *

```

```

46 *      *
47 *      VERIFICATION:
48 *
49 *      ! date ! auteur ! but
50 *      -----
51 *      ! ! !
52 *      $
53 *      *
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *      $
58 *      */
59
60 // La classe Mat_abstraite est une classe abstraite qui permet de cacher les
61 // implementations spécifiques a un type particulier de matrices.
62 // Cette classe fournit un ensemble de methodes qui seront implementees dans
63 // les classes derivees caracterisant un type particulier de matrice.
64
65
66 #ifndef MAT_ABSTRAITE_H
67 #define MAT_ABSTRAITE_H
68
69 #include "mvvtp_GR.h" // classe template MV++
70 #include "Vecteur.h"
71 #include "Tableau_T.h"
72 #include "Enum_matrice.h"
73 #include "Enum_type_resolution_matri.h"
74 #include "ExceptionsMatrices.h"
75 #include "Coordonnee3.h"
76
77
78 class VeurPropre;
79 /** @defgroup Les_classes_Matrices
80 *
81 * BUT: les différentes classes de matrices.
82 *
83 *
84 * \author Gérard Rio
85 * \version 1.0
86 * \date 23/01/97
87 * \brief les différentes classes de matrices.
88 *
89 */
90
91 /// @addtogroup Les_classes_Matrices
92 /// @{
93 ///
94
95
96 class Mat_abstraite
97 {
98 protected :
99 // grandeur par défaut déclarées avant car on s'en sert dans les passages de paramètres
100 static const double tol_defaut; // tolérance sur le résidu dans les méthodes itératives
101 static const int maxit_defaut ; // maximum d'itération dans les méthodes itératives
102 static const int restart_defaut ;// maximum de restart itération (nb de vecteur sauvegardé)
103
104 public :
105
106 // Constructeur par défaut
107 Mat_abstraite (Enum_matrice type_mat = RIEN_MATRICE,
108 Enum_type_resolution_matri type_resol=RIEN_TYPE_RESOLUTION_MATRI,
109 Enum_preconditionnement type_precondi= RIEN_PRECONDITIONNEMENT ):
110 type_matrice(type_mat),type_resolution(type_resol),
111 type_preconditionnement(type_precondi)
112 {}
113 // de copie
114 Mat_abstraite (const Mat_abstraite& a) :
115 type_matrice(a.type_matrice),type_resolution(a.type_resolution),
116 type_preconditionnement(a.type_preconditionnement)
117 {};
118
119 // DESTRUCTEUR VIRTUEL :
120
121 virtual ~Mat_abstraite ()
122 {};
123
124
125 // METHODES VIRTUELLES PURES :
126
127 // surcharge de l'opérateur d'affectation
128 // IMPORTANT : le fonctionnement de l'implantation dépend de la classe
129 // dérivée : en particulier il peut y avoir redimensionnement automatique
130 // de la matrice en fonction de l'attribut, cas par exemple des matrices
131 // pleines, ou message d'erreur car les tailles ne sont pas identiques

```

```

132 // exemple de la classe mat_band
133 virtual Mat_abstraite & operator = ( const Mat_abstraite & ) = 0;
134
135 // transfert des informations de *this dans la matrice passée en paramètre
136 // la matrice paramètre est au préalable, mise à 0.
137 virtual void Transfert_vers_mat( Mat_abstraite & A ) = 0;
138
139 //-----
140 // --- plusieurs fonctions virtuelles qui agissent en général sur la matrice -----
141 //-----
142 // ici l'idée est d'éviter de construire une nouvelle matrice pour des questions
143 // d'encombrement, c'est surtout des méthodes utiles pour le stockage de matrice
144 // de raideur. On met le nombre mini de fonction, pour pouvoir faire des expressions.
145 // Cependant aucune de ces fonctions n'est en désaccord avec les fonctions membres
146 // qui crée une matrice en résultat (cf. Mat_pleine)
147 // pour les 2 premières méthodes : += et -= :
148 // les matrices arguments sont en général du mêmes type que celui de la matrice this
149 // mais cela fonctionne également avec comme argument une matrice diagonale (pour tous les
types)
150
151 // Surcharge de l'opérateur += : addition d'une matrice a la matrice courante
152 virtual void operator+= (const Mat_abstraite& mat_pl) = 0;
153
154 // Surcharge de l'opérateur -= : soustraction d'une matrice a la matrice courante
155 virtual void operator-= (const Mat_abstraite& mat_pl) = 0;
156
157 // Surcharge de l'opérateur *= : multiplication de la matrice courante par un scalaire
158 virtual void operator*= (const double r) = 0;
159
160 //-----
161 // --- fin de plusieurs fonctions virtuelles qui agissent en général sur la matrice --
162 //-----
163
164 // Surcharge de l'opérateur == : test d'egalite entre deux matrices
165 virtual int operator==(const Mat_abstraite& mat_pl) const =0;
166
167 inline int operator!=(const Mat_abstraite& mat_pl) const
168 // Surcharge de l'opérateur != : test de non egalite entre deux matrices
169 // Renvoie 1 si les deux matrices ne sont pas egales
170 // Renvoie 0 sinon
171 { if ( (*this)==mat_pl )
172     return 0;
173     else
174     return 1;
175 };
176
177 // fonction permettant de creer une nouvelle instance d'element
178 // dérivé. la nouvelle instance = *this, il y a donc utilisation du constructeur de copie
179 virtual Mat_abstraite * NouvelElement() const = 0;
180
181 // Affichage des valeurs de la matrice
182 virtual void Affiche () const =0;
183
184 // Initialisation des valeurs des composantes
185 virtual void Initialise (double val_init) =0;
186
187 // Liberation de la place memoire
188 virtual void Libere () =0;
189
190 // Retour du nombre de lignes de la matrice vue comme une matrice_rectangulaire
191 virtual int Nb_ligne() const =0;
192
193 // Retour du nombre de colonnes de la matrice vue comme une matrice_rectangulaire
194 virtual int Nb_colonne() const =0;
195
196 // Symetrie de la matrice
197 virtual int Symetrie () const =0;
198
199 // Acces aux valeurs de la matrices en écriture
200 // i : indice de ligne, j : indice de colonne
201 virtual double& operator () (int i, int j) =0;
202
203 // ramène true si la place (i,j) existe, false sinon
204 // ici on ne test pas le fait que i et j puissent être négatif ou pas
205 virtual bool Existe(int i, int j) const = 0;
206
207 // Acces aux valeurs de la matrices en lecture
208 // cas ou l'on ne veut pas modifier les valeurs
209 // i : indice de ligne, j : indice de colonne
210 // !!! important, si l'élément n'existe pas du au type de stockage
211 // utilisé, (mais que les indices sont possible) le retour est 0
212 virtual double operator () ( int i, int j) const =0;
213
214 // Retourne la ieme ligne de la matrice
215 // lorsque implemente ( a verifier )
216 virtual Vecteur& Ligne_set(int i) = 0;
217

```

```

218 // Retourne la ieme ligne de la matrice
219 virtual Vecteur Ligne(int i) const = 0;
220
221 // Retourne la ieme ligne de la matrice
222 // sous le format de stockage propre a la matrice
223 // donc a n'utiliser que comme sauvegarde en parralele
224 // avec la fonction RemplaceLigne
225 virtual Vecteur LigneSpe(int i) const = 0;
226 // remplace la ligne de la matrice par la ligne fournie
227 virtual void RemplaceLigneSpe(int i,const Vecteur & v) = 0;
228
229 //met une valeur identique sur toute la ligne
230 virtual void MetVallLigne(int i,double val) = 0;
231
232 // Retourne la jeme colonne de la matrice
233 virtual Vecteur Colonne(int j) const = 0;
234
235 // Retourne la jeme colonne de la matrice
236 // sous le format de stockage propre a la matrice
237 // donc a n'utiliser que comme sauvegarde en parralele
238 // avec la fonction RemplaceColonne
239 virtual Vecteur ColonneSpe(int j) const = 0;
240 // remplace la Colonne de la matrice par la colonne fournie
241 virtual void RemplaceColonneSpe(int j,const Vecteur & v) = 0;
242
243 //met une valeur identique sur toute la colonne
244 virtual void MetValColonne(int j,double val) = 0;
245
246 // Resolution du systeme Ax=b
247 //1) avec en sortie un new vecteur
248 virtual Vecteur Resol_syst (const Vecteur& b,const double &tol = tol_default
249 ,const int maxit = maxit_default,const int restart = restart_default) =0;
250 //2) avec en sortie le vecteur d'entree
251 virtual Vecteur& Resol_systID (Vecteur& b,const double &tol = tol_default
252 ,const int maxit = maxit_default,const int restart = restart_default) =0;
253 //3) avec en entrée un tableau de vecteur second membre et
254 // en sortie un nouveau tableau de vecteurs
255 virtual Tableau <Vecteur> Resol_syst
256 (const Tableau <Vecteur>& b,const double &tol = tol_default
257 ,const int maxit = maxit_default,const int restart = restart_default) =0;
258 //4) avec en entrée un tableau de vecteur second membre et
259 // en sortie le tableau de vecteurs d'entree
260 virtual Tableau <Vecteur>& Resol_systID
261 (Tableau <Vecteur>& b,const double &tol = tol_default
262 ,const int maxit = maxit_default,const int restart = restart_default) =0;
263 //5) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
264 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
265 // deux étant identiques
266 virtual Vecteur& Resol_systID_2 (const Vecteur& b,Vecteur& vortie
267 ,const double &tol = tol_default
268 ,const int maxit = maxit_default
269 ,const int restart = restart_default) =0;
270 // ===== RÉOLUTION EN DEUX TEMPS ===== :
271 // 1) préparation de la matrice donc modification de la matrice éventuellement
272 // par exemple pour les matrices bandes avec cholesky : triangulation
273 virtual void Preparation_resol() =0 ;
274 // 2) *** résolution sans modification de la matrice DOIT ÊTRE PRÉCÉDÉ DE L'APPEL DE
275 // Preparation_resol
276 // a) avec en sortie un new vecteur
277 virtual Vecteur Simple_Resol_syst (const Vecteur& b,const double &tol = tol_default
278 ,const int maxit = maxit_default,const int restart = restart_default) const =0 ;
279 // b) avec en sortie le vecteur d'entree
280 virtual Vecteur& Simple_Resol_systID (Vecteur& b,const double &tol = tol_default
281 ,const int maxit = maxit_default,const int restart = restart_default) const =0 ;
282 // c) avec en entrée un tableau de vecteur second membre et
283 // en sortie un nouveau tableau de vecteurs
284 virtual Tableau <Vecteur> Simple_Resol_syst
285 (const Tableau <Vecteur>& b,const double &tol = tol_default
286 ,const int maxit = maxit_default,const int restart = restart_default) const =0 ;
287 // d) avec en entrée un tableau de vecteur second membre et
288 // en sortie le tableau de vecteurs d'entree
289 virtual Tableau <Vecteur>& Simple_Resol_systID
290 (Tableau <Vecteur>& b,const double &tol = tol_default
291 ,const int maxit = maxit_default,const int restart = restart_default) const =0 ;
292 // e) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
293 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
294 // deux étant identiques
295 virtual Vecteur& Simple_Resol_systID_2 (const Vecteur& b,Vecteur& vortie
296 ,const double &tol = tol_default
297 ,const int maxit = maxit_default
298 ,const int restart = restart_default) const =0 ;
299 // ===== FIN RÉOLUTION EN DEUX TEMPS ===== :
300
301 // Multiplication d'un vecteur par une matrice ( (vec)t * A )
302 virtual Vecteur Prod_vec_mat ( const Vecteur& vec) const =0;
303 // idem mais on utilise la place du second vecteur pour le résultat
304 virtual Vecteur& Prod_vec_mat ( const Vecteur& vec, Vecteur & resul) const =0;

```

```

305
306 // Multiplication d'une matrice par un vecteur ( A * vec )
307 virtual Vecteur Prod_mat_vec ( const Vecteur& vec) const =0;
308 // idem mais on utilise la place du second vecteur pour le résultat
309 virtual Vecteur& Prod_mat_vec ( const Vecteur& vec, Vecteur & resul) const =0;
310
311 // Multiplication d'une ligne iligne de la matrice avec un vecteur de
312 // dimension = le nombre de colonne de la matrice
313 virtual double Prod_Ligne_vec ( int iligne,const Vecteur& vec) const =0;
314
315 // Multiplication d'un vecteur avec une colonne icol de la matrice
316 // dimension = le nombre de ligne de la matrice
317 virtual double Prod_vec_col( int icol,const Vecteur& vec) const =0;
318
319 // calcul du produit : (vec_1)^T * A * (vect_2)
320 virtual double vectT_mat_vec(const Vecteur& vec1, const Vecteur& vec2) const =0;
321
322 // retourne la place que prend la matrice en entier
323 virtual int Place() const = 0;
324
325 // ----- méthode non virtuelle pur -> générales -----
326
327
328 // Affiche une partie de la matrice (util pour le debug)
329 // min_ et max_ sont les bornes de la sous_matrice
330 // pas_ indique le pas en i et j pour les indices
331 virtual void Affiche1(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j) const ;
332
333 // Affiche une partie de la matrice idem si dessus
334 // mais avec un nombre de digit (>7) = nd
335 // si < 7 ne fait rien
336 virtual void Affiche2
337     (int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j,int nd) const ;
338
339 // changement du choix de la méthode de résolution si c'est possible
340 // peut être surchargé par les classes dérivées en fonction des spécificités
341 virtual void Change_Choix_resolution(Enum_type_resolution_matri type_resol,
342     Enum_preconditionnement type_precondi)
343     { type_resolution = type_resol; type_preconditionnement = type_precondi;};
344
345 // retourne le type de la matrice
346 inline const Enum_matrice Type_matrice() const {return type_matrice;};
347
348 // affichage à l'écran après une demande interactive de la matrice
349 // entete: une chaine explicative, a afficher en entête
350 void Affichage_ecran(string entete) const;
351
352 // calcul, récupération et affichage éventuelle
353 // des mini, maxi, et en valeur absolue la moyenne de la diagonale de la matrice
354 // en retour: le min, le max et la moyenne en valeur absolue
355 Coordonnee3 MinMaxMoy(bool affiche) const;
356
357 // limite la valeur mini de la diagonale: si un terme de la diagonale
358 // est inférieure à la limite passée en argument (seuil_bas), d'une manière arbitraire
359 // le terme à mis à une valeur égale au paramètre passé en paramètre val_a_imposer
360 // s'il y a eu changement retour de true
361 bool Limitation_min_diag(double seuil_bas, double val_a_imposer) ;
362
363
364 //===== quelques méthodes spécifiques à la classe MV_Vector<double>=====
365 // définit la surcharge de multiplication d'une matrice par un MV_Vector
366 // ici de type constant, ramène un nouveau MV_Vector<double>
367 MV_Vector <double> operator * (const MV_Vector <double> & vec) const ;
368 // multiplication matrice transposée fois vecteur ce qui est équivalent à
369 // la transposée du résultat de : multiplication vecteur transposé fois matrice
370 virtual MV_Vector <double> trans_mult(const MV_Vector <double> &x) const ;
371 //===== fin des quelques méthodes spécifiques à la classe MV_Vector <double>=====
372
373 //calcul des valeurs propres par la méthode des puissances itérées
374 // inverses.
375 // --> Utilisable que pour des matrices carrées
376 // Intéressant lorsque l'on veut les plus petites
377 // valeurs propres, en nombre restreint car seule les premières valeurs
378 // sont calculées avec une précision convenable. On obtient également
379 // les vecteurs propres correspondant.
380 // résolution de : ((*this) - lambda KG) X = 0
381 // en entrée : KG, VP dont la dimension est défini et fourni le nombre
382 // de valeurs propres que l'on veut calculer
383 // On considère que les conditions limites sont déjà appliquées à (*this) et KG.
384 // c'est à dire des 1 sur la diagonale pour (*this) et des 0 sur la diagonale pour KG
385 // en sortie : VP contient les valeurs propres et vecteurs propres
386
387 Tableau <VeurPropre>* V_Propres(Mat_abstraite& KG,Tableau <VeurPropre> & VP );
388
389 protected :
390
391 // VARIABLES PROTEGEES :

```

```

392     Enum_matrice type_matrice; // le type de la matrice, défini dans les classes dérivée
393     Enum_type_resolution_matri type_resolution; // le type de résolution envisagée
394     Enum_preconditionnement type_preconditionnement; // le type éventuelle de preconditionnement
395
396                                     // pour la méthode "résidu minimal généralisé"
397
398     // Méthodes protégées :
399     // Résolution du système Ax=b , méthodes générales pouvant être appelées par les classes dérivée
400     // en entrée : b : comme second membre
401     //           tol : tolérance sur le résidu
402     //           maxit : le maximum d'itération
403     //           restart : le Maximum de restart iterations (vecteur sauvegardé) dans le cas
404     //                   de la méthode résidu minimal généralisé (GENE_MINI_RESIDUAL)
405     // en sortie : sol de même dimension que sol
406     void Resolution_syst
407         (const Vecteur &b, Vecteur &sol, const double &tol, const int maxit = 300
408          , const int restart = 32) const ;
409     // idem avec en entrée un tableau de vecteur second membre et
410     // en sortie un tableau de vecteurs solution
411     void Resolution_syst
412         (const Tableau <Vecteur>& b, Tableau <Vecteur> &sol, const double &tol
413          , const int maxit = 300
414          , const int restart = 32) const ;
415
416 };
417 /// @} // end of group
418
419
420 #include "VeurPropre.h"
421
422 #endif
423
424

```

## 7.378 Mat\_pleine.h

```

1 // FICHER : Mat_pleine.h
2 // CLASSE : Mat_pleine
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *   DATE:      23/01/97
35 *
36 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *
41 * La classe Mat_pleine derive de la classe Mat_abstraite et permet de declarer
42 * une matrice pleine dont les composantes sont de type double. Toute instance
43 * de cette classe n'a pas de restrictions particulieres et est capable de représenter
44 * n'importe quel type de matrices.
45 *
46 * Une telle matrice est stockee a l'aide d'un tableau de vecteurs à l'aide des classes
47 * Tableau<T> et Vecteur de facon a beneficier des methodes introduites au niveau de
48 * ces dernieres.
49 *

```

```

50 *      *
51 *      VERIFICATION:
52 *
53 *      ! date ! auteur ! but
54 *      -----
55 *      ! ! ! !
56 *      $
57 *      *
58 *      MODIFICATIONS:
59 *      ! date ! auteur ! but
60 *      -----
61 *      $
62 *      *****/
63
64 // La classe Mat_pleine derive de la classe Mat_abstraite et permet de declarer
65 // une matrice pleine dont les composantes sont de type double. Toute instance
66 // de cette classe n'a pas de restrictions particulieres et est capable de représenter
67 // n'importe quel type de matrices.
68 //
69 // Une telle matrice est stockee a l'aide d'un tableau de vecteurs à l'aide des classes
70 // Tableau<T> et Vecteur de facon a beneficier des methodes introduites au niveau de
71 // ces dernieres.
72
73
74 #ifndef MAT_PLEINE_H
75 #define MAT_PLEINE_H
76
77
78 #include "Mat_abstraite.h"
79 #include "Tableau_T.h"
80 #include "Vecteur.h"
81 #include "Coordonnee.h"
82 #include "Coordonnee3.h"
83
84 class Mat_abstraite;
85
86 /// @addtogroup Les_classes_Matrices
87 /// @{
88 ///
89
90
91 class Mat_pleine : public Mat_abstraite
92 {
93     // surcharge de l'operator de lecture typée
94     friend istream & operator » (istream &, Mat_pleine &);
95     // surcharge de l'operator d'écriture typée
96     friend ostream & operator « (ostream &, const Mat_pleine &);
97
98     public :
99         // CONSTRUCTEURS :
100
101         // Constructeur par défaut
102         Mat_pleine ();
103
104         // Constructeur se servant d'un nombre de lignes et de colonnes, et éventuellement
105         // d'une valeur d'initialisation
106         Mat_pleine (int nb_ligne,int nb_colonne,double val_init=0.0);
107
108         // Constructeur de copie
109         Mat_pleine (const Mat_pleine& mat_pl);
110
111         // DESTRUCTEUR :
112
113         virtual ~Mat_pleine ();
114
115
116         // METHODES :
117
118         // fonction permettant de creer une nouvelle instance d'element
119         Mat_abstraite * NouvelElement() const ;
120
121         // surcharge de l'opérateur d'affectation : cas de matrices abstraites
122         // il y a ajustement de la taille de la matrice en fonction de mat_pl
123         Mat_abstraite & operator = ( const Mat_abstraite & mat_pl);
124
125         // transfert des informations de *this dans la matrice passée en paramètre
126         // la matrice paramètre est au préalable, mise à 0.
127         void Transfert_vers_mat ( Mat_abstraite & A );
128
129         //-----
130         // --- plusieurs fonctions virtuelles qui agissent en général sur la matrice -----
131         //-----
132         // ici l'idée est d'éviter de construire une nouvelle matrice pour des questions
133         // d'encombrement, c'est surtout des méthodes utiles pour le stockage de matrice
134         // de raideur. On met le nombre mini de fonction, pour pouvoir faire des expressions.
135         // Cependant aucune de ces fonctions n'est en désaccord avec les fonctions membres

```



```

136 // qui crée une matrice en résultat (cf. Mat_pleine)
137
138 // Surcharge de l'opérateur += : addition d'une matrice a la matrice courante
139 void operator+= (const Mat_abstraite& mat_pl);
140
141 // Surcharge de l'opérateur -= : soustraction d'une matrice a la matrice courante
142 void operator-= (const Mat_abstraite& mat_pl);
143
144 //-----
145 // --- fin de plusieurs fonctions virtuelles qui agissent en général sur la matrice ---
146 //-----
147
148 // Surcharge de l'opérateur == : test d'égalité entre deux matrices
149 int operator== (const Mat_abstraite& mat_pl) const ;
150
151 // Surcharge de l'opérateur = : affectation d'une matrice a une autre
152 // il y a ajustement de la taille de la matrice en fonction de mat_pl
153 Mat_pleine& operator= (const Mat_pleine& mat_pl);
154
155 // Affichage des données de la matrice
156 void Affiche () const ;
157
158 // Initialisation des composantes de la matrice (par défaut a 0, sinon a val_init)
159 void Initialise (double val_init=0.0);
160
161 // Initialisation d'une matrice a un nombre de lignes et de colonnes ainsi que
162 // ces composantes a 0 par défaut ou a val_init sinon
163 void Initialise (int nb_ligne,int nb_colonne,double val_init=0.0);
164
165 inline void Libere ()
166 // A la suite de l'appel de cette methode, la matrice est identique a ce qu'elle
167 // serait a la suite d'un appel du constructeur par défaut
168 { val.Libere(); };
169
170 // Retour du nombre de lignes de la matrice vue comme une matrice_rectangulaire
171 inline int Nb_ligne () const
172 { return val.Taille(); };
173
174 // Retour du nombre de colonnes de la matrice vue comme une matrice_rectangulaire
175 // N.B. : La taille des vecteurs du tableau val est supposee etre la meme pour tous
176 // les vecteurs du tableau (aucun test n'est realise pour le verifier)
177 inline int Nb_colonne () const
178 { if ( Nb_ligne()!=0 )
179     return val(1).Taille();
180     else
181     return 0;
182 };
183
184 // Retourne la ieme ligne de la matrice
185 inline Vecteur Ligne (int i) const
186 { return val(i); };
187
188 // Retourne la ieme ligne de la matrice
189 // sous le format de stockage propre a la matrice
190 // donc a n'utiliser que comme sauvegarde en parrallele
191 // avec la fonction RemplaceLigne
192 inline Vecteur LigneSpe (int i) const
193 { return val(i); };
194 // remplace la ligne de la matrice par la ligne fournie
195 void RemplaceLigneSpe(int i,const Vecteur & v);
196 //met une valeur identique sur toute la ligne
197 void MetValLigne(int i,double x);
198
199 // Retourne la jieme colonne de la matrice
200 inline Vecteur Colonne (int j) const
201 { Vecteur result(Nb_ligne());
202   for (int i=1;i<=Nb_ligne();i++)
203     result(i)=(*this)(i,j);
204   return result;
205 };
206
207 // Retourne la jeme colonne de la matrice
208 // sous le format de stockage propre a la matrice
209 // donc a n'utiliser que comme sauvegarde en parrallele
210 // avec la fonction RemplaceColonne
211 inline Vecteur ColonneSpe (int j) const
212 // Retourne la jieme colonne de la matrice
213 { return Colonne(j);
214 };
215 // remplace la Colonne de la matrice par la colonne fournie
216 void RemplaceColonneSpe(int j,const Vecteur & v);
217 //met une valeur identique sur toute la colonne
218 void MetValColonne(int j,double y);
219
220 // Test sur la symetrie de la matrice
221 int Symetrie () const ;
222 // affiche les termes non symétriques de la matrice s'il y en a

```

```

223     void AfficheNonSymetries() const;
224
225     // Resolution du systeme Ax=b par la methode de Cholesky, de cramer ou appel de la résolution
générale
226     // par exemple gradient conjugué
227     //1) avec en sortie un new vecteur
228     Vecteur Resol_syst (const Vecteur& b,const double &tol = tol_defaut
229     ,const int maxit = maxit_defaut,const int restart = restart_defaut);
230     //2) avec en sortie le vecteur d'entree
231     Vecteur& Resol_systID (Vecteur& b,const double &tol = tol_defaut
232     ,const int maxit = maxit_defaut,const int restart = restart_defaut);
233     //3) avec en entrée un tableau de vecteur second membre et
234     //    en sortie un nouveau tableau de vecteurs
235     Tableau <Vecteur> Resol_syst
236     (const Tableau <Vecteur>& b,const double &tol = tol_defaut
237     ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
238     //4) avec en entrée un tableau de vecteur second membre et
239     //    en sortie le tableau de vecteurs d'entree
240     Tableau <Vecteur>& Resol_systID
241     (Tableau <Vecteur>& b,const double &tol = tol_defaut
242     ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
243     //5) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
244     //    et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
245     //    deux étant identiques
246     Vecteur& Resol_systID_2 (const Vecteur& b,Vecteur& vortie
247     ,const double &tol = tol_defaut,const int maxit = maxit_defaut
248     ,const int restart = restart_defaut);
249
250     // ===== RÉOLUTION EN DEUX TEMPS ===== :
251     //    1) préparation de la matrice donc modification de la matrice éventuellement
252     //    par exemple pour les matrices bandes avec cholesky : triangulation
253     void Preparation_resol();
254     //    2) *** résolution sans modification de la matrice DOIT ÊTRE PRÉCÉDÉ DE L'APPEL DE
255     //    Preparation_resol
256     //    a) avec en sortie un new vecteur
257     Vecteur Simple_Resol_syst (const Vecteur& b,const double &tol = tol_defaut
258     ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
259     //    b) avec en sortie le vecteur d'entree
260     Vecteur& Simple_Resol_systID (Vecteur& b,const double &tol = tol_defaut
261     ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
262     //    c) avec en entrée un tableau de vecteur second membre et
263     //    en sortie un nouveau tableau de vecteurs
264     Tableau <Vecteur> Simple_Resol_syst
265     (const Tableau <Vecteur>& b,const double &tol = tol_defaut
266     ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
267     //    d) avec en entrée un tableau de vecteur second membre et
268     //    en sortie le tableau de vecteurs d'entree
269     Tableau <Vecteur>& Simple_Resol_systID
270     (Tableau <Vecteur>& b,const double &tol = tol_defaut
271     ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
272     //    e) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
273     //    et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
274     //    deux étant identiques
275     Vecteur& Simple_Resol_systID_2 (const Vecteur& b,Vecteur& vortie
276     ,const double &tol = tol_defaut
277     ,const int maxit = maxit_defaut
278     ,const int restart = restart_defaut) const ;
279     // ===== FIN RÉOLUTION EN DEUX TEMPS ===== :
280
281     // Produit d'un vecteur par une matrice
282     Vecteur Prod_vec_mat (const Vecteur& vec) const;
283     // idem mais on utilise la place du second vecteur pour le résultat
284     Vecteur& Prod_vec_mat (const Vecteur& vec, Vecteur & res) const ;
285     // Produit d'une matrice par un vecteur
286     Vecteur Prod_mat_vec (const Vecteur& vec) const;
287     // idem mais on utilise la place du second vecteur pour le résultat
288     Vecteur& Prod_mat_vec (const Vecteur& vec, Vecteur & res) const;
289     // Détermine la transpose d'une matrice
290     Mat_pleine Transpose() const;
291
292     // determine l'inverse d'une matrice carre
293     // actuellement uniquement implemente pour dim = 1,2,3
294     Mat_pleine Inverse() const;
295     // idem mais en retournant l'inverse dans la matrice passée en paramètre
296     // qui doit être de même type et de même dimension que this
297     Mat_pleine& Inverse(Mat_pleine& res) const;
298
299     // détermine le déterminant d'une matrice
300     // actuellement uniquement implemente pour dim = 1,2,3
301     double Determinant() const;
302
303     // Surcharge de l'operateur + : addition de deux matrices
304     Mat_pleine operator+ (const Mat_pleine& mat_pl) const;
305
306     // Surcharge de l'operateur - : soustraction entre deux matrices
307     Mat_pleine operator- (const Mat_pleine& mat_pl) const;
308

```

```

309     // Surcharge de l'opérateur - : oppose d'une matrice
310     Mat_pleine operator- () const;
311
312     // Surcharge de l'opérateur += : addition d'une matrice a la matrice courante
313     void operator+= (const Mat_pleine& mat_pl);
314
315     // Surcharge de l'opérateur -= : soustraction d'une matrice a la matrice courante
316     void operator-= (const Mat_pleine& mat_pl);
317
318     // Surcharge de l'opérateur * : multiplication de deux matrices
319     Mat_pleine operator* (const Mat_pleine& mat_pl) const;
320
321     // Surcharge de l'opérateur *= : multiplication de la matrice courante par un scalaire
322     void operator*= (const double r);
323
324     inline Vecteur operator* (const Vecteur& vec) const
325     // Realise la multiplication d'une matrice par un vecteur
326     { return Prod_mat_vec((Vecteur&) vec); };
327
328     // Surcharge de l'opérateur * : multiplication d'une matrice par un scalaire
329     Mat_pleine operator* (const double coeff) const;
330
331     // Surcharge de l'opérateur == : test d'egalite entre deux matrices
332     int operator== (const Mat_pleine& mat_pl) const ;
333
334     inline int operator!= (const Mat_pleine& mat_pl) const
335     // Surcharge de l'opérateur != : test de non egalite entre deux matrices
336     // Renvoie 1 si les deux matrices ne sont pas egales
337     // Renvoie 0 sinon
338     { if ( (*this)==mat_pl )
339         return 0;
340         else
341             return 1;
342     };
343
344     inline Vecteur& Ligne_set(int i)
345     // Retourne la ieme ligne de la matrice
346     { return val(i); };
347
348     inline Vecteur& operator() (int i)
349     // Retourne également de la ieme ligne de la matrice
350     // (on a changé: un moment on ne ramenait qu'une copie, mais c'est trop dangereux)
351     // ( car naturellement on s'attend à avoir la ligne et cela engendre des erreurs)
352     // ( très difficile à trouver, car tant que la copie persiste c'est ok, et quand le destructeur)
353     // ( est appelé -> plus rien !! )
354     { return val(i); };
355
356     inline Vecteur operator() (int i) const
357     // Retourne une copie de la ieme ligne de la matrice
358     // utile quand on travaille avec des const
359     { return val(i); };
360
361     inline double& operator () (int i,int j)
362     // Retourne la ieme jieme composante de la matrice
363     { return val(i)(j); };
364
365     // ramène true si la place (i,j) existe, false sinon
366     // ici on ne test pas le fait que i et j puissent être négatif ou pas
367     inline bool Existe(int , int ) const {return true;};
368
369     // Acces aux valeurs de la matrices
370     // cas ou l'on ne veut pas modifier les valeurs
371     // i : indice de ligne, j : indice de colonne
372     inline double operator () ( int i, int j) const
373     { return val(i)(j); };
374
375     // Multiplication d'une ligne iligne de la matrice avec un vecteur de
376     // dimension = le nombre de colonne de la matrice
377     double Prod_Ligne_vec ( int iligne, const Vecteur& vec) const;
378
379     // Multiplication d'un vecteur avec une colonne icol de la matrice
380     // dimension = le nombre de ligne de la matrice
381     double Prod_vec_col( int icol, const Vecteur& vec) const;
382
383     // calcul du produit : (vect_1)^T * A * (vect_2)
384     double vectT_mat_vec(const Vecteur& vec1, const Vecteur& vec2) const ;
385
386     // mise a zero de tous les composantes de la matrice
387     void Zero()
388     { int imax = val.Taille();for (int i=1; i<=imax;i++) val(i).Zero();};
389
390     // retourne la place que prend la matrice en entier
391     int Place() const { return (2*(Nb_ligne() * Nb_colonne()));};
392
393     // ----- méthodes particulières (en particulier non virtuelles) -----
394
395     // ramène un tableau de coordonnées (avec 3 variances possibles) correspondant à la matrice

```

```

396 // tab(i) = la colonne i de la matrice
397 Tableau <Coordonnee > Coordonnee_Base_associee() const;
398 Tableau <CoordonneeH > CoordonneeH_Base_associee() const;
399 Tableau <CoordonneeB > CoordonneeB_Base_associee() const;
400 // idem mais une seule colonne i
401 Coordonnee Coordonnee_Base_associee(int i) const;
402 CoordonneeH CoordonneeH_Base_associee(int i) const;
403 CoordonneeB CoordonneeB_Base_associee(int i) const;
404
405
406 // ramène le maxi en valeur absolue des valeurs de la matrice, et les indices associées
407 double MaxiValAbs(int & i, int & j) const;
408 // ramène le maxi en valeur absolue de la somme des valeurs absolues de chaque ligne,
409 // et l'indice de ligne associé
410 // permet d'avoir une approximation de la valeur propre maximale de la matrice via le
411 // théorème de Gerschgorin :  $|\lambda| < \text{Max}_i (|\lambda_{ii}|)$ 
412 // avec:  $|\lambda_{ii}| < \text{Max}_j \text{somme}_j |k_{ij}|$ 
413 double Maxi_ligne_ValAbs(int & i) const;
414
415
416 protected :
417
418
419     Tableau<Vecteur> val; // Valeurs des composantes
420     // val(i) = une ligne
421
422     // calcul de la matrice triangulée dans le cadre de la méthode de cholesky
423     // utilisation particulière : la matrice résultat B est défini dans le programme
424     // appelant.
425     void Triangulation (int N,Mat_pleine& B);
426     // résolution du problème triangularisé
427     // second membre b, et résultat res
428     void Resolution (int N, const Mat_pleine& B, const Vecteur& b,Vecteur& res) const ;
429     // résolution directe par la méthode de cramer (pour les petits systèmes dim 1,2,3)
430     void Cramer( const Vecteur& b,Vecteur& res) const;
431     // idem pour un tableau de vecteur
432     void Cramer( const Tableau <Vecteur>& b,Tableau <Vecteur>& res) const;
433     // symétrisation de la matrice (il faut qu'elle soit carrée
434     void Symetrisation();
435
436 };
437 /// @} // end of group
438
439
440 inline Vecteur operator* (const Vecteur& vec,const Mat_pleine& mat_pl)
441 // Permet de realiser la multiplication entre un vecteur et une matrice
442 { return mat_pl.Prod_vec_mat((Vecteur&) vec);
443 };
444
445 inline Mat_pleine operator* (const double coeff,const Mat_pleine& mat_pl)
446 // Permet de realiser la multiplication entre un scalaire et une matrice
447 { return (mat_pl*coeff);
448 };
449
450 #include "Mat_abstraite.h"
451
452 #ifndef MISE_AU_POINT
453 #include "Mat_pleine.cc"
454 #define Mat_pleine_H_deja_inclus
455 #endif
456
457
458 #endif

```

## 7.379 MatBand.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //

```

```

20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: definition de matrices bandes symetriques a valeurs reelles.*
39 *   Seule la partie inferieure est stockee.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *****/
54 #ifndef MATBAND_H
55 #define MATBAND_H
56
57 // #include "Debug.h"
58 #include <iostream>
59 #include "Mat_abstraite.h"
60 #include "MathUtil.h"
61
62
63 /// @addtogroup Les_classes_Matrices
64 /// @{
65 ///
66
67 class MatBand : public Mat_abstraite
68 {
69     // surcharge de l'operator de lecture typée
70     friend istream & operator » (istream &, MatBand &);
71     // surcharge de l'operator d'écriture typée
72     friend ostream & operator « (ostream &, const MatBand &);
73 public :
74     // CONSTRUCTEURS :
75     MatBand (); // par default
76     MatBand (Enum_matrice type_mat , int lb, int dim ); // def d'une matrice bande
77     MatBand (Enum_matrice type_mat, int lb, int dim , double a); // def d'une matrice bande avec
78     // initialisation des composantes a la valeur a
79     MatBand (const MatBand& m) ; // de copie, il y a creation d'une deuxieme matrice
80
81     // DESTRUCTEUR :
82     ~MatBand ();
83
84     // fonction permettant de creer une nouvelle instance d'element
85     Mat_abstraite * NouvelElement() const;
86
87     // METHODES PUBLIQUES :
88
89     // surcharge de l'opérateur d'affectation : cas de matrices abstraites
90     Mat_abstraite & operator = ( const Mat_abstraite &);
91
92     // transfert des informations de *this dans la matrice passée en paramètre
93     // la matrice paramètre est au préalable, mise à 0.
94     void Transfert_vers_mat( Mat_abstraite & A );
95     // surcharge de l'opérateur d'affectation : cas de matrices bandes
96     MatBand & operator = ( const MatBand &);
97
98     //-----
99     // --- plusieurs fonctions virtuelles qui agissent en général sur la matrice -----
100    //-----
101    // ici l'idée est d'éviter de construire une nouvelle matrice pour des questions
102    // d'encombrement, c'est surtout des méthodes utiles pour le stockage de matrice
103    // de raideur. On met le nombre mini de fonction, pour pouvoir faire des expressions.
104    // Cependant aucune de ces fonctions n'est en désaccord avec les fonctions membres
105    // qui crée une matrice en résultat (cf. Mat_pleine)

```

```

106 // pour les 2 premières méthodes : += et -= :
107 // les matrices arguments sont en général du mêmes type que celui de la matrice this
108 // mais cela fonctionne également avec comme argument une matrice diagonale (pour tous les types)
109
110 // Surcharge de l'opérateur += : addition d'une matrice a la matrice courante
111 void operator+= (const Mat_abstraite& mat_pl);
112
113 // Surcharge de l'opérateur -= : soustraction d'une matrice a la matrice courante
114 void operator-= (const Mat_abstraite& mat_pl);
115
116 // Surcharge de l'opérateur *= : multiplication de la matrice courante par un scalaire
117 void operator*= (const double r);
118
119 //-----
120 // --- fin de plusieurs fonctions virtuelles qui agissent en général sur la matrice --
121 //-----
122
123 // Surcharge de l'opérateur == : test d'egalite entre deux matrices
124 int operator== (const Mat_abstraite& mat_pl) const ;
125
126 // acces aux composantes comme pour une matrice carree
127 // l'utilisateur doit gerer les depassements de taille
128 // il y a un message en debuggage
129 double& operator ()(int i, int j );
130
131 // ramène true si la place (i,j) existe, false sinon
132 // ici on ne test pas le fait que i et j puissent être négatif ou pas
133 inline bool Existe(int i, int j) const {return ((abs(i-j)-1b) < 0);};
134
135 // acces en lecture seul, aux composantes comme pour une matrice carree
136 // l'utilisateur doit gerer les depassements de taille
137 // il y a un message en debuggage
138 // si l'indice est possible (c'est à dire >0 et < à la dimension) mais
139 // hors la bande -> retour 0
140 double operator () (int i, int j ) const ;
141
142 // Retourne la ieme ligne de la matrice
143 // pas util dans le cas des bande donc
144 // non implemente
145 Vecteur& Ligne_set(int i)
146 { cout << " fonction non implementee : "
147   << " Vecteur& MatBand::operator() (int i) " << endl;
148   Sortie(1);
149   i = i; // pour ne pas avoir de message a la compilation !!
150   Vecteur* toto; toto = new Vecteur();
151   return *toto;
152 };
153
154 // Retourne la ieme ligne de la matrice uniquement en lecture
155 // pas utile dans le cas des bandes donc
156 // non implementee
157 Vecteur operator() (int i) const
158 { cout << " fonction non implementee : "
159   << " Vecteur& MatBand::operator() (int i) " << endl;
160   i = i; // pour ne pas avoir de message a la compilation !!
161   return Vecteur(0);
162 };
163
164 // Retourne la ieme ligne de la matrice
165 Vecteur Ligne(int i) const ;
166
167 // Retourne la ieme ligne de la matrice
168 // sous le format de stokage propre a la matrice
169 // donc a n'utiliser que comme sauvegarde en parralele
170 // avec la fonction RemplaceLigne
171 Vecteur LigneSpe(int i) const ;
172 // remplace la ligne de la matrice par la ligne fournie
173 void RemplaceLigneSpe(int i,const Vecteur & v);
174
175 //met une valeur identique sur toute la ligne
176 void MetValLigne(int i,double val);
177
178 // Retourne la jeme colonne de la matrice
179 Vecteur Colonne(int j) const ;
180
181 // Retourne la jeme colonne de la matrice
182 // sous le format de stokage propre a la matrice
183 // donc a n'utiliser que comme sauvegarde en parralele
184 // avec la fonction RemplaceColonne
185 Vecteur ColonneSpe(int j) const ;
186 // remplace la Colonne de la matrice par la ligne fournie
187 void RemplaceColonneSpe(int j, const Vecteur & v);
188 //met une valeur identique sur toute la colonne
189
190 void MetValColonne(int j,double val);
191
192 // Affichage des valeurs de la matrice

```

```

193 // uniquement le valeurs de la bande inferieur
194 void Affiche() const ;
195 // Affiche une partie de la matrice (util pour le debug)
196 // min_ et max_ sont les bornes de la sous_matrice
197 // pas_ indique le pas en i et j pour les indices
198 void Affiche1(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j) const ;
199 // Affiche une partie de la matrice idem si dessus
200 // mais avec un nombre de digit (>7) = nd
201 // si < 7 ne fait rien
202 void Affiche2(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j,int nd) const ;
203
204 void Change_taille(int lb, int dim ); // changement de la taille de la matrice
205 inline int Nb_colonne () const{ return lb;} ; // retour de la largeur de bande
206 inline int Nb_ligne() const { return dim;} ; // retour de la dimension de la matrice
207 void Initialise (double a); // initialisation de la matrice a la valeur "a"
208 void Libere () ; // Liberation de la place memoire
209 int Symetrie () const { return 1; } // la matrice est Symetrique
210
211 // Resolution du systeme Ax=b
212 // la verification de taille n'est faite que pour le debug
213 //1) avec en sortie un new vecteur
214 Vecteur Resol_syst ( const Vecteur& b,const double &tol = tol_defaut
215 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
216 //2) avec en sortie le vecteur d'entree
217 Vecteur& Resol_systID ( Vecteur& b,const double &tol = tol_defaut
218 ,const int maxit = maxit_defaut,const int restart = restart_defaut);
219 //3) avec en entree un tableau de vecteur second membre et
220 // en sortie un nouveau tableau de vecteurs
221 Tableau <Vecteur> Resol_syst (const Tableau <Vecteur>& b,const double &tol = tol_defaut
222 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
223 //4) avec en entree un tableau de vecteur second membre et
224 // en sortie le tableau de vecteurs d'entree
225 Tableau <Vecteur>& Resol_systID (Tableau <Vecteur>& b,const double &tol = tol_defaut
226 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
227 //5) avec en sortie le dernier vecteur d'entree, le premier etant le second membre
228 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
229 // deux étant identiques
230 Vecteur& Resol_systID_2 (const Vecteur& b,Vecteur& vortie
231 , const double &tol = tol_defaut,const int maxit = maxit_defaut
232 ,const int restart = restart_defaut);
233
234 // ===== RÉOLUTION EN DEUX TEMPS ===== :
235 // 1) préparation de la matrice donc modification de la matrice éventuellement
236 // par exemple pour les matrices bandes avec cholesky : triangulation
237 void Preparation_resol();
238 // 2) *** résolution sans modification de la matrice DOIT ÊTRE PRÉCÉDÉ DE L'APPEL DE
239 // Preparation_resol
240 // a) avec en sortie un new vecteur
241 Vecteur Simple_Resol_syst (const Vecteur& b,const double &tol = tol_defaut
242 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
243 // b) avec en sortie le vecteur d'entree
244 Vecteur& Simple_Resol_systID (Vecteur& b,const double &tol = tol_defaut
245 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
246 // c) avec en entree un tableau de vecteur second membre et
247 // en sortie un nouveau tableau de vecteurs
248 Tableau <Vecteur> Simple_Resol_syst
249 (const Tableau <Vecteur>& b,const double &tol = tol_defaut
250 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
251 // d) avec en entree un tableau de vecteur second membre et
252 // en sortie le tableau de vecteurs d'entree
253 Tableau <Vecteur>& Simple_Resol_systID
254 (Tableau <Vecteur>& b,const double &tol = tol_defaut
255 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
256 // e) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
257 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
258 // deux étant identiques
259 Vecteur& Simple_Resol_systID_2 (const Vecteur& b,Vecteur& vortie
260 , const double &tol = tol_defaut
261 ,const int maxit = maxit_defaut
262 ,const int restart = restart_defaut) const ;
263 // ===== FIN RÉOLUTION EN DEUX TEMPS ===== :
264
265 // Multiplication d'un vecteur par une matrice ( (vec)t * A )
266 Vecteur Prod_vec_mat ( const Vecteur& vec) const ;
267 // idem mais on utilise la place du second vecteur pour le résultat
268 Vecteur& Prod_vec_mat ( const Vecteur& vec, Vecteur & resul) const ;
269 // Multiplication d'une matrice par un vecteur ( A * vec )
270 Vecteur Prod_mat_vec ( const Vecteur& vec) const ;
271 // idem mais on utilise la place du second vecteur pour le résultat
272 Vecteur& Prod_mat_vec ( const Vecteur& vec, Vecteur & resul) const ;
273 // Multiplication d'une ligne iligne de la matrice avec un vecteur de
274 // dimension = le nombre de colonne de la matrice
275 double Prod_Ligne_vec ( int iligne, const Vecteur& vec) const ;
276 // Multiplication d'un vecteur avec une colonne icol de la matrice
277 // dimension = le nombre de ligne de la matrice
278 double Prod_vec_col( int icol, const Vecteur& vec) const ;
279

```

```

280 // calcul du produit : (vec_1)^T * A * (vect_2)
281 double vectT_mat_vec(const Vecteur& vec1, const Vecteur& vec2) const ;
282
283 // retourne la place que prend la matrice en entier
284 int Place() const { return (dim * lb );};
285
286 // ++++ méthodes spécifique à la classe
287 // Resolution du systeme Ax=b, sans triangulation, c'est-à-dire que l'on
288 // considère que la matrice stocke actuellement la triangulation
289 Vecteur& Resol_systID_sans_triangu ( Vecteur& b);
290
291
292 protected :
293
294 // VARIABLES PROTEGEES :
295 double * pt; // pointeur de la matrice
296 int lb; // largeur de la bande
297 int dim; // dimension de la matrice
298
299 // METHODES PROTEGEES :
300 //reduction de la matrice de raideur avant resolution
301 void REDUCT (MatBand & TRAIID,int LB,int NDDL,Vecteur& CC);
302 // resolution
303 void RESOLT(const MatBand & TRAIID,Vecteur & TSM,int NDDL,int LB) const;
304 // enchainement de la resolution avec controle des pivots
305 void ResoBand (MatBand & TRAIID,int LB,int NDDL,Vecteur& CC);
306 // idem précédent mais avec plusieurs seconds membres
307 void ResoBand (MatBand & TRAIID,int LB,int NDDL,Tableau <Vecteur>& CC);
308 // acces aux coordonnées bandes : utilisés uniquement pour la résolution:
309 // de reduct et rosolt
310 // l'adressage est différent de celui de la surcharge de l'opérateur ()
311 // donc à ne pas utiliser autre part !!
312 inline double& c(int ii, int jj )
313 {
314     #ifdef MISE_AU_POINT
315     if (ii>lb || jj>dim )
316     {cout << "erreur d acces aux composantes bande";
317     cout << " ii = " << ii << " jj = " << jj << '\n';
318     cout << "double& c(int ii, int jj )" << endl;
319     }
320     #endif
321     return pt[(jj-1)*lb+ii-1];
322 };
323 // idem comme acces aux coordonnées bandes mais en constante
324 inline const double& cConst(int ii, int jj ) const
325 {
326     #ifdef MISE_AU_POINT
327     if (ii>lb || jj>dim )
328     {cout << "erreur d acces aux composantes bande";
329     cout << " ii = " << ii << " jj = " << jj << '\n';
330     cout << "double& cConst(int ii, int jj )" << endl;
331     }
332     #endif
333     return pt[(jj-1)*lb+ii-1];
334 };
335 };
336 /// @} // end of group
337
338 #endif

```

## 7.380 MatDiag.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.

```



```

24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      30/09/2001
32 *
33 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: definition de matrices diagonales a valeurs reelles.
39 *   On ne stocke que la diagonale.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *****/
54 #ifndef MATDIAG_H
55 #define MATDIAG_H
56
57 // #include "Debug.h"
58 #include <iostream>
59 #include "Mat_abstraite.h"
60 #include "MathUtil.h"
61
62 /// @addtogroup Les_classes_Matrices
63 /// @{
64 ///
65
66
67 class MatDiag : public Mat_abstraite
68 {
69     // surcharge de l'operator de lecture typée
70     friend istream & operator » (istream &, MatDiag &);
71     // surcharge de l'operator d'écriture typée
72     friend ostream & operator « (ostream &, const MatDiag &);
73 public :
74     // CONSTRUCTEURS :
75     MatDiag (); // par défaut
76     MatDiag ( int dim ); // def d'une matrice diagonale d'une taille donnée
77     MatDiag (int dim , double a); // def d'une matrice diagonale avec
78     // initialisation des composantes a la valeur a
79     MatDiag (const MatDiag& m) ; // de copie, il y a creation d'une deuxieme matrice
80
81     // DESTRUCTEUR :
82     ~MatDiag ();
83
84     // fonction permettant de creer une nouvelle instance d'element
85     Mat_abstraite * NouvelElement() const;
86
87     // METHODES PUBLIQUES :
88
89     // surcharge de l'opérateur d'affectation : cas de matrices abstraites
90     Mat_abstraite & operator = ( const Mat_abstraite &);
91
92     // transfert des informations de *this dans la matrice passée en paramètre
93     // la matrice paramètre est au préalable, mise à 0.
94     void Transfert_vers_mat ( Mat_abstraite & A );
95
96     // surcharge de l'opérateur d'affectation : cas de matrices diagonales
97     MatDiag & operator = ( const MatDiag &);
98     // surcharge de l'opérateur d'affectation avec un vecteur
99     MatDiag & operator = ( const Vecteur &);
100
101     //-----
102     // --- plusieurs fonctions virtuelles qui agissent en général sur la matrice -----
103     //-----
104     // ici l'idée est d'éviter de construire une nouvelle matrice pour des questions
105     // d'encombrement, c'est surtout des méthodes utiles pour le stockage de matrice
106     // de raideur. On met le nombre mini de fonction, pour pouvoir faire des expressions.
107     // Cependant aucune de ces fonctions n'est en désaccord avec les fonctions membres
108     // qui crée une matrice en résultat (cf. Mat_pleine)
109

```

```

110 // Surcharge de l'operateur += : addition d'une matrice a la matrice courante
111 void operator+= (const Mat_abstraite& mat_pl);
112
113 // Surcharge de l'operateur -= : soustraction d'une matrice a la matrice courante
114 void operator-= (const Mat_abstraite& mat_pl);
115
116 // Surcharge de l'operateur *= : multiplication de la matrice courante par un scalaire
117 void operator*= (const double r);
118
119 //-----
120 // --- fin de plusieurs fonctions virtuelles qui agissent en général sur la matrice ---
121 //-----
122
123 // Surcharge de l'operateur == : test d'egalite entre deux matrices
124 int operator==(const Mat_abstraite& mat_pl) const ;
125
126 // acces aux composantes comme pour une matrice carree
127 // l'utilisateur doit gerer les depassements de taille
128 // il y a un message en debuggage
129 // la première fonction est en fait a remplacer par set_element(i,j)
130 // quand c'est possible
131 double& operator ()(int i, int j );
132
133 // ramène true si la place (i,j) existe, false sinon
134 // ici on ne test pas le fait que i et j puissent être négatif ou pas
135 inline bool Existe(int i, int j) const {return (i==j);};
136
137 // acces en lecture seul, aux composantes comme pour une matrice carree
138 // l'utilisateur doit gerer les depassements de taille
139 // il y a un message en debuggage
140 // si l'indice est possible (c'est à dire >0 et < à la dimension) mais
141 // hors de la diagonale -> retour 0
142 double operator () (int i, int j ) const ;
143 // modification d'une valeur
144 // on impose la valeur val à la position (i,j) en fait ici seul la position
145 // i=j est possible
146 void set_element(int i, int j, double val);
147
148 // Retourne la ieme ligne de la matrice
149 // pas util dans le cas des matrices diagonales donc
150 // non implemente
151 Vecteur& Ligne_set(int i)
152 { cout <<" fonction non implementee : "
153     << " Vecteur& MatDiag::operator() (int i) " << endl;
154   Sortie(1);
155   i = i; // pour ne pas avoir de message a la compilation !!
156   Vecteur* toto = new Vecteur();
157   return *toto;
158 };
159
160 // Retourne la ieme ligne de la matrice uniquement en lecture
161 // pas util dans le cas des matrices diagonales donc
162 // non implemente
163 Vecteur operator() (int i) const
164 { cout <<" fonction non implementee : "
165     << " Vecteur& MatDiag::operator() (int i) " << endl;
166   Sortie(1);
167   i = i; // pour ne pas avoir de message a la compilation !!
168   return Vecteur(0);
169 };
170
171 // Retourne la ieme ligne de la matrice
172 // on considère une matrice carrée équivalente
173 Vecteur Ligne(int i) const ;
174
175 // Retourne la ieme ligne de la matrice
176 // sous le format de stockage propre a la matrice
177 // donc a n'utiliser que comme sauvegarde en parralele
178 // avec la fonction RemplaceLigne
179 Vecteur LigneSpe(int i) const ;
180 // remplace la ligne de la matrice par la ligne fournie
181 void RemplaceLigneSpe(int i,const Vecteur & v);
182
183 //met une valeur identique sur toute la ligne
184 void MetVallLigne(int i,double val);
185
186 // Retourne la jeme colonne de la matrice
187 // on considère une matrice carrée associée
188 Vecteur Colonne(int j) const ;
189
190 // Retourne la jeme colonne de la matrice
191 // sous le format de stockage propre a la matrice
192 // donc a n'utiliser que comme sauvegarde en parralele
193 // avec la fonction RemplaceColonne
194 Vecteur ColonneSpe(int j) const ;
195 // remplace la Colonne de la matrice par la ligne fournie
196 void RemplaceColonneSpe(int j, const Vecteur & v);

```

```

197 //met une valeur identique sur toute la colonne
198
199 void MetValColonne(int j,double val);
200
201 // Affichage des valeurs de la matrice
202 // uniquement le valeurs de la diagonale
203 void Affiche () const ;
204 // Affiche une partie de la matrice (util pour le debug)
205 // min_ et max_ sont les bornes de la sous_matrice
206 // pas_ indique le pas en i et j pour les indices
207 void Affiche1(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j) const ;
208 // Affiche une partie de la matrice idem si dessus
209 // mais avec un nombre de digit (>7) = nd
210 // si < 7 ne fait rien
211 void Affiche2(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j,int nd) const ;
212
213 void Change_taille(int dim ); // changement de la taille de la matrice
214 inline int Nb_colonne () const{ return 1;} ;
215 inline int Nb_ligne() const { return diago.Taille();}; // retour de la dimension de la matrice
216 void Initialise (double a); // initialisation de la matrice a la valeur "a"
217 void Libere () ; // Liberation de la place memoire
218 int Symetrie () const { return 1; } // la matrice est Symetrique
219
220 // Resolution du systeme Ax=b
221 // la verification de taille n'est faite que pour le debug
222 //1) avec en sortie un new vecteur
223 Vecteur Resol_syst ( const Vecteur& b,const double &tol = tol_defaut
224 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
225 //2) avec en sortie le vecteur d'entree
226 Vecteur& Resol_systID ( Vecteur& b,const double &tol = tol_defaut
227 ,const int maxit = maxit_defaut,const int restart = restart_defaut);
228 //3) avec en entrée un tableau de vecteur second membre et
229 // en sortie un nouveau tableau de vecteurs
230 Tableau <Vecteur> Resol_syst (const Tableau <Vecteur>& b,const double &tol = tol_defaut
231 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
232 //4) avec en entrée un tableau de vecteur second membre et
233 // en sortie le tableau de vecteurs d'entree
234 Tableau <Vecteur>& Resol_systID (Tableau <Vecteur>& b,const double &tol = tol_defaut
235 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
236 //5) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
237 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
238 // deux étant identiques
239 Vecteur& Resol_systID_2 (const Vecteur& b,Vecteur& vortie
240 ,const double &tol = tol_defaut,const int maxit = maxit_defaut
241 ,const int restart = restart_defaut);
242
243 // ===== RÉOLUTION EN DEUX TEMPS ===== :
244 // 1) préparation de la matrice donc modification de la matrice éventuellement
245 // par exemple pour les matrices bandes avec cholesky : triangulation
246 void Preparation_resol();
247 // 2) *** résolution sans modification de la matrice DOIT ÊTRE PRÉCÉDÉ DE L'APPEL DE
248 // Preparation_resol
249 // a) avec en sortie un new vecteur
250 Vecteur Simple_Resol_syst (const Vecteur& b,const double &tol = tol_defaut
251 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
252 // b) avec en sortie le vecteur d'entree
253 Vecteur& Simple_Resol_systID (Vecteur& b,const double &tol = tol_defaut
254 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
255 // c) avec en entrée un tableau de vecteur second membre et
256 // en sortie un nouveau tableau de vecteurs
257 Tableau <Vecteur> Simple_Resol_syst
258 (const Tableau <Vecteur>& b,const double &tol = tol_defaut
259 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
260 // d) avec en entrée un tableau de vecteur second membre et
261 // en sortie le tableau de vecteurs d'entree
262 Tableau <Vecteur>& Simple_Resol_systID
263 (Tableau <Vecteur>& b,const double &tol = tol_defaut
264 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
265 // e) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
266 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
267 // deux étant identiques
268 Vecteur& Simple_Resol_systID_2 (const Vecteur& b,Vecteur& vortie
269 ,const double &tol = tol_defaut
270 ,const int maxit = maxit_defaut
271 ,const int restart = restart_defaut) const ;
272 // ===== FIN RÉOLUTION EN DEUX TEMPS ===== :
273
274 // Multiplication d'un vecteur par une matrice ( (vec)t * A )
275 Vecteur Prod_vec_mat ( const Vecteur& vec) const ;
276 // idem mais on utilise la place du second vecteur pour le résultat
277 Vecteur& Prod_vec_mat ( const Vecteur& vec, Vecteur & resul) const ;
278 // Multiplication d'une matrice par un vecteur ( A * vec )
279 Vecteur Prod_mat_vec ( const Vecteur& vec) const ;
280 // idem mais on utilise la place du second vecteur pour le résultat
281 Vecteur& Prod_mat_vec ( const Vecteur& vec, Vecteur & resul) const ;
282 // Multiplication d'une ligne iligne de la matrice avec un vecteur de
283 // dimension = le nombre de colonne de la matrice

```

```

284 double Prod_Ligne_vec ( int iligne, const Vecteur& vec) const ;
285     // Multiplication d'un vecteur avec une colonne icol de la matrice
286     // dimension = le nombre de ligne de la matrice
287 double Prod_vec_col( int icol, const Vecteur& vec) const ;
288
289 // calcul du produit : (vec_1)^T * A * (vect_2)
290 double vectT_mat_vec(const Vecteur& vec1, const Vecteur& vec2) const ;
291
292 // retourne la place que prend la matrice en entier
293 int Place() const { return (diago.Taille() * 2);};
294
295 // ===== fonction spécifique des matrices diagonales =====
296     // surcharge d'opérateur += avec un vecteur
297     MatDiag & operator += ( const Vecteur &);
298     // surcharge d'opérateur -= avec un vecteur
299     MatDiag & operator -= ( const Vecteur &);
300     // ramène un conteneur vecteur contenant la diagonale de la matrice
301     const Vecteur& Vecteur_MatDiag() const {return diago;};
302
303 protected :
304
305     // VARIABLES PROTEGEES :
306     Vecteur diago; // la diagonale
307     Vecteur inv_diago; // l'inverse de la diagonale
308     bool inversion; // dit si l'inversion est utilisable ou pas
309
310     // METHODES PROTEGEES :
311     // enchainement de la resolution
312     void ResoDiag (const Vecteur& BB,Vecteur& CC);
313     // idem précédent mais avec plusieurs seconds membres
314     void ResoDiag (const Tableau <Vecteur>& BB,Tableau <Vecteur>& CC);
315     // resolution seule sans modification de la matrice
316     void Const_ResoDiag (const Vecteur& BB,Vecteur& CC) const ;
317     // idem précédent mais avec plusieurs seconds membres
318     void Const_ResoDiag (const Tableau <Vecteur>& BB,Tableau <Vecteur>& CC) const ;
319     // affichage d'un élément quelconque -> 0 si diff de la diagonale
320     double Affiche_elem(int i,int j) const { if ((i==j)&&(i!=0)&&(i<=diago.Taille())) return diago(i);
321         else return 0.;};
322 };
323 /// @} // end of group
324
325 #endif

```

## 7.381 Mat\_creuse\_CompCol.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: definition de matrices creuses a valeurs reelles.
39 *   Le stockage est quelconque (pas symétrique par exemple).
40 *   Ici le stockage est de type en colonne compressée.

```

```
41 *                                     $ *
42 *      *
43 *      VERIFICATION: *
44 *      *
45 *      ! date !   auteur !           but           ! *
46 *      ----- *
47 *      !           !           !           !           ! *
48 *      *                                     $ *
49 *      *
50 *      MODIFICATIONS: *
51 *      ! date !   auteur !           but           ! *
52 *      ----- *
53 *      *                                     $ *
54 *      *****/
55 #ifndef MAT_CREUSE_COMP COL_H
56 #define MAT_CREUSE_COMP COL_H
57
58 // #include "Debug.h"
59 #include <iostream>
60 #include "Mat_abstraite.h"
61 #include "MathUtil.h"
62
63 #include "compcol_double.h"
64 #include "compro_w_double.h"
65 #include "coord_double.h"
66
67 #include "Vecteur.h"
68 #include "Tableau2_T.h"
69
70 /// @addtogroup Les_classes_Matrices
71 /// @{
72 ///
73
74
75 class Mat_creuse_CompCol : public Mat_abstraite , public CompCol_Mat_double
76 { // surcharge de l'operator de lecture typée
77     friend istream & operator » (istream &, Mat_creuse_CompCol &);
78     // surcharge de l'operator d'écriture typée
79     friend ostream & operator « (ostream &, const Mat_creuse_CompCol &);
80 public :
81     // CONSTRUCTEURS :
82     Mat_creuse_CompCol (); // par défaut
83
84     // constructeur permettant la création d'une matrice creuse
85     // ici aucun élément de la matrice n'est créé, il faut ensuite utiliser l'initialisation
86     // pour créer les éléments non nulles dans la matrice
87     // M : nombre de lignes
88     // N : nombre de colonnes
89     //
90     Mat_creuse_CompCol(int M, int N);
91
92     // constructeur permettant la création d'une matrice creuse complete
93     // à partir de la donnée d'un ensemble de sous matrices carrées
94     // les petites matrices sont carrées, ils sont définis par un tableau ligne qui contient
95     // la position d'un terme de la petite matrice dans la grande matrice
96     // exemple : pet_mat(k)(i), pet_mat(k)(j) : indique le numéro de ligne et le numéro de colonne
97     // d'un élément non nul dans la grande matrice
98     // M : nombre de lignes
99     // N : nombre de colonnes
100    //
101    Mat_creuse_CompCol(int M, int N, const Tableau < Tableau <int> >& petites_matrices);
102
103    // de copie à partir d'une même instance
104    Mat_creuse_CompCol(const Mat_creuse_CompCol &S) ;
105
106    // de copie à partir d'une sparse matrice compressée par ligne
107    // Mat_creuse_CompCol(const Mat_creuse_CompRow &R) ;
108    // de copie à partir d'une sparse matrice non compressée
109    // Mat_creuse_CompCol(const Mat_creuse_Coord &CO) ;
110
111    // DESTRUCTEUR :
112    ~Mat_creuse_CompCol () {} ;
113
114
115    // fonction permettant de créer une nouvelle instance d'élément du même type
116    Mat_abstraite * NouvelElement() const;
117
118    // METHODES PUBLIQUES :
119
120    // surcharge de l'opérateur d'affectation : cas de matrices abstraites
121    Mat_abstraite & operator = ( const Mat_abstraite &);
122
123    // transfert des informations de *this dans la matrice passée en paramètre
124    // la matrice paramètre est au préalable, mise à 0.
125    void Transfert_vers_mat( Mat_abstraite & A );
126
```

```

127     // surcharge de l'opérateur d'affectation : cas de matrices creuses
128     Mat_creuse_CompCol & operator = ( const Mat_creuse_CompCol &);
129
130     //-----
131     // --- plusieurs fonctions virtuelles qui agissent en général sur la matrice -----
132     //-----
133     // ici l'idée est d'éviter de construire une nouvelle matrice pour des questions
134     // d'encombrement, c'est surtout des méthodes utiles pour le stockage de matrice
135     // de raideur. On met le nombre mini de fonction, pour pouvoir faire des expressions.
136     // Cependant aucune de ces fonctions n'est en désaccord avec les fonctions membres
137     // qui crée une matrice en résultat (cf. Mat_pleine)
138
139     // Surcharge de l'opérateur += : addition d'une matrice a la matrice courante
140     // cette méthode n'est possible si et seulement les termes existant de mat_pl
141     void operator+= (const Mat_abstraite& mat_pl);
142
143     // Surcharge de l'opérateur -= : soustraction d'une matrice a la matrice courante
144     // cette méthode n'est possible si et seulement les termes existant de mat_pl
145     void operator-= (const Mat_abstraite& mat_pl);
146
147     // Surcharge de l'opérateur *= : multiplication de la matrice courante par un scalaire
148     void operator*= (const double r);
149
150     //-----
151     // --- fin de plusieurs fonctions virtuelles qui agissent en général sur la matrice --
152     //-----
153
154     // Surcharge de l'opérateur == : test d'égalite entre deux matrices
155     int operator== (const Mat_abstraite& mat_pl) const ;
156
157     // acces aux composantes comme pour une matrice carree
158     // il y a un message en cas de mauvaises valeurs
159     double& operator () (int i, int j )
160     { return CompCol_Mat_double::set(i-1,j-1);};
161
162
163     // ramène true si la place (i,j) existe, false sinon
164     // ici on ne test pas le fait que i et j puissent être négatif ou pas
165     inline bool Existe(int i, int j) const
166     {return Exista(i,j);};
167
168
169     // acces en lecture seul, aux composantes comme pour une matrice carree
170     // il y a un message en cas de mauvais indice
171     // si les indices sont possibles mais que l'élément n'existe pas retour 0
172     double operator () (int i, int j ) const
173     {return CompCol_Mat_double::operator () (i-1,j-1);};
174
175     // Retourne la ieme ligne de la matrice en lecture / écriture
176     // pas util dans le cas des matrices creuses donc
177     // non implemente
178     Vecteur& Ligne_set (int i)
179     { cout << " fonction non implementee : "
180       << " Vecteur& Mat_creuse_CompCol::operator() (int i) " << endl;
181       Sortie(1);
182       i = i; // pour ne pas avoir de message a la compilation !!
183       Vecteur* toto = new Vecteur();
184       return *toto;
185     };
186
187     // Retourne la ieme ligne de la matrice uniquement en lecture
188     // pas util dans le cas des matrices creuses donc
189     // non implemente
190     Vecteur operator() (int i) const
191     { cout << " fonction non implementee : "
192       << " Vecteur& Mat_creuse_CompCol::operator() (int i) " << endl;
193       i = i; // pour ne pas avoir de message a la compilation !!
194       return Vecteur(0);
195     };
196
197     // Retourne la ieme ligne de la matrice (lent !!!)
198     Vecteur Ligne(int i) const ;
199
200     // Retourne la ieme ligne de la matrice (lent !!!)
201     // sous le format de stokage propre a la matrice
202     // donc a n'utiliser que comme sauvegarde en parralele
203     // avec la fonction RemplaceLigne
204     Vecteur LigneSpe(int i) const ;
205     // remplace la ligne de la matrice par la ligne fournie (lent !!!)
206     void RemplaceLigneSpe(int i,const Vecteur & v);
207
208     //met une valeur identique sur toute la ligne (lent !!!)
209     void MetValLigne(int i,double val);
210
211     // Retourne la jeme colonne de la matrice (rapide)
212     Vecteur Colonne(int j) const ;
213

```

```

214 // Retourne la jeme colonne de la matrice (rapide)
215 // sous le format de stokage propre a la matrice
216 // donc a n'utiliser que comme sauvegarde en parrallele
217 // avec la fonction RemplaceColonne
218 Vecteur ColonneSpe(int j) const ;
219 // remplace la Colonne de la matrice par la ligne fournie (rapide)
220 void RemplaceColonneSpe(int j, const Vecteur & v);
221 //met une valeur identique sur toute la colonne (rapide)
222 void MetValColonne(int j,double val);
223
224 // Affichage des valeurs de la matrice
225 void Affiche () const ;
226 // Affiche une partie de la matrice (util pour le debug)
227 // min_ et max_ sont les bornes de la sous_matrice
228 // pas_ indique le pas en i et j pour les indices
229 void Affiche1(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j) const ;
230 // Affiche une partie de la matrice idem si dessus
231 // mais avec un nombre de digit (>7) = nd
232 // si < 7 ne fait rien
233 void Affiche2(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j,int nd) const ;
234
235 // changement de la taille de la matrice : c-a-d permet de redimensionner
236 // la matrice creuse
237 // 1) cas d'un utilisateur qui connait un ensemble de petite matrice qui doivent être contenue
238 // dans la matrice creuse, la matrice initiale est effacée
239 // les petites matrices sont carrées, ils sont définit par un tableau ligne qui contiend
240 // la position d'un terme de la petite matrice dans la grande matrice
241 // exemple : pet_mat(k)(i), pet_mat(k)(j) : indique le numéro de ligne et le numéro de colonne
242 // d'un élément non nul dans la grande matrice
243 // M et N : nb ligne et nb colonne,
244 void Change_taille(const int M, const int N ,const Tableau < Tableau <int> >& petites_matrices);
245
246 // 2) cas d'un utilisateur qui connait le stockage par colonne
247 // M et N : nb ligne et nb colonne,
248 // pointeur_colonne : donne dans le vecteur de stockage global le début de chaque colonne
249 // nz : nb de termes non nulles dans la matrice
250 void Change_taille(const int M, const int N ,const Tableau<int>& pointeur_colonne,const int nz =
0);
251
252 // ramène le nombre de colonne
253 inline int Nb_colonne () const { return CompCol_Mat_double::dim(1); } ;
254 // ramène le nombre de ligne
255 inline int Nb_ligne() const { return CompCol_Mat_double::dim(0); };
256 // initialisation toutes les valeurs de la matrice a la valeur "a"
257 void Initialise (double a);
258 // Liberation de la place memoire
259 void Libere () ;
260 // test si la matrice est symétrique ou pas
261 // ici pour l'instant on considère qu'elle ne l'est pas
262 int Symetrie () const { return 0; }
263
264 // Resolution du systeme Ax=b
265 // la verification de taille n'est faite que pour le debug
266 //1) avec en sortie un new vecteur
267 Vecteur Resol_syst ( const Vecteur& b,const double &tol = tol_defaut
268 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
269 //2) avec en sortie le vecteur d'entree
270 Vecteur& Resol_systID ( Vecteur& b,const double &tol = tol_defaut
271 ,const int maxit = maxit_defaut,const int restart = restart_defaut);
272 //3) avec en entrée un tableau de vecteur second membre et
273 // en sortie un nouveau tableau de vecteurs
274 Tableau <Vecteur> Resol_syst
275 (const Tableau <Vecteur>& b,const double &tol = tol_defaut
276 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
277 //4) avec en entrée un tableau de vecteur second membre et
278 // en sortie le tableau de vecteurs d'entree
279 Tableau <Vecteur>& Resol_systID
280 (Tableau <Vecteur>& b,const double &tol = tol_defaut
281 ,const int maxit = maxit_defaut,const int restart = restart_defaut) ;
282 //5) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
283 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
284 // deux étant identiques
285 Vecteur& Resol_systID_2 (const Vecteur& b,Vecteur& vortie
286 , const double &tol = tol_defaut,const int maxit = maxit_defaut
287 ,const int restart = restart_defaut);
288
289 // ===== RÉOLUTION EN DEUX TEMPS ===== :
290 // 1) préparation de la matrice donc modification de la matrice éventuellement
291 // par exemple pour les matrices bandes avec cholesky : triangulation
292 void Preparation_resol();
293 // 2) *** résolution sans modification de la matrice DOIT ÊTRE PRÉCÉDÉ DE L'APPEL DE
294 // Preparation_resol
295 // a) avec en sortie un new vecteur
296 Vecteur Simple_Resol_syst (const Vecteur& b,const double &tol = tol_defaut
297 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
298 // b) avec en sortie le vecteur d'entree
299 Vecteur& Simple_Resol_systID (Vecteur& b,const double &tol = tol_defaut

```

```

300         ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
301         // c) avec en entrée un tableau de vecteur second membre et
302         //      en sortie un nouveau tableau de vecteurs
303     Tableau <Vecteur> Simple_Resol_syst
304         (const Tableau <Vecteur>& b,const double &tol = tol_defaut
305         ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
306         // d) avec en entrée un tableau de vecteur second membre et
307         //      en sortie le tableau de vecteurs d'entree
308     Tableau <Vecteur>& Simple_Resol_systID
309         (Tableau <Vecteur>& b,const double &tol = tol_defaut
310         ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
311         // e) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
312         //      et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
313         //      deux étant identiques
314     Vecteur& Simple_Resol_systID_2 (const Vecteur& b,Vecteur& vortie
315         , const double &tol = tol_defaut
316         ,const int maxit = maxit_defaut
317         ,const int restart = restart_defaut) const ;
318     // ===== FIN RÉOLUTION EN DEUX TEMPS ===== :
319
320     // Multiplication d'un vecteur par une matrice ( (vec)t * A )
321     Vecteur Prod_vec_mat ( const Vecteur& vec) const ;
322     // idem mais on utilise la place du second vecteur pour le résultat
323     Vecteur& Prod_vec_mat ( const Vecteur& vec, Vecteur & resul) const ;
324     // Multiplication d'une matrice par un vecteur ( A * vec )
325     Vecteur Prod_mat_vec ( const Vecteur& vec) const ;
326     // idem mais on utilise la place du second vecteur pour le résultat
327     Vecteur& Prod_mat_vec ( const Vecteur& vec, Vecteur & resul) const;
328     // Multiplication d'une ligne iligne de la matrice avec un vecteur de
329     // dimension = le nombre de colonne de la matrice
330     double Prod_Ligne_vec ( int iligne, const Vecteur& vec) const ;
331     // Multiplication d'un vecteur avec une colonne icol de la matrice
332     // dimension = le nombre de ligne de la matrice
333     double Prod_vec_col( int icol, const Vecteur& vec) const ;
334
335     // calcul du produit : (vec_1)^T * A * (vect_2)
336     double vectT_mat_vec(const Vecteur& vecl, const Vecteur& vec2) const ;
337
338     // retourne la place que prend la matrice en entier
339     int Place() const { return (NumNonzeros() * 3 + dim(1) + 5);};
340
341 // ===== quelques méthodes spécifiques à la classe MV_Vector<double>=====
342 // définie la surcharge de multiplication d'une matrice par un MV_Vector
343 // ici de type constant, ramène un nouveau MV_Vector<double>
344 inline virtual MV_Vector<double> operator * (const MV_Vector<double> & vec) const
345 { return this->CompCol_Mat_double::operator *(vec); };
346 // multiplication matrice transposée fois vecteur ce qui est équivalent à
347 // la transposée du résultat de : multiplication vecteur transposé fois matrice
348 inline virtual MV_Vector<double> trans_mult(const MV_Vector<double> &x) const
349 { return this->CompCol_Mat_double::trans_mult(x); };
350 // ===== fin des quelques méthodes spécifiques à la classe MV_Vector<double>=====
351
352 protected :
353
354     // VARIABLES PROTEGEES :
355 };
356 /// @} // end of group
357
358 #endif

```

## 7.382 clapack.h

```

1 /*
2 =====
3 Definitions and prototypes for LAPACK as provided Apple Computer.
4
5 Documentation of the LAPACK interfaces, including reference implementations, can be found on the web
6 starting from the LAPACK FAQ page at this URL (verified live as of April 2002):
7     http://netlib.org/lapack/faq.html
8
9 A hardcopy maanual is:
10     LAPACK Users' Guide, Third Edition.
11     @BOOK{laug,
12         AUTHOR = {Anderson, E. and Bai, Z. and Bischof, C. and
13                 Blackford, S. and Demmel, J. and Dongarra, J. and
14                 Du Croz, J. and Greenbaum, A. and Hammarling, S. and
15                 McKenney, A. and Sorensen, D.},
16         TITLE = {{LAPACK} Users' Guide},
17         EDITION = {Third},
18         PUBLISHER = {Society for Industrial and Applied Mathematics},
19         YEAR = {1999},
20         ADDRESS = {Philadelphia, PA},
21         ISBN = {0-89871-447-8 (paperback)} }
22
23 =====

```



```

24 */
25 #ifndef __CLAPACK_H
26 #define __CLAPACK_H
27
28 #ifdef __cplusplus
29 extern "C" {
30 #endif
31
32
33 #if defined(__LP64__) /* In LP64 match sizes with the 32 bit ABI */
34 typedef int      __CLPK_integer;
35 typedef int      __CLPK_logical;
36 typedef float    __CLPK_real;
37 typedef double   __CLPK_doublereal;
38 typedef          __CLPK_logical (*__CLPK_L_fp)();
39 typedef int      __CLPK_ftnlen;
40 #else
41 typedef long int  __CLPK_integer;
42 typedef long int  __CLPK_logical;
43 typedef float     __CLPK_real;
44 typedef double    __CLPK_doublereal;
45 typedef          __CLPK_logical (*__CLPK_L_fp)();
46 typedef long int  __CLPK_ftnlen;
47 #endif
48
49 typedef struct { __CLPK_real r, i; } __CLPK_complex;
50 typedef struct { __CLPK_doublereal r, i; } __CLPK_doublecomplex;
51
52 /* Subroutine */ int cbsdqr(char *uplo, __CLPK_integer *n, __CLPK_integer *ncvt, __CLPK_integer *
53   nru, __CLPK_integer *ncc, __CLPK_real *d__, __CLPK_real *e, __CLPK_complex *vt, __CLPK_integer *ldvt,
54   __CLPK_complex *u, __CLPK_integer *ldu, __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_real *rwork,
55   __CLPK_integer *info);
56
57 /* Subroutine */ int cgbbrd(char *vect, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *ncc,
58   __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *d__,
59   __CLPK_real *e, __CLPK_complex *q, __CLPK_integer *ldq, __CLPK_complex *pt, __CLPK_integer *ldpt,
60   __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer
   *info);
61
62 /* Subroutine */ int cgbcon(char *norm, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku,
63   __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_real *anorm, __CLPK_real
   *rcond,
64   __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
65
66 /* Subroutine */ int cgbequ(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
   *ku,
67   __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *rowcnd,
   __CLPK_real
68   *colcnd, __CLPK_real *amax, __CLPK_integer *info);
69
70 /* Subroutine */ int cgbrfs(char *trans, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *
71   ku, __CLPK_integer *nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex *afb,
   __CLPK_integer *
72   ldafb, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *x,
   __CLPK_integer *
73   idx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *
74   info);
75
76 /* Subroutine */ int cgbsv(__CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_integer *
77   nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_complex *b,
   __CLPK_integer *
78   ldb, __CLPK_integer *info);
79
80 /* Subroutine */ int cgbsvx(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *kl,
81   __CLPK_integer *ku, __CLPK_integer *nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex
   *afb,
82   __CLPK_integer *ldafb, __CLPK_integer *ipiv, char *equed, __CLPK_real *r__, __CLPK_real *c__,
   __CLPK_complex *b,
83   __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *rcond,
   __CLPK_real
84   *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
85
86 /* Subroutine */ int cgbtf2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
   *ku,
87   __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_integer *info);
88
89 /* Subroutine */ int cgbtrf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
   *ku,
90   __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_integer *info);
91
92 /* Subroutine */ int cgbtrs(char *trans, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *
93   ku, __CLPK_integer *nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv,
   __CLPK_complex
94   *b, __CLPK_integer *ldb, __CLPK_integer *info);
95
96 /* Subroutine */ int cgebak(char *job, char *side, __CLPK_integer *n, __CLPK_integer *ilo,

```

```

97  __CLPK_integer *ihi, __CLPK_real *scale, __CLPK_integer *m, __CLPK_complex *v, __CLPK_integer *ldv,
98  __CLPK_integer *info);
99
100 /* Subroutine */ int cgebal(char *job, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
101  __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real *scale, __CLPK_integer *info);
102
103 /* Subroutine */ int cgebd2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
104  *lda,
105  __CLPK_real *d__, __CLPK_real *e, __CLPK_complex *tauq, __CLPK_complex *taup, __CLPK_complex *work,
106  __CLPK_integer *info);
107 /* Subroutine */ int cgebrd(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
108  *lda,
109  __CLPK_real *d__, __CLPK_real *e, __CLPK_complex *tauq, __CLPK_complex *taup, __CLPK_complex *work,
110  __CLPK_integer *lwork, __CLPK_integer *info);
111 /* Subroutine */ int cgecon(char *norm, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
112  __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer
113  *info);
114 /* Subroutine */ int cgeequ(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
115  *lda,
116  __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *rowcnd, __CLPK_real *colcnd, __CLPK_real *amax,
117  __CLPK_integer *info);
118 /* Subroutine */ int cgees(char *jobvs, char *sort, __CLPK_L_fp select, __CLPK_integer *n,
119  __CLPK_complex *a, __CLPK_integer *lda, __CLPK_integer *sdim, __CLPK_complex *w, __CLPK_complex *vs,
120  __CLPK_integer *ldvs, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork,
121  __CLPK_logical *
122  bwork, __CLPK_integer *info);
123 /* Subroutine */ int cgeesx(char *jobvs, char *sort, __CLPK_L_fp select, char *
124  sense, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_integer *sdim,
125  __CLPK_complex *
126  w, __CLPK_complex *vs, __CLPK_integer *ldvs, __CLPK_real *rconde, __CLPK_real *rcondv,
127  __CLPK_complex *
128  work, __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_logical *bwork, __CLPK_integer *info);
129 /* Subroutine */ int cgeev(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_complex *a,
130  __CLPK_integer *lda, __CLPK_complex *w, __CLPK_complex *vl, __CLPK_integer *ldvl, __CLPK_complex
131  *vr,
132  __CLPK_integer *ldvr, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork,
133  __CLPK_integer *
134  info);
135 /* Subroutine */ int cgeevx(char *balanc, char *jobvl, char *jobvr, char *
136  sense, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *w, __CLPK_complex
137  *vl,
138  __CLPK_integer *ldvl, __CLPK_complex *vr, __CLPK_integer *ldvr, __CLPK_integer *ilo, __CLPK_integer
139  *ihi,
140  __CLPK_real *scale, __CLPK_real *abnrm, __CLPK_real *rconde, __CLPK_real *rcondv, __CLPK_complex
141  *work,
142  __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *info);
143 /* Subroutine */ int cgegsv(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_complex *a,
144  __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *alpha, __CLPK_complex
145  *beta,
146  __CLPK_complex *vl, __CLPK_integer *ldvl, __CLPK_complex *vr, __CLPK_integer *ldvr, __CLPK_complex
147  *
148  work, __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *info);
149 /* Subroutine */ int cgehd2(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_complex
150  *
151  a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
152 /* Subroutine */ int cgehrd(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_complex
153  *
154  a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork,
155  __CLPK_integer
156  *info);
157 /* Subroutine */ int cgelq2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
158  *lda,
159  __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
160 /* Subroutine */ int cgelqf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
161  *lda,
162  __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);

```

```
161
162 /* Subroutine */ int cgels(char *trans, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *
163 nrhs, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
*
164 work, __CLPK_integer *lwork, __CLPK_integer *info);
165
166 /* Subroutine */ int cgelsx(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex
*
167 a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *jpvt, __CLPK_real
*rcond,
168 __CLPK_integer *rank, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
169
170 /* Subroutine */ int cgelsy(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex
*
171 a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *jpvt, __CLPK_real
*rcond,
172 __CLPK_integer *rank, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork,
__CLPK_integer *
173 info);
174
175 /* Subroutine */ int cgeql2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
176 __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
177
178 /* Subroutine */ int cgeqlf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
179 __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
180
181 /* Subroutine */ int cgeqp3(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
182 __CLPK_integer *jpvt, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real
*
183 rwork, __CLPK_integer *info);
184
185 /* Subroutine */ int cgeqpf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
186 __CLPK_integer *jpvt, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer
*
187 info);
188
189 /* Subroutine */ int cgeqr2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
190 __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
191
192 /* Subroutine */ int cgeqrf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
193 __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
194
195 /* Subroutine */ int cgerfs(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
196 a, __CLPK_integer *lda, __CLPK_complex *af, __CLPK_integer *ldaf, __CLPK_integer *ipiv,
__CLPK_complex *
197 b, __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real
*berr,
198 __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
199
200 /* Subroutine */ int cgerq2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
201 __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
202
203 /* Subroutine */ int cgerqf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
204 __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
205
206 /* Subroutine */ int cges2(__CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *
207 rhs, __CLPK_integer *ipiv, __CLPK_integer *jpiv, __CLPK_real *scale);
208
209 /* Subroutine */ int cgesv(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *a, __CLPK_integer *
210 lda, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
211
212 /* Subroutine */ int cgesvx(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *
213 nrhs, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *af, __CLPK_integer *ldaf,
__CLPK_integer *
214 ipiv, char *equed, __CLPK_real *r__, __CLPK_real *c__, __CLPK_complex *b, __CLPK_integer *ldb,
__CLPK_complex *x,
215 __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr,
__CLPK_complex *work,
216 __CLPK_real *rwork, __CLPK_integer *info);
217
218 /* Subroutine */ int cgetc2(__CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_integer *
219 ipiv, __CLPK_integer *jpiv, __CLPK_integer *info);
220
221 /* Subroutine */ int cgetf2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
222 __CLPK_integer *ipiv, __CLPK_integer *info);
223
224 /* Subroutine */ int cgetrf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
225 __CLPK_integer *ipiv, __CLPK_integer *info);
226
```

```

227 /* Subroutine */ int cgetri(__CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_integer *
228     ipiv, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
229
230 /* Subroutine */ int cgetrs(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
231     a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer
232     *
233     info);
234
235 /* Subroutine */ int cggbak(char *job, char *side, __CLPK_integer *n, __CLPK_integer *ilo,
236     __CLPK_integer *ihi, __CLPK_real *lscale, __CLPK_real *rscale, __CLPK_integer *m, __CLPK_complex *v,
237     __CLPK_integer *ldv, __CLPK_integer *info);
238
239 /* Subroutine */ int cggbal(char *job, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
240     __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real
241     *lscale,
242     __CLPK_real *rscale, __CLPK_real *work, __CLPK_integer *info);
243
244 /* Subroutine */ int cgges(char *jobvsl, char *jobvsr, char *sort, __CLPK_L_fp
245     selctg, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer
246     *
247     ldb, __CLPK_integer *sdim, __CLPK_complex *alpha, __CLPK_complex *beta, __CLPK_complex *vsl,
248     __CLPK_integer *ldvsl, __CLPK_complex *vsr, __CLPK_integer *ldvsr, __CLPK_complex *work,
249     __CLPK_integer *
250     lwork, __CLPK_real *rwork, __CLPK_logical *bwork, __CLPK_integer *info);
251
252 /* Subroutine */ int cggesx(char *jobvsl, char *jobvsr, char *sort, __CLPK_L_fp
253     selctg, char *sense, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b,
254     __CLPK_integer *ldb, __CLPK_integer *sdim, __CLPK_complex *alpha, __CLPK_complex *beta,
255     __CLPK_complex *
256     vsl, __CLPK_integer *ldvsl, __CLPK_complex *vsr, __CLPK_integer *ldvsr, __CLPK_real *rconde,
257     __CLPK_real
258     *rcondv, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *iwork,
259     __CLPK_integer *liwork, __CLPK_logical *bwork, __CLPK_integer *info);
260
261 /* Subroutine */ int cggev(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_complex *a,
262     __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *alpha, __CLPK_complex
263     *beta,
264     __CLPK_complex *vl, __CLPK_integer *ldvl, __CLPK_complex *vr, __CLPK_integer *ldvr, __CLPK_complex
265     *
266     work, __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *info);
267
268 /* Subroutine */ int cggevx(char *balanc, char *jobvl, char *jobvr, char *
269     sense, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer
270     *ldb,
271     __CLPK_complex *alpha, __CLPK_complex *beta, __CLPK_complex *vl, __CLPK_integer *ldvl,
272     __CLPK_complex *
273     vr, __CLPK_integer *ldvr, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real *lscale, __CLPK_real
274     *
275     rscale, __CLPK_real *abnrm, __CLPK_real *bbnrm, __CLPK_real *rconde, __CLPK_real *rcondv,
276     __CLPK_complex
277     *work, __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *iwork, __CLPK_logical *bwork,
278     __CLPK_integer *info);
279
280 /* Subroutine */ int cggglm(__CLPK_integer *n, __CLPK_integer *m, __CLPK_integer *p, __CLPK_complex *a,
281     __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *d__, __CLPK_complex *x,
282     __CLPK_complex *y,
283     __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
284
285 /* Subroutine */ int cgghrd(char *compq, char *compz, __CLPK_integer *n, __CLPK_integer *
286     ilo, __CLPK_integer *ihi, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer
287     *ldb,
288     __CLPK_complex *q, __CLPK_integer *ldq, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_integer
289     *info);
290
291 /* Subroutine */ int cgglse(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *p, __CLPK_complex *a,
292     __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *c__, __CLPK_complex
293     *d__,
294     __CLPK_complex *x, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
295
296 /* Subroutine */ int cggqrf(__CLPK_integer *n, __CLPK_integer *m, __CLPK_integer *p, __CLPK_complex *a,
297     __CLPK_integer *lda, __CLPK_complex *taua, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
298     *taub,
299     __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
300
301 /* Subroutine */ int cggqrq(__CLPK_integer *m, __CLPK_integer *p, __CLPK_integer *n, __CLPK_complex *a,
302     __CLPK_integer *lda, __CLPK_complex *taua, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
303     *taub,
304     __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
305
306 /* Subroutine */ int cggsvd(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
307     __CLPK_integer *n, __CLPK_integer *p, __CLPK_integer *k, __CLPK_integer *l, __CLPK_complex *a,
308     __CLPK_integer *

```

```

290   lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_real *alpha, __CLPK_real *beta, __CLPK_complex
291   *u,
292   __CLPK_integer *ldu, __CLPK_complex *v, __CLPK_integer *ldv, __CLPK_complex *q, __CLPK_integer *ldq,
293   __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *iwork, __CLPK_integer *info);
294 /* Subroutine */ int cggsvp(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
295   __CLPK_integer *p, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b,
296   __CLPK_integer
297   *ldb, __CLPK_real *tola, __CLPK_real *tolb, __CLPK_integer *k, __CLPK_integer *l, __CLPK_complex *u,
298   __CLPK_integer *ldu, __CLPK_complex *v, __CLPK_integer *ldv, __CLPK_complex *q, __CLPK_integer *ldq,
299   __CLPK_integer *iwork, __CLPK_real *rwork, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer
300   *
301   info);
302 /* Subroutine */ int cgtcon(char *norm, __CLPK_integer *n, __CLPK_complex *dl, __CLPK_complex *
303   d_, __CLPK_complex *du, __CLPK_complex *du2, __CLPK_integer *ipiv, __CLPK_real *anorm, __CLPK_real
304   *
305   rcond, __CLPK_complex *work, __CLPK_integer *info);
306 /* Subroutine */ int cgtrfs(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
307   dl, __CLPK_complex *d_, __CLPK_complex *du, __CLPK_complex *dlf, __CLPK_complex *df, __CLPK_complex
308   *
309   *
310   *
311   *
312   *
313   *
314   *
315   *
316   *
317   *
318   *
319   *
320   *
321   *
322   *
323   *
324   *
325   *
326   *
327   *
328   *
329   *
330   *
331   *
332   *
333   *
334   *
335   *
336   *
337   *
338   *
339   *
340   *
341   *
342   *
343   *
344   *
345   *
346   *
347   *
348   *
349   *
350   *
351   *
352   *

```

```

353     __CLPK_integer *info);
354
355 /* Subroutine */ int chbgvx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
356   __CLPK_integer *ka, __CLPK_integer *kb, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex
   *bb,
357   __CLPK_integer *ldbb, __CLPK_complex *q, __CLPK_integer *ldq, __CLPK_real *vl, __CLPK_real *vu,
   __CLPK_integer *
358   il, __CLPK_integer *iu, __CLPK_real *abstol, __CLPK_integer *m, __CLPK_real *w, __CLPK_complex *z__,
   __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *iwork, __CLPK_integer
   *
359   * ifail, __CLPK_integer *info);
360
361
362 /* Subroutine */ int chbtrd_(char *vect, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
363   __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *d__, __CLPK_real *e, __CLPK_complex *q,
   __CLPK_integer *
364   ldq, __CLPK_complex *work, __CLPK_integer *info);
365
366 /* Subroutine */ int checon_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
367   __CLPK_integer *ipiv, __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_integer
   *
368   info);
369
370 /* Subroutine */ int cheev_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_complex *a,
371   __CLPK_integer *lda, __CLPK_real *w, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real
   *rwork,
372   __CLPK_integer *info);
373
374 /* Subroutine */ int cheevd_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_complex *a,
375   __CLPK_integer *lda, __CLPK_real *w, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real
   *rwork,
376   __CLPK_integer *lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
377
378 /* Subroutine */ int cheevr_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
379   __CLPK_complex *a, __CLPK_integer *lda, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il,
   __CLPK_integer *
380   iu, __CLPK_real *abstol, __CLPK_integer *m, __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer
   *ldz,
381   __CLPK_integer *isuppz, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork,
   __CLPK_integer *
382   lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
383
384 /* Subroutine */ int cheevx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
385   __CLPK_complex *a, __CLPK_integer *lda, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il,
   __CLPK_integer *
386   iu, __CLPK_real *abstol, __CLPK_integer *m, __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer
   *ldz,
387   __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *iwork,
   __CLPK_integer *
388   ifail, __CLPK_integer *info);
389
390 /* Subroutine */ int chegs2_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n, __CLPK_complex *
391   a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
392
393 /* Subroutine */ int chegst_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n, __CLPK_complex *
394   a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
395
396 /* Subroutine */ int chegv_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
397   n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_real *w,
   __CLPK_complex *work,
398   __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *info);
399
400 /* Subroutine */ int chegvd_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
401   n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_real *w,
   __CLPK_complex *work,
402   __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *lrwork,
   __CLPK_integer *
403   iwork, __CLPK_integer *liwork, __CLPK_integer *info);
404
405 /* Subroutine */ int chegvx_(__CLPK_integer *itype, char *jobz, char *range, char *
406   uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer
   *ldb,
407   __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu, __CLPK_real *abstol,
   __CLPK_integer *
408   m, __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_integer
   *lwork,
409   __CLPK_real *rwork, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
410
411 /* Subroutine */ int cherfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
412   a, __CLPK_integer *lda, __CLPK_complex *af, __CLPK_integer *ldaf, __CLPK_integer *ipiv,
   __CLPK_complex *
413   b, __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real
   *berr,
414   __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
415
416 /* Subroutine */ int chesv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *a,
417   __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
   *work,
418   __CLPK_integer *lwork, __CLPK_integer *info);

```

```

419
420 /* Subroutine */ int chesvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
421   nrhs, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *af, __CLPK_integer *ldaf,
   __CLPK_integer *
422   ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real
   *rcond,
423   __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real
   *rwork,
424   __CLPK_integer *info);
425
426 /* Subroutine */ int chetf2_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
427   __CLPK_integer *ipiv, __CLPK_integer *info);
428
429 /* Subroutine */ int chetrd_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
430   __CLPK_real *d__, __CLPK_real *e, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork,
   __CLPK_integer *info);
431
432
433 /* Subroutine */ int chetrf_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
434   __CLPK_integer *ipiv, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
435
436 /* Subroutine */ int chetri_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
437   __CLPK_integer *ipiv, __CLPK_complex *work, __CLPK_integer *info);
438
439 /* Subroutine */ int chetrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
440   a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *
   *
441   info);
442
443 /* Subroutine */ int chgeqz_(char *jobz, char *compq, char *compz, __CLPK_integer *n,
444   __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b,
   __CLPK_integer *ldb,
445   __CLPK_complex *alpha, __CLPK_complex *beta, __CLPK_complex *q, __CLPK_integer
   *ldq,
446   __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real
   *
   rwork, __CLPK_integer *info);
447
448
449 /* Subroutine */ int chpcon_(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_integer *
450   ipiv, __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_integer *info);
451
452 /* Subroutine */ int chpev_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_complex *ap,
453   __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_real *rwork,
   __CLPK_integer *info);
454
455
456 /* Subroutine */ int chpevd_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_complex *ap,
457   __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_integer *
   lwork,
458   __CLPK_real *rwork, __CLPK_integer *lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork,
   __CLPK_integer *info);
459
460
461 /* Subroutine */ int chpevx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
462   __CLPK_complex *ap, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu,
   __CLPK_real *
463   abstol, __CLPK_integer *m, __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_complex
   *
   *
464   work, __CLPK_real *rwork, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
465
466 /* Subroutine */ int chpgst_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n, __CLPK_complex *
467   ap, __CLPK_complex *bp, __CLPK_integer *info);
468
469 /* Subroutine */ int chpgv_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
470   n, __CLPK_complex *ap, __CLPK_complex *bp, __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer *ldz,
   __CLPK_complex *work,
471   __CLPK_real *rwork, __CLPK_integer *info);
472
473 /* Subroutine */ int chpgvd_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
474   n, __CLPK_complex *ap, __CLPK_complex *bp, __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer *ldz,
   __CLPK_complex *work,
475   __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *lrwork,
   __CLPK_integer *
476   iwork, __CLPK_integer *liwork, __CLPK_integer *info);
477
478 /* Subroutine */ int chpgvx_(__CLPK_integer *itype, char *jobz, char *range, char *
479   uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_complex *bp, __CLPK_real *vl, __CLPK_real *vu,
   __CLPK_integer *il,
480   __CLPK_integer *iu, __CLPK_real *abstol, __CLPK_integer *m, __CLPK_real *w,
   __CLPK_complex *
   *
481   z__, __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *iwork,
   __CLPK_integer *ifail,
482   __CLPK_integer *info);
483
484 /* Subroutine */ int chprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
485   ap, __CLPK_complex *afp, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb,
   __CLPK_complex *x,
486   __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_real
   *rwork,
487   __CLPK_integer *info);
488

```



```

489 /* Subroutine */ int chpsv(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
490 ap, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
491
492 /* Subroutine */ int chpsvx(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
493 nrhs, __CLPK_complex *ap, __CLPK_complex *afp, __CLPK_integer *ipiv, __CLPK_complex *b,
__CLPK_integer *
494 ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real
*berr,
495 __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
496
497 /* Subroutine */ int chptrd(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_real *d,
498 __CLPK_real *e, __CLPK_complex *tau, __CLPK_integer *info);
499
500 /* Subroutine */ int chptrf(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_integer *
501 ipiv, __CLPK_integer *info);
502
503 /* Subroutine */ int chptri(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_integer *
504 ipiv, __CLPK_complex *work, __CLPK_integer *info);
505
506 /* Subroutine */ int chptrs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
507 ap, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
508
509 /* Subroutine */ int chsein(char *side, char *eigsrc, char *initv, __CLPK_logical *
510 select, __CLPK_integer *n, __CLPK_complex *h, __CLPK_integer *ldh, __CLPK_complex *w,
__CLPK_complex *
511 vl, __CLPK_integer *ldvl, __CLPK_complex *vr, __CLPK_integer *ldvr, __CLPK_integer *mm,
__CLPK_integer *
512 m, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *ifail1, __CLPK_integer *ifailr,
__CLPK_integer *info);
513
514
515 /* Subroutine */ int chseqr(char *job, char *compz, __CLPK_integer *n, __CLPK_integer *ilo,
516 __CLPK_integer *ihi, __CLPK_complex *h, __CLPK_integer *ldh, __CLPK_complex *w, __CLPK_complex
*z,
517 __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
518
519 /* Subroutine */ int clabrd(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_complex
*a,
520 __CLPK_integer *lda, __CLPK_real *d, __CLPK_real *e, __CLPK_complex *tauq, __CLPK_complex *taup,
__CLPK_complex *x,
521 __CLPK_integer *ldx, __CLPK_complex *y, __CLPK_integer *ldy);
522
523 /* Subroutine */ int clacgv(__CLPK_integer *n, __CLPK_complex *x, __CLPK_integer *incx);
524
525 /* Subroutine */ int clacon(__CLPK_integer *n, __CLPK_complex *v, __CLPK_complex *x, __CLPK_real *est,
526 __CLPK_integer *kase);
527
528 /* Subroutine */ int clacp2(char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a,
529 __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb);
530
531 /* Subroutine */ int clacpy(char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a,
532 __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb);
533
534 /* Subroutine */ int clacrm(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
*lda,
535 __CLPK_real *b, __CLPK_integer *ldb, __CLPK_complex *c, __CLPK_integer *ldc, __CLPK_real *rwork);
536
537 /* Subroutine */ int clacrt(__CLPK_integer *n, __CLPK_complex *cx, __CLPK_integer *incx, __CLPK_complex
*
538 cy, __CLPK_integer *incy, __CLPK_complex *c, __CLPK_complex *s);
539
540 /* Subroutine */ int claed0(__CLPK_integer *qsiz, __CLPK_integer *n, __CLPK_real *d, __CLPK_real *e,
541 __CLPK_complex *q, __CLPK_integer *ldq, __CLPK_complex *qstore, __CLPK_integer *ldqs, __CLPK_real
*rwork,
542 __CLPK_integer *iwork, __CLPK_integer *info);
543
544 /* Subroutine */ int claed7(__CLPK_integer *n, __CLPK_integer *cutpnt, __CLPK_integer *qsiz,
545 __CLPK_integer *tlvl, __CLPK_integer *curlvl, __CLPK_integer *curpbm, __CLPK_real *d,
__CLPK_complex *
546 q, __CLPK_integer *ldq, __CLPK_real *rho, __CLPK_integer *indxq, __CLPK_real *qstore, __CLPK_integer
*
547 qpnr, __CLPK_integer *prmptr, __CLPK_integer *perm, __CLPK_integer *givptr, __CLPK_integer *
548 givcol, __CLPK_real *givnum, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *iwork,
__CLPK_integer *info);
549
550
551 /* Subroutine */ int claed8(__CLPK_integer *k, __CLPK_integer *n, __CLPK_integer *qsiz, __CLPK_complex
*
552 q, __CLPK_integer *ldq, __CLPK_real *d, __CLPK_real *rho, __CLPK_integer *cutpnt, __CLPK_real
*z,
553 __CLPK_real *dlamda, __CLPK_complex *q2, __CLPK_integer *ldq2, __CLPK_real *w, __CLPK_integer
*indxp,
554 __CLPK_integer *indx, __CLPK_integer *indxq, __CLPK_integer *perm, __CLPK_integer *givptr,
__CLPK_integer *givcol,
555 __CLPK_real *givnum, __CLPK_integer *info);
556
557 /* Subroutine */ int claein(__CLPK_logical *rightv, __CLPK_logical *noinit, __CLPK_integer *n,
558 __CLPK_complex *h, __CLPK_integer *ldh, __CLPK_complex *w, __CLPK_complex *v, __CLPK_complex *b,
559 __CLPK_integer *ldb, __CLPK_real *rwork, __CLPK_real *eps3, __CLPK_real *smlnum, __CLPK_integer
*info);
560

```



```

561 /* Subroutine */ int claesy(__CLPK_complex *a, __CLPK_complex *b, __CLPK_complex *c__, __CLPK_complex *
562     rtl, __CLPK_complex *rt2, __CLPK_complex *evscal, __CLPK_complex *csl, __CLPK_complex *snl);
563
564 /* Subroutine */ int claev2(__CLPK_complex *a, __CLPK_complex *b, __CLPK_complex *c__, __CLPK_real
565     *rtl,
566     __CLPK_real *rt2, __CLPK_real *csl, __CLPK_complex *snl);
567 /* Subroutine */ int clags2(__CLPK_logical *upper, __CLPK_real *a1, __CLPK_complex *a2, __CLPK_real
568     *a3,
569     __CLPK_real *b1, __CLPK_complex *b2, __CLPK_real *b3, __CLPK_real *csu, __CLPK_complex *snu,
570     __CLPK_real *csv,
571     __CLPK_complex *snv, __CLPK_real *csq, __CLPK_complex *snq);
572 /* Subroutine */ int clagtm(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *
573     alpha, __CLPK_complex *dl, __CLPK_complex *d__, __CLPK_complex *du, __CLPK_complex *x,
574     __CLPK_integer *
575     idx, __CLPK_real *beta, __CLPK_complex *b, __CLPK_integer *ldb);
576 /* Subroutine */ int clahef(char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_integer *kb,
577     __CLPK_complex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_complex *w, __CLPK_integer
578     *ldw,
579     __CLPK_integer *info);
580 /* Subroutine */ int clahqr(__CLPK_logical *wantt, __CLPK_logical *wantz, __CLPK_integer *n,
581     __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_complex *h__, __CLPK_integer *ldh, __CLPK_complex
582     *w,
583     __CLPK_integer *iloz, __CLPK_integer *ihiz, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_integer
584     *
585     info);
586 /* Subroutine */ int clahrd(__CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *nb, __CLPK_complex
587     *a,
588     __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *t, __CLPK_integer *ldt, __CLPK_complex *y,
589     __CLPK_integer *ldy);
590 /* Subroutine */ int claicl(__CLPK_integer *job, __CLPK_integer *j, __CLPK_complex *x, __CLPK_real
591     *sest,
592     __CLPK_complex *w, __CLPK_complex *gamma, __CLPK_real *sestpr, __CLPK_complex *s, __CLPK_complex
593     *c__);
594 /* Subroutine */ int clals0(__CLPK_integer *icompq, __CLPK_integer *nl, __CLPK_integer *nr,
595     __CLPK_integer *sqre, __CLPK_integer *nrhs, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
596     *bx,
597     __CLPK_integer *ldbxx, __CLPK_integer *perm, __CLPK_integer *givptr, __CLPK_integer *givcol,
598     __CLPK_integer *ldgcol, __CLPK_real *givnum, __CLPK_integer *ldgnum, __CLPK_real *poles, __CLPK_real
599     *
600     difl, __CLPK_real *difr, __CLPK_real *z__, __CLPK_integer *k, __CLPK_real *c__, __CLPK_real *s,
601     __CLPK_real *
602     rwork, __CLPK_integer *info);
603 /* Subroutine */ int clalsa(__CLPK_integer *icompq, __CLPK_integer *smlsiz, __CLPK_integer *n,
604     __CLPK_integer *nrhs, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *bx, __CLPK_integer
605     *ldbxx,
606     __CLPK_real *u, __CLPK_integer *ldu, __CLPK_real *vt, __CLPK_integer *k, __CLPK_real *difl,
607     __CLPK_real *difr,
608     __CLPK_real *z__, __CLPK_real *poles, __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_integer
609     *
610     ldgcol, __CLPK_integer *perm, __CLPK_real *givnum, __CLPK_real *c__, __CLPK_real *s, __CLPK_real
611     *rwork,
612     __CLPK_integer *iwork, __CLPK_integer *info);
613 /* Subroutine */ int clapll(__CLPK_integer *n, __CLPK_complex *x, __CLPK_integer *incx, __CLPK_complex
614     *
615     y, __CLPK_integer *incy, __CLPK_real *ssmin);
616 /* Subroutine */ int clapmt(__CLPK_logical *forwrd, __CLPK_integer *m, __CLPK_integer *n,
617     __CLPK_complex
618     *x, __CLPK_integer *ldx, __CLPK_integer *k);
619 /* Subroutine */ int claqgb(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
620     *ku,
621     __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *rowcnd,
622     __CLPK_real
623     *colcnd, __CLPK_real *amax, char *equad);
624 /* Subroutine */ int claqge(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
625     *lda,
626     __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *rowcnd, __CLPK_real *colcnd, __CLPK_real *amax,
627     char *
628     equed);
629 /* Subroutine */ int claghb(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_complex *ab,
630     __CLPK_integer
631     *ldab, __CLPK_real *s, __CLPK_real *scnd, __CLPK_real *amax, char *equad);
632 /* Subroutine */ int claghe(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
633     __CLPK_real
634     *s, __CLPK_real *scnd, __CLPK_real *amax, char *equad);

```

```

624
625 /* Subroutine */ int claqhp(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_real *s,
626   __CLPK_real *scond, __CLPK_real *amax, char *equet);
627
628 /* Subroutine */ int claqp2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *offset,
629   __CLPK_complex
630   *a, __CLPK_integer *lda, __CLPK_integer *jpvt, __CLPK_complex *tau, __CLPK_real *vn1, __CLPK_real
631   *vn2,
632   __CLPK_complex *work);
633
634 /* Subroutine */ int claqps(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *offset,
635   __CLPK_integer
636   *nb, __CLPK_integer *kb, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_integer *jpvt,
637   __CLPK_complex *
638   tau, __CLPK_real *vn1, __CLPK_real *vn2, __CLPK_complex *auxv, __CLPK_complex *f, __CLPK_integer
639   *ldf);
640
641 /* Subroutine */ int claqsb(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_complex *ab,
642   __CLPK_integer *ldab, __CLPK_real *s, __CLPK_real *scond, __CLPK_real *amax, char *equet);
643
644 /* Subroutine */ int claqsp(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_real *s,
645   __CLPK_real *scond, __CLPK_real *amax, char *equet);
646
647 /* Subroutine */ int claqsy(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
648   __CLPK_real *s, __CLPK_real *scond, __CLPK_real *amax, char *equet);
649
650 /* Subroutine */ int clarlv(__CLPK_integer *n, __CLPK_integer *b1, __CLPK_integer *bn, __CLPK_real *
651   sigma, __CLPK_real *d__, __CLPK_real *l, __CLPK_real *ld, __CLPK_real *lld, __CLPK_real *gersch,
652   __CLPK_complex
653   *z__, __CLPK_real *ztz, __CLPK_real *mingma, __CLPK_integer *r__, __CLPK_integer *isuppz,
654   __CLPK_real *
655   work);
656
657 /* Subroutine */ int clar2v(__CLPK_integer *n, __CLPK_complex *x, __CLPK_complex *y, __CLPK_complex
658   *z__,
659   __CLPK_integer *incx, __CLPK_real *c__, __CLPK_complex *s, __CLPK_integer *incc);
660
661 /* Subroutine */ int clarcm(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
662   __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_real
663   *rwork);
664
665 /* Subroutine */ int clarf(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *v,
666   __CLPK_integer *incv, __CLPK_complex *tau, __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_complex
667   *
668   work);
669
670 /* Subroutine */ int clarfb(char *side, char *trans, char *direct, char *
671   storev, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex *v, __CLPK_integer
672   *ldv,
673   __CLPK_complex *t, __CLPK_integer *ldt, __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_complex
674   *work,
675   __CLPK_integer *ldwork);
676
677 /* Subroutine */ int clarfg(__CLPK_integer *n, __CLPK_complex *alpha, __CLPK_complex *x, __CLPK_integer
678   *
679   incx, __CLPK_complex *tau);
680
681 /* Subroutine */ int clarft(char *direct, char *storev, __CLPK_integer *n, __CLPK_integer *
682   k, __CLPK_complex *v, __CLPK_integer *ldv, __CLPK_complex *tau, __CLPK_complex *t, __CLPK_integer
683   *ldt);
684
685 /* Subroutine */ int clarfx(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *v,
686   __CLPK_complex *tau, __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_complex *work);
687
688 /* Subroutine */ int clargv(__CLPK_integer *n, __CLPK_complex *x, __CLPK_integer *incx, __CLPK_complex
689   *
690   y, __CLPK_integer *incy, __CLPK_real *c__, __CLPK_integer *incc);
691
692 /* Subroutine */ int clarv(__CLPK_integer *idist, __CLPK_integer *iseed, __CLPK_integer *n,
693   __CLPK_complex *x);
694
695 /* Subroutine */ int clarrv(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *l, __CLPK_integer
696   *isplit,
697   __CLPK_integer *m, __CLPK_real *w, __CLPK_integer *iblock, __CLPK_real *gersch, __CLPK_real *tol,
698   __CLPK_complex *z__,
699   __CLPK_integer *ldz, __CLPK_integer *isuppz, __CLPK_real *work, __CLPK_integer
700   *
701   iwork, __CLPK_integer *info);
702
703 /* Subroutine */ int clartg(__CLPK_complex *f, __CLPK_complex *g, __CLPK_real *cs, __CLPK_complex *sn,
704   __CLPK_complex *r__);
705
706 /* Subroutine */ int clartv(__CLPK_integer *n, __CLPK_complex *x, __CLPK_integer *incx, __CLPK_complex
707   *
708   y, __CLPK_integer *incy, __CLPK_real *c__, __CLPK_complex *s, __CLPK_integer *incc);
709
710 /* Subroutine */ int clarz(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *l,
711   __CLPK_complex *v, __CLPK_integer *incv, __CLPK_complex *tau, __CLPK_complex *c__, __CLPK_integer

```

```

        *ldc,
        __CLPK_complex *work);
693
694
695 /* Subroutine */ int clarzb(char *side, char *trans, char *direct, char *
696   storev, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *l, __CLPK_complex
        *v,
697   __CLPK_integer *ldv, __CLPK_complex *t, __CLPK_integer *ldt, __CLPK_complex *c_, __CLPK_integer
        *ldc,
698   __CLPK_complex *work, __CLPK_integer *ldwork);
699
700 /* Subroutine */ int clarzt(char *direct, char *storev, __CLPK_integer *n, __CLPK_integer *
701   k, __CLPK_complex *v, __CLPK_integer *ldv, __CLPK_complex *tau, __CLPK_complex *t, __CLPK_integer
        *ldt);
702
703 /* Subroutine */ int clascl(char *type_, __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_real *
704   cfrom, __CLPK_real *cto, __CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
        *lda,
705   __CLPK_integer *info);
706
707 /* Subroutine */ int claset(char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *
708   alpha, __CLPK_complex *beta, __CLPK_complex *a, __CLPK_integer *lda);
709
710 /* Subroutine */ int clar_ (char *side, char *pivot, char *direct, __CLPK_integer *m,
711   __CLPK_integer *n, __CLPK_real *c_, __CLPK_real *s, __CLPK_complex *a, __CLPK_integer *lda);
712
713 /* Subroutine */ int classq(__CLPK_integer *n, __CLPK_complex *x, __CLPK_integer *incx, __CLPK_real *
714   scale, __CLPK_real *sumsq);
715
716 /* Subroutine */ int claswp(__CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_integer *
717   k1, __CLPK_integer *k2, __CLPK_integer *ipiv, __CLPK_integer *incx);
718
719 /* Subroutine */ int clasyf(char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_integer *kb,
720   __CLPK_complex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_complex *w, __CLPK_integer
        *ldw,
721   __CLPK_integer *info);
722
723 /* Subroutine */ int clatbs(char *uplo, char *trans, char *diag, char *
724   normin, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_complex *ab, __CLPK_integer *ldab,
        __CLPK_complex *
725   x, __CLPK_real *scale, __CLPK_real *cnorm, __CLPK_integer *info);
726
727 /* Subroutine */ int clatdf(__CLPK_integer *ijob, __CLPK_integer *n, __CLPK_complex *z_,
        __CLPK_integer
728   *ldz, __CLPK_complex *rhs, __CLPK_real *rdsum, __CLPK_real *rdscal, __CLPK_integer *ipiv,
        __CLPK_integer
729   *jpiv);
730
731 /* Subroutine */ int clatps(char *uplo, char *trans, char *diag, char *
732   normin, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_complex *x, __CLPK_real *scale, __CLPK_real
        *cnorm,
733   __CLPK_integer *info);
734
735 /* Subroutine */ int clatr_ (char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_complex *a,
736   __CLPK_integer *lda, __CLPK_real *e, __CLPK_complex *tau, __CLPK_complex *w, __CLPK_integer *ldw);
737
738 /* Subroutine */ int clatrs(char *uplo, char *trans, char *diag, char *
739   normin, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *x, __CLPK_real
        *scale,
740   __CLPK_real *cnorm, __CLPK_integer *info);
741
742 /* Subroutine */ int clatr_ (char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *l, __CLPK_complex *a,
        __CLPK_integer *lda,
743   __CLPK_complex *tau, __CLPK_complex *work);
744
745 /* Subroutine */ int clatzm(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *v,
746   __CLPK_integer *incv, __CLPK_complex *tau, __CLPK_complex *c1, __CLPK_complex *c2, __CLPK_integer
        *ldc,
747   __CLPK_complex *work);
748
749 /* Subroutine */ int clauu2(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
        __CLPK_integer *info);
750
751 /* Subroutine */ int clauum(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
        __CLPK_integer *info);
752
753 /* Subroutine */ int cpbcon(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_complex *ab,
754   __CLPK_integer *ldab, __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_real
        *rwork,
755   __CLPK_integer *info);
756
757 /* Subroutine */ int cpbequ(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_complex *ab,
758   __CLPK_integer *ldab, __CLPK_real *s, __CLPK_real *scond, __CLPK_real *amax, __CLPK_integer *info);
759
760 /* Subroutine */ int cpbrfs(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
761   nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex *afb, __CLPK_integer *ldafb,
        __CLPK_complex *b,
762   __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *ferr,
        __CLPK_real *

```

```

765     berr, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
766
767 /* Subroutine */ int cpbstf_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_complex *ab,
768     __CLPK_integer *ldab, __CLPK_integer *info);
769
770 /* Subroutine */ int cpbsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
771     nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex *b, __CLPK_integer *ldb,
772     __CLPK_integer *
773     info);
774
775 /* Subroutine */ int cpbsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
776     __CLPK_integer *nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex *afb, __CLPK_integer
777     *
778     ldafb, char *equed, __CLPK_real *s, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *x,
779     __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work,
780     __CLPK_real *rwork, __CLPK_integer *info);
781
782 /* Subroutine */ int cpbtf2_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_complex *ab,
783     __CLPK_integer *ldab, __CLPK_integer *info);
784
785 /* Subroutine */ int cpbtrf_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_complex *ab,
786     __CLPK_integer *ldab, __CLPK_integer *info);
787
788 /* Subroutine */ int cpbtrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
789     nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex *b, __CLPK_integer *ldb,
790     __CLPK_integer *
791     info);
792
793 /* Subroutine */ int cpocon_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
794     __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer
795     *info);
796
797 /* Subroutine */ int cpoequ_(__CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_real *s,
798     __CLPK_real *scond, __CLPK_real *amax, __CLPK_integer *info);
799
800 /* Subroutine */ int cporfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
801     a, __CLPK_integer *lda, __CLPK_complex *af, __CLPK_integer *ldaf, __CLPK_complex *b, __CLPK_integer
802     *ldb,
803     __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work,
804     __CLPK_real *rwork,
805     __CLPK_integer *info);
806
807 /* Subroutine */ int cposv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *a,
808     __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
809
810 /* Subroutine */ int cposvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
811     nrhs, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *af, __CLPK_integer *ldaf, char *
812     equed, __CLPK_real *s, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer
813     *ldx,
814     __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_real *rwork,
815     __CLPK_integer *info);
816
817 /* Subroutine */ int cpotf2_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
818     __CLPK_integer *info);
819
820 /* Subroutine */ int cpotrf_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
821     __CLPK_integer *info);
822
823 /* Subroutine */ int cpotri_(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
824     __CLPK_integer *info);
825
826 /* Subroutine */ int cpotrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
827     a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
828
829 /* Subroutine */ int cppcon_(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_real *anorm,
830     __CLPK_real *rcond, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
831
832 /* Subroutine */ int cppequ_(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_real *s,
833     __CLPK_real *scond, __CLPK_real *amax, __CLPK_integer *info);
834
835 /* Subroutine */ int cprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
836     ap, __CLPK_complex *afp, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer
837     *ldx,
838     __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer
839     *info);
840
841 /* Subroutine */ int cppsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
842     ap, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
843
844 /* Subroutine */ int cppsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
845     nrhs, __CLPK_complex *ap, __CLPK_complex *afp, char *equed, __CLPK_real *s, __CLPK_complex *b,
846     __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr,
847     __CLPK_real
848     *berr, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
849
850 /* Subroutine */ int cpptrf_(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_integer *

```

```

841     info);
842
843 /* Subroutine */ int cpptri_(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_integer *
844     info);
845
846 /* Subroutine */ int cpptrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
847     ap, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
848
849 /* Subroutine */ int cptcon__(__CLPK_integer *n, __CLPK_real *d__, __CLPK_complex *e, __CLPK_real *anorm,
850     __CLPK_real *rcond, __CLPK_real *rwork, __CLPK_integer *info);
851
852 /* Subroutine */ int cptrfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *d__,
853     __CLPK_complex *e, __CLPK_real *df, __CLPK_complex *ef, __CLPK_complex *b, __CLPK_integer *ldb,
854     __CLPK_complex
855     *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_real
856     *rwork,
857     __CLPK_integer *info);
858
859 /* Subroutine */ int cptsv__(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *d__, __CLPK_complex
860     *e,
861     __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
862
863 /* Subroutine */ int cptsvx_(char *fact, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *d__,
864     __CLPK_complex *e, __CLPK_real *df, __CLPK_complex *ef, __CLPK_complex *b, __CLPK_integer *ldb,
865     __CLPK_complex
866     *x, __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex
867     *work,
868     __CLPK_real *rwork, __CLPK_integer *info);
869
870 /* Subroutine */ int cptrf__(__CLPK_integer *n, __CLPK_real *d__, __CLPK_complex *e, __CLPK_integer
871     *info);
872
873 /* Subroutine */ int cptrfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *d__,
874     __CLPK_complex *e, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
875
876 /* Subroutine */ int cptts2__(__CLPK_integer *iuplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real
877     *
878     d__, __CLPK_complex *e, __CLPK_complex *b, __CLPK_integer *ldb);
879
880 /* Subroutine */ int crot__(__CLPK_integer *n, __CLPK_complex *cx, __CLPK_integer *incx, __CLPK_complex *
881     cy,
882     __CLPK_integer *incy, __CLPK_real *c__, __CLPK_complex *s);
883
884 /* Subroutine */ int cspcon_(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_integer *
885     ipiv,
886     __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_integer *info);
887
888 /* Subroutine */ int cspmv_(char *uplo, __CLPK_integer *n, __CLPK_complex *alpha, __CLPK_complex *
889     ap,
890     __CLPK_complex *x, __CLPK_integer *incx, __CLPK_complex *beta, __CLPK_complex *y, __CLPK_integer
891     *
892     incy);
893
894 /* Subroutine */ int cspr_(char *uplo, __CLPK_integer *n, __CLPK_complex *alpha, __CLPK_complex *x,
895     __CLPK_integer
896     *incx, __CLPK_complex *ap);
897
898 /* Subroutine */ int csprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
899     ap,
900     __CLPK_complex *afp, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb,
901     __CLPK_complex *x,
902     __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_real
903     *rwork,
904     __CLPK_integer *info);
905
906 /* Subroutine */ int cspsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
907     ap,
908     __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
909
910 /* Subroutine */ int cspsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
911     nrhs,
912     __CLPK_complex *ap, __CLPK_complex *afp, __CLPK_integer *ipiv, __CLPK_complex *b,
913     __CLPK_integer *
914     ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real
915     *berr,
916     __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
917
918 /* Subroutine */ int csprtf_(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_integer *
919     ipiv,
920     __CLPK_integer *info);
921
922 /* Subroutine */ int csptri_(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_integer *
923     ipiv,
924     __CLPK_complex *work, __CLPK_integer *info);
925
926 /* Subroutine */ int csptrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
927     ap,
928     __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer *info);
929
930 /* Subroutine */ int csrot__(__CLPK_integer *n, __CLPK_complex *cx, __CLPK_integer *incx, __CLPK_complex
931     *
932     cy, __CLPK_integer *incy, __CLPK_real *c__, __CLPK_real *s);
933
934 /* Subroutine */ int csrsl__(__CLPK_integer *n, __CLPK_real *sa, __CLPK_complex *sx, __CLPK_integer
935     *
936     incx);
937
938

```

```

913 /* Subroutine */ int cstedc(char *compz, __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e,
914   __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *
915   rwork, __CLPK_integer *lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *
916   info);
917
918 /* Subroutine */ int cstein(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e, __CLPK_integer *m,
   __CLPK_real
919   *w, __CLPK_integer *iblock, __CLPK_integer *isplit, __CLPK_complex *z__, __CLPK_integer *ldz,
920   __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
921
922 /* Subroutine */ int csteqr(char *compz, __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e,
923   __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *info);
924
925 /* Subroutine */ int csycon(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
926   __CLPK_integer *ipiv, __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_integer
   *
927   info);
928
929 /* Subroutine */ int csymv(char *uplo, __CLPK_integer *n, __CLPK_complex *alpha, __CLPK_complex *
930   a, __CLPK_integer *lda, __CLPK_complex *x, __CLPK_integer *incx, __CLPK_complex *beta,
931   __CLPK_complex *y,
932   __CLPK_integer *incy);
933 /* Subroutine */ int csyr(char *uplo, __CLPK_integer *n, __CLPK_complex *alpha, __CLPK_complex *x,
934   __CLPK_integer *incx, __CLPK_complex *a, __CLPK_integer *lda);
935
936 /* Subroutine */ int csyrfs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
937   a, __CLPK_integer *lda, __CLPK_complex *af, __CLPK_integer *ldaf, __CLPK_integer *ipiv,
938   __CLPK_complex *
939   b, __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real
   *berr,
940   __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
941 /* Subroutine */ int csysv(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *a,
942   __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
   *work,
943   __CLPK_integer *lwork, __CLPK_integer *info);
944
945 /* Subroutine */ int csysvx(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
946   nrhs, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *af, __CLPK_integer *ldaf,
947   __CLPK_integer *
948   ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real
   *rcond,
949   __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real
   *rwork,
950   __CLPK_integer *info);
951 /* Subroutine */ int csytf2(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
952   __CLPK_integer *ipiv, __CLPK_integer *info);
953
954 /* Subroutine */ int csytrf(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
955   __CLPK_integer *ipiv, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
956
957 /* Subroutine */ int csytri(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
958   __CLPK_integer *ipiv, __CLPK_complex *work, __CLPK_integer *info);
959
960 /* Subroutine */ int csytrs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex *
961   a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer
   *
962   info);
963
964 /* Subroutine */ int ctbcon(char *norm, char *uplo, char *diag, __CLPK_integer *n,
965   __CLPK_integer *kd, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *rcond, __CLPK_complex
   *work,
966   __CLPK_real *rwork, __CLPK_integer *info);
967
968 /* Subroutine */ int ctbrfs(char *uplo, char *trans, char *diag, __CLPK_integer *n,
969   __CLPK_integer *kd, __CLPK_integer *nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex
   *b,
970   __CLPK_integer *ldb, __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr,
971   __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
972
973 /* Subroutine */ int ctbtrs(char *uplo, char *trans, char *diag, __CLPK_integer *n,
974   __CLPK_integer *kd, __CLPK_integer *nrhs, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex
   *b,
975   __CLPK_integer *ldb, __CLPK_integer *info);
976
977 /* Subroutine */ int ctgvec(char *side, char *howmny, __CLPK_logical *select,
978   __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb,
979   __CLPK_complex *v1, __CLPK_integer *ldv1, __CLPK_complex *vr, __CLPK_integer *ldvr, __CLPK_integer
   *mm,
980   __CLPK_integer *m, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *info);
981
982 /* Subroutine */ int ctgex2(__CLPK_logical *wantq, __CLPK_logical *wantz, __CLPK_integer *n,
983   __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *q,
984   __CLPK_integer *ldq, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_integer *j1, __CLPK_integer
   *info);

```

```

985
986 /* Subroutine */ int ctgexc(__CLPK_logical *wantq, __CLPK_logical *wantz, __CLPK_integer *n,
987   __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex *q,
988   __CLPK_integer *ldq, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_integer *ifst, __CLPK_integer
   *
989   ilst, __CLPK_integer *info);
990
991 /* Subroutine */ int ctgsen(__CLPK_integer *ijob, __CLPK_logical *wantq, __CLPK_logical *wantz,
992   __CLPK_logical *select, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex
   *b,
993   __CLPK_integer *ldb, __CLPK_complex *alpha, __CLPK_complex *beta, __CLPK_complex *q, __CLPK_integer
   *ldq,
994   __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_integer *m, __CLPK_real *pl, __CLPK_real *pr,
   __CLPK_real *
995   dif, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork,
996   __CLPK_integer *info);
997
998 /* Subroutine */ int ctgsja(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
999   __CLPK_integer *p, __CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *l, __CLPK_complex *a,
   __CLPK_integer *
1000   lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_real *tola, __CLPK_real *tolb, __CLPK_real
   *alpha,
1001   __CLPK_real *beta, __CLPK_complex *u, __CLPK_integer *ldu, __CLPK_complex *v, __CLPK_integer *ldv,
1002   __CLPK_complex *q, __CLPK_integer *ldq, __CLPK_complex *work, __CLPK_integer *ncycle,
   __CLPK_integer *
1003   info);
1004
1005 /* Subroutine */ int ctgsna(char *job, char *howmny, __CLPK_logical *select,
1006   __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb,
   __CLPK_complex *
1007   vl, __CLPK_integer *ldvl, __CLPK_complex *vr, __CLPK_integer *ldvr, __CLPK_real *s,
   __CLPK_real
1008   *dif, __CLPK_integer *mm, __CLPK_integer *m, __CLPK_complex *work, __CLPK_integer *lwork,
   __CLPK_integer
1009   *iwork, __CLPK_integer *info);
1010
1011 /* Subroutine */ int ctgsy2(char *trans, __CLPK_integer *ijob, __CLPK_integer *m, __CLPK_integer *
1012   n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
   *c__,
1013   __CLPK_integer *ldc, __CLPK_complex *d__, __CLPK_integer *ldd, __CLPK_complex *e, __CLPK_integer
   *lde,
1014   __CLPK_complex *f, __CLPK_integer *ldf, __CLPK_real *scale, __CLPK_real *rdsum, __CLPK_real
   *rdscale,
1015   __CLPK_integer *info);
1016
1017 /* Subroutine */ int ctgsyl(char *trans, __CLPK_integer *ijob, __CLPK_integer *m, __CLPK_integer *
1018   n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
   *c__,
1019   __CLPK_integer *ldc, __CLPK_complex *d__, __CLPK_integer *ldd, __CLPK_complex *e, __CLPK_integer
   *lde,
1020   __CLPK_complex *f, __CLPK_integer *ldf, __CLPK_real *scale, __CLPK_real *dif, __CLPK_complex *work,
   __CLPK_integer
1021   *lwork, __CLPK_integer *iwork, __CLPK_integer *info);
1022
1023 /* Subroutine */ int ctqcon(char *norm, char *uplo, char *diag, __CLPK_integer *n,
1024   __CLPK_complex *ap, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer
   *info);
1025
1026 /* Subroutine */ int ctrfs(char *uplo, char *trans, char *diag, __CLPK_integer *n,
1027   __CLPK_integer *nrhs, __CLPK_complex *ap, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_complex
   *x,
1028   __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work, __CLPK_real
   *rwork,
1029   __CLPK_integer *info);
1030
1031 /* Subroutine */ int ctptri(char *uplo, char *diag, __CLPK_integer *n, __CLPK_complex *ap,
1032   __CLPK_integer *info);
1033
1034 /* Subroutine */ int ctptrs(char *uplo, char *trans, char *diag, __CLPK_integer *n,
1035   __CLPK_integer *nrhs, __CLPK_complex *ap, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_integer
   *info);
1036
1037 /* Subroutine */ int ctrcon(char *norm, char *uplo, char *diag, __CLPK_integer *n,
1038   __CLPK_complex *a, __CLPK_integer *lda, __CLPK_real *rcond, __CLPK_complex *work, __CLPK_real
   *rwork,
1039   __CLPK_integer *info);
1040
1041 /* Subroutine */ int ctrevc(char *side, char *howmny, __CLPK_logical *select,
1042   __CLPK_integer *n, __CLPK_complex *t, __CLPK_integer *ldt, __CLPK_complex *vl, __CLPK_integer
   *ldvl,
1043   __CLPK_complex *vr, __CLPK_integer *ldvr, __CLPK_integer *mm, __CLPK_integer *m, __CLPK_complex
   *work,
1044   __CLPK_real *rwork, __CLPK_integer *info);
1045
1046 /* Subroutine */ int ctrex_ (char *compq, __CLPK_integer *n, __CLPK_complex *t, __CLPK_integer *
1047   ldt, __CLPK_complex *q, __CLPK_integer *ldq, __CLPK_integer *ifst, __CLPK_integer *ilst,
   __CLPK_integer *
1048   info);

```



```

1049
1050 /* Subroutine */ int ctrrfs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
1051   __CLPK_integer *nrhs, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer
   *ldb,
1052   __CLPK_complex *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_complex *work,
   __CLPK_real
1053   *rwork, __CLPK_integer *info);
1054
1055 /* Subroutine */ int ctrsen_(char *job, char *compq, __CLPK_logical *select, __CLPK_integer
1056   *n, __CLPK_complex *t, __CLPK_integer *ldt, __CLPK_complex *q, __CLPK_integer *ldq, __CLPK_complex
   *w,
1057   __CLPK_integer *m, __CLPK_real *s, __CLPK_real *sep, __CLPK_complex *work, __CLPK_integer *lwork,
   __CLPK_integer *info);
1059
1060 /* Subroutine */ int ctrsna_(char *job, char *howmny, __CLPK_logical *select,
1061   __CLPK_integer *n, __CLPK_complex *t, __CLPK_integer *ldt, __CLPK_complex *vl, __CLPK_integer
   *ldvl,
1062   __CLPK_complex *vr, __CLPK_integer *ldvr, __CLPK_real *s, __CLPK_real *sep, __CLPK_integer *mm,
   __CLPK_integer *
1063   m, __CLPK_complex *work, __CLPK_integer *ldwork, __CLPK_real *rwork, __CLPK_integer *info);
1064
1065 /* Subroutine */ int ctrsyl_(char *trana, char *tranb, __CLPK_integer *isgn, __CLPK_integer
1066   *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer
   *ldb,
1067   __CLPK_complex *c, __CLPK_integer *ldc, __CLPK_real *scale, __CLPK_integer *info);
1068
1069 /* Subroutine */ int ctrti2_(char *uplo, char *diag, __CLPK_integer *n, __CLPK_complex *a,
1070   __CLPK_integer *lda, __CLPK_integer *info);
1071
1072 /* Subroutine */ int ctrtri_(char *uplo, char *diag, __CLPK_integer *n, __CLPK_complex *a,
1073   __CLPK_integer *lda, __CLPK_integer *info);
1074
1075 /* Subroutine */ int ctrtrs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
1076   __CLPK_integer *nrhs, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer
   *ldb,
1077   __CLPK_integer *info);
1078
1079 /* Subroutine */ int ctzrqf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
   *lda,
1080   __CLPK_complex *tau, __CLPK_integer *info);
1081
1082 /* Subroutine */ int ctzrzf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer
   *lda,
1083   __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
1084
1085 /* Subroutine */ int cung2l_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex
   *a,
1086   __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
1087
1088 /* Subroutine */ int cung2r_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex
   *a,
1089   __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
1090
1091 /* Subroutine */ int cungbr_(char *vect, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
1092   __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer
   *lwork,
1093   __CLPK_integer *info);
1094
1095 /* Subroutine */ int cunghr_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi,
   __CLPK_complex *
1096   a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork,
   __CLPK_integer
1097   *info);
1098
1099 /* Subroutine */ int cungl2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex
   *a,
1100   __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
1101
1102 /* Subroutine */ int cunglq_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex
   *a,
1103   __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork,
   __CLPK_integer *
1104   info);
1105
1106 /* Subroutine */ int cungql_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex
   *a,
1107   __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork,
   __CLPK_integer *
1108   info);
1109
1110 /* Subroutine */ int cungqr_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex
   *a,
1111   __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork,
   __CLPK_integer *
1112   info);
1113
1114 /* Subroutine */ int cungr2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex

```



```

    *a,
1115     __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *info);
1116
1117 /* Subroutine */ int cungrq__(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex
    *a,
1118     __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork,
    __CLPK_integer *
1119     info);
1120
1121 /* Subroutine */ int cungrt__(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
    __CLPK_complex *tau, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
1122
1123
1124 /* Subroutine */ int cunm2l__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1125     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *info);
1126
1127
1128 /* Subroutine */ int cunm2r__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1129     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *info);
1130
1131
1132 /* Subroutine */ int cunmbr__(char *vect, char *side, char *trans, __CLPK_integer *m,
    __CLPK_integer *n, __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau,
    __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *lwork,
    __CLPK_integer *
1133     info);
1134
1135
1136 /* Subroutine */ int cunmhr__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex
    *tau,
1137     __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *lwork,
    __CLPK_integer *
1138     info);
1139
1140
1141 /* Subroutine */ int cunml2__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1142     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *info);
1143
1144
1145 /* Subroutine */ int cunmlq__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1146     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
1147
1148
1149 /* Subroutine */ int cunmq1__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1150     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
1151
1152
1153 /* Subroutine */ int cunmqr__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1154     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
1155
1156
1157 /* Subroutine */ int cunmr2__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1158     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *info);
1159
1160
1161 /* Subroutine */ int cunmr3__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_integer *l, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau,
    __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *info);
1162
1163
1164 /* Subroutine */ int cunmrq__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1165     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
1166
1167
1168 /* Subroutine */ int cunmrz__(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
    __CLPK_integer *k, __CLPK_integer *l, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau,
    __CLPK_complex *c__, __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *lwork,
    __CLPK_integer *
1169     info);
1170
1171
1172 /* Subroutine */ int cunmtr__(char *side, char *uplo, char *trans, __CLPK_integer *m,
    __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_complex *tau, __CLPK_complex
    *c__,
1173     __CLPK_integer *ldc, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_integer *info);
1174
1175
1176 /* Subroutine */ int cupgtr__(char *uplo, __CLPK_integer *n, __CLPK_complex *ap, __CLPK_complex *
    tau, __CLPK_complex *q, __CLPK_integer *ldq, __CLPK_complex *work, __CLPK_integer *info);
1177
1178
1179 /* Subroutine */ int cupmtr__(char *side, char *uplo, char *trans, __CLPK_integer *m,
    __CLPK_integer *n, __CLPK_complex *ap, __CLPK_complex *tau, __CLPK_complex *c__, __CLPK_integer
    *ldc,
1180     __CLPK_complex *work, __CLPK_integer *info);
1181
1182
1183
1184

```

```

1185
1186 /* Subroutine */ int dbdsdc(char *uplo, char *compq, __CLPK_integer *n, __CLPK_doublereal *
1187   d, __CLPK_doublereal *e, __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *vt,
1188   __CLPK_integer *ldvt, __CLPK_doublereal *q, __CLPK_integer *iq, __CLPK_doublereal *work,
   __CLPK_integer *
1189   iwork, __CLPK_integer *info);
1190
1191 /* Subroutine */ int dbdsqr(char *uplo, __CLPK_integer *n, __CLPK_integer *ncvt, __CLPK_integer *
1192   nru, __CLPK_integer *ncc, __CLPK_doublereal *d, __CLPK_doublereal *e, __CLPK_doublereal *vt,
1193   __CLPK_integer *ldvt, __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *c,
   __CLPK_integer *
1194   ldc, __CLPK_doublereal *work, __CLPK_integer *info);
1195
1196 /* Subroutine */ int ddisna(char *job, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *
1197   d, __CLPK_doublereal *sep, __CLPK_integer *info);
1198
1199 /* Subroutine */ int dgbrbd(char *vect, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *ncc,
1200   __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_doublereal *ab, __CLPK_integer *ldab,
   __CLPK_doublereal *
1201   d, __CLPK_doublereal *e, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *pt,
1202   __CLPK_integer *ldpt, __CLPK_doublereal *c, __CLPK_integer *ldc, __CLPK_doublereal *work,
1203   __CLPK_integer *info);
1204
1205 /* Subroutine */ int dgbrcon(char *norm, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku,
1206   __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_doublereal *anorm,
1207   __CLPK_doublereal *rcond, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
1208
1209 /* Subroutine */ int dgbequ(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
   *ku,
1210   __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *r, __CLPK_doublereal *c,
1211   __CLPK_doublereal *rowcnd, __CLPK_doublereal *colcnd, __CLPK_doublereal *amax, __CLPK_integer *
   info);
1212
1213
1214 /* Subroutine */ int dgbrfs(char *trans, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *
1215   ku, __CLPK_integer *nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *afb,
1216   __CLPK_integer *ldafb, __CLPK_integer *ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb,
1217   __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
1218   __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
1219
1220 /* Subroutine */ int dgbsv(__CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_integer *
1221   nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_doublereal *b,
1222   __CLPK_integer *ldb, __CLPK_integer *info);
1223
1224 /* Subroutine */ int dgbsvx(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *kl,
1225   __CLPK_integer *ku, __CLPK_integer *nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab,
1226   __CLPK_doublereal *afb, __CLPK_integer *ldafb, __CLPK_integer *ipiv, char *equed,
1227   __CLPK_doublereal *r, __CLPK_doublereal *c, __CLPK_doublereal *b, __CLPK_integer *ldb,
1228   __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *ferr,
1229   __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
1230
1231 /* Subroutine */ int dgbt2f2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
   *ku,
1232   __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_integer *info);
1233
1234 /* Subroutine */ int dgbrtf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
   *ku,
1235   __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_integer *info);
1236
1237 /* Subroutine */ int dgbrts(char *trans, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *
1238   ku, __CLPK_integer *nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv,
1239   __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
1240
1241 /* Subroutine */ int dgebak(char *job, char *side, __CLPK_integer *n, __CLPK_integer *ilo,
1242   __CLPK_integer *ihi, __CLPK_doublereal *scale, __CLPK_integer *m, __CLPK_doublereal *v,
   __CLPK_integer *
1243   ldv, __CLPK_integer *info);
1244
1245 /* Subroutine */ int dgebal(char *job, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
1246   lda, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *scale, __CLPK_integer *info);
1247
1248 /* Subroutine */ int dgebd2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
   *
1249   lda, __CLPK_doublereal *d, __CLPK_doublereal *e, __CLPK_doublereal *tauq, __CLPK_doublereal *
1250   tau, __CLPK_doublereal *work, __CLPK_integer *info);
1251
1252 /* Subroutine */ int dgebrd(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
   *
1253   lda, __CLPK_doublereal *d, __CLPK_doublereal *e, __CLPK_doublereal *tauq, __CLPK_doublereal *
1254   tau, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1255
1256 /* Subroutine */ int dgecon(char *norm, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
1257   lda, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *work, __CLPK_integer *
1258   iwork, __CLPK_integer *info);
1259
1260 /* Subroutine */ int dgeequ(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
   *
1261   lda, __CLPK_doublereal *r, __CLPK_doublereal *c, __CLPK_doublereal *rowcnd, __CLPK_doublereal

```

```

1262     *colcnd, __CLPK_doublereal *amax, __CLPK_integer *info);
1263
1264 /* Subroutine */ int dgees_(char *jobvs, char *sort, __CLPK_L_fp select, __CLPK_integer *n,
1265     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_integer *sdim, __CLPK_doublereal *wr,
1266     __CLPK_doublereal *wi, __CLPK_doublereal *vs, __CLPK_integer *ldvs, __CLPK_doublereal *work,
1267     __CLPK_integer *lwork, __CLPK_logical *bwork, __CLPK_integer *info);
1268
1269 /* Subroutine */ int dgeesx_(char *jobvs, char *sort, __CLPK_L_fp select, char *
1270     sense, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_integer *sdim,
1271     __CLPK_doublereal *wr, __CLPK_doublereal *wi, __CLPK_doublereal *vs, __CLPK_integer *ldvs,
1272     __CLPK_doublereal *rconde, __CLPK_doublereal *rcondv, __CLPK_doublereal *work, __CLPK_integer *
1273     lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_logical *bwork, __CLPK_integer *info);
1274
1275 /* Subroutine */ int dgeev_(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_doublereal *
1276     a, __CLPK_integer *lda, __CLPK_doublereal *wr, __CLPK_doublereal *wi, __CLPK_doublereal *vl,
1277     __CLPK_integer *ldvl, __CLPK_doublereal *vr, __CLPK_integer *ldvr, __CLPK_doublereal *work,
1278     __CLPK_integer *lwork, __CLPK_integer *info);
1279
1280 /* Subroutine */ int dgeevx_(char *balanc, char *jobvl, char *jobvr, char *
1281     sense, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *wr,
1282     __CLPK_doublereal *wi, __CLPK_doublereal *vl, __CLPK_integer *ldvl, __CLPK_doublereal *vr,
1283     __CLPK_integer *ldvr, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *scale,
1284     __CLPK_doublereal *abnrm, __CLPK_doublereal *rconde, __CLPK_doublereal *rcondv, __CLPK_doublereal
1285     *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *info);
1286
1287 /* Subroutine */ int dgegs_(char *jobvsl, char *jobvsr, __CLPK_integer *n,
1288     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
1289     __CLPK_doublereal *
1290     alphar, __CLPK_doublereal *alphai, __CLPK_doublereal *beta, __CLPK_doublereal *vsl,
1291     __CLPK_integer *ldvsl, __CLPK_doublereal *vsr, __CLPK_integer *ldvsr, __CLPK_doublereal *work,
1292     __CLPK_integer *lwork, __CLPK_integer *info);
1293
1294 /* Subroutine */ int dgegv_(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_doublereal *
1295     a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *alphar,
1296     __CLPK_doublereal *alphai, __CLPK_doublereal *beta, __CLPK_doublereal *vl, __CLPK_integer *ldvl,
1297     __CLPK_doublereal *vr, __CLPK_integer *ldvr, __CLPK_doublereal *work, __CLPK_integer *lwork,
1298     __CLPK_integer *info);
1299
1300 /* Subroutine */ int dgehd2_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi,
1301     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work,
1302     __CLPK_integer *info);
1303
1304 /* Subroutine */ int dgehrd_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi,
1305     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work,
1306     __CLPK_integer *lwork, __CLPK_integer *info);
1307 /* Subroutine */ int dgelq2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
1308     *
1309     lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *info);
1310 /* Subroutine */ int dgelqf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
1311     *
1312     lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1313
1314 /* Subroutine */ int dgels_(char *trans, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *
1315     nrhs, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
1316     __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1317
1318 /* Subroutine */ int dgelsd_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs,
1319     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
1320     __CLPK_doublereal *
1321     s, __CLPK_doublereal *rcond, __CLPK_integer *rank, __CLPK_doublereal *work, __CLPK_integer *lwork,
1322     __CLPK_integer *iwork, __CLPK_integer *info);
1323
1324 /* Subroutine */ int dgelss_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs,
1325     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
1326     __CLPK_doublereal *
1327     s, __CLPK_doublereal *rcond, __CLPK_integer *rank, __CLPK_doublereal *work, __CLPK_integer *lwork,
1328     __CLPK_integer *info);
1329
1330 /* Subroutine */ int dgelsx_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs,
1331     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
1332     __CLPK_integer *
1333     jpvt, __CLPK_doublereal *rcond, __CLPK_integer *rank, __CLPK_doublereal *work, __CLPK_integer *
1334     lwork, __CLPK_integer *info);
1335
1336 /* Subroutine */ int dgeql2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
1337     *
1338     lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *info);
1339
1340 /* Subroutine */ int dgeqlf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer

```

```
*
1341   lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1342
1343 /* Subroutine */ int dgeqp3(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1344   lda, __CLPK_integer *jpvvt, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork,
1345   __CLPK_integer *info);
1346
1347 /* Subroutine */ int dgeqpf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1348   lda, __CLPK_integer *jpvvt, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *info);
1349
1350 /* Subroutine */ int dgeqr2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1351   lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *info);
1352
1353 /* Subroutine */ int dgeqrf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1354   lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1355
1356 /* Subroutine */ int dgerfs(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
1357   __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *af, __CLPK_integer *ldaf,
1358   __CLPK_integer *
1359   ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
1360   __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork,
1361   __CLPK_integer *info);
1362 /* Subroutine */ int dgerq2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1363   lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *info);
1364
1365 /* Subroutine */ int dgerqf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1366   lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1367
1368 /* Subroutine */ int dgesc2(__CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda,
1369   __CLPK_doublereal *rhs, __CLPK_integer *ipiv, __CLPK_integer *jpiv, __CLPK_doublereal *scale);
1370
1371 /* Subroutine */ int dgesdd(char *jobz, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *
1372   a, __CLPK_integer *lda, __CLPK_doublereal *s, __CLPK_doublereal *u, __CLPK_integer *ldu,
1373   __CLPK_doublereal *vt, __CLPK_integer *ldvt, __CLPK_doublereal *work, __CLPK_integer *lwork,
1374   __CLPK_integer *iwork, __CLPK_integer *info);
1375
1376 /* Subroutine */ int dgesv(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal *a,
1377   __CLPK_integer
1378   *lda, __CLPK_integer *ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
1379 /* Subroutine */ int dgesvd(char *jobu, char *jobvt, __CLPK_integer *m, __CLPK_integer *n,
1380   __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *s, __CLPK_doublereal *u,
1381   __CLPK_integer *
1382   ldu, __CLPK_doublereal *vt, __CLPK_integer *ldvt, __CLPK_doublereal *work, __CLPK_integer *lwork,
1383   __CLPK_integer *info);
1384 /* Subroutine */ int dgesvx(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *
1385   nrhs, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *af, __CLPK_integer *ldaf,
1386   __CLPK_integer *ipiv, char *equed, __CLPK_doublereal *r__, __CLPK_doublereal *c__,
1387   __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
1388   __CLPK_doublereal *
1389   rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *
1390   iwork, __CLPK_integer *info);
1391 /* Subroutine */ int dgetc2(__CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda,
1392   __CLPK_integer
1393   *ipiv, __CLPK_integer *jpiv, __CLPK_integer *info);
1394 /* Subroutine */ int dgetf2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1395   lda, __CLPK_integer *ipiv, __CLPK_integer *info);
1396
1397 /* Subroutine */ int dgetrf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1398   lda, __CLPK_integer *ipiv, __CLPK_integer *info);
1399
1400 /* Subroutine */ int dgetri(__CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda,
1401   __CLPK_integer
1402   *ipiv, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1403 /* Subroutine */ int dgetrs(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
1404   __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublereal *b,
1405   __CLPK_integer *
1406   ldb, __CLPK_integer *info);
1407 /* Subroutine */ int dggbak(char *job, char *side, __CLPK_integer *n, __CLPK_integer *ilo,
1408   __CLPK_integer *ihi, __CLPK_doublereal *lscale, __CLPK_doublereal *rscale, __CLPK_integer *m,
1409   __CLPK_doublereal *v,
1410   __CLPK_integer *ldv, __CLPK_integer *info);
1411 /* Subroutine */ int dggbal(char *job, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
```

```

1412     lda, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *ilo, __CLPK_integer *ihi,
1413     __CLPK_doublereal *lscale, __CLPK_doublereal *rscale, __CLPK_doublereal *work, __CLPK_integer *
1414     info);
1415
1416 /* Subroutine */ int dggex_(char *jobvsl, char *jobvsr, char *sort, __CLPK_L_fp
1417     delctg, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
1418     __CLPK_integer *ldb, __CLPK_integer *sdim, __CLPK_doublereal *alpha, __CLPK_doublereal *alphai,
1419     __CLPK_doublereal *beta, __CLPK_doublereal *vsl, __CLPK_integer *ldvsl, __CLPK_doublereal *vsr,
1420     __CLPK_integer *ldvsr, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_logical *bwork,
1421     __CLPK_integer *info);
1422
1423 /* Subroutine */ int dggesx_(char *jobvsl, char *jobvsr, char *sort, __CLPK_L_fp
1424     delctg, char *sense, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda,
1425     __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *sdim, __CLPK_doublereal *alpha,
1426     __CLPK_doublereal *alphai, __CLPK_doublereal *beta, __CLPK_doublereal *vsl, __CLPK_integer *ldvsl,
1427     __CLPK_doublereal *vsr, __CLPK_integer *ldvsr, __CLPK_doublereal *rconde, __CLPK_doublereal *
1428     rcondv, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *
1429     liwork, __CLPK_logical *bwork, __CLPK_integer *info);
1430
1431 /* Subroutine */ int dggev_(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_doublereal *
1432     a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *alpha,
1433     __CLPK_doublereal *alphai, __CLPK_doublereal *beta, __CLPK_doublereal *vl, __CLPK_integer *ldvl,
1434     __CLPK_doublereal *vr, __CLPK_integer *ldvr, __CLPK_doublereal *work, __CLPK_integer *lwork,
1435     __CLPK_integer *info);
1436
1437 /* Subroutine */ int dggevx_(char *balanc, char *jobvl, char *jobvr, char *
1438     sense, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
1439     __CLPK_integer *ldb, __CLPK_doublereal *alpha, __CLPK_doublereal *alphai, __CLPK_doublereal *
1440     beta, __CLPK_doublereal *vl, __CLPK_integer *ldvl, __CLPK_doublereal *vr, __CLPK_integer *ldvr,
1441     __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *lscale, __CLPK_doublereal *rscale,
1442     __CLPK_doublereal *abnrm, __CLPK_doublereal *bbnrm, __CLPK_doublereal *rconde, __CLPK_doublereal *
1443     rcondv, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_logical *
1444     bwork, __CLPK_integer *info);
1445
1446 /* Subroutine */ int dggglm_(__CLPK_integer *n, __CLPK_integer *m, __CLPK_integer *p, __CLPK_doublereal
1447     *
1448     a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *d__,
1449     __CLPK_doublereal *x, __CLPK_doublereal *y, __CLPK_doublereal *work, __CLPK_integer *lwork,
1450     __CLPK_integer *info);
1451
1452 /* Subroutine */ int dgghrd_(char *compq, char *compz, __CLPK_integer *n, __CLPK_integer *
1453     ilo, __CLPK_integer *ihi, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
1454     __CLPK_integer *ldb, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *z__,
1455     __CLPK_integer *
1456     ldz, __CLPK_integer *info);
1457
1458 /* Subroutine */ int dggls_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *p, __CLPK_doublereal
1459     *
1460     a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *c__,
1461     __CLPK_doublereal *d__, __CLPK_doublereal *x, __CLPK_doublereal *work, __CLPK_integer *lwork,
1462     __CLPK_integer *info);
1463
1464 /* Subroutine */ int dggqrf_(__CLPK_integer *n, __CLPK_integer *m, __CLPK_integer *p, __CLPK_doublereal
1465     *
1466     a, __CLPK_integer *lda, __CLPK_doublereal *taua, __CLPK_doublereal *b, __CLPK_integer *ldb,
1467     __CLPK_doublereal *taub, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1468
1469 /* Subroutine */ int dggrqf_(__CLPK_integer *m, __CLPK_integer *p, __CLPK_integer *n, __CLPK_doublereal
1470     *
1471     a, __CLPK_integer *lda, __CLPK_doublereal *taua, __CLPK_doublereal *b, __CLPK_integer *ldb,
1472     __CLPK_doublereal *taub, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1473
1474 /* Subroutine */ int dggsvd_(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
1475     __CLPK_integer *n, __CLPK_integer *p, __CLPK_integer *k, __CLPK_integer *l, __CLPK_doublereal *a,
1476     __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *alpha,
1477     __CLPK_doublereal *beta, __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *v,
1478     __CLPK_integer
1479     *ldv, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *work, __CLPK_integer *iwork,
1480     __CLPK_integer *info);
1481
1482 /* Subroutine */ int dggsvp_(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
1483     __CLPK_integer *p, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal
1484     *b,
1485     __CLPK_integer *ldb, __CLPK_doublereal *tola, __CLPK_doublereal *tolb, __CLPK_integer *k,
1486     __CLPK_integer
1487     *l, __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *v, __CLPK_integer *ldv,
1488     __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_integer *iwork, __CLPK_doublereal *tau,
1489     __CLPK_doublereal *work, __CLPK_integer *info);
1490
1491 /* Subroutine */ int dgtcon_(char *norm, __CLPK_integer *n, __CLPK_doublereal *dl,
1492     __CLPK_doublereal *d__, __CLPK_doublereal *du, __CLPK_doublereal *du2, __CLPK_integer *ipiv,
1493     __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *work, __CLPK_integer *
1494     iwork, __CLPK_integer *info);
1495
1496 /* Subroutine */ int dgtrfs_(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
1497     __CLPK_doublereal *dl, __CLPK_doublereal *d__, __CLPK_doublereal *du, __CLPK_doublereal *dlf,
1498     __CLPK_doublereal *df, __CLPK_doublereal *duf, __CLPK_doublereal *du2, __CLPK_integer *ipiv,

```

```

1491   __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
1492   __CLPK_doublereal *
1493   ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *
1494   info);
1495 /* Subroutine */ int dgtsv__(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal *dl,
1496   __CLPK_doublereal *d__, __CLPK_doublereal *du, __CLPK_doublereal *b, __CLPK_integer *ldb,
1497   __CLPK_integer
1498   *info);
1499 /* Subroutine */ int dgtsvx__(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *
1500   nrhs, __CLPK_doublereal *dl, __CLPK_doublereal *d__, __CLPK_doublereal *du, __CLPK_doublereal *
1501   dlf, __CLPK_doublereal *df, __CLPK_doublereal *duf, __CLPK_doublereal *du2, __CLPK_integer *ipiv,
1502   __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
1503   __CLPK_doublereal *
1504   rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *
1505   iwork, __CLPK_integer *info);
1506 /* Subroutine */ int dgtrf__(__CLPK_integer *n, __CLPK_doublereal *dl, __CLPK_doublereal *d__,
1507   __CLPK_doublereal *du, __CLPK_doublereal *du2, __CLPK_integer *ipiv, __CLPK_integer *info);
1508
1509 /* Subroutine */ int dgtrr__(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
1510   __CLPK_doublereal *dl, __CLPK_doublereal *d__, __CLPK_doublereal *du, __CLPK_doublereal *du2,
1511   __CLPK_integer *ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
1512
1513 /* Subroutine */ int dgts2__(__CLPK_integer *itrans, __CLPK_integer *n, __CLPK_integer *nrhs,
1514   __CLPK_doublereal *dl, __CLPK_doublereal *d__, __CLPK_doublereal *du, __CLPK_doublereal *du2,
1515   __CLPK_integer *ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb);
1516
1517 /* Subroutine */ int dhgeqz__(char *job, char *compq, char *compz, __CLPK_integer *n,
1518   __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *a, __CLPK_integer *lda,
1519   __CLPK_doublereal *
1520   b, __CLPK_integer *ldb, __CLPK_doublereal *alphar, __CLPK_doublereal *alphai, __CLPK_doublereal *
1521   beta, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *z__, __CLPK_integer *ldz,
1522   __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
1523
1524 /* Subroutine */ int dhsein__(char *side, char *eigsrc, char *initv, __CLPK_logical *
1525   select, __CLPK_integer *n, __CLPK_doublereal *h__, __CLPK_integer *ldh, __CLPK_doublereal *wr,
1526   __CLPK_doublereal *wi, __CLPK_doublereal *vl, __CLPK_integer *ldvl, __CLPK_doublereal *vr,
1527   __CLPK_integer *ldvr, __CLPK_integer *mm, __CLPK_integer *m, __CLPK_doublereal *work,
1528   __CLPK_integer *
1529   ifail, __CLPK_integer *ifailr, __CLPK_integer *info);
1530
1531 /* Subroutine */ int dhseqr__(char *job, char *compz, __CLPK_integer *n, __CLPK_integer *ilo,
1532   __CLPK_integer *ihi, __CLPK_doublereal *h__, __CLPK_integer *ldh, __CLPK_doublereal *wr,
1533   __CLPK_doublereal *wi, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
1534   __CLPK_integer *lwork, __CLPK_integer *info);
1535
1536 /* Subroutine */ int dlabad__(__CLPK_doublereal *small, __CLPK_doublereal *large);
1537
1538 /* Subroutine */ int dlabrd__(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nb,
1539   __CLPK_doublereal *
1540   a, __CLPK_integer *lda, __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublereal *tauq,
1541   __CLPK_doublereal *taup, __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *y,
1542   __CLPK_integer
1543   *ldy);
1544
1545 /* Subroutine */ int dlacon__(__CLPK_integer *n, __CLPK_doublereal *v, __CLPK_doublereal *x,
1546   __CLPK_integer *isgn, __CLPK_doublereal *est, __CLPK_integer *kase);
1547
1548 /* Subroutine */ int dlacpy__(char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *
1549   a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb);
1550
1551 /* Subroutine */ int dladiv__(__CLPK_doublereal *a, __CLPK_doublereal *b, __CLPK_doublereal *c__,
1552   __CLPK_doublereal *d__, __CLPK_doublereal *p, __CLPK_doublereal *q);
1553
1554 /* Subroutine */ int dlac2__(__CLPK_doublereal *a, __CLPK_doublereal *b, __CLPK_doublereal *c__,
1555   __CLPK_doublereal *rt1, __CLPK_doublereal *rt2);
1556
1557 /* Subroutine */ int dlaebz__(__CLPK_integer *ijob, __CLPK_integer *nitmax, __CLPK_integer *n,
1558   __CLPK_integer *mmax, __CLPK_integer *minp, __CLPK_integer *nbmin, __CLPK_doublereal *abstol,
1559   __CLPK_doublereal *reltol, __CLPK_doublereal *pivmin, __CLPK_doublereal *d__, __CLPK_doublereal *
1560   e, __CLPK_doublereal *e2, __CLPK_integer *nval, __CLPK_doublereal *ab, __CLPK_doublereal *c__,
1561   __CLPK_integer *mout, __CLPK_integer *nab, __CLPK_doublereal *work, __CLPK_integer *iwork,
1562   __CLPK_integer *info);
1563
1564 /* Subroutine */ int dlaed0__(__CLPK_integer *icompq, __CLPK_integer *qsiz, __CLPK_integer *n,
1565   __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublereal *q, __CLPK_integer *ldq,
1566   __CLPK_doublereal *qstore, __CLPK_integer *ldqs, __CLPK_doublereal *work, __CLPK_integer *iwork,
1567   __CLPK_integer *info);
1568
1569 /* Subroutine */ int dlaed1__(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *q,
1570   __CLPK_integer *ldq, __CLPK_integer *indxq, __CLPK_doublereal *rho, __CLPK_integer *cutpnt,
1571   __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
1572
1573 /* Subroutine */ int dlaed2__(__CLPK_integer *k, __CLPK_integer *n, __CLPK_integer *n1,
1574   __CLPK_doublereal *

```

```

1570     d__, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_integer *indxq, __CLPK_doublereal *rho,
1571     __CLPK_doublereal *z__, __CLPK_doublereal *dlamda, __CLPK_doublereal *w, __CLPK_doublereal *q2,
1572     __CLPK_integer *indx, __CLPK_integer *indx, __CLPK_integer *indx, __CLPK_integer *coltyp,
1573     __CLPK_integer *info);
1574
1575 /* Subroutine */ int dlaed3(__CLPK_integer *k, __CLPK_integer *n, __CLPK_integer *n1,
    __CLPK_doublereal *
1576     d__, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *rho, __CLPK_doublereal *dlamda,
1577     __CLPK_doublereal *q2, __CLPK_integer *indx, __CLPK_integer *ctot, __CLPK_doublereal *w,
1578     __CLPK_doublereal *s, __CLPK_integer *info);
1579
1580 /* Subroutine */ int dlaed4(__CLPK_integer *n, __CLPK_integer *i__, __CLPK_doublereal *d__,
1581     __CLPK_doublereal *z__, __CLPK_doublereal *delta, __CLPK_doublereal *rho, __CLPK_doublereal *dlam,
1582     __CLPK_integer *info);
1583
1584 /* Subroutine */ int dlaed5(__CLPK_integer *i__, __CLPK_doublereal *d__, __CLPK_doublereal *z__,
1585     __CLPK_doublereal *delta, __CLPK_doublereal *rho, __CLPK_doublereal *dlam);
1586
1587 /* Subroutine */ int dlaed6(__CLPK_integer *kniter, __CLPK_logical *orgati, __CLPK_doublereal *
1588     rho, __CLPK_doublereal *d__, __CLPK_doublereal *z__, __CLPK_doublereal *finit, __CLPK_doublereal *
1589     tau, __CLPK_integer *info);
1590
1591 /* Subroutine */ int dlaed7(__CLPK_integer *icompq, __CLPK_integer *n, __CLPK_integer *qsiz,
1592     __CLPK_integer *tlvls, __CLPK_integer *curlvl, __CLPK_integer *curpbm, __CLPK_doublereal *d__,
1593     __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_integer *indxq, __CLPK_doublereal *rho,
1594     __CLPK_integer
1595     *cutpnt, __CLPK_doublereal *qstore, __CLPK_integer *qpnr, __CLPK_integer *prmptr, __CLPK_integer *
1596     perm, __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_doublereal *givnum,
1597     __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
1598
1599 /* Subroutine */ int dlaed8(__CLPK_integer *icompq, __CLPK_integer *k, __CLPK_integer *n,
    __CLPK_integer
1600     *qsiz, __CLPK_doublereal *d__, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_integer *indxq,
1601     __CLPK_doublereal *rho, __CLPK_integer *cutpnt, __CLPK_doublereal *z__, __CLPK_doublereal *dlamda,
1602     __CLPK_doublereal *q2, __CLPK_integer *ldq2, __CLPK_doublereal *w, __CLPK_integer *perm,
1603     __CLPK_integer
1604     *givptr, __CLPK_integer *givcol, __CLPK_doublereal *givnum, __CLPK_integer *indx, __CLPK_integer
1605     *info);
1606
1607 /* Subroutine */ int dlaed9(__CLPK_integer *k, __CLPK_integer *kstart, __CLPK_integer *kstop,
    __CLPK_integer *n,
1608     __CLPK_doublereal *d__, __CLPK_doublereal *q, __CLPK_integer *ldq,
    __CLPK_doublereal *
1609     rho, __CLPK_doublereal *dlamda, __CLPK_doublereal *w, __CLPK_doublereal *s, __CLPK_integer *lds,
    __CLPK_integer *info);
1610
1611 /* Subroutine */ int dlaeda(__CLPK_integer *n, __CLPK_integer *tlvls, __CLPK_integer *curlvl,
    __CLPK_integer *curpbm,
1612     __CLPK_integer *prmptr, __CLPK_integer *perm, __CLPK_integer *givptr,
    __CLPK_integer *givcol,
1613     __CLPK_doublereal *givnum, __CLPK_doublereal *q, __CLPK_integer *qpnr,
    __CLPK_doublereal *z__,
1614     __CLPK_doublereal *ztemp, __CLPK_integer *info);
1615
1616 /* Subroutine */ int dlaein(__CLPK_logical *rightv, __CLPK_logical *noinit, __CLPK_integer *n,
    __CLPK_doublereal *h__,
1617     __CLPK_integer *ldh, __CLPK_doublereal *wr, __CLPK_doublereal *wi,
    __CLPK_doublereal *vr,
1618     __CLPK_doublereal *vi, __CLPK_doublereal *b, __CLPK_integer *ldb,
    __CLPK_doublereal *work,
1619     __CLPK_doublereal *eps3, __CLPK_doublereal *smlnum, __CLPK_doublereal *
    bignum, __CLPK_integer *info);
1620
1621 /* Subroutine */ int dlaev2(__CLPK_doublereal *a, __CLPK_doublereal *b, __CLPK_doublereal *c__,
    __CLPK_doublereal *rt1,
1622     __CLPK_doublereal *rt2, __CLPK_doublereal *csl, __CLPK_doublereal *sn1);
1623
1624 /* Subroutine */ int dlaexc(__CLPK_logical *wantq, __CLPK_integer *n, __CLPK_doublereal *t,
    __CLPK_integer *ldt,
1625     __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_integer *j1, __CLPK_integer
    *n1,
1626     __CLPK_integer *n2, __CLPK_doublereal *work, __CLPK_integer *info);
1627
1628 /* Subroutine */ int dlag2(__CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
    __CLPK_integer *ldb,
1629     __CLPK_doublereal *safmin, __CLPK_doublereal *scale1, __CLPK_doublereal *
    scale2, __CLPK_doublereal *wr1,
1630     __CLPK_doublereal *wr2, __CLPK_doublereal *wi);
1631
1632 /* Subroutine */ int dlags2(__CLPK_logical *upper, __CLPK_doublereal *a1, __CLPK_doublereal *a2,
    __CLPK_doublereal *a3,
1633     __CLPK_doublereal *b1, __CLPK_doublereal *b2, __CLPK_doublereal *b3,
    __CLPK_doublereal *b4,
1634     __CLPK_doublereal *csu, __CLPK_doublereal *snu, __CLPK_doublereal *csv, __CLPK_doublereal *snv,
    __CLPK_doublereal *csq,
1635     __CLPK_doublereal *snq);
1636
1637 /* Subroutine */ int dlagtf(__CLPK_integer *n, __CLPK_doublereal *a, __CLPK_doublereal *lambda,
    __CLPK_doublereal *b,
1638     __CLPK_doublereal *c__, __CLPK_doublereal *tol, __CLPK_doublereal *d__,
    __CLPK_integer *in,
1639     __CLPK_integer *info);
1640
1641 /* Subroutine */ int dlagtm(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
    __CLPK_doublereal *alpha,
1642     __CLPK_doublereal *dl, __CLPK_doublereal *d__, __CLPK_doublereal *du,
    __CLPK_doublereal *x,
1643     __CLPK_integer *ldx, __CLPK_doublereal *beta, __CLPK_doublereal *b,
    __CLPK_doublereal *ldb);
1644
1645 /* Subroutine */ int dlagts(__CLPK_integer *job, __CLPK_integer *n, __CLPK_doublereal *a,
    __CLPK_doublereal *b,
1646     __CLPK_doublereal *c__, __CLPK_doublereal *d__, __CLPK_integer *in,
    __CLPK_doublereal *y,
1647     __CLPK_doublereal *tol, __CLPK_integer *info);
1648
1649

```



```

1650 /* Subroutine */ int dlagv2(__CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
1651 __CLPK_integer *ldb, __CLPK_doublereal *alphar, __CLPK_doublereal *alpha, __CLPK_doublereal *
1652 beta, __CLPK_doublereal *csl, __CLPK_doublereal *snl, __CLPK_doublereal *csr, __CLPK_doublereal *
1653 snr);
1654
1655 /* Subroutine */ int dlahqr(__CLPK_logical *wantt, __CLPK_logical *wantz, __CLPK_integer *n,
1656 __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *h__, __CLPK_integer *ldh,
__CLPK_doublereal
1657 *wr, __CLPK_doublereal *wi, __CLPK_integer *iloz, __CLPK_integer *ihiz, __CLPK_doublereal *z__,
1658 __CLPK_integer *ldz, __CLPK_integer *info);
1659
1660 /* Subroutine */ int dlahrd(__CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *nb,
__CLPK_doublereal *
1661 a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *t, __CLPK_integer *ldt,
1662 __CLPK_doublereal *y, __CLPK_integer *ldy);
1663
1664 /* Subroutine */ int dlaicl(__CLPK_integer *job, __CLPK_integer *j, __CLPK_doublereal *x,
1665 __CLPK_doublereal *sest, __CLPK_doublereal *w, __CLPK_doublereal *gamma, __CLPK_doublereal *
1666 sestpr, __CLPK_doublereal *s, __CLPK_doublereal *c__);
1667
1668 /* Subroutine */ int dlaln2(__CLPK_logical *ltrans, __CLPK_integer *na, __CLPK_integer *nw,
1669 __CLPK_doublereal *smin, __CLPK_doublereal *ca, __CLPK_doublereal *a, __CLPK_integer *lda,
1670 __CLPK_doublereal *d1, __CLPK_doublereal *d2, __CLPK_doublereal *b, __CLPK_integer *ldb,
1671 __CLPK_doublereal *wr, __CLPK_doublereal *wi, __CLPK_doublereal *x, __CLPK_integer *ldx,
1672 __CLPK_doublereal *scale, __CLPK_doublereal *xnrm, __CLPK_integer *info);
1673
1674 /* Subroutine */ int dlals0(__CLPK_integer *icompq, __CLPK_integer *nl, __CLPK_integer *nr,
1675 __CLPK_integer *sqre, __CLPK_integer *nrhs, __CLPK_doublereal *b, __CLPK_integer *ldb,
__CLPK_doublereal
1676 *bx, __CLPK_integer *ldb, __CLPK_integer *perm, __CLPK_integer *givptr, __CLPK_integer *givcol,
1677 __CLPK_integer *ldgcol, __CLPK_doublereal *givnum, __CLPK_integer *ldgnum, __CLPK_doublereal *
1678 poles, __CLPK_doublereal *difl, __CLPK_doublereal *difr, __CLPK_doublereal *z__, __CLPK_integer *
1679 k, __CLPK_doublereal *c__, __CLPK_doublereal *s, __CLPK_doublereal *work, __CLPK_integer *info);
1680
1681 /* Subroutine */ int dlalsa(__CLPK_integer *icompq, __CLPK_integer *smlsiz, __CLPK_integer *n,
1682 __CLPK_integer *nrhs, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *bx,
__CLPK_integer *
1683 ldbx, __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *vt, __CLPK_integer *k,
1684 __CLPK_doublereal *difl, __CLPK_doublereal *difr, __CLPK_doublereal *z__, __CLPK_doublereal *
1685 poles, __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_integer *ldgcol, __CLPK_integer *
1686 perm, __CLPK_doublereal *givnum, __CLPK_doublereal *c__, __CLPK_doublereal *s, __CLPK_doublereal *
1687 work, __CLPK_integer *iwork, __CLPK_integer *info);
1688
1689 /* Subroutine */ int dlalsd(char *uplo, __CLPK_integer *smlsiz, __CLPK_integer *n, __CLPK_integer
1690 *nrhs, __CLPK_doublereal *d, __CLPK_doublereal *e, __CLPK_doublereal *b, __CLPK_integer *ldb,
1691 __CLPK_doublereal *rcond, __CLPK_integer *rank, __CLPK_doublereal *work, __CLPK_integer *iwork,
1692 __CLPK_integer *info);
1693
1694 /* Subroutine */ int dlamc1(__CLPK_integer *beta, __CLPK_integer *t, __CLPK_logical *rnd,
__CLPK_logical
1695 *ieeel);
1696
1697 /* Subroutine */ int dlamc2(__CLPK_integer *beta, __CLPK_integer *t, __CLPK_logical *rnd,
1698 __CLPK_doublereal *eps, __CLPK_integer *emin, __CLPK_doublereal *rmin, __CLPK_integer *emax,
1699 __CLPK_doublereal *rmax);
1700
1701 /* Subroutine */ int dlamc4(__CLPK_integer *emin, __CLPK_doublereal *start, __CLPK_integer *base);
1702
1703 /* Subroutine */ int dlamc5(__CLPK_integer *beta, __CLPK_integer *p, __CLPK_integer *emin,
1704 __CLPK_logical *ieee, __CLPK_integer *emax, __CLPK_doublereal *rmax);
1705
1706 /* Subroutine */ int dlamrg(__CLPK_integer *n1, __CLPK_integer *n2, __CLPK_doublereal *a,
__CLPK_integer
1707 *dtrd1, __CLPK_integer *dtrd2, __CLPK_integer *index);
1708
1709 /* Subroutine */ int dlanv2(__CLPK_doublereal *a, __CLPK_doublereal *b, __CLPK_doublereal *c__,
1710 __CLPK_doublereal *d__, __CLPK_doublereal *rtlr, __CLPK_doublereal *rtli, __CLPK_doublereal *rt2r,
1711 __CLPK_doublereal *rt2i, __CLPK_doublereal *cs, __CLPK_doublereal *sn);
1712
1713 /* Subroutine */ int dlapll(__CLPK_integer *n, __CLPK_doublereal *x, __CLPK_integer *incx,
1714 __CLPK_doublereal *y, __CLPK_integer *incy, __CLPK_doublereal *ssmin);
1715
1716 /* Subroutine */ int dlapmt(__CLPK_logical *forwrd, __CLPK_integer *m, __CLPK_integer *n,
1717 __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_integer *k);
1718
1719 /* Subroutine */ int dlaggb(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
*ku,
1720 __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *r__, __CLPK_doublereal *c__,
1721 __CLPK_doublereal *rowcnd, __CLPK_doublereal *colcnd, __CLPK_doublereal *amax, char *equad);
1722
1723 /* Subroutine */ int dlaqge(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
*
1724 lda, __CLPK_doublereal *r__, __CLPK_doublereal *c__, __CLPK_doublereal *rowcnd, __CLPK_doublereal
*colcnd,
1725 __CLPK_doublereal *amax, char *equad);
1726
1727 /* Subroutine */ int dlaqp2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *offset,
1728 __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_integer *jpvt, __CLPK_doublereal *tau,

```



```

1729     __CLPK_doublereal *vn1, __CLPK_doublereal *vn2, __CLPK_doublereal *work);
1730
1731 /* Subroutine */ int dlaqps__(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *offset,
    __CLPK_integer
1732     *nb, __CLPK_integer *kb, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_integer *jpvt,
1733     __CLPK_doublereal *tau, __CLPK_doublereal *vn1, __CLPK_doublereal *vn2, __CLPK_doublereal *auxv,
1734     __CLPK_doublereal *f, __CLPK_integer *ldf);
1735
1736 /* Subroutine */ int dlaqsb__(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_doublereal *
1737     ab, __CLPK_integer *ldab, __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax,
1738     char *equet);
1739
1740 /* Subroutine */ int dlaqsp__(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap,
1741     __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax, char *equet);
1742
1743 /* Subroutine */ int dlaqsy__(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
1744     lda, __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax, char *equet);
1745
1746 /* Subroutine */ int dlaqtr__(__CLPK_logical *ltran, __CLPK_logical *lreal, __CLPK_integer *n,
1747     __CLPK_doublereal *t, __CLPK_integer *ldt, __CLPK_doublereal *b, __CLPK_doublereal *w,
    __CLPK_doublereal
1748     *scale, __CLPK_doublereal *x, __CLPK_doublereal *work, __CLPK_integer *info);
1749
1750 /* Subroutine */ int dlar1v__(__CLPK_integer *n, __CLPK_integer *b1, __CLPK_integer *bn,
    __CLPK_doublereal
1751     *sigma, __CLPK_doublereal *d__, __CLPK_doublereal *l, __CLPK_doublereal *ld, __CLPK_doublereal *
1752     lld, __CLPK_doublereal *gersch, __CLPK_doublereal *z__, __CLPK_doublereal *ztz, __CLPK_doublereal
1753     *mingma, __CLPK_integer *r__, __CLPK_integer *isuppz, __CLPK_doublereal *work);
1754
1755 /* Subroutine */ int dlar2v__(__CLPK_integer *n, __CLPK_doublereal *x, __CLPK_doublereal *y,
1756     __CLPK_doublereal *z__, __CLPK_integer *incx, __CLPK_doublereal *c__, __CLPK_doublereal *s,
1757     __CLPK_integer *incc);
1758
1759 /* Subroutine */ int dlarf__(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *v,
1760     __CLPK_integer *incv, __CLPK_doublereal *tau, __CLPK_doublereal *c__, __CLPK_integer *ldc,
1761     __CLPK_doublereal *work);
1762
1763 /* Subroutine */ int dlarfb__(char *side, char *trans, char *direct, char *
1764     storev, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublereal *v,
    __CLPK_integer *
1765     ldv, __CLPK_doublereal *t, __CLPK_integer *ldt, __CLPK_doublereal *c__, __CLPK_integer *ldc,
1766     __CLPK_doublereal *work, __CLPK_integer *ldwork);
1767
1768 /* Subroutine */ int dlarfg__(__CLPK_integer *n, __CLPK_doublereal *alpha, __CLPK_doublereal *x,
1769     __CLPK_integer *incx, __CLPK_doublereal *tau);
1770
1771 /* Subroutine */ int dlarft__(char *direct, char *storev, __CLPK_integer *n, __CLPK_integer *
1772     k, __CLPK_doublereal *v, __CLPK_integer *ldv, __CLPK_doublereal *tau, __CLPK_doublereal *t,
1773     __CLPK_integer *ldt);
1774
1775 /* Subroutine */ int dlarfx__(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *
1776     v, __CLPK_doublereal *tau, __CLPK_doublereal *c__, __CLPK_integer *ldc, __CLPK_doublereal *work);
1777
1778 /* Subroutine */ int dlargv__(__CLPK_integer *n, __CLPK_doublereal *x, __CLPK_integer *incx,
1779     __CLPK_doublereal *y, __CLPK_integer *incy, __CLPK_doublereal *c__, __CLPK_integer *incc);
1780
1781 /* Subroutine */ int dlarnv__(__CLPK_integer *idist, __CLPK_integer *iseed, __CLPK_integer *n,
1782     __CLPK_doublereal *x);
1783
1784 /* Subroutine */ int dlarrb__(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *l,
1785     __CLPK_doublereal *ld, __CLPK_doublereal *lld, __CLPK_integer *ifirst, __CLPK_integer *ilast,
1786     __CLPK_doublereal *sigma, __CLPK_doublereal *reltol, __CLPK_doublereal *w, __CLPK_doublereal *
1787     wgap, __CLPK_doublereal *werr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *
1788     info);
1789
1790 /* Subroutine */ int dlarre__(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *e,
1791     __CLPK_doublereal *tol, __CLPK_integer *nsplit, __CLPK_integer *isplit, __CLPK_integer *m,
1792     __CLPK_doublereal *w, __CLPK_doublereal *woff, __CLPK_doublereal *gersch, __CLPK_doublereal *work,
1793     __CLPK_integer *info);
1794
1795 /* Subroutine */ int dlarrf__(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *l,
1796     __CLPK_doublereal *ld, __CLPK_doublereal *lld, __CLPK_integer *ifirst, __CLPK_integer *ilast,
1797     __CLPK_doublereal *w, __CLPK_doublereal *dplus, __CLPK_doublereal *lplus, __CLPK_doublereal *work,
1798     __CLPK_integer *iwork, __CLPK_integer *info);
1799
1800 /* Subroutine */ int dlarrv__(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *l,
1801     __CLPK_integer *isplit, __CLPK_integer *m, __CLPK_doublereal *w, __CLPK_integer *iblock,
1802     __CLPK_doublereal *gersch, __CLPK_doublereal *tol, __CLPK_doublereal *z__, __CLPK_integer *ldz,
1803     __CLPK_integer *isuppz, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
1804
1805 /* Subroutine */ int dlartg__(__CLPK_doublereal *f, __CLPK_doublereal *g, __CLPK_doublereal *cs,
1806     __CLPK_doublereal *sn, __CLPK_doublereal *r__);
1807
1808 /* Subroutine */ int dlartv__(__CLPK_integer *n, __CLPK_doublereal *x, __CLPK_integer *incx,
1809     __CLPK_doublereal *y, __CLPK_integer *incy, __CLPK_doublereal *c__, __CLPK_doublereal *s,
    __CLPK_integer
1810     *incc);

```

```

1811
1812 /* Subroutine */ int dlaruv(__CLPK_integer *iseed, __CLPK_integer *n, __CLPK_doublereal *x);
1813
1814 /* Subroutine */ int dlarz(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *l,
1815   __CLPK_doublereal *v, __CLPK_integer *incv, __CLPK_doublereal *tau, __CLPK_doublereal *c__,
1816   __CLPK_integer *ldc, __CLPK_doublereal *work);
1817
1818 /* Subroutine */ int dlarzb(char *side, char *trans, char *direct, char *
1819   storev, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *l,
1820   __CLPK_doublereal *v,
1821   __CLPK_integer *ldv, __CLPK_doublereal *t, __CLPK_integer *ldt, __CLPK_doublereal *c__,
1822   __CLPK_integer *
1823   ldc, __CLPK_doublereal *work, __CLPK_integer *ldwork);
1824
1825 /* Subroutine */ int dlarzt(char *direct, char *storev, __CLPK_integer *n, __CLPK_integer *
1826   k, __CLPK_doublereal *v, __CLPK_integer *ldv, __CLPK_doublereal *tau, __CLPK_doublereal *t,
1827   __CLPK_integer *ldt);
1828
1829 /* Subroutine */ int dlas2(__CLPK_doublereal *f, __CLPK_doublereal *g, __CLPK_doublereal *h__,
1830   __CLPK_doublereal *ssmin, __CLPK_doublereal *ssmax);
1831
1832 /* Subroutine */ int dlascl(char *type__, __CLPK_integer *kl, __CLPK_integer *ku,
1833   __CLPK_doublereal *cfrom, __CLPK_doublereal *cto, __CLPK_integer *m, __CLPK_integer *n,
1834   __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_integer *info);
1835
1836 /* Subroutine */ int dlasd0(__CLPK_integer *n, __CLPK_integer *sqre, __CLPK_doublereal *d__,
1837   __CLPK_doublereal *e, __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *vt,
1838   __CLPK_integer *
1839   ldvt, __CLPK_integer *smlsiz, __CLPK_integer *iwork, __CLPK_doublereal *work, __CLPK_integer *
1840   info);
1841
1842 /* Subroutine */ int dlasd1(__CLPK_integer *nl, __CLPK_integer *nr, __CLPK_integer *sqre,
1843   __CLPK_doublereal *d__, __CLPK_doublereal *alpha, __CLPK_doublereal *beta, __CLPK_doublereal *u,
1844   __CLPK_integer *ldu, __CLPK_doublereal *vt, __CLPK_integer *ldvt, __CLPK_integer *idxq,
1845   __CLPK_integer *
1846   iwork, __CLPK_doublereal *work, __CLPK_integer *info);
1847
1848 /* Subroutine */ int dlasd2(__CLPK_integer *nl, __CLPK_integer *nr, __CLPK_integer *sqre,
1849   __CLPK_integer
1850   *k, __CLPK_doublereal *d__, __CLPK_doublereal *z__, __CLPK_doublereal *alpha, __CLPK_doublereal *
1851   beta, __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *vt, __CLPK_integer *ldvt,
1852   __CLPK_doublereal *dsigma, __CLPK_doublereal *u2, __CLPK_integer *ldu2, __CLPK_doublereal *vt2,
1853   __CLPK_integer *ldvt2, __CLPK_integer *idxp, __CLPK_integer *idx, __CLPK_integer *idxc,
1854   __CLPK_integer *
1855   idxq, __CLPK_integer *coltyp, __CLPK_integer *info);
1856
1857 /* Subroutine */ int dlasd3(__CLPK_integer *nl, __CLPK_integer *nr, __CLPK_integer *sqre,
1858   __CLPK_integer
1859   *k, __CLPK_doublereal *d__, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *dsigma,
1860   __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *u2, __CLPK_integer *ldu2,
1861   __CLPK_doublereal *vt, __CLPK_integer *ldvt, __CLPK_doublereal *vt2, __CLPK_integer *ldvt2,
1862   __CLPK_integer *idxc, __CLPK_integer *ctot, __CLPK_doublereal *z__, __CLPK_integer *info);
1863
1864 /* Subroutine */ int dlasd4(__CLPK_integer *n, __CLPK_integer *i__, __CLPK_doublereal *d__,
1865   __CLPK_doublereal *z__, __CLPK_doublereal *delta, __CLPK_doublereal *rho, __CLPK_doublereal *
1866   sigma, __CLPK_doublereal *work, __CLPK_integer *info);
1867
1868 /* Subroutine */ int dlasd5(__CLPK_integer *i__, __CLPK_doublereal *d__, __CLPK_doublereal *z__,
1869   __CLPK_doublereal *delta, __CLPK_doublereal *rho, __CLPK_doublereal *dsigma, __CLPK_doublereal *
1870   work);
1871
1872 /* Subroutine */ int dlasd6(__CLPK_integer *icompq, __CLPK_integer *nl, __CLPK_integer *nr,
1873   __CLPK_integer *sqre, __CLPK_doublereal *d__, __CLPK_doublereal *vf, __CLPK_doublereal *vl,
1874   __CLPK_doublereal *alpha, __CLPK_doublereal *beta, __CLPK_integer *idxq, __CLPK_integer *perm,
1875   __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_integer *ldgcol, __CLPK_doublereal *givnum,
1876   __CLPK_integer *ldgnum, __CLPK_doublereal *poles, __CLPK_doublereal *difl, __CLPK_doublereal *
1877   difr, __CLPK_doublereal *z__, __CLPK_integer *k, __CLPK_doublereal *c__, __CLPK_doublereal *s,
1878   __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
1879
1880 /* Subroutine */ int dlasd7(__CLPK_integer *icompq, __CLPK_integer *nl, __CLPK_integer *nr,
1881   __CLPK_integer *sqre, __CLPK_integer *k, __CLPK_doublereal *d__, __CLPK_doublereal *z__,
1882   __CLPK_doublereal *zw, __CLPK_doublereal *vf, __CLPK_doublereal *vfw, __CLPK_doublereal *vl,
1883   __CLPK_doublereal *vlw, __CLPK_doublereal *alpha, __CLPK_doublereal *beta, __CLPK_doublereal *
1884   dsigma, __CLPK_integer *idx, __CLPK_integer *idxp, __CLPK_integer *idxq, __CLPK_integer *perm,
1885   __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_integer *ldgcol, __CLPK_doublereal *givnum,
1886   __CLPK_integer *ldgnum, __CLPK_doublereal *c__, __CLPK_doublereal *s, __CLPK_integer *info);
1887
1888 /* Subroutine */ int dlasd8(__CLPK_integer *icompq, __CLPK_integer *k, __CLPK_doublereal *d__,
1889   __CLPK_doublereal *z__, __CLPK_doublereal *vf, __CLPK_doublereal *vl, __CLPK_doublereal *difl,
1890   __CLPK_doublereal *difr, __CLPK_integer *lddifr, __CLPK_doublereal *dsigma, __CLPK_doublereal *
1891   work, __CLPK_integer *info);
1892
1893 /* Subroutine */ int dlasd9(__CLPK_integer *icompq, __CLPK_integer *ldu, __CLPK_integer *k,
1894   __CLPK_doublereal *d__, __CLPK_doublereal *z__, __CLPK_doublereal *vf, __CLPK_doublereal *vl,
1895   __CLPK_doublereal *difl, __CLPK_doublereal *difr, __CLPK_doublereal *dsigma, __CLPK_doublereal *
1896   work, __CLPK_integer *info);
1897
1898

```

```

1891 /* Subroutine */ int dlasda(__CLPK_integer *icompq, __CLPK_integer *smlsiz, __CLPK_integer *n,
1892   __CLPK_integer *sqre, __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublereal *u,
   __CLPK_integer
1893   *ldu, __CLPK_doublereal *vt, __CLPK_integer *k, __CLPK_doublereal *difl, __CLPK_doublereal *difr,
1894   __CLPK_doublereal *z__, __CLPK_doublereal *poles, __CLPK_integer *givptr, __CLPK_integer *givcol,
1895   __CLPK_integer *ldgcol, __CLPK_integer *perm, __CLPK_doublereal *givnum, __CLPK_doublereal *c__,
1896   __CLPK_doublereal *s, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
1897
1898 /* Subroutine */ int dlasdq(char *uplo, __CLPK_integer *sqre, __CLPK_integer *n, __CLPK_integer *
1899   ncv, __CLPK_integer *nru, __CLPK_integer *ncc, __CLPK_doublereal *d__, __CLPK_doublereal *e,
1900   __CLPK_doublereal *vt, __CLPK_integer *ldvt, __CLPK_doublereal *u, __CLPK_integer *ldu,
1901   __CLPK_doublereal *c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *info);
1902
1903 /* Subroutine */ int dlasdt(__CLPK_integer *n, __CLPK_integer *lvl, __CLPK_integer *nd, __CLPK_integer
   *
1904   inode, __CLPK_integer *ndiml, __CLPK_integer *ndimr, __CLPK_integer *msub);
1905
1906 /* Subroutine */ int dlaset(char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *
1907   alpha, __CLPK_doublereal *beta, __CLPK_doublereal *a, __CLPK_integer *lda);
1908
1909 /* Subroutine */ int dlasq1(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *e,
1910   __CLPK_doublereal *work, __CLPK_integer *info);
1911
1912 /* Subroutine */ int dlasq2(__CLPK_integer *n, __CLPK_doublereal *z__, __CLPK_integer *info);
1913
1914 /* Subroutine */ int dlasq3(__CLPK_integer *i0, __CLPK_integer *n0, __CLPK_doublereal *z__,
1915   __CLPK_integer *pp, __CLPK_doublereal *dmin__, __CLPK_doublereal *sigma, __CLPK_doublereal *desig,
1916   __CLPK_doublereal *qmax, __CLPK_integer *nfail, __CLPK_integer *iter, __CLPK_integer *ndiv,
1917   __CLPK_logical *ieee);
1918
1919 /* Subroutine */ int dlasq4(__CLPK_integer *i0, __CLPK_integer *n0, __CLPK_doublereal *z__,
1920   __CLPK_integer *pp, __CLPK_integer *n0in, __CLPK_doublereal *dmin__, __CLPK_doublereal *dminl,
1921   __CLPK_doublereal *dmin2, __CLPK_doublereal *dn, __CLPK_doublereal *dn1, __CLPK_doublereal *dn2,
1922   __CLPK_doublereal *tau, __CLPK_integer *ttype);
1923
1924 /* Subroutine */ int dlasq5(__CLPK_integer *i0, __CLPK_integer *n0, __CLPK_doublereal *z__,
1925   __CLPK_integer *pp, __CLPK_doublereal *tau, __CLPK_doublereal *dmin__, __CLPK_doublereal *dminl,
1926   __CLPK_doublereal *dmin2, __CLPK_doublereal *dn, __CLPK_doublereal *dnml, __CLPK_doublereal *dnm2,
1927   __CLPK_logical *ieee);
1928
1929 /* Subroutine */ int dlasq6(__CLPK_integer *i0, __CLPK_integer *n0, __CLPK_doublereal *z__,
1930   __CLPK_integer *pp, __CLPK_doublereal *dmin__, __CLPK_doublereal *dminl, __CLPK_doublereal *dmin2,
1931   __CLPK_doublereal *dn, __CLPK_doublereal *dnml, __CLPK_doublereal *dnm2);
1932
1933 /* Subroutine */ int dlasr(char *side, char *pivot, char *direct, __CLPK_integer *m,
1934   __CLPK_integer *n, __CLPK_doublereal *c__, __CLPK_doublereal *s, __CLPK_doublereal *a,
   __CLPK_integer *
1935   lda);
1936
1937 /* Subroutine */ int dlasrt(char *id, __CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_integer *
1938   info);
1939
1940 /* Subroutine */ int dlassq(__CLPK_integer *n, __CLPK_doublereal *x, __CLPK_integer *incx,
1941   __CLPK_doublereal *scale, __CLPK_doublereal *sumsq);
1942
1943 /* Subroutine */ int dlasv2(__CLPK_doublereal *f, __CLPK_doublereal *g, __CLPK_doublereal *h__,
1944   __CLPK_doublereal *ssmin, __CLPK_doublereal *ssmax, __CLPK_doublereal *snr, __CLPK_doublereal *
1945   csr, __CLPK_doublereal *snl, __CLPK_doublereal *csl);
1946
1947 /* Subroutine */ int dlaswp(__CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda,
   __CLPK_integer
1948   *kl, __CLPK_integer *k2, __CLPK_integer *ipiv, __CLPK_integer *incx);
1949
1950 /* Subroutine */ int dlasz2(__CLPK_logical *ltranl, __CLPK_logical *ltranr, __CLPK_integer *isgn,
1951   __CLPK_integer *n1, __CLPK_integer *n2, __CLPK_doublereal *tl, __CLPK_integer *ldtl,
   __CLPK_doublereal *
1952   tr, __CLPK_integer *ldtr, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *scale,
1953   __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *xnorm, __CLPK_integer *info);
1954
1955 /* Subroutine */ int dlasyf(char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_integer *kb,
1956   __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublereal *w,
   __CLPK_integer *
1957   ldw, __CLPK_integer *info);
1958
1959 /* Subroutine */ int dlatbs(char *uplo, char *trans, char *diag, char *
1960   normin, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_doublereal *ab, __CLPK_integer *ldab,
1961   __CLPK_doublereal *x, __CLPK_doublereal *scale, __CLPK_doublereal *cnorm, __CLPK_integer *info);
1962
1963 /* Subroutine */ int dlatdf(__CLPK_integer *ijob, __CLPK_integer *n, __CLPK_doublereal *z__,
1964   __CLPK_integer *ldz, __CLPK_doublereal *rhs, __CLPK_doublereal *rdsum, __CLPK_doublereal *rdscale,
1965   __CLPK_integer *ipiv, __CLPK_integer *jpiv);
1966
1967 /* Subroutine */ int dlatps(char *uplo, char *trans, char *diag, char *
1968   normin, __CLPK_integer *n, __CLPK_doublereal *ap, __CLPK_doublereal *x, __CLPK_doublereal *scale,
1969   __CLPK_doublereal *cnorm, __CLPK_integer *info);
1970
1971 /* Subroutine */ int dlatrd(char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_doublereal *

```

```
1972     a, __CLPK_integer *lda, __CLPK_doublereal *e, __CLPK_doublereal *tau, __CLPK_doublereal *w,
1973     __CLPK_integer *ldw);
1974
1975 /* Subroutine */ int dlatrs_(char *uplo, char *trans, char *diag, char *
1976     normin, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *x,
1977     __CLPK_doublereal *scale, __CLPK_doublereal *cnorm, __CLPK_integer *info);
1978
1979 /* Subroutine */ int dlatrz_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *l, __CLPK_doublereal
1980     *
1981     a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work);
1982
1983 /* Subroutine */ int dlatzm_(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *
1984     v, __CLPK_integer *incv, __CLPK_doublereal *tau, __CLPK_doublereal *c1, __CLPK_doublereal *c2,
1985     __CLPK_integer *ldc, __CLPK_doublereal *work);
1986
1987 /* Subroutine */ int dlau2_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
1988     lda, __CLPK_integer *info);
1989
1990 /* Subroutine */ int dlauu_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
1991     lda, __CLPK_integer *info);
1992
1993 /* Subroutine */ int dopgtr_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap,
1994     __CLPK_doublereal *tau, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *work,
1995     __CLPK_integer *info);
1996
1997 /* Subroutine */ int dopmtr_(char *side, char *uplo, char *trans, __CLPK_integer *m,
1998     __CLPK_integer *n, __CLPK_doublereal *ap, __CLPK_doublereal *tau, __CLPK_doublereal *c__,
1999     __CLPK_integer
2000     *ldc, __CLPK_doublereal *work, __CLPK_integer *info);
2001
2002 /* Subroutine */ int dorg2l_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublereal
2003     *
2004     a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *info);
2005
2006 /* Subroutine */ int dorg2r_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublereal
2007     *
2008     a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *info);
2009
2010 /* Subroutine */ int dorgbr_(char *vect, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
2011     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work,
2012     __CLPK_integer *lwork, __CLPK_integer *info);
2013
2014 /* Subroutine */ int dorghr_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi,
2015     __CLPK_doublereal *a1, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work,
2016     __CLPK_integer *lwork, __CLPK_integer *info);
2017
2018 /* Subroutine */ int dorgl2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublereal
2019     *
2020     a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *info);
2021
2022 /* Subroutine */ int dorglq_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublereal
2023     *
2024     a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork,
2025     __CLPK_integer *info);
2026
2027 /* Subroutine */ int dorgql_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublereal
2028     *
2029     a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork,
2030     __CLPK_integer *info);
2031
2032 /* Subroutine */ int dorgqr_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublereal
2033     *
2034     a, __CLPK_integer *lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork,
2035     __CLPK_integer *info);
2036
2037 /* Subroutine */ int dorgtr_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2038     lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2039
2040 /* Subroutine */ int dorm2l_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2041     __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
2042     __CLPK_doublereal *
2043     c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *info);
2044
2045 /* Subroutine */ int dorm2r_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2046     __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
2047     __CLPK_doublereal *
2048     c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *info);
```

```

2047 /* Subroutine */ int dormbr_(char *vect, char *side, char *trans, __CLPK_integer *m,
2048   __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal
   *tau,
2049   __CLPK_doublereal *c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *lwork,
2050   __CLPK_integer *info);
2051
2052 /* Subroutine */ int dormhr_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2053   __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *a, __CLPK_integer *lda,
   __CLPK_doublereal *
2054   tau, __CLPK_doublereal *c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *lwork,
2055   __CLPK_integer *info);
2056
2057 /* Subroutine */ int dorml2_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2058   __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
   __CLPK_doublereal *
2059   c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *info);
2060
2061 /* Subroutine */ int dormlq_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2062   __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
   __CLPK_doublereal *
2063   c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2064
2065 /* Subroutine */ int dormql_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2066   __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
   __CLPK_doublereal *
2067   c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2068
2069 /* Subroutine */ int dormqr_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2070   __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
   __CLPK_doublereal *
2071   c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2072
2073 /* Subroutine */ int dormr2_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2074   __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
   __CLPK_doublereal *
2075   c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *info);
2076
2077 /* Subroutine */ int dormr3_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2078   __CLPK_integer *k, __CLPK_integer *l, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal
   *tau,
2079   __CLPK_doublereal *c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *info);
2080
2081 /* Subroutine */ int dormrq_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2082   __CLPK_integer *k, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
   __CLPK_doublereal *
2083   c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2084
2085 /* Subroutine */ int dormrz_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
2086   __CLPK_integer *k, __CLPK_integer *l, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal
   *tau,
2087   __CLPK_doublereal *c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *lwork,
2088   __CLPK_integer *info);
2089
2090 /* Subroutine */ int dormtr_(char *side, char *uplo, char *trans, __CLPK_integer *m,
2091   __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *tau,
   __CLPK_doublereal *
2092   c__, __CLPK_integer *ldc, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2093
2094 /* Subroutine */ int dpbcon_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_doublereal *
2095   ab, __CLPK_integer *ldab, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *
2096   work, __CLPK_integer *iwork, __CLPK_integer *info);
2097
2098 /* Subroutine */ int dpbequ_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_doublereal *
2099   ab, __CLPK_integer *ldab, __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax,
2100   __CLPK_integer *info);
2101
2102 /* Subroutine */ int dprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
2103   nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *afb, __CLPK_integer *ldafb,
   __CLPK_doublereal *
2104   b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
   __CLPK_doublereal *
2105   ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *
2106   info);
2107
2108 /* Subroutine */ int dpbstf_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_doublereal *
2109   ab, __CLPK_integer *ldab, __CLPK_integer *info);
2110
2111 /* Subroutine */ int dpbsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
2112   nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *b, __CLPK_integer *ldb,
   __CLPK_integer *
2113   info);
2114
2115 /* Subroutine */ int dpbsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
2116   __CLPK_integer *nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *afb,
   __CLPK_integer *
2117   ldafb, char *equed, __CLPK_doublereal *s, __CLPK_doublereal *b, __CLPK_integer *
2118   ldb, __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *ferr,
   __CLPK_doublereal *
2119   berr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
2120
2121 /* Subroutine */ int dpbtf2_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_doublereal *

```

```

2122     ab, __CLPK_integer *ldab, __CLPK_integer *info);
2123
2124 /* Subroutine */ int dpbtrf_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_doublereal *
2125     ab, __CLPK_integer *ldab, __CLPK_integer *info);
2126
2127 /* Subroutine */ int dpbtrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
2128     nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *b, __CLPK_integer *ldb,
2129     __CLPK_integer *info);
2130
2131 /* Subroutine */ int dpocon_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2132     lda, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *work, __CLPK_integer *
2133     iwork, __CLPK_integer *info);
2134
2135 /* Subroutine */ int dpoequ_(__CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda,
2136     __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax, __CLPK_integer *info);
2137
2138 /* Subroutine */ int dporfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
2139     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *af, __CLPK_integer *ldaf,
2140     __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
2141     __CLPK_doublereal *
2142     ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *
2143     info);
2144
2145 /* Subroutine */ int dposv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal
2146     *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
2147
2148 /* Subroutine */ int dposvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
2149     nrhs, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *af, __CLPK_integer *ldaf,
2150     char *equed, __CLPK_doublereal *s, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *
2151     x, __CLPK_integer *ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *
2152     berr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
2153
2154 /* Subroutine */ int dpotf2_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2155     lda, __CLPK_integer *info);
2156
2157 /* Subroutine */ int dpotrf_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2158     lda, __CLPK_integer *info);
2159
2160 /* Subroutine */ int dpotri_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2161     lda, __CLPK_integer *info);
2162
2163 /* Subroutine */ int dpotrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
2164     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
2165     __CLPK_integer *
2166     info);
2167
2168 /* Subroutine */ int dppcon_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap,
2169     __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *work, __CLPK_integer *
2170     iwork, __CLPK_integer *info);
2171
2172 /* Subroutine */ int dppequ_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap,
2173     __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax, __CLPK_integer *info);
2174
2175 /* Subroutine */ int dpprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
2176     __CLPK_doublereal *ap, __CLPK_doublereal *afp, __CLPK_doublereal *b, __CLPK_integer *ldb,
2177     __CLPK_doublereal *x,
2178     __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
2179     __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
2180
2181 /* Subroutine */ int dppsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal
2182     *ap, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
2183
2184 /* Subroutine */ int dppsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
2185     nrhs, __CLPK_doublereal *ap, __CLPK_doublereal *afp, char *equed, __CLPK_doublereal *s,
2186     __CLPK_doublereal *b,
2187     __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
2188     __CLPK_doublereal *
2189     rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *
2190     iwork, __CLPK_integer *info);
2191
2192 /* Subroutine */ int dpptrf_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap, __CLPK_integer *
2193     info);
2194
2195 /* Subroutine */ int dpptri_(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap, __CLPK_integer *
2196     info);
2197
2198 /* Subroutine */ int dpptrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
2199     __CLPK_doublereal *ap, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
2200
2201 /* Subroutine */ int dptcon_(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *e,
2202     __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *work, __CLPK_integer *info);
2203
2204 /* Subroutine */ int dpteqr_(char *compz, __CLPK_integer *n, __CLPK_doublereal *d__,
2205     __CLPK_doublereal *e, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2206     __CLPK_integer *info);
2207
2208 /* Subroutine */ int dptrfs_(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal *d__,
2209     __CLPK_doublereal *e, __CLPK_doublereal *df, __CLPK_doublereal *ef, __CLPK_doublereal *b,
2210     __CLPK_integer

```

```

2205     *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
2206     __CLPK_doublereal *work, __CLPK_integer *info);
2207
2208 /* Subroutine */ int dptsv(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal *d__,
2209     __CLPK_doublereal *e, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
2210
2211 /* Subroutine */ int dptsvx(char *fact, __CLPK_integer *n, __CLPK_integer *nrhs,
2212     __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublereal *df, __CLPK_doublereal *ef,
2213     __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
2214     __CLPK_doublereal *
2215     rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *
2216     info);
2217 /* Subroutine */ int dpstrf(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *e,
2218     __CLPK_integer *info);
2219
2220 /* Subroutine */ int dpstrs(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal *d__,
2221     __CLPK_doublereal *e, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
2222
2223 /* Subroutine */ int dpsts2(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal *d__,
2224     __CLPK_doublereal *e, __CLPK_doublereal *b, __CLPK_integer *ldb);
2225
2226 /* Subroutine */ int drscl(__CLPK_integer *n, __CLPK_doublereal *sa, __CLPK_doublereal *sx,
2227     __CLPK_integer *incx);
2228
2229 /* Subroutine */ int dsbev(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
2230     __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *w, __CLPK_doublereal *z__,
2231     __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *info);
2232
2233 /* Subroutine */ int dsbevd(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
2234     __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *w, __CLPK_doublereal *z__,
2235     __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *iwork,
2236     __CLPK_integer *liwork, __CLPK_integer *info);
2237
2238 /* Subroutine */ int dsbevz(char *jobz, char *range, char *uplo, __CLPK_integer *n,
2239     __CLPK_integer *kd, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *q,
2240     __CLPK_integer *
2241     ldq, __CLPK_doublereal *vl, __CLPK_doublereal *vu, __CLPK_integer *il, __CLPK_integer *iu,
2242     __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w, __CLPK_doublereal *z__,
2243     __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *ifail,
2244     __CLPK_integer *info);
2245
2246 /* Subroutine */ int dsbgst(char *vect, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
2247     __CLPK_integer *kb, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *bb,
2248     __CLPK_integer *
2249     ldbb, __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *work, __CLPK_integer *info);
2250
2251 /* Subroutine */ int dsbgv(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
2252     __CLPK_integer *kb, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *bb,
2253     __CLPK_integer *
2254     ldbb, __CLPK_doublereal *w, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2255     __CLPK_integer *info);
2256
2257 /* Subroutine */ int dsbgvd(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
2258     __CLPK_integer *kb, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *bb,
2259     __CLPK_integer *
2260     ldbb, __CLPK_doublereal *w, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2261     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
2262
2263 /* Subroutine */ int dsbgvx(char *jobz, char *range, char *uplo, __CLPK_integer *n,
2264     __CLPK_integer *ka, __CLPK_integer *kb, __CLPK_doublereal *ab, __CLPK_integer *ldab,
2265     __CLPK_doublereal *
2266     bb, __CLPK_integer *ldbb, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *vl,
2267     __CLPK_doublereal *vu, __CLPK_integer *il, __CLPK_integer *iu, __CLPK_doublereal *abstol,
2268     __CLPK_integer *
2269     *m, __CLPK_doublereal *w, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2270     __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
2271
2272 /* Subroutine */ int dsbtrd(char *vect, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
2273     __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *d__, __CLPK_doublereal *e,
2274     __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *work, __CLPK_integer *info);
2275
2276 /* Subroutine */ int dspcon(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap, __CLPK_integer *
2277     ipiv, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *work, __CLPK_integer *
2278     iwork, __CLPK_integer *info);
2279
2280 /* Subroutine */ int dspev(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_doublereal *
2281     ap, __CLPK_doublereal *w, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2282     __CLPK_integer *info);
2283
2284 /* Subroutine */ int dspevd(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_doublereal *
2285     ap, __CLPK_doublereal *w, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2286     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
2287
2288 /* Subroutine */ int dspevx(char *jobz, char *range, char *uplo, __CLPK_integer *n,
2289     __CLPK_doublereal *ap, __CLPK_doublereal *vl, __CLPK_doublereal *vu, __CLPK_integer *il,
2290     __CLPK_integer *

```



```

2284     iu, __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w, __CLPK_doublereal *z__,
2285     __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *ifail,
2286     __CLPK_integer *info);
2287
2288 /* Subroutine */ int dspgst(__CLPK_integer *itype, char *uplo, __CLPK_integer *n,
2289     __CLPK_doublereal *ap, __CLPK_doublereal *bp, __CLPK_integer *info);
2290
2291 /* Subroutine */ int dspgv(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
2292     n, __CLPK_doublereal *ap, __CLPK_doublereal *bp, __CLPK_doublereal *w, __CLPK_doublereal *z__,
2293     __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *info);
2294
2295 /* Subroutine */ int dspgvd(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
2296     n, __CLPK_doublereal *ap, __CLPK_doublereal *bp, __CLPK_doublereal *w, __CLPK_doublereal *z__,
2297     __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *iwork,
2298     __CLPK_integer *liwork, __CLPK_integer *info);
2299
2300 /* Subroutine */ int dspgvx(__CLPK_integer *itype, char *jobz, char *range, char *
2301     uplo, __CLPK_integer *n, __CLPK_doublereal *ap, __CLPK_doublereal *bp, __CLPK_doublereal *vl,
2302     __CLPK_doublereal *vu, __CLPK_integer *il, __CLPK_integer *iu, __CLPK_doublereal *abstol,
2303     __CLPK_integer *m, __CLPK_doublereal *w, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2304     __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
2305
2306 /* Subroutine */ int dsprfs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
2307     __CLPK_doublereal *ap, __CLPK_doublereal *afp, __CLPK_integer *ipiv, __CLPK_doublereal *b,
2308     __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr,
2309     __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
2310
2311 /* Subroutine */ int dspsv(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal
2312     *ap, __CLPK_integer *ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_integer *info);
2313
2314 /* Subroutine */ int dspsvx(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
2315     nrhs, __CLPK_doublereal *ap, __CLPK_doublereal *afp, __CLPK_integer *ipiv, __CLPK_doublereal *b,
2316     __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *rcond,
2317     __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork,
2318     __CLPK_integer *info);
2319
2320 /* Subroutine */ int dsptrd(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap,
2321     __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublereal *tau, __CLPK_integer *info);
2322
2323 /* Subroutine */ int dsprtf(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap, __CLPK_integer *
2324     ipiv, __CLPK_integer *info);
2325
2326 /* Subroutine */ int dsptri(char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap, __CLPK_integer *
2327     ipiv, __CLPK_doublereal *work, __CLPK_integer *info);
2328
2329 /* Subroutine */ int dsptrs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
2330     __CLPK_doublereal *ap, __CLPK_integer *ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb,
2331     __CLPK_integer *
2332     info);
2333
2334 /* Subroutine */ int dstebz(char *range, char *order, __CLPK_integer *n, __CLPK_doublereal
2335     *vl, __CLPK_doublereal *vu, __CLPK_integer *il, __CLPK_integer *iu, __CLPK_doublereal *abstol,
2336     __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_integer *m, __CLPK_integer *nsplit,
2337     __CLPK_doublereal *w, __CLPK_integer *iblock, __CLPK_integer *isplit, __CLPK_doublereal *work,
2338     __CLPK_integer *iwork, __CLPK_integer *info);
2339
2340 /* Subroutine */ int dstedc(char *compz, __CLPK_integer *n, __CLPK_doublereal *d__,
2341     __CLPK_doublereal *e, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2342     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
2343
2344 /* Subroutine */ int dstegr(char *jobz, char *range, __CLPK_integer *n, __CLPK_doublereal *
2345     d__, __CLPK_doublereal *e, __CLPK_doublereal *vl, __CLPK_doublereal *vu, __CLPK_integer *il,
2346     __CLPK_integer *iu, __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w,
2347     __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_integer *isuppz, __CLPK_doublereal *work,
2348     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
2349
2350 /* Subroutine */ int dstein(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *e,
2351     __CLPK_integer *m, __CLPK_doublereal *w, __CLPK_integer *iblock, __CLPK_integer *isplit,
2352     __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *iwork,
2353     __CLPK_integer *ifail, __CLPK_integer *info);
2354
2355 /* Subroutine */ int dsteqr(char *compz, __CLPK_integer *n, __CLPK_doublereal *d__,
2356     __CLPK_doublereal *e, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2357     __CLPK_integer *info);
2358
2359 /* Subroutine */ int dsterf(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *e,
2360     __CLPK_integer *info);
2361
2362 /* Subroutine */ int dstev(char *jobz, __CLPK_integer *n, __CLPK_doublereal *d__,
2363     __CLPK_doublereal *e, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2364     __CLPK_integer *info);
2365
2366 /* Subroutine */ int dstevd(char *jobz, __CLPK_integer *n, __CLPK_doublereal *d__,
2367     __CLPK_doublereal *e, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
2368     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);

```



```

2369 /* Subroutine */ int dstevr(char *jobz, char *range, __CLPK_integer *n, __CLPK_doublereal *
2370 d, __CLPK_doublereal *e, __CLPK_doublereal *vl, __CLPK_doublereal *vu, __CLPK_integer *il,
2371 __CLPK_integer *iu, __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w,
2372 __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_integer *isuppz, __CLPK_doublereal *work,
2373 __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
2374
2375 /* Subroutine */ int dstevx(char *jobz, char *range, __CLPK_integer *n, __CLPK_doublereal *
2376 d, __CLPK_doublereal *e, __CLPK_doublereal *vl, __CLPK_doublereal *vu, __CLPK_integer *il,
2377 __CLPK_integer *iu, __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w,
2378 __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *iwork,
2379 __CLPK_integer *ifail, __CLPK_integer *info);
2380
2381 /* Subroutine */ int dsycon(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2382 lda, __CLPK_integer *ipiv, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *
2383 work, __CLPK_integer *iwork, __CLPK_integer *info);
2384
2385 /* Subroutine */ int dsyev(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_doublereal *a,
2386 __CLPK_integer *lda, __CLPK_doublereal *w, __CLPK_doublereal *work, __CLPK_integer *lwork,
2387 __CLPK_integer *info);
2388
2389 /* Subroutine */ int dsyevd(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_doublereal *
2390 a, __CLPK_integer *lda, __CLPK_doublereal *w, __CLPK_doublereal *work, __CLPK_integer *lwork,
2391 __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
2392
2393 /* Subroutine */ int dsyevr(char *jobz, char *range, char *uplo, __CLPK_integer *n,
2394 __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *vl, __CLPK_doublereal *vu,
__CLPK_integer *
2395 il, __CLPK_integer *iu, __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w,
2396 __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_integer *isuppz, __CLPK_doublereal *work,
2397 __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
2398
2399 /* Subroutine */ int dsyevx(char *jobz, char *range, char *uplo, __CLPK_integer *n,
2400 __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *vl, __CLPK_doublereal *vu,
__CLPK_integer *
2401 il, __CLPK_integer *iu, __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w,
2402 __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *lwork,
2403 __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
2404
2405 /* Subroutine */ int dsygs2(__CLPK_integer *itype, char *uplo, __CLPK_integer *n,
2406 __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
__CLPK_integer *
2407 info);
2408
2409 /* Subroutine */ int dsygst(__CLPK_integer *itype, char *uplo, __CLPK_integer *n,
2410 __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
__CLPK_integer *
2411 info);
2412
2413 /* Subroutine */ int dsygv(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
2414 n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
2415 __CLPK_doublereal *w, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2416
2417 /* Subroutine */ int dsygvd(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
2418 n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
2419 __CLPK_doublereal *w, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *iwork,
2420 __CLPK_integer *liwork, __CLPK_integer *info);
2421
2422 /* Subroutine */ int dsygvx(__CLPK_integer *itype, char *jobz, char *range, char *
2423 uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
__CLPK_integer *
2424 *ldb, __CLPK_doublereal *vl, __CLPK_doublereal *vu, __CLPK_integer *il, __CLPK_integer *iu,
2425 __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w, __CLPK_doublereal *z__,
2426 __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *iwork,
2427 __CLPK_integer *ifail, __CLPK_integer *info);
2428
2429 /* Subroutine */ int dsyrfs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
2430 __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *af, __CLPK_integer *ldaf,
__CLPK_integer *
2431 ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx,
2432 __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork,
2433 __CLPK_integer *info);
2434
2435 /* Subroutine */ int dsysv(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal
2436 *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb,
2437 __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2438
2439 /* Subroutine */ int dsysvx(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
2440 nrhs, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *af, __CLPK_integer *ldaf,
2441 __CLPK_integer *ipiv, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *x,
__CLPK_integer *
2442 ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
2443 __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *info);
2444
2445 /* Subroutine */ int dsytd2(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2446 lda, __CLPK_doublereal *d, __CLPK_doublereal *e, __CLPK_doublereal *tau, __CLPK_integer *info);
2447
2448 /* Subroutine */ int dsytf2(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *

```

```

2449     lda, __CLPK_integer *ipiv, __CLPK_integer *info);
2450
2451 /* Subroutine */ int dsytrd(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2452     lda, __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublereal *tau, __CLPK_doublereal *
2453     work, __CLPK_integer *lwork, __CLPK_integer *info);
2454
2455 /* Subroutine */ int dsytrf(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2456     lda, __CLPK_integer *ipiv, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2457
2458 /* Subroutine */ int dsytri(char *uplo, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *
2459     lda, __CLPK_integer *ipiv, __CLPK_doublereal *work, __CLPK_integer *info);
2460
2461 /* Subroutine */ int dsytrs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
2462     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublereal *b,
2463     __CLPK_integer *
2464     ldb, __CLPK_integer *info);
2465
2466 /* Subroutine */ int dtbcon(char *norm, char *uplo, char *diag, __CLPK_integer *n,
2467     __CLPK_integer *kd, __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *rcond,
2468     __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
2469
2470 /* Subroutine */ int dtbrfs(char *uplo, char *trans, char *diag, __CLPK_integer *n,
2471     __CLPK_integer *kd, __CLPK_integer *nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab,
2472     __CLPK_doublereal
2473     *b, __CLPK_integer *ldb, __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr,
2474     __CLPK_doublereal *berr, __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
2475
2476 /* Subroutine */ int dtbtrs(char *uplo, char *trans, char *diag, __CLPK_integer *n,
2477     __CLPK_integer *kd, __CLPK_integer *nrhs, __CLPK_doublereal *ab, __CLPK_integer *ldab,
2478     __CLPK_doublereal
2479     *b, __CLPK_integer *ldb, __CLPK_integer *info);
2480
2481 /* Subroutine */ int dtgevc(char *side, char *howmny, __CLPK_logical *select,
2482     __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer
2483     *ldb,
2484     __CLPK_doublereal *vl, __CLPK_integer *ldvl, __CLPK_doublereal *vr, __CLPK_integer *ldvr,
2485     __CLPK_integer
2486     *mm, __CLPK_integer *m, __CLPK_doublereal *work, __CLPK_integer *info);
2487
2488 /* Subroutine */ int dtgex2(__CLPK_logical *wantq, __CLPK_logical *wantz, __CLPK_integer *n,
2489     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
2490     __CLPK_doublereal *
2491     q, __CLPK_integer *ldq, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_integer *j1,
2492     __CLPK_integer *
2493     n1, __CLPK_integer *n2, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2494
2495 /* Subroutine */ int dtgexc(__CLPK_logical *wantq, __CLPK_logical *wantz, __CLPK_integer *n,
2496     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
2497     __CLPK_doublereal *
2498     q, __CLPK_integer *ldq, __CLPK_doublereal *z__, __CLPK_integer *ldz, __CLPK_integer *ifst,
2499     __CLPK_integer *
2500     ilst, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2501
2502 /* Subroutine */ int dtgsen(__CLPK_integer *ijob, __CLPK_logical *wantq, __CLPK_logical *wantz,
2503     __CLPK_logical *select, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda,
2504     __CLPK_doublereal *
2505     b, __CLPK_integer *ldb, __CLPK_doublereal *alphar, __CLPK_doublereal *alphai, __CLPK_doublereal *
2506     beta, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_doublereal *z__, __CLPK_integer *ldz,
2507     __CLPK_integer *m,
2508     __CLPK_doublereal *pl, __CLPK_doublereal *pr, __CLPK_doublereal *dif,
2509     __CLPK_doublereal *work,
2510     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork,
2511     __CLPK_integer *info);
2512
2513 /* Subroutine */ int dtgsja(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
2514     __CLPK_integer *p, __CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *l, __CLPK_doublereal *a,
2515     __CLPK_integer *lda,
2516     __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublereal *tola,
2517     __CLPK_doublereal *tolb,
2518     __CLPK_doublereal *alpha, __CLPK_doublereal *beta, __CLPK_doublereal *u,
2519     __CLPK_integer *ldu,
2520     __CLPK_doublereal *v, __CLPK_integer *ldv, __CLPK_doublereal *q,
2521     __CLPK_integer *
2522     ldq, __CLPK_doublereal *work, __CLPK_integer *ncycle, __CLPK_integer *info);
2523
2524 /* Subroutine */ int dtgsna(char *job, char *howmny, __CLPK_logical *select,
2525     __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer
2526     *ldb,
2527     __CLPK_doublereal *vl, __CLPK_integer *ldvl, __CLPK_doublereal *vr, __CLPK_integer *ldvr,
2528     __CLPK_doublereal *s,
2529     __CLPK_doublereal *dif, __CLPK_integer *mm, __CLPK_integer *m,
2530     __CLPK_doublereal *
2531     work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *info);
2532
2533 /* Subroutine */ int dtgsy2(char *trans, __CLPK_integer *ijob, __CLPK_integer *m, __CLPK_integer *
2534     n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,
2535     __CLPK_doublereal *c__,
2536     __CLPK_integer *ldc, __CLPK_doublereal *d__, __CLPK_integer *ldd,
2537     __CLPK_doublereal *e,
2538     __CLPK_integer *lde, __CLPK_doublereal *f, __CLPK_integer *ldf,
2539     __CLPK_doublereal *
2540     scale, __CLPK_doublereal *rdsum, __CLPK_doublereal *rdscal, __CLPK_integer *iwork, __CLPK_integer
2541     *pq, __CLPK_integer *info);
2542
2543 /* Subroutine */ int dtgsyl(char *trans, __CLPK_integer *ijob, __CLPK_integer *m, __CLPK_integer *
2544     n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb,

```

```

2523     __CLPK_doublereal *c__, __CLPK_integer *ldc, __CLPK_doublereal *d__, __CLPK_integer *ldd,
2524     __CLPK_doublereal *e, __CLPK_integer *lde, __CLPK_doublereal *f, __CLPK_integer *ldf,
    __CLPK_doublereal *
2525     scale, __CLPK_doublereal *dif, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *
2526     iwork, __CLPK_integer *info);
2527
2528 /* Subroutine */ int dtpcon_(char *norm, char *uplo, char *diag, __CLPK_integer *n,
2529     __CLPK_doublereal *ap, __CLPK_doublereal *rcond, __CLPK_doublereal *work, __CLPK_integer *iwork,
2530     __CLPK_integer *info);
2531
2532 /* Subroutine */ int dtprfs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
2533     __CLPK_integer *nrhs, __CLPK_doublereal *ap, __CLPK_doublereal *b, __CLPK_integer *ldb,
2534     __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
2535     __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
2536
2537 /* Subroutine */ int dtptri_(char *uplo, char *diag, __CLPK_integer *n, __CLPK_doublereal *
2538     ap, __CLPK_integer *info);
2539
2540 /* Subroutine */ int dtptsr_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
2541     __CLPK_integer *nrhs, __CLPK_doublereal *ap, __CLPK_doublereal *b, __CLPK_integer *ldb,
    __CLPK_integer *
2542     info);
2543
2544 /* Subroutine */ int dtrcon_(char *norm, char *uplo, char *diag, __CLPK_integer *n,
2545     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *rcond, __CLPK_doublereal *work,
2546     __CLPK_integer *iwork, __CLPK_integer *info);
2547
2548 /* Subroutine */ int dtrevc_(char *side, char *howmny, __CLPK_logical *select,
2549     __CLPK_integer *n, __CLPK_doublereal *t, __CLPK_integer *ldt, __CLPK_doublereal *vl, __CLPK_integer
    *
2550     ldvl, __CLPK_doublereal *vr, __CLPK_integer *ldvr, __CLPK_integer *mm, __CLPK_integer *m,
2551     __CLPK_doublereal *work, __CLPK_integer *info);
2552
2553 /* Subroutine */ int dtrexc_(char *compq, __CLPK_integer *n, __CLPK_doublereal *t, __CLPK_integer *
2554     ldt, __CLPK_doublereal *q, __CLPK_integer *ldq, __CLPK_integer *ifst, __CLPK_integer *ilst,
2555     __CLPK_doublereal *work, __CLPK_integer *info);
2556
2557 /* Subroutine */ int dtrrfs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
2558     __CLPK_integer *nrhs, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
    __CLPK_integer *
2559     ldb, __CLPK_doublereal *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
2560     __CLPK_doublereal *work, __CLPK_integer *iwork, __CLPK_integer *info);
2561
2562 /* Subroutine */ int dtrsen_(char *job, char *compq, __CLPK_logical *select, __CLPK_integer
2563     *n, __CLPK_doublereal *t, __CLPK_integer *ldt, __CLPK_doublereal *q, __CLPK_integer *ldq,
2564     __CLPK_doublereal *wr, __CLPK_doublereal *wi, __CLPK_integer *m, __CLPK_doublereal *s,
    __CLPK_doublereal
2565     *sep, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *
2566     liwork, __CLPK_integer *info);
2567
2568 /* Subroutine */ int dtrsna_(char *job, char *howmny, __CLPK_logical *select,
2569     __CLPK_integer *n, __CLPK_doublereal *t, __CLPK_integer *ldt, __CLPK_doublereal *vl, __CLPK_integer
    *
2570     ldvl, __CLPK_doublereal *vr, __CLPK_integer *ldvr, __CLPK_doublereal *s, __CLPK_doublereal *sep,
2571     __CLPK_integer *mm, __CLPK_integer *m, __CLPK_doublereal *work, __CLPK_integer *ldwork,
    __CLPK_integer *
2572     iwork, __CLPK_integer *info);
2573
2574 /* Subroutine */ int dtrsyl_(char *trana, char *tranb, __CLPK_integer *isgn, __CLPK_integer
2575     *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
    __CLPK_integer *
2576     ldb, __CLPK_doublereal *c__, __CLPK_integer *ldc, __CLPK_doublereal *scale, __CLPK_integer *info);
2577
2578 /* Subroutine */ int dtrti2_(char *uplo, char *diag, __CLPK_integer *n, __CLPK_doublereal *
2579     a, __CLPK_integer *lda, __CLPK_integer *info);
2580
2581 /* Subroutine */ int dtrtri_(char *uplo, char *diag, __CLPK_integer *n, __CLPK_doublereal *
2582     a, __CLPK_integer *lda, __CLPK_integer *info);
2583
2584 /* Subroutine */ int dtrtrs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
2585     __CLPK_integer *nrhs, __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *b,
    __CLPK_integer *
2586     ldb, __CLPK_integer *info);
2587
2588 /* Subroutine */ int dtzrqf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
    *
2589     lda, __CLPK_doublereal *tau, __CLPK_integer *info);
2590
2591 /* Subroutine */ int dtzrzf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
    *
2592     lda, __CLPK_doublereal *tau, __CLPK_doublereal *work, __CLPK_integer *lwork, __CLPK_integer *info);
2593
2594 __CLPK_integer icmax1(__CLPK_integer *n, __CLPK_complex *cx, __CLPK_integer *incx);
2595
2596 __CLPK_integer ieeeck__(__CLPK_integer *ispec, __CLPK_real *zero, __CLPK_real *one);
2597
2598 __CLPK_integer ilaenv__(__CLPK_integer *ispec, char *name__, char *opts, __CLPK_integer *n1,

```

```

2599     __CLPK_integer *n2, __CLPK_integer *n3, __CLPK_integer *n4, __CLPK_ftnlen name_len, __CLPK_ftnlen
2600     opts_len);
2601
2602     __CLPK_integer izmax1(__CLPK_integer *n, __CLPK_doublecomplex *cx, __CLPK_integer *incx);
2603
2604     /* Subroutine */ int sbsdsc(char *uplo, char *compq, __CLPK_integer *n, __CLPK_real *d__,
2605     __CLPK_real *e, __CLPK_real *u, __CLPK_integer *ldu, __CLPK_real *vt, __CLPK_integer *ldvt,
2606     __CLPK_real *q,
2607     __CLPK_integer *iq, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
2608
2609     /* Subroutine */ int sbsdqr(char *uplo, __CLPK_integer *n, __CLPK_integer *ncvt, __CLPK_integer *
2610     nru, __CLPK_integer *ncc, __CLPK_real *d__, __CLPK_real *e, __CLPK_real *vt, __CLPK_integer *ldvt,
2611     __CLPK_real *
2612     u, __CLPK_integer *ldu, __CLPK_real *c__, __CLPK_integer *ldc, __CLPK_real *work, __CLPK_integer
2613     *info);
2614
2615     /* Subroutine */ int sdisna(char *job, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *d__,
2616     __CLPK_real *sep, __CLPK_integer *info);
2617
2618     /* Subroutine */ int sgbrd(char *vect, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *ncc,
2619     __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *d__,
2620     __CLPK_real *
2621     e, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *pt, __CLPK_integer *ldpt, __CLPK_real *c__,
2622     __CLPK_integer
2623     *ldc, __CLPK_real *work, __CLPK_integer *info);
2624
2625     /* Subroutine */ int sgbcon(char *norm, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku,
2626     __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_real *anorm, __CLPK_real
2627     *rcond,
2628     __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
2629
2630     /* Subroutine */ int sgbequ(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
2631     *ku,
2632     __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *rowcnd,
2633     __CLPK_real *
2634     colcnd, __CLPK_real *amax, __CLPK_integer *info);
2635
2636     /* Subroutine */ int sgrfs(char *trans, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *
2637     ku, __CLPK_integer *nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *afb, __CLPK_integer
2638     *ldafb,
2639     __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx,
2640     __CLPK_real *
2641     ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
2642
2643     /* Subroutine */ int sgbsv(__CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_integer *
2644     nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer
2645     *ldb,
2646     __CLPK_integer *info);
2647
2648     /* Subroutine */ int sgbsvx(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *kl,
2649     __CLPK_integer *ku, __CLPK_integer *nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *afb,
2650     __CLPK_integer *ldafb,
2651     __CLPK_integer *ipiv, char *equed, __CLPK_real *r__, __CLPK_real *c__,
2652     __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx, __CLPK_real *rcond,
2653     __CLPK_real *ferr,
2654     __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
2655
2656     /* Subroutine */ int sgbtf2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
2657     *ku,
2658     __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_integer *info);
2659
2660     /* Subroutine */ int sgbtrf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
2661     *ku,
2662     __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_integer *info);
2663
2664     /* Subroutine */ int sgbtrs(char *trans, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *
2665     ku, __CLPK_integer *nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_real
2666     *b,
2667     __CLPK_integer *ldb, __CLPK_integer *info);
2668
2669     /* Subroutine */ int sgebak(char *job, char *side, __CLPK_integer *n, __CLPK_integer *ilo,
2670     __CLPK_integer *ihi, __CLPK_real *scale, __CLPK_integer *m, __CLPK_real *v, __CLPK_integer *ldv,
2671     __CLPK_integer
2672     *info);
2673
2674     /* Subroutine */ int sgebal(char *job, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2675     __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real *scale, __CLPK_integer *info);
2676
2677     /* Subroutine */ int sgebd2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2678     __CLPK_real *d__, __CLPK_real *e, __CLPK_real *tauq, __CLPK_real *taup, __CLPK_real *work,
2679     __CLPK_integer *info);
2680
2681     /* Subroutine */ int sgebrd(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2682     __CLPK_real *d__, __CLPK_real *e, __CLPK_real *tauq, __CLPK_real *taup, __CLPK_real *work,
2683     __CLPK_integer *

```

```

2665     lwork, __CLPK_integer *info);
2666
2667 /* Subroutine */ int sgecon_(char *norm, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2668     __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
    *info);
2669
2670 /* Subroutine */ int sgeequ_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2671     __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *rowcnd, __CLPK_real *colcnd, __CLPK_real *amax,
    __CLPK_integer
    *info);
2672
2673
2674 /* Subroutine */ int sgees_(char *jobvs, char *sort, __CLPK_L_fp select, __CLPK_integer *n,
2675     __CLPK_real *a, __CLPK_integer *lda, __CLPK_integer *sdim, __CLPK_real *wr, __CLPK_real *wi,
    __CLPK_real *vs,
    __CLPK_integer *ldvs, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_logical *bwork,
    __CLPK_integer *
    info);
2677
2678
2679 /* Subroutine */ int sgeesx_(char *jobvs, char *sort, __CLPK_L_fp select, char *
2680     sense, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_integer *sdim, __CLPK_real
    *wr,
    __CLPK_real *wi, __CLPK_real *vs, __CLPK_integer *ldvs, __CLPK_real *rconde, __CLPK_real *rcondv,
    __CLPK_real *
    work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_logical *bwork,
    __CLPK_integer *info);
2682
2683
2684
2685 /* Subroutine */ int sgeev_(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_real *a,
2686     __CLPK_integer *lda, __CLPK_real *wr, __CLPK_real *wi, __CLPK_real *vl, __CLPK_integer *ldvl,
    __CLPK_real *vr,
    __CLPK_integer *ldvr, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2688
2689 /* Subroutine */ int sgeevx_(char *balanc, char *jobvl, char *jobvr, char *
2690     sense, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *wr, __CLPK_real *wi,
    __CLPK_real *
    vl, __CLPK_integer *ldvl, __CLPK_real *vr, __CLPK_integer *ldvr, __CLPK_integer *ilo,
    __CLPK_integer *
    ihi, __CLPK_real *scale, __CLPK_real *abnrm, __CLPK_real *rconde, __CLPK_real *rcondv, __CLPK_real
    *work,
    __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *info);
2693
2694
2695 /* Subroutine */ int sgegs_(char *jobvsl, char *jobvsr, __CLPK_integer *n, __CLPK_real *a,
2696     __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *alphar, __CLPK_real *alpai,
    __CLPK_real
    *beta, __CLPK_real *vsl, __CLPK_integer *ldvsl, __CLPK_real *vsr, __CLPK_integer *ldvsr,
    __CLPK_real *
    work, __CLPK_integer *lwork, __CLPK_integer *info);
2698
2699
2700 /* Subroutine */ int sggev_(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_real *a,
2701     __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *alphar, __CLPK_real *alpai,
    __CLPK_real
    *beta, __CLPK_real *vl, __CLPK_integer *ldvl, __CLPK_real *vr, __CLPK_integer *ldvr, __CLPK_real
    *work,
    __CLPK_integer *lwork, __CLPK_integer *info);
2703
2704
2705 /* Subroutine */ int sgehd2_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real
    *a,
    __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
2706
2707
2708 /* Subroutine */ int sgehrd_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real
    *a,
    __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
    *info);
2709
2710
2711 /* Subroutine */ int sgelq2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2712     __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
2713
2714 /* Subroutine */ int sgelqf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2715     __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2716
2717 /* Subroutine */ int sgels_(char *trans, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *
2718     nrhs, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *work,
    __CLPK_integer *lwork,
    __CLPK_integer *info);
2719
2720
2721 /* Subroutine */ int sgelsd_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real
    *a,
    __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *s, __CLPK_real *rcond,
    __CLPK_integer *
    rank, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *info);
2723
2724
2725 /* Subroutine */ int sgelss_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real
    *a,
    __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *s, __CLPK_real *rcond,
    __CLPK_integer *
    rank, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2727

```

```
2728
2729 /* Subroutine */ int sgelsx(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real
2730 *a,
2731 __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *jpvt, __CLPK_real *rcond,
2732 __CLPK_integer *rank, __CLPK_real *work, __CLPK_integer *info);
2733 /* Subroutine */ int sgelsy(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real
2734 *a,
2735 __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *jpvt, __CLPK_real *rcond,
2736 __CLPK_integer *rank, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2737 /* Subroutine */ int sgeql2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2738 __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
2739
2740 /* Subroutine */ int sgeqlf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2741 __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2742
2743 /* Subroutine */ int sgeqp3(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2744 __CLPK_integer *jpvt, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
2745 *info);
2746 /* Subroutine */ int sgeqpf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2747 __CLPK_integer *jpvt, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
2748
2749 /* Subroutine */ int sgeqr2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2750 __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
2751
2752 /* Subroutine */ int sgeqrf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2753 __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2754
2755 /* Subroutine */ int sgerfs(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a,
2756 __CLPK_integer *lda, __CLPK_real *af, __CLPK_integer *ldaf, __CLPK_integer *ipiv, __CLPK_real *b,
2757 __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr,
2758 __CLPK_real *
2759 work, __CLPK_integer *iwork, __CLPK_integer *info);
2760 /* Subroutine */ int sgerq2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2761 __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
2762
2763 /* Subroutine */ int sgerqf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2764 __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2765
2766 /* Subroutine */ int sgesc2(__CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *rhs,
2767 __CLPK_integer *ipiv, __CLPK_integer *jpiv, __CLPK_real *scale);
2768
2769 /* Subroutine */ int sgesdd(char *jobz, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a,
2770 __CLPK_integer *lda, __CLPK_real *s, __CLPK_real *u, __CLPK_integer *ldu, __CLPK_real *vt,
2771 __CLPK_integer *ldvt,
2772 __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *info);
2773 /* Subroutine */ int sgesv(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a, __CLPK_integer
2774 *lda,
2775 __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
2776 /* Subroutine */ int sgesvd(char *jobu, char *jobvt, __CLPK_integer *m, __CLPK_integer *n,
2777 __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *s, __CLPK_real *u, __CLPK_integer *ldu,
2778 __CLPK_real *vt,
2779 __CLPK_integer *ldvt, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2780 /* Subroutine */ int sgesvx(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *
2781 nrhs, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *af, __CLPK_integer *ldaf, __CLPK_integer
2782 *ipiv,
2783 char *equed, __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real
2784 *x,
2785 __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work,
2786 __CLPK_integer *iwork, __CLPK_integer *info);
2787
2788 /* Subroutine */ int sgetc2(__CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_integer
2789 *ipiv,
2790 __CLPK_integer *jpiv, __CLPK_integer *info);
2791
2792 /* Subroutine */ int sgetf2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
```

```

2793     __CLPK_integer *ipiv, __CLPK_integer *info);
2794
2795 /* Subroutine */ int sgetri(__CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_integer
2796 *ipiv,
2797     __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2798
2799 /* Subroutine */ int sgetrs(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a,
2800     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer
2801 *info);
2802
2803 /* Subroutine */ int sgbak(char *job, char *side, __CLPK_integer *n, __CLPK_integer *ilo,
2804     __CLPK_integer *ihi, __CLPK_real *lscale, __CLPK_real *rscale, __CLPK_integer *m, __CLPK_real *v,
2805     __CLPK_integer *ldv, __CLPK_integer *info);
2806
2807 /* Subroutine */ int sgbal(char *job, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
2808     __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real *lscale,
2809     __CLPK_real *rscale, __CLPK_real *work, __CLPK_integer *info);
2810
2811 /* Subroutine */ int sgges(char *jobvsl, char *jobvsr, char *sort, __CLPK_L_fp
2812     selctg, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer
2813 *ldb,
2814     __CLPK_integer *sdim, __CLPK_real *alpar, __CLPK_real *alphai, __CLPK_real *beta, __CLPK_real
2815 *vsl,
2816     __CLPK_integer *ldvsl, __CLPK_real *vsr, __CLPK_integer *ldvsr, __CLPK_real *work, __CLPK_integer
2817 *lwork,
2818     __CLPK_logical *bwork, __CLPK_integer *info);
2819
2820 /* Subroutine */ int sggesx(char *jobvsl, char *jobvsr, char *sort, __CLPK_L_fp
2821     selctg, char *sense, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b,
2822     __CLPK_integer *ldb, __CLPK_integer *sdim, __CLPK_real *alpar, __CLPK_real *alphai, __CLPK_real
2823 *beta,
2824     __CLPK_real *vsl, __CLPK_integer *ldvsl, __CLPK_real *vsr, __CLPK_integer *ldvsr, __CLPK_real
2825 *rconde,
2826     __CLPK_real *rcondv, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork,
2827     __CLPK_integer *
2828     liwork, __CLPK_logical *bwork, __CLPK_integer *info);
2829
2830 /* Subroutine */ int sggev(char *jobvl, char *jobvr, __CLPK_integer *n, __CLPK_real *a,
2831     __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *alpar, __CLPK_real *alphai,
2832     __CLPK_real *beta, __CLPK_real *vl, __CLPK_integer *ldvl, __CLPK_real *vr, __CLPK_integer *ldvr,
2833     __CLPK_real
2834 *work,
2835     __CLPK_integer *lwork, __CLPK_integer *info);
2836
2837 /* Subroutine */ int sggevx(char *balanc, char *jobvl, char *jobvr, char *
2838     sense, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
2839     __CLPK_real
2840 *alpar, __CLPK_real *alphai, __CLPK_real *beta, __CLPK_real *vl, __CLPK_integer *ldvl, __CLPK_real
2841 *vr,
2842     __CLPK_integer *ldvr, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real *lscale, __CLPK_real
2843 *rscale,
2844     __CLPK_real *abnrm, __CLPK_real *bbnrm, __CLPK_real *rconde, __CLPK_real *rcondv, __CLPK_real
2845 *work,
2846     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_logical *bwork, __CLPK_integer *info);
2847
2848 /* Subroutine */ int sggglm(__CLPK_integer *n, __CLPK_integer *m, __CLPK_integer *p, __CLPK_real *a,
2849     __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *d, __CLPK_real *x,
2850     __CLPK_real *y,
2851     __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2852
2853 /* Subroutine */ int sgghrd(char *compq, char *compz, __CLPK_integer *n, __CLPK_integer *
2854     ilo, __CLPK_integer *ihi, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
2855     __CLPK_real
2856 *q, __CLPK_integer *ldq, __CLPK_real *z, __CLPK_integer *ldz, __CLPK_integer *info);
2857
2858 /* Subroutine */ int sgglsd(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *p, __CLPK_real *a,
2859     __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *c, __CLPK_real *d,
2860     __CLPK_real *x,
2861     __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2862
2863 /* Subroutine */ int sggqrf(__CLPK_integer *n, __CLPK_integer *m, __CLPK_integer *p, __CLPK_real *a,
2864     __CLPK_integer *lda, __CLPK_real *taua, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *taub,
2865     __CLPK_real *
2866     work, __CLPK_integer *lwork, __CLPK_integer *info);
2867
2868 /* Subroutine */ int sggrqf(__CLPK_integer *m, __CLPK_integer *p, __CLPK_integer *n, __CLPK_real *a,
2869     __CLPK_integer *lda, __CLPK_real *taua, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *taub,
2870     __CLPK_real *
2871     work, __CLPK_integer *lwork, __CLPK_integer *info);
2872
2873 /* Subroutine */ int sggsvd(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
2874     __CLPK_integer *n, __CLPK_integer *p, __CLPK_integer *k, __CLPK_integer *l, __CLPK_real *a,
2875     __CLPK_integer *lda,
2876     __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *alpha, __CLPK_real *beta, __CLPK_real *u,
2877     __CLPK_integer *
2878     ldu, __CLPK_real *v, __CLPK_integer *ldv, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *work,

```



```

2858     __CLPK_integer *iwork, __CLPK_integer *info);
2859
2860 /* Subroutine */ int sggsvp_(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
2861     __CLPK_integer *p, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b,
2862     __CLPK_integer *ldb,
2863     __CLPK_real *tola, __CLPK_real *tolb, __CLPK_integer *k, __CLPK_integer *l, __CLPK_real *u,
2864     __CLPK_integer *ldu,
2865     __CLPK_real *v, __CLPK_integer *ldv, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_integer *iwork,
2866     __CLPK_real *
2867     tau, __CLPK_real *work, __CLPK_integer *info);
2868
2869 /* Subroutine */ int sgtcon_(char *norm, __CLPK_integer *n, __CLPK_real *dl, __CLPK_real *d__,
2870     __CLPK_real *du, __CLPK_real *du2, __CLPK_integer *ipiv, __CLPK_real *anorm, __CLPK_real *rcond,
2871     __CLPK_real *
2872     work, __CLPK_integer *iwork, __CLPK_integer *info);
2873
2874 /* Subroutine */ int sgrfs_(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *dl,
2875     __CLPK_real *d__, __CLPK_real *du, __CLPK_real *dlf, __CLPK_real *df, __CLPK_real *duf,
2876     __CLPK_real *du2,
2877     __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx,
2878     __CLPK_real *
2879     ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
2880
2881 /* Subroutine */ int sgtsv_(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *dl, __CLPK_real *d__,
2882     __CLPK_real *du, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
2883
2884 /* Subroutine */ int sgtsvx_(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *
2885     nrhs, __CLPK_real *dl, __CLPK_real *d__, __CLPK_real *du, __CLPK_real *dlf, __CLPK_real *df,
2886     __CLPK_real *duf,
2887     __CLPK_real *du2, __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x,
2888     __CLPK_integer *
2889     ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer
2890     *iwork,
2891     __CLPK_integer *info);
2892
2893 /* Subroutine */ int sgtrf_(__CLPK_integer *n, __CLPK_real *dl, __CLPK_real *d__, __CLPK_real *du,
2894     __CLPK_real *
2895     du2, __CLPK_integer *ipiv, __CLPK_integer *info);
2896
2897 /* Subroutine */ int sgtrrs_(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *dl,
2898     __CLPK_real *d__, __CLPK_real *du, __CLPK_real *du2, __CLPK_integer *ipiv, __CLPK_real *b,
2899     __CLPK_integer *ldb,
2900     __CLPK_integer *info);
2901
2902 /* Subroutine */ int sgts2_(__CLPK_integer *itrans, __CLPK_integer *n, __CLPK_integer *nrhs,
2903     __CLPK_real
2904     *dl, __CLPK_real *d__, __CLPK_real *du, __CLPK_real *du2, __CLPK_integer *ipiv, __CLPK_real *b,
2905     __CLPK_integer *
2906     ldb);
2907
2908 /* Subroutine */ int shgeqz_(char *job, char *compq, char *compz, __CLPK_integer *n,
2909     __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b,
2910     __CLPK_integer *
2911     ldb, __CLPK_real *alphar, __CLPK_real *alphai, __CLPK_real *beta, __CLPK_real *q, __CLPK_integer
2912     *ldq,
2913     __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
2914     *info);
2915
2916 /* Subroutine */ int shsein_(char *side, char *eigsrc, char *initv, __CLPK_logical *
2917     select, __CLPK_integer *n, __CLPK_real *h__, __CLPK_integer *ldh, __CLPK_real *wr, __CLPK_real *wi,
2918     __CLPK_real *
2919     *vl, __CLPK_integer *ldvl, __CLPK_real *vr, __CLPK_integer *ldvr, __CLPK_integer *mm,
2920     __CLPK_integer *m,
2921     __CLPK_real *work, __CLPK_integer *ifaill, __CLPK_integer *ifailr, __CLPK_integer *info);
2922
2923 /* Subroutine */ int shseqr_(char *job, char *compz, __CLPK_integer *n, __CLPK_integer *ilo,
2924     __CLPK_integer *ihi, __CLPK_real *h__, __CLPK_integer *ldh, __CLPK_real *wr, __CLPK_real *wi,
2925     __CLPK_real *z__,
2926     __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
2927
2928 /* Subroutine */ int slabad_(__CLPK_real *small, __CLPK_real *large);
2929
2930 /* Subroutine */ int slabrd_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_real *a,
2931     __CLPK_integer *lda, __CLPK_real *d__, __CLPK_real *e, __CLPK_real *tauq, __CLPK_real *taup,
2932     __CLPK_real *x,
2933     __CLPK_integer *ldx, __CLPK_real *y, __CLPK_integer *ldy);
2934
2935 /* Subroutine */ int slacon_(__CLPK_integer *n, __CLPK_real *v, __CLPK_real *x, __CLPK_integer *isgn,
2936     __CLPK_real *est, __CLPK_integer *kase);
2937
2938 /* Subroutine */ int slacpy_(char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a,
2939     __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb);
2940
2941 /* Subroutine */ int sladiv_(__CLPK_real *a, __CLPK_real *b, __CLPK_real *c__, __CLPK_real *d__,
2942     __CLPK_real *p,
2943     __CLPK_real *q);

```



```

2923
2924 /* Subroutine */ int slae2__(__CLPK_real *a, __CLPK_real *b, __CLPK_real *c__, __CLPK_real *rt1,
    __CLPK_real *rt2);
2925
2926 /* Subroutine */ int slaebz__(__CLPK_integer *ijob, __CLPK_integer *nitmax, __CLPK_integer *n,
2927     __CLPK_integer *mmax, __CLPK_integer *minp, __CLPK_integer *nbmin, __CLPK_real *abstol, __CLPK_real
    *
2928     reltol, __CLPK_real *pivmin, __CLPK_real *d__, __CLPK_real *e, __CLPK_real *e2, __CLPK_integer
    *nval,
2929     __CLPK_real *ab, __CLPK_real *c__, __CLPK_integer *mout, __CLPK_integer *nab, __CLPK_real *work,
    __CLPK_integer
2930     *iwork, __CLPK_integer *info);
2931
2932 /* Subroutine */ int slaed0__(__CLPK_integer *icompq, __CLPK_integer *qsiz, __CLPK_integer *n,
    __CLPK_real
2933     *d__, __CLPK_real *e, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *qstore, __CLPK_integer
    *ldqs,
2934     __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
2935
2936 /* Subroutine */ int slaed1__(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *q, __CLPK_integer *ldq,
2937     __CLPK_integer *indxq, __CLPK_real *rho, __CLPK_integer *cutpnt, __CLPK_real *work, __CLPK_integer
    *
2938     iwork, __CLPK_integer *info);
2939
2940 /* Subroutine */ int slaed2__(__CLPK_integer *k, __CLPK_integer *n, __CLPK_integer *n1, __CLPK_real
    *d__,
2941     __CLPK_real *q, __CLPK_integer *ldq, __CLPK_integer *indxq, __CLPK_real *rho, __CLPK_real *z__,
    __CLPK_real *
2942     dlamda, __CLPK_real *w, __CLPK_real *q2, __CLPK_integer *indx, __CLPK_integer *indxc,
    __CLPK_integer *
2943     indxp, __CLPK_integer *coltyp, __CLPK_integer *info);
2944
2945 /* Subroutine */ int slaed3__(__CLPK_integer *k, __CLPK_integer *n, __CLPK_integer *n1, __CLPK_real
    *d__,
2946     __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *rho, __CLPK_real *dlamda, __CLPK_real *q2,
    __CLPK_integer *
2947     indx, __CLPK_integer *ctot, __CLPK_real *w, __CLPK_real *s, __CLPK_integer *info);
2948
2949 /* Subroutine */ int slaed4__(__CLPK_integer *n, __CLPK_integer *i__, __CLPK_real *d__, __CLPK_real
    *z__,
2950     __CLPK_real *delta, __CLPK_real *rho, __CLPK_real *dlam, __CLPK_integer *info);
2951
2952 /* Subroutine */ int slaed5__(__CLPK_integer *i__, __CLPK_real *d__, __CLPK_real *z__, __CLPK_real
    *delta,
2953     __CLPK_real *rho, __CLPK_real *dlam);
2954
2955 /* Subroutine */ int slaed6__(__CLPK_integer *kniter, __CLPK_logical *orgati, __CLPK_real *rho,
2956     __CLPK_real *d__, __CLPK_real *z__, __CLPK_real *finit, __CLPK_real *tau, __CLPK_integer *info);
2957
2958 /* Subroutine */ int slaed7__(__CLPK_integer *icompq, __CLPK_integer *n, __CLPK_integer *qsiz,
2959     __CLPK_integer *tlvl, __CLPK_integer *curlvl, __CLPK_integer *curpbm, __CLPK_real *d__,
    __CLPK_real *q,
2960     __CLPK_integer *ldq, __CLPK_integer *indxq, __CLPK_real *rho, __CLPK_integer *cutpnt, __CLPK_real *
    qstore, __CLPK_integer *qptr, __CLPK_integer *prmptr, __CLPK_integer *perm, __CLPK_integer *
2961     givptr, __CLPK_integer *givcol, __CLPK_real *givnum, __CLPK_real *work, __CLPK_integer *iwork,
    __CLPK_integer *info);
2962
2963
2964
2965 /* Subroutine */ int slaed8__(__CLPK_integer *icompq, __CLPK_integer *k, __CLPK_integer *n,
    __CLPK_integer
2966     *qsiz, __CLPK_real *d__, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_integer *indxq, __CLPK_real
    *rho,
2967     __CLPK_integer *cutpnt, __CLPK_real *z__, __CLPK_real *dlamda, __CLPK_real *q2, __CLPK_integer
    *ldq2,
2968     __CLPK_real *w, __CLPK_integer *perm, __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_real *
    givnum, __CLPK_integer *indxp, __CLPK_integer *indx, __CLPK_integer *info);
2969
2970
2971 /* Subroutine */ int slaed9__(__CLPK_integer *k, __CLPK_integer *kstart, __CLPK_integer *kstop,
2972     __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *rho,
    __CLPK_real *dlamda,
2973     __CLPK_real *w, __CLPK_real *s, __CLPK_integer *lds, __CLPK_integer *info);
2974
2975 /* Subroutine */ int slaeda__(__CLPK_integer *n, __CLPK_integer *tlvl, __CLPK_integer *curlvl,
2976     __CLPK_integer *curpbm, __CLPK_integer *prmptr, __CLPK_integer *perm, __CLPK_integer *givptr,
    __CLPK_integer *givcol,
2977     __CLPK_real *givnum, __CLPK_real *q, __CLPK_integer *qptr, __CLPK_real
    *z__,
2978     __CLPK_real *ztemp, __CLPK_integer *info);
2979
2980 /* Subroutine */ int slaein__(__CLPK_logical *rightv, __CLPK_logical *noinit, __CLPK_integer *n,
2981     __CLPK_real *h__, __CLPK_integer *ldh, __CLPK_real *wr, __CLPK_real *wi, __CLPK_real *vr,
    __CLPK_real *vi,
2982     __CLPK_integer *ldb, __CLPK_real *work, __CLPK_real *eps3, __CLPK_real *smlnum, __CLPK_real
    *bignum,
2983     __CLPK_integer *info);
2984
2985 /* Subroutine */ int slaev2__(__CLPK_real *a, __CLPK_real *b, __CLPK_real *c__, __CLPK_real *rt1,
    __CLPK_real *
2986     rt2, __CLPK_real *cs1, __CLPK_real *sn1);

```

```

2987
2988 /* Subroutine */ int slaexc(__CLPK_logical *wantq, __CLPK_integer *n, __CLPK_real *t, __CLPK_integer *
2989   ldt, __CLPK_real *q, __CLPK_integer *ldg, __CLPK_integer *jl, __CLPK_integer *nl, __CLPK_integer
   *n2,
2990   __CLPK_real *work, __CLPK_integer *info);
2991
2992 /* Subroutine */ int slag2(__CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
2993   __CLPK_real *safmin, __CLPK_real *scale1, __CLPK_real *scale2, __CLPK_real *wr1, __CLPK_real *wr2,
   __CLPK_real *
2994   wi);
2995
2996 /* Subroutine */ int slags2(__CLPK_logical *upper, __CLPK_real *a1, __CLPK_real *a2, __CLPK_real *a3,
2997   __CLPK_real *b1, __CLPK_real *b2, __CLPK_real *b3, __CLPK_real *csu, __CLPK_real *snu, __CLPK_real
   *csv, __CLPK_real *
2998   snv, __CLPK_real *csq, __CLPK_real *snq);
2999
3000 /* Subroutine */ int slagtf(__CLPK_integer *n, __CLPK_real *a, __CLPK_real *lambda, __CLPK_real *b,
   __CLPK_real
3001   *c, __CLPK_real *tol, __CLPK_real *d, __CLPK_integer *in, __CLPK_integer *info);
3002
3003 /* Subroutine */ int slagtm(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *
3004   alpha, __CLPK_real *dl, __CLPK_real *d, __CLPK_real *du, __CLPK_real *x, __CLPK_integer *ldx,
   __CLPK_real *
3005   beta, __CLPK_real *b, __CLPK_integer *ldb);
3006
3007 /* Subroutine */ int slagts(__CLPK_integer *job, __CLPK_integer *n, __CLPK_real *a, __CLPK_real *b,
   __CLPK_real
3008   *c, __CLPK_real *d, __CLPK_integer *in, __CLPK_real *y, __CLPK_real *tol, __CLPK_integer
   *info);
3009
3010 /* Subroutine */ int slagv2(__CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
3011   __CLPK_real *alphar, __CLPK_real *alphai, __CLPK_real *beta, __CLPK_real *csl, __CLPK_real *snl,
   __CLPK_real *
3012   csr, __CLPK_real *snr);
3013
3014 /* Subroutine */ int slahr(__CLPK_logical *wantt, __CLPK_logical *wantz, __CLPK_integer *n,
3015   __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real *h, __CLPK_integer *ldh, __CLPK_real *wr,
   __CLPK_real *
3016   wi, __CLPK_integer *iloz, __CLPK_integer *ihiz, __CLPK_real *z, __CLPK_integer *ldz,
   __CLPK_integer *
3017   info);
3018
3019 /* Subroutine */ int slahrd(__CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *nb, __CLPK_real *a,
3020   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *t, __CLPK_integer *ldt, __CLPK_real *y,
   __CLPK_integer *ldy);
3021
3022 /* Subroutine */ int slaicl(__CLPK_integer *job, __CLPK_integer *j, __CLPK_real *x, __CLPK_real *sest,
   __CLPK_real *
3023   w, __CLPK_real *gamma, __CLPK_real *sestpr, __CLPK_real *s, __CLPK_real *c);
3024
3025 /* Subroutine */ int slaln2(__CLPK_logical *ltrans, __CLPK_integer *na, __CLPK_integer *nw,
   __CLPK_real *
3026   smin, __CLPK_real *ca, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *d1, __CLPK_real *d2,
   __CLPK_real *b,
3027   __CLPK_integer *ldb, __CLPK_real *wr, __CLPK_real *wi, __CLPK_real *x, __CLPK_integer *ldx,
   __CLPK_real *
3028   scale,
   __CLPK_real *xnrm, __CLPK_integer *info);
3029
3030 /* Subroutine */ int slals0(__CLPK_integer *icompq, __CLPK_integer *nl, __CLPK_integer *nr,
   __CLPK_integer *
3031   sqre, __CLPK_integer *nrhs, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *bx,
   __CLPK_integer *
3032   ldbx, __CLPK_integer *perm, __CLPK_integer *givptr, __CLPK_integer *givcol,
   __CLPK_integer *
3033   ldgcol, __CLPK_real *givnum, __CLPK_integer *ldgnum, __CLPK_real *poles,
   __CLPK_real *
3034   difl, __CLPK_real *difr, __CLPK_real *z, __CLPK_integer *k, __CLPK_real *c, __CLPK_real *s,
   __CLPK_real *
3035   work, __CLPK_integer *info);
3036
3037 /* Subroutine */ int slalsa(__CLPK_integer *icompq, __CLPK_integer *smlsiz, __CLPK_integer *n,
   __CLPK_integer *
3038   nrhs, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *bx, __CLPK_integer *ldbx,
   __CLPK_real *
3039   u, __CLPK_integer *ldu, __CLPK_real *vt, __CLPK_integer *k, __CLPK_real *difl, __CLPK_real *difr,
   __CLPK_real *
3040   z, __CLPK_real *poles, __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_integer *ldgcol,
   __CLPK_integer *
3041   perm, __CLPK_real *givnum, __CLPK_real *c, __CLPK_real *s, __CLPK_real *work,
   __CLPK_integer *
3042   iwork, __CLPK_integer *info);
3043
3044 /* Subroutine */ int slalsd(char *uplo, __CLPK_integer *smlsiz, __CLPK_integer *n, __CLPK_integer
   *nrhs,
3045   __CLPK_real *d, __CLPK_real *e, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *rcond,
   __CLPK_integer *
3046   rank, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
3047
3048 /* Subroutine */ int slamcl(__CLPK_integer *beta, __CLPK_integer *t, __CLPK_logical *rnd,
   __CLPK_logical
3049   *ieeel);
3050
3051 /* Subroutine */ int slamc2(__CLPK_integer *beta, __CLPK_integer *t, __CLPK_logical *rnd, __CLPK_real
   *

```

```
3052     eps, __CLPK_integer *emin, __CLPK_real *rmin, __CLPK_integer *emax, __CLPK_real *rmax);
3053
3054 /* Subroutine */ int slamc4(__CLPK_integer *emin, __CLPK_real *start, __CLPK_integer *base);
3055
3056 /* Subroutine */ int slamc5(__CLPK_integer *beta, __CLPK_integer *p, __CLPK_integer *emin,
3057     __CLPK_logical *ieee, __CLPK_integer *emax, __CLPK_real *rmax);
3058
3059 /* Subroutine */ int slamrg(__CLPK_integer *n1, __CLPK_integer *n2, __CLPK_real *a, __CLPK_integer *
3060     strd1, __CLPK_integer *strd2, __CLPK_integer *index);
3061
3062 /* Subroutine */ int slanv2(__CLPK_real *a, __CLPK_real *b, __CLPK_real *c__, __CLPK_real *d__,
3063     __CLPK_real *
3064     rtlr, __CLPK_real *rtli, __CLPK_real *rt2r, __CLPK_real *rt2i, __CLPK_real *cs, __CLPK_real *sn);
3065 /* Subroutine */ int slapll(__CLPK_integer *n, __CLPK_real *x, __CLPK_integer *incx, __CLPK_real *y,
3066     __CLPK_integer *incy, __CLPK_real *ssmin);
3067
3068 /* Subroutine */ int slapmt(__CLPK_logical *forwrd, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real
3069     *x,
3070     __CLPK_integer *ldx, __CLPK_integer *k);
3071
3072 /* Subroutine */ int slaqgb(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
3073     *ku,
3074     __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *rowcnd,
3075     __CLPK_real *
3076     colcnd, __CLPK_real *amax, char *equet);
3077
3078 /* Subroutine */ int slaqge(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3079     __CLPK_real *r__, __CLPK_real *c__, __CLPK_real *rowcnd, __CLPK_real *colcnd, __CLPK_real *amax,
3080     char *
3081     equet);
3082
3083 /* Subroutine */ int slaqp2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *offset, __CLPK_real
3084     *a,
3085     __CLPK_integer *lda, __CLPK_integer *jpv, __CLPK_real *tau, __CLPK_real *vn1, __CLPK_real *vn2,
3086     __CLPK_real *
3087     work);
3088
3089 /* Subroutine */ int slaqps(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *offset,
3090     __CLPK_integer
3091     *nb, __CLPK_integer *kb, __CLPK_real *a, __CLPK_integer *lda, __CLPK_integer *jpv, __CLPK_real
3092     *tau,
3093     __CLPK_real *vn1, __CLPK_real *vn2, __CLPK_real *auxv, __CLPK_real *f, __CLPK_integer *ldf);
3094
3095 /* Subroutine */ int slaqsb(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_real *ab,
3096     __CLPK_integer *ldab,
3097     __CLPK_real *s, __CLPK_real *scond, __CLPK_real *amax, char *equet);
3098
3099 /* Subroutine */ int slaqsp(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_real *s,
3100     __CLPK_real *
3101     scond, __CLPK_real *amax, char *equet);
3102
3103 /* Subroutine */ int slaqsy(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3104     __CLPK_real *s,
3105     __CLPK_real *scond, __CLPK_real *amax, char *equet);
3106
3107 /* Subroutine */ int slaqtr(__CLPK_logical *ltran, __CLPK_logical *lreal, __CLPK_integer *n,
3108     __CLPK_real
3109     *t, __CLPK_integer *ldt, __CLPK_real *b, __CLPK_real *w, __CLPK_real *scale, __CLPK_real *x,
3110     __CLPK_real *work,
3111     __CLPK_integer *info);
3112
3113 /* Subroutine */ int slarl1v(__CLPK_integer *n, __CLPK_integer *b1, __CLPK_integer *bn, __CLPK_real *
3114     sigma, __CLPK_real *d__,
3115     __CLPK_real *l, __CLPK_real *ld, __CLPK_real *lld, __CLPK_real *gersch,
3116     __CLPK_real *
3117     z__, __CLPK_real *ztz, __CLPK_real *mingma, __CLPK_integer *r__, __CLPK_integer *isuppz,
3118     __CLPK_real *
3119     work);
3120
3121 /* Subroutine */ int slar2v(__CLPK_integer *n, __CLPK_real *x, __CLPK_real *y, __CLPK_real *z__,
3122     __CLPK_integer
3123     *incx, __CLPK_real *c__, __CLPK_real *s, __CLPK_integer *incc);
3124
3125 /* Subroutine */ int slarf(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *v,
3126     __CLPK_integer *incv,
3127     __CLPK_real *tau, __CLPK_real *c__, __CLPK_integer *ldc, __CLPK_real *work);
3128
3129 /* Subroutine */ int slarfb(char *side, char *trans, char *direct, char *
3130     storev, __CLPK_integer *m,
3131     __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *v, __CLPK_integer
3132     *ldv,
3133     __CLPK_real *t, __CLPK_integer *ldt, __CLPK_real *c__, __CLPK_integer *ldc, __CLPK_real *work,
3134     __CLPK_integer *
3135     ldwork);
3136
3137 /* Subroutine */ int slarfg(__CLPK_integer *n, __CLPK_real *alpha, __CLPK_real *x, __CLPK_integer
3138     *incx,
3139     __CLPK_real *tau);
3140
3141 /* Subroutine */ int slarft(char *direct, char *storev, __CLPK_integer *n, __CLPK_integer *
```

```

3120     k, __CLPK_real *v, __CLPK_integer *ldv, __CLPK_real *tau, __CLPK_real *t, __CLPK_integer *ldt);
3121
3122 /* Subroutine */ int slarfz_(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *v,
3123     __CLPK_real *tau, __CLPK_real *c__, __CLPK_integer *ldc, __CLPK_real *work);
3124
3125 /* Subroutine */ int slargv_(__CLPK_integer *n, __CLPK_real *x, __CLPK_integer *incx, __CLPK_real *y,
3126     __CLPK_integer *incy, __CLPK_real *c__, __CLPK_integer *incc);
3127
3128 /* Subroutine */ int slarnv_(__CLPK_integer *idist, __CLPK_integer *iseed, __CLPK_integer *n,
3129     __CLPK_real
3130     *x);
3131 /* Subroutine */ int slarrb_(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *l, __CLPK_real *ld,
3132     __CLPK_real *
3133     lld, __CLPK_integer *ifirst, __CLPK_integer *ilast, __CLPK_real *sigma, __CLPK_real *reltol,
3134     __CLPK_real
3135     *w, __CLPK_real *wgap, __CLPK_real *werr, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
3136     *info);
3137
3138 /* Subroutine */ int slarre_(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e, __CLPK_real *tol,
3139     __CLPK_integer *nsplit, __CLPK_integer *isplit, __CLPK_integer *m, __CLPK_real *w, __CLPK_real
3140     *woff,
3141     __CLPK_real *gersch, __CLPK_real *work, __CLPK_integer *info);
3142
3143 /* Subroutine */ int slarrf_(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *l, __CLPK_real *ld,
3144     __CLPK_real *
3145     lld, __CLPK_integer *ifirst, __CLPK_integer *ilast, __CLPK_real *w, __CLPK_real *dplus, __CLPK_real
3146     *
3147     lplus, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
3148
3149 /* Subroutine */ int slarrv_(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *l, __CLPK_integer
3150     *isplit,
3151     __CLPK_integer *m, __CLPK_real *w, __CLPK_integer *iblock, __CLPK_real *gersch, __CLPK_real *tol,
3152     __CLPK_real *
3153     z__, __CLPK_integer *ldz, __CLPK_integer *isuppz, __CLPK_real *work, __CLPK_integer *iwork,
3154     __CLPK_integer *info);
3155
3156 /* Subroutine */ int slartg_(__CLPK_real *f, __CLPK_real *g, __CLPK_real *cs, __CLPK_real *sn,
3157     __CLPK_real *r__);
3158
3159 /* Subroutine */ int slartv_(__CLPK_integer *n, __CLPK_real *x, __CLPK_integer *incx, __CLPK_real *y,
3160     __CLPK_integer *incy, __CLPK_real *c__, __CLPK_real *s, __CLPK_integer *incc);
3161
3162 /* Subroutine */ int slaruv_(__CLPK_integer *iseed, __CLPK_integer *n, __CLPK_real *x);
3163
3164 /* Subroutine */ int slarz_(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *l,
3165     __CLPK_real *v, __CLPK_integer *incv, __CLPK_real *tau, __CLPK_real *c__, __CLPK_integer *ldc,
3166     __CLPK_real *
3167     work);
3168
3169 /* Subroutine */ int slarzb_(char *side, char *trans, char *direct, char *
3170     storev, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *l, __CLPK_real *v,
3171     __CLPK_integer *ldv, __CLPK_real *t, __CLPK_integer *ldt, __CLPK_real *c__, __CLPK_integer *ldc,
3172     __CLPK_real *
3173     work, __CLPK_integer *ldwork);
3174
3175 /* Subroutine */ int slarzt_(char *direct, char *storev, __CLPK_integer *n, __CLPK_integer *
3176     k, __CLPK_real *v, __CLPK_integer *ldv, __CLPK_real *tau, __CLPK_real *t, __CLPK_integer *ldt);
3177
3178 /* Subroutine */ int slas2_(__CLPK_real *f, __CLPK_real *g, __CLPK_real *h__, __CLPK_real *ssmin,
3179     __CLPK_real *
3180     ssmx);
3181
3182 /* Subroutine */ int slascl_(char *type__, __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_real *
3183     cfrom, __CLPK_real *cto, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3184     __CLPK_integer *info);
3185
3186 /* Subroutine */ int slasd0_(__CLPK_integer *n, __CLPK_integer *sqre, __CLPK_real *d__, __CLPK_real *e,
3187     __CLPK_real *u, __CLPK_integer *ldu, __CLPK_real *vt, __CLPK_integer *ldvt, __CLPK_integer *smlsiz,
3188     __CLPK_integer *iwork, __CLPK_real *work, __CLPK_integer *info);
3189
3190 /* Subroutine */ int slasd1_(__CLPK_integer *nl, __CLPK_integer *nr, __CLPK_integer *sqre, __CLPK_real
3191     *
3192     d__, __CLPK_real *alpha, __CLPK_real *beta, __CLPK_real *u, __CLPK_integer *ldu, __CLPK_real *vt,
3193     __CLPK_integer *ldvt, __CLPK_integer *idxq, __CLPK_integer *iwork, __CLPK_real *work,
3194     __CLPK_integer *
3195     info);
3196
3197 /* Subroutine */ int slasd2_(__CLPK_integer *nl, __CLPK_integer *nr, __CLPK_integer *sqre,
3198     __CLPK_integer
3199     *k, __CLPK_real *d__, __CLPK_real *z__, __CLPK_real *alpha, __CLPK_real *beta, __CLPK_real *u,
3200     __CLPK_integer *
3201     ldu, __CLPK_real *vt, __CLPK_integer *ldvt, __CLPK_real *dsigma, __CLPK_real *u2, __CLPK_integer

```

```

*ldu2,
3186   __CLPK_real *vt2, __CLPK_integer *ldvt2, __CLPK_integer *idxp, __CLPK_integer *idx, __CLPK_integer
*idxc,
3187   __CLPK_integer *idxq, __CLPK_integer *coltyp, __CLPK_integer *info);
3188
3189 /* Subroutine */ int slasd3(__CLPK_integer *nl, __CLPK_integer *nr, __CLPK_integer *sqre,
__CLPK_integer
3190 *k, __CLPK_real *d__, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *dsigma, __CLPK_real *u,
__CLPK_integer *
3191 ldu, __CLPK_real *u2, __CLPK_integer *ldu2, __CLPK_real *vt, __CLPK_integer *ldvt, __CLPK_real
*vt2,
3192 __CLPK_integer *ldvt2, __CLPK_integer *idxc, __CLPK_integer *ctot, __CLPK_real *z__, __CLPK_integer
*
3193 info);
3194
3195 /* Subroutine */ int slasd4(__CLPK_integer *n, __CLPK_integer *i__, __CLPK_real *d__, __CLPK_real
*z__,
3196 __CLPK_real *delta, __CLPK_real *rho, __CLPK_real *sigma, __CLPK_real *work, __CLPK_integer *info);
3197
3198 /* Subroutine */ int slasd5(__CLPK_integer *i__, __CLPK_real *d__, __CLPK_real *z__, __CLPK_real
*delta,
3199 __CLPK_real *rho, __CLPK_real *dsigma, __CLPK_real *work);
3200
3201 /* Subroutine */ int slasd6(__CLPK_integer *icompq, __CLPK_integer *nl, __CLPK_integer *nr,
__CLPK_integer *sqre,
3202 __CLPK_real *d__, __CLPK_real *vfv, __CLPK_real *vvl, __CLPK_real *alpha,
__CLPK_real *beta,
3203 __CLPK_integer *idxq, __CLPK_integer *perm, __CLPK_integer *givptr, __CLPK_integer *givcol,
__CLPK_integer *ldgcol,
3204 __CLPK_real *givnum, __CLPK_integer *ldgnum, __CLPK_real *poles,
__CLPK_real *
3205 difl, __CLPK_real *difr, __CLPK_real *z__, __CLPK_integer *k, __CLPK_real *c__, __CLPK_real *s,
__CLPK_real *
3206 work, __CLPK_integer *iwork, __CLPK_integer *info);
3207
3208 /* Subroutine */ int slasd7(__CLPK_integer *icompq, __CLPK_integer *nl, __CLPK_integer *nr,
__CLPK_integer *sqre,
3209 __CLPK_integer *k, __CLPK_real *d__, __CLPK_real *z__, __CLPK_real *zw,
__CLPK_real *vfv,
3210 __CLPK_real *vfw, __CLPK_real *vvl, __CLPK_real *vwl, __CLPK_real *alpha, __CLPK_real *beta,
__CLPK_real *dsigma,
3211 __CLPK_integer *idx, __CLPK_integer *idxp, __CLPK_integer *idxq, __CLPK_integer *perm,
__CLPK_integer *
3212 givptr, __CLPK_integer *givcol, __CLPK_integer *ldgcol, __CLPK_real *givnum, __CLPK_integer *
3213 ldgnum, __CLPK_real *c__, __CLPK_real *s, __CLPK_integer *info);
3214
3215 /* Subroutine */ int slasd8(__CLPK_integer *icompq, __CLPK_integer *k, __CLPK_real *d__, __CLPK_real *
3216 z__, __CLPK_real *vfv, __CLPK_real *vvl, __CLPK_real *difl, __CLPK_real *difr, __CLPK_integer
*lddifr,
3217 __CLPK_real *dsigma, __CLPK_real *work, __CLPK_integer *info);
3218
3219 /* Subroutine */ int slasd9(__CLPK_integer *icompq, __CLPK_integer *ldu, __CLPK_integer *k,
__CLPK_real *
3220 d__, __CLPK_real *z__, __CLPK_real *vfv, __CLPK_real *vvl, __CLPK_real *difl, __CLPK_real *difr,
__CLPK_real *
3221 dsigma, __CLPK_real *work, __CLPK_integer *info);
3222
3223 /* Subroutine */ int slasda(__CLPK_integer *icompq, __CLPK_integer *smlsiz, __CLPK_integer *n,
__CLPK_integer *sqre,
3224 __CLPK_real *d__, __CLPK_real *e, __CLPK_real *u, __CLPK_integer *ldu,
__CLPK_real *vt,
3225 __CLPK_integer *k, __CLPK_real *difl, __CLPK_real *difr, __CLPK_real *z__, __CLPK_real *poles,
__CLPK_integer *
3226 givptr, __CLPK_integer *givcol, __CLPK_integer *ldgcol, __CLPK_integer *perm, __CLPK_real *givnum,
__CLPK_integer *
3227 ldgnum, __CLPK_real *c__, __CLPK_real *s, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
3228
3229 /* Subroutine */ int slasdq(char *uplo, __CLPK_integer *sqre, __CLPK_integer *n, __CLPK_integer *
3230 ncv, __CLPK_integer *nru, __CLPK_integer *ncc, __CLPK_real *d__, __CLPK_real *e, __CLPK_real *vt,
__CLPK_integer *ldvt,
3231 __CLPK_integer *ldu, __CLPK_real *u, __CLPK_integer *ldu, __CLPK_real *c__, __CLPK_integer *ldc,
__CLPK_real *
3232 work, __CLPK_integer *info);
3233
3234 /* Subroutine */ int slasdt(__CLPK_integer *n, __CLPK_integer *lvl, __CLPK_integer *nd, __CLPK_integer
*
3235 inode, __CLPK_integer *ndiml, __CLPK_integer *ndimr, __CLPK_integer *msub);
3236
3237 /* Subroutine */ int slaset(char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *alpha,
__CLPK_real *beta,
3238 __CLPK_real *a, __CLPK_integer *lda);
3239
3240 /* Subroutine */ int slasql(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e, __CLPK_real *work,
__CLPK_integer *info);
3241
3242
3243 /* Subroutine */ int slasq2(__CLPK_integer *n, __CLPK_real *z__, __CLPK_integer *info);
3244
3245 /* Subroutine */ int slasq3(__CLPK_integer *i0, __CLPK_integer *n0, __CLPK_real *z__, __CLPK_integer
*pp,
3246 __CLPK_real *dmin__, __CLPK_real *sigma, __CLPK_real *desig, __CLPK_real *qmax, __CLPK_integer
*nfail,
3247 __CLPK_integer *iter, __CLPK_integer *ndiv, __CLPK_logical *ieee);
3248
3249 /* Subroutine */ int slasq4(__CLPK_integer *i0, __CLPK_integer *n0, __CLPK_real *z__, __CLPK_integer

```

```

*pp,
3250  __CLPK_integer *n0in, __CLPK_real *dmin__, __CLPK_real *dmin1, __CLPK_real *dmin2, __CLPK_real
*dn,
3251  __CLPK_real *dn1, __CLPK_real *dn2, __CLPK_real *tau, __CLPK_integer *ttype);
3252
3253 /* Subroutine */ int slasq5(__CLPK_integer *i0, __CLPK_integer *n0, __CLPK_real *z__, __CLPK_integer
*pp,
3254  __CLPK_real *tau, __CLPK_real *dmin__, __CLPK_real *dmin1, __CLPK_real *dmin2, __CLPK_real *dn,
__CLPK_real *
3255  dnm1, __CLPK_real *dnm2, __CLPK_logical *ieee);
3256
3257 /* Subroutine */ int slasq6(__CLPK_integer *i0, __CLPK_integer *n0, __CLPK_real *z__, __CLPK_integer
*pp,
3258  __CLPK_real *dmin__, __CLPK_real *dmin1, __CLPK_real *dmin2, __CLPK_real *dn, __CLPK_real *dnm1,
__CLPK_real *
3259  dnm2);
3260
3261 /* Subroutine */ int slasr(char *side, char *pivot, char *direct, __CLPK_integer *m,
3262  __CLPK_integer *n, __CLPK_real *c__, __CLPK_real *s, __CLPK_real *a, __CLPK_integer *lda);
3263
3264 /* Subroutine */ int slasrt(char *id, __CLPK_integer *n, __CLPK_real *d__, __CLPK_integer *info);
3265
3266 /* Subroutine */ int slassq(__CLPK_integer *n, __CLPK_real *x, __CLPK_integer *incx, __CLPK_real
*scale,
3267  __CLPK_real *sumsq);
3268
3269 /* Subroutine */ int slasv2(__CLPK_real *f, __CLPK_real *g, __CLPK_real *h__, __CLPK_real *ssmin,
__CLPK_real *
3270  ssmx, __CLPK_real *snr, __CLPK_real *csr, __CLPK_real *snl, __CLPK_real *csl);
3271
3272 /* Subroutine */ int slaswp(__CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_integer
*k1,
3273  __CLPK_integer *k2, __CLPK_integer *ipiv, __CLPK_integer *incx);
3274
3275 /* Subroutine */ int slasy2(__CLPK_logical *ltrn1, __CLPK_logical *ltrn2, __CLPK_integer *isgn,
3276  __CLPK_integer *n1, __CLPK_integer *n2, __CLPK_real *t1, __CLPK_integer *ldt1, __CLPK_real *tr,
__CLPK_integer *
3277  ldr, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *scale, __CLPK_real *x, __CLPK_integer *ldx,
__CLPK_real
3278  *xnrm, __CLPK_integer *info);
3279
3280 /* Subroutine */ int slasyf(char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_integer *kb,
3281  __CLPK_real *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_real *w, __CLPK_integer *ldw,
__CLPK_integer
3282  *info);
3283
3284 /* Subroutine */ int slatbs(char *uplo, char *trans, char *diag, char *
3285  normin, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real
*x,
3286  __CLPK_real *scale, __CLPK_real *cnorm, __CLPK_integer *info);
3287
3288 /* Subroutine */ int slatdf(__CLPK_integer *ijob, __CLPK_integer *n, __CLPK_real *z__, __CLPK_integer
*
3289  ldz, __CLPK_real *rhs, __CLPK_real *rdsum, __CLPK_real *rdscal, __CLPK_integer *ipiv,
__CLPK_integer *
3290  jpiv);
3291
3292 /* Subroutine */ int slatps(char *uplo, char *trans, char *diag, char *
3293  normin, __CLPK_integer *n, __CLPK_real *ap, __CLPK_real *x, __CLPK_real *scale, __CLPK_real *cnorm,
__CLPK_integer
3294  *info);
3295
3296 /* Subroutine */ int slatrd(char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_real *a,
3297  __CLPK_integer *lda, __CLPK_real *e, __CLPK_real *tau, __CLPK_real *w, __CLPK_integer *ldw);
3298
3299 /* Subroutine */ int slatrs(char *uplo, char *trans, char *diag, char *
3300  normin, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *x, __CLPK_real *scale,
__CLPK_real
3301  *cnorm, __CLPK_integer *info);
3302
3303 /* Subroutine */ int slatrz(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *l, __CLPK_real *a,
3304  __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work);
3305
3306 /* Subroutine */ int slatzm(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *v,
3307  __CLPK_integer *incv, __CLPK_real *tau, __CLPK_real *c1, __CLPK_real *c2, __CLPK_integer *ldc,
__CLPK_real *
3308  work);
3309
3310 /* Subroutine */ int slauu2(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3311  __CLPK_integer *info);
3312
3313 /* Subroutine */ int slauum(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3314  __CLPK_integer *info);
3315
3316 /* Subroutine */ int sopgtr(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_real *tau,
3317  __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *work, __CLPK_integer *info);
3318

```

```
3319 /* Subroutine */ int somptr(char *side, char *uplo, char *trans, __CLPK_integer *m,
3320   __CLPK_integer *n, __CLPK_real *ap, __CLPK_real *tau, __CLPK_real *c__, __CLPK_integer *ldc,
   __CLPK_real *work,
3321   __CLPK_integer *info);
3322
3323 /* Subroutine */ int sorg2l(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a,
3324   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
3325
3326 /* Subroutine */ int sorg2r(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a,
3327   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
3328
3329 /* Subroutine */ int sorgbr(char *vect, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
3330   __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork,
   __CLPK_integer
3331   *info);
3332
3333 /* Subroutine */ int sorgbr(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real
   *a,
3334   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *info);
3335
3336 /* Subroutine */ int sorgl2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a,
3337   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
3338
3339 /* Subroutine */ int sorglq(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a,
3340   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *info);
3341
3342 /* Subroutine */ int sorgql(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a,
3343   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *info);
3344
3345 /* Subroutine */ int sorgqr(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a,
3346   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *info);
3347
3348 /* Subroutine */ int sorgr2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a,
3349   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *info);
3350
3351 /* Subroutine */ int sorgrq(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a,
3352   __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *info);
3353
3354 /* Subroutine */ int sorgtr(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3355   __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3356
3357 /* Subroutine */ int sorm2l(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3358   __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
   __CLPK_integer *ldc,
3359   __CLPK_real *work, __CLPK_integer *info);
3360
3361 /* Subroutine */ int sorm2r(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3362   __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
   __CLPK_integer *ldc,
3363   __CLPK_real *work, __CLPK_integer *info);
3364
3365 /* Subroutine */ int sormbr(char *vect, char *side, char *trans, __CLPK_integer *m,
3366   __CLPK_integer *n, __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau,
   __CLPK_real *c__,
3367   __CLPK_integer *ldc, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3368
3369 /* Subroutine */ int sormhr(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3370   __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau,
   __CLPK_real *
3371   c__, __CLPK_integer *ldc, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3372
3373 /* Subroutine */ int sorml2(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3374   __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
   __CLPK_integer *ldc,
3375   __CLPK_real *work, __CLPK_integer *info);
3376
3377 /* Subroutine */ int sormlq(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3378   __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
   __CLPK_integer *ldc,
3379   __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3380
3381 /* Subroutine */ int sormql(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3382   __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
   __CLPK_integer *ldc,
3383   __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3384
3385 /* Subroutine */ int sormqr(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3386   __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
   __CLPK_integer *ldc,
3387   __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3388
3389 /* Subroutine */ int sormr2(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
```

```

3390   __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
      __CLPK_integer *ldc,
3391   __CLPK_real *work, __CLPK_integer *info);
3392
3393 /* Subroutine */ int sormr3_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3394   __CLPK_integer *k, __CLPK_integer *l, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau,
      __CLPK_real *c__,
3395   __CLPK_integer *ldc, __CLPK_real *work, __CLPK_integer *info);
3396
3397 /* Subroutine */ int sormrq_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3398   __CLPK_integer *k, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
      __CLPK_integer *ldc,
3399   __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3400
3401 /* Subroutine */ int sormrz_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
3402   __CLPK_integer *k, __CLPK_integer *l, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau,
      __CLPK_real *c__,
3403   __CLPK_integer *ldc, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3404
3405 /* Subroutine */ int sormtr_(char *side, char *uplo, char *trans, __CLPK_integer *m,
3406   __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *tau, __CLPK_real *c__,
      __CLPK_integer *ldc,
3407   __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3408
3409 /* Subroutine */ int spbcon_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_real *ab,
3410   __CLPK_integer *ldab, __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer
      *iwork,
3411   __CLPK_integer *info);
3412
3413 /* Subroutine */ int spequ_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_real *ab,
3414   __CLPK_integer *ldab, __CLPK_real *s, __CLPK_real *scond, __CLPK_real *amax, __CLPK_integer *info);
3415
3416 /* Subroutine */ int sprbfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
3417   nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *afb, __CLPK_integer *ldafb, __CLPK_real
      *b,
3418   __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr,
      __CLPK_real *
3419   work, __CLPK_integer *iwork, __CLPK_integer *info);
3420
3421 /* Subroutine */ int spbstf_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_real *ab,
3422   __CLPK_integer *ldab, __CLPK_integer *info);
3423
3424 /* Subroutine */ int spbsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
3425   nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer
      *info);
3426
3427 /* Subroutine */ int spbsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
3428   __CLPK_integer *nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *afb, __CLPK_integer
      *ldafb,
3429   char *equed, __CLPK_real *s, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer
      *ldx,
3430   __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *iwork,
      __CLPK_integer *info);
3431
3432
3433 /* Subroutine */ int spbtf2_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_real *ab,
3434   __CLPK_integer *ldab, __CLPK_integer *info);
3435
3436 /* Subroutine */ int spbtrf_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_real *ab,
3437   __CLPK_integer *ldab, __CLPK_integer *info);
3438
3439 /* Subroutine */ int spbtrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
3440   nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer
      *info);
3441
3442 /* Subroutine */ int spocon_(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3443   __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
      *info);
3444
3445 /* Subroutine */ int spoqu_(__CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *s,
      __CLPK_real
3446   *scond, __CLPK_real *amax, __CLPK_integer *info);
3447
3448 /* Subroutine */ int sporfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a,
3449   __CLPK_integer *lda, __CLPK_real *af, __CLPK_integer *ldaf, __CLPK_real *b, __CLPK_integer *ldb,
      __CLPK_real *x,
3450   __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer
      *iwork,
3451   __CLPK_integer *info);
3452
3453 /* Subroutine */ int sposv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a,
3454   __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3455
3456 /* Subroutine */ int sposvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
3457   nrhs, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *af, __CLPK_integer *ldaf, char *equed,
      __CLPK_real *s,
3458   __CLPK_real *s, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx,
      __CLPK_real *rcond,

```



```
3459     __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
3460     *info);
3461 /* Subroutine */ int spotf2(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3462     __CLPK_integer *info);
3463
3464 /* Subroutine */ int spotrf(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3465     __CLPK_integer *info);
3466
3467 /* Subroutine */ int spotri(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3468     __CLPK_integer *info);
3469
3470 /* Subroutine */ int spotrs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a,
3471     __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3472
3473 /* Subroutine */ int sppcon(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_real *anorm,
3474     __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
3475
3476 /* Subroutine */ int spequ(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_real *s,
3477     __CLPK_real *
3478     scond, __CLPK_real *amax, __CLPK_integer *info);
3479
3480 /* Subroutine */ int sprfs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *ap,
3481     __CLPK_real *afp, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx,
3482     __CLPK_real *ferr,
3483     __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
3484
3485 /* Subroutine */ int sppsv(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *ap,
3486     __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3487
3488 /* Subroutine */ int sppsvx(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
3489     nrhs, __CLPK_real *ap, __CLPK_real *afp, char *equad, __CLPK_real *s, __CLPK_real *b,
3490     __CLPK_integer *
3491     ldb, __CLPK_real *x, __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr,
3492     __CLPK_real
3493     *work, __CLPK_integer *iwork, __CLPK_integer *info);
3494
3495 /* Subroutine */ int sptrf(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_integer *info);
3496
3497 /* Subroutine */ int sptri(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_integer *info);
3498
3499 /* Subroutine */ int sptrs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *ap,
3500     __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3501
3502 /* Subroutine */ int sptcon(__CLPK_integer *n, __CLPK_real *d, __CLPK_real *e, __CLPK_real *anorm,
3503     __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer *info);
3504
3505 /* Subroutine */ int sptqr(char *compz, __CLPK_integer *n, __CLPK_real *d, __CLPK_real *e,
3506     __CLPK_real *z, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *info);
3507
3508 /* Subroutine */ int sprfs(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *d, __CLPK_real *e,
3509     __CLPK_real *df, __CLPK_real *ef, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x,
3510     __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work,
3511     __CLPK_integer *info);
3512
3513 /* Subroutine */ int sptrf(__CLPK_integer *n, __CLPK_real *d, __CLPK_real *e, __CLPK_integer *info);
3514
3515 /* Subroutine */ int sptrs(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *d, __CLPK_real *e,
3516     __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3517
3518 /* Subroutine */ int sptts2(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *d, __CLPK_real *e,
3519     __CLPK_real *b, __CLPK_integer *ldb);
3520
3521 /* Subroutine */ int srscl(__CLPK_integer *n, __CLPK_real *sa, __CLPK_real *sx, __CLPK_integer *incx);
3522
3523 /* Subroutine */ int ssbev(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
3524     __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *w, __CLPK_real *z, __CLPK_integer *ldz,
3525     __CLPK_real *work,
3526     __CLPK_integer *info);
3527
3528 /* Subroutine */ int ssbevd(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
3529     __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *w, __CLPK_real *z, __CLPK_integer *ldz,
3530     __CLPK_real *work,
3531     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
```

```

3532
3533 /* Subroutine */ int ssbev_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
3534   __CLPK_integer *kd, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *q, __CLPK_integer *ldq,
   __CLPK_real *vl,
3535   __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu, __CLPK_real *abstol, __CLPK_integer *m,
   __CLPK_real *
3536   w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
   *
3537   ifail, __CLPK_integer *info);
3538
3539 /* Subroutine */ int ssbgst_(char *vect, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
3540   __CLPK_integer *kb, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *bb, __CLPK_integer *ldb,
   __CLPK_real *
3541   x, __CLPK_integer *ldx, __CLPK_real *work, __CLPK_integer *info);
3542
3543 /* Subroutine */ int ssbgv_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
3544   __CLPK_integer *kb, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *bb, __CLPK_integer *ldb,
   __CLPK_real *
3545   w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *info);
3546
3547 /* Subroutine */ int ssbgvd_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
3548   __CLPK_integer *kb, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *bb, __CLPK_integer *ldb,
   __CLPK_real *
3549   w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *
3550   iwork, __CLPK_integer *liwork, __CLPK_integer *info);
3551
3552 /* Subroutine */ int ssbgvx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
3553   __CLPK_integer *ka, __CLPK_integer *kb, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *bb,
   __CLPK_integer *
3554   ldb, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il,
   __CLPK_integer
3555   *iu, __CLPK_real *abstol, __CLPK_integer *m, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz,
   __CLPK_real
3556   *work, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
3557
3558 /* Subroutine */ int ssbtrd_(char *vect, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
3559   __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *d__, __CLPK_real *e, __CLPK_real *q,
   __CLPK_integer *ldq,
3560   __CLPK_real *work, __CLPK_integer *info);
3561
3562 /* Subroutine */ int sscon_(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_integer *ipiv,
3563   __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
   *info);
3564
3565 /* Subroutine */ int sspev_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_real *ap,
3566   __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *info);
3567
3568 /* Subroutine */ int sspevd_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_real *ap,
3569   __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *lwork,
   __CLPK_integer
3570   *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
3571
3572 /* Subroutine */ int sspevx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
3573   __CLPK_real *ap, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu,
   __CLPK_real *abstol,
3574   __CLPK_integer *m, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work,
   __CLPK_integer *
3575   iwork, __CLPK_integer *ifail, __CLPK_integer *info);
3576
3577 /* Subroutine */ int sspgst_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n, __CLPK_real *ap,
3578   __CLPK_real *bp, __CLPK_integer *info);
3579
3580 /* Subroutine */ int sspgv_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
3581   n, __CLPK_real *ap, __CLPK_real *bp, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz,
   __CLPK_real *work,
3582   __CLPK_integer *info);
3583
3584 /* Subroutine */ int sspgvd_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
3585   n, __CLPK_real *ap, __CLPK_real *bp, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz,
   __CLPK_real *work,
3586   __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
3587
3588 /* Subroutine */ int sspgvx_(__CLPK_integer *itype, char *jobz, char *range, char *
3589   uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_real *bp, __CLPK_real *vl, __CLPK_real *vu,
   __CLPK_integer *il,
3590   __CLPK_integer *iu, __CLPK_real *abstol, __CLPK_integer *m, __CLPK_real *w, __CLPK_real *z__,
   __CLPK_integer *
3591   ldz, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
3592
3593 /* Subroutine */ int ssprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *ap,
3594   __CLPK_real *afp, __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x,
   __CLPK_integer *
3595   ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
   *
3596   info);
3597

```

```
3598 /* Subroutine */ int sspsv(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *ap,
3599   __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3600
3601 /* Subroutine */ int sspsvx(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
3602   nrhs, __CLPK_real *ap, __CLPK_real *afp, __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb,
   __CLPK_real
3603   *x, __CLPK_integer *ldx, __CLPK_real *rcond, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real
   *work,
3604   __CLPK_integer *iwork, __CLPK_integer *info);
3605
3606 /* Subroutine */ int ssptrd(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_real *d__,
3607   __CLPK_real *e, __CLPK_real *tau, __CLPK_integer *info);
3608
3609 /* Subroutine */ int ssptrf(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_integer *ipiv,
3610   __CLPK_integer *info);
3611
3612 /* Subroutine */ int ssptri(char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_integer *ipiv,
3613   __CLPK_real *work, __CLPK_integer *info);
3614
3615 /* Subroutine */ int ssptrs(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *ap,
3616   __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3617
3618 /* Subroutine */ int sstebz(char *range, char *order, __CLPK_integer *n, __CLPK_real *vl,
3619   __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu, __CLPK_real *abstol, __CLPK_real *d__,
   __CLPK_real *e,
3620   __CLPK_integer *m, __CLPK_integer *nsplit, __CLPK_real *w, __CLPK_integer *iblock, __CLPK_integer *
   isplit, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer *info);
3622
3623 /* Subroutine */ int sstedc(char *compz, __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e,
3624   __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *iwork,
3625   __CLPK_integer *liwork, __CLPK_integer *info);
3626
3627 /* Subroutine */ int sstegr(char *jobz, char *range, __CLPK_integer *n, __CLPK_real *d__,
3628   __CLPK_real *e, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu,
   __CLPK_real *abstol,
3629   __CLPK_integer *m, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_integer *isuppz,
   __CLPK_real *
3630   work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
3631
3632 /* Subroutine */ int sstein(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e, __CLPK_integer *m,
   __CLPK_real
3633   *w, __CLPK_integer *iblock, __CLPK_integer *isplit, __CLPK_real *z__, __CLPK_integer *ldz,
   __CLPK_real *
3634   work, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
3635
3636 /* Subroutine */ int ssteqr(char *compz, __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e,
3637   __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *info);
3638
3639 /* Subroutine */ int ssterf(__CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e, __CLPK_integer
   *info);
3640
3641 /* Subroutine */ int sstev(char *jobz, __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e,
   __CLPK_real *
3642   z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *info);
3643
3644 /* Subroutine */ int sstevd(char *jobz, __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e,
   __CLPK_real
3645   *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork,
   __CLPK_integer *liwork,
3646   __CLPK_integer *info);
3647
3648 /* Subroutine */ int sstevr(char *jobz, char *range, __CLPK_integer *n, __CLPK_real *d__,
3649   __CLPK_real *e, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu,
   __CLPK_real *abstol,
3650   __CLPK_integer *m, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_integer *isuppz,
   __CLPK_real *
3651   work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
3652
3653 /* Subroutine */ int sstevx(char *jobz, char *range, __CLPK_integer *n, __CLPK_real *d__,
3654   __CLPK_real *e, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu,
   __CLPK_real *abstol,
3655   __CLPK_integer *m, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work,
   __CLPK_integer *
3656   iwork, __CLPK_integer *ifail, __CLPK_integer *info);
3657
3658 /* Subroutine */ int ssycon(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3659   __CLPK_integer *ipiv, __CLPK_real *anorm, __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer
   *iwork,
3660   __CLPK_integer *info);
3661
3662 /* Subroutine */ int ssyev(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_real *a,
3663   __CLPK_integer *lda, __CLPK_real *w, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *info);
3664
3665 /* Subroutine */ int ssyevd(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_real *a,
3666   __CLPK_integer *lda, __CLPK_real *w, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer
   *iwork,
```

```

3667     __CLPK_integer *liwork, __CLPK_integer *info);
3668
3669 /* Subroutine */ int sseyvr_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
3670   __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il,
   __CLPK_integer *iu,
3671   __CLPK_real *abstol, __CLPK_integer *m, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz,
   __CLPK_integer *
3672   isuppz, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork,
3673   __CLPK_integer *info);
3674
3675 /* Subroutine */ int sseyvx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
3676   __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il,
   __CLPK_integer *iu,
3677   __CLPK_real *abstol, __CLPK_integer *m, __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz,
   __CLPK_real *
3678   work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
3679
3680 /* Subroutine */ int ssygs2_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n, __CLPK_real *a,
3681   __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3682
3683 /* Subroutine */ int ssygst_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n, __CLPK_real *a,
3684   __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3685
3686 /* Subroutine */ int ssygv_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
3687   n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *w,
   __CLPK_real *work,
3688   __CLPK_integer *lwork, __CLPK_integer *info);
3689
3690 /* Subroutine */ int ssygvd_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
3691   n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *w,
   __CLPK_real *work,
3692   __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
3693
3694 /* Subroutine */ int ssygvx_(__CLPK_integer *itype, char *jobz, char *range, char *
3695   uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
   __CLPK_real *
3696   vl, __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu, __CLPK_real *abstol, __CLPK_integer
   *m,
3697   __CLPK_real *w, __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *lwork,
   __CLPK_integer
3698   *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
3699
3700 /* Subroutine */ int ssyrfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a,
3701   __CLPK_integer *lda, __CLPK_real *af, __CLPK_integer *ldaf, __CLPK_integer *ipiv, __CLPK_real *b,
3702   __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr,
   __CLPK_real *
3703   work, __CLPK_integer *iwork, __CLPK_integer *info);
3704
3705 /* Subroutine */ int ssysv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a,
3706   __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *work,
3707   __CLPK_integer *lwork, __CLPK_integer *info);
3708
3709 /* Subroutine */ int ssysvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
3710   nrhs, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *af, __CLPK_integer *ldaf, __CLPK_integer
   *ipiv,
3711   __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x, __CLPK_integer *ldx, __CLPK_real *rcond,
   __CLPK_real *ferr,
3712   __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer
   *
3713   info);
3714
3715 /* Subroutine */ int ssytd2_(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3716   __CLPK_real *d__, __CLPK_real *e, __CLPK_real *tau, __CLPK_integer *info);
3717
3718 /* Subroutine */ int ssytf2_(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3719   __CLPK_integer *ipiv, __CLPK_integer *info);
3720
3721 /* Subroutine */ int ssytrd_(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3722   __CLPK_real *d__, __CLPK_real *e, __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork,
   __CLPK_integer *
3723   info);
3724
3725 /* Subroutine */ int ssytrf_(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3726   __CLPK_integer *ipiv, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3727
3728 /* Subroutine */ int ssytri_(char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3729   __CLPK_integer *ipiv, __CLPK_real *work, __CLPK_integer *info);
3730
3731 /* Subroutine */ int ssytrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_real *a,
3732   __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer
   *info);
3733
3734 /* Subroutine */ int stbcon_(char *norm, char *uplo, char *diag, __CLPK_integer *n,
3735   __CLPK_integer *kd, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *rcond, __CLPK_real *work,
3736   __CLPK_integer *iwork, __CLPK_integer *info);
3737
3738 /* Subroutine */ int stbrfs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,

```

```

3739   __CLPK_integer *kd, __CLPK_integer *nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *b,
3740   __CLPK_integer
3741   *ldb, __CLPK_real *x, __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work,
3742
3743   __CLPK_integer *iwork, __CLPK_integer *info);
3744
3745 /* Subroutine */ int stbtrs(char *uplo, char *trans, char *diag, __CLPK_integer *n,
3746   __CLPK_integer *kd, __CLPK_integer *nrhs, __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *b,
3747   __CLPK_integer
3748   *ldb, __CLPK_integer *info);
3749
3750 /* Subroutine */ int stgevc(char *side, char *howmny, __CLPK_logical *select,
3751   __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
3752   __CLPK_real *vl,
3753   __CLPK_integer *ldvl, __CLPK_real *vr, __CLPK_integer *ldvr, __CLPK_integer *mm, __CLPK_integer *m,
3754   __CLPK_real
3755   *work, __CLPK_integer *info);
3756
3757 /* Subroutine */ int stgex2(__CLPK_logical *wantq, __CLPK_logical *wantz, __CLPK_integer *n,
3758   __CLPK_real
3759   *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *q, __CLPK_integer *ldq,
3760   __CLPK_real *
3761   z__, __CLPK_integer *ldz, __CLPK_integer *jl, __CLPK_integer *nl, __CLPK_integer *n2, __CLPK_real
3762   *work,
3763   __CLPK_integer *lwork, __CLPK_integer *info);
3764
3765 /* Subroutine */ int stgexc(__CLPK_logical *wantq, __CLPK_logical *wantz, __CLPK_integer *n,
3766   __CLPK_real
3767   *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *q, __CLPK_integer *ldq,
3768   __CLPK_real *
3769   z__, __CLPK_integer *ldz, __CLPK_integer *ifst, __CLPK_integer *ilst, __CLPK_real *work,
3770   __CLPK_integer *
3771   lwork, __CLPK_integer *info);
3772
3773 /* Subroutine */ int stgsen(__CLPK_integer *ijob, __CLPK_logical *wantq, __CLPK_logical *wantz,
3774   __CLPK_logical *select, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b,
3775   __CLPK_integer *
3776   ldb, __CLPK_real *alpha, __CLPK_real *alphai, __CLPK_real *beta, __CLPK_real *q, __CLPK_integer
3777   *ldq,
3778   __CLPK_real *z__, __CLPK_integer *ldz, __CLPK_integer *m, __CLPK_real *pl, __CLPK_real *pr,
3779   __CLPK_real *dif,
3780   __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork,
3781   __CLPK_integer *
3782   info);
3783
3784 /* Subroutine */ int stgsja(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
3785   __CLPK_integer *p, __CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *l, __CLPK_real *a,
3786   __CLPK_integer *lda,
3787   __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *tola, __CLPK_real *tolb, __CLPK_real *alpha,
3788   __CLPK_real *
3789   beta, __CLPK_real *u, __CLPK_integer *ldu, __CLPK_real *v, __CLPK_integer *ldv, __CLPK_real *q,
3790   __CLPK_integer *
3791   ldq, __CLPK_real *work, __CLPK_integer *ncycle, __CLPK_integer *info);
3792
3793 /* Subroutine */ int stgsna(char *job, char *howmny, __CLPK_logical *select,
3794   __CLPK_integer *p, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
3795   __CLPK_real *vl,
3796   __CLPK_integer *ldvl, __CLPK_real *vr, __CLPK_integer *ldvr, __CLPK_real *s, __CLPK_real *dif,
3797   __CLPK_integer *
3798   mm, __CLPK_integer *m, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork,
3799   __CLPK_integer *
3800   info);
3801
3802 /* Subroutine */ int stgsy2(char *trans, __CLPK_integer *ijob, __CLPK_integer *m, __CLPK_integer *
3803   n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *c__,
3804   __CLPK_integer *
3805   ldc, __CLPK_real *d__, __CLPK_integer *ldd, __CLPK_real *e, __CLPK_integer *lde, __CLPK_real *f,
3806   __CLPK_integer
3807   *ldf, __CLPK_real *scale, __CLPK_real *rdsum, __CLPK_real *rdscal, __CLPK_integer *iwork,
3808   __CLPK_integer
3809   *pq, __CLPK_integer *info);
3810
3811 /* Subroutine */ int stgsyl(char *trans, __CLPK_integer *ijob, __CLPK_integer *m, __CLPK_integer *
3812   n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *c__,
3813   __CLPK_integer *
3814   ldc, __CLPK_real *d__, __CLPK_integer *ldd, __CLPK_real *e, __CLPK_integer *lde, __CLPK_real *f,
3815   __CLPK_integer
3816   *ldf, __CLPK_real *scale, __CLPK_real *dif, __CLPK_real *work, __CLPK_integer *lwork,
3817   __CLPK_integer *
3818   iwork, __CLPK_integer *info);
3819
3820 /* Subroutine */ int stpcon(char *norm, char *uplo, char *diag, __CLPK_integer *n,
3821   __CLPK_real *ap, __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
3822   *info);
3823
3824 /* Subroutine */ int stprfs(char *uplo, char *trans, char *diag, __CLPK_integer *n,
3825   __CLPK_integer *nrhs, __CLPK_real *ap, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_real *x,

```

```

    __CLPK_integer *ldx,
3798    __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer *iwork, __CLPK_integer
    *info);
3799
3800 /* Subroutine */ int stptri_(char *uplo, char *diag, __CLPK_integer *n, __CLPK_real *ap,
3801    __CLPK_integer *info);
3802
3803 /* Subroutine */ int stptrs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
3804    __CLPK_integer *nrhs, __CLPK_real *ap, __CLPK_real *b, __CLPK_integer *ldb, __CLPK_integer *info);
3805
3806 /* Subroutine */ int strcon_(char *norm, char *uplo, char *diag, __CLPK_integer *n,
3807    __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *rcond, __CLPK_real *work, __CLPK_integer *iwork,
3808    __CLPK_integer *info);
3809
3810 /* Subroutine */ int strevc_(char *side, char *howmny, __CLPK_logical *select,
3811    __CLPK_integer *n, __CLPK_real *t, __CLPK_integer *ldt, __CLPK_real *vl, __CLPK_integer *ldvl,
    __CLPK_real *vr,
3812    __CLPK_integer *ldvr, __CLPK_integer *mm, __CLPK_integer *m, __CLPK_real *work, __CLPK_integer
    *info);
3813
3814 /* Subroutine */ int strexc_(char *compq, __CLPK_integer *n, __CLPK_real *t, __CLPK_integer *ldt,
3815    __CLPK_real *q, __CLPK_integer *ldq, __CLPK_integer *ifst, __CLPK_integer *ilst, __CLPK_real *work,
    __CLPK_integer *info);
3816
3817
3818 /* Subroutine */ int strfs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
3819    __CLPK_integer *nrhs, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
    __CLPK_real *x,
3820    __CLPK_integer *ldx, __CLPK_real *ferr, __CLPK_real *berr, __CLPK_real *work, __CLPK_integer
    *iwork,
3821    __CLPK_integer *info);
3822
3823 /* Subroutine */ int strsen_(char *job, char *compq, __CLPK_logical *select, __CLPK_integer
3824    *n, __CLPK_real *t, __CLPK_integer *ldt, __CLPK_real *q, __CLPK_integer *ldq, __CLPK_real *wr,
    __CLPK_real *wi,
3825    __CLPK_integer *m, __CLPK_real *s, __CLPK_real *sep, __CLPK_real *work, __CLPK_integer *lwork,
    __CLPK_integer *
3826    iwork, __CLPK_integer *liwork, __CLPK_integer *info);
3827
3828 /* Subroutine */ int strzna_(char *job, char *howmny, __CLPK_logical *select,
3829    __CLPK_integer *n, __CLPK_real *t, __CLPK_integer *ldt, __CLPK_real *vl, __CLPK_integer *ldvl,
    __CLPK_real *vr,
3830    __CLPK_integer *ldvr, __CLPK_real *s, __CLPK_real *sep, __CLPK_integer *mm, __CLPK_integer *m,
    __CLPK_real *
3831    work, __CLPK_integer *ldwork, __CLPK_integer *iwork, __CLPK_integer *info);
3832
3833 /* Subroutine */ int strsyl_(char *trana, char *tranb, __CLPK_integer *isgn, __CLPK_integer
3834    *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
    __CLPK_real *
3835    c__, __CLPK_integer *ldc, __CLPK_real *scale, __CLPK_integer *info);
3836
3837 /* Subroutine */ int strti2_(char *uplo, char *diag, __CLPK_integer *n, __CLPK_real *a,
3838    __CLPK_integer *lda, __CLPK_integer *info);
3839
3840 /* Subroutine */ int strtri_(char *uplo, char *diag, __CLPK_integer *n, __CLPK_real *a,
3841    __CLPK_integer *lda, __CLPK_integer *info);
3842
3843 /* Subroutine */ int strtrs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
3844    __CLPK_integer *nrhs, __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *b, __CLPK_integer *ldb,
    __CLPK_integer *
3845    info);
3846
3847 /* Subroutine */ int stzrqf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3848    __CLPK_real *tau, __CLPK_integer *info);
3849
3850 /* Subroutine */ int stzrzf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
3851    __CLPK_real *tau, __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *info);
3852
3853 /* Subroutine */ int xerbla_(char *sname, __CLPK_integer *info);
3854
3855 /* Subroutine */ int zbsdqr_(char *uplo, __CLPK_integer *n, __CLPK_integer *ncvt, __CLPK_integer *
3856    nru, __CLPK_integer *ncc, __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublecomplex *vt,
    __CLPK_integer *ldvt,
3857    __CLPK_doublecomplex *u, __CLPK_integer *ldu, __CLPK_doublecomplex *c__,
    __CLPK_integer *ldc,
3858    __CLPK_doublereal *rwork, __CLPK_integer *info);
3859
3860 /* Subroutine */ int zdrot_(__CLPK_integer *n, __CLPK_doublecomplex *cx, __CLPK_integer *incx,
3861    __CLPK_doublecomplex *cy, __CLPK_integer *incy, __CLPK_doublereal *c__, __CLPK_doublereal *s);
3862
3863 /* Subroutine */ int zdrcsl_(__CLPK_integer *n, __CLPK_doublereal *sa, __CLPK_doublecomplex *sx,
3864    __CLPK_integer *incx);
3865
3866 /* Subroutine */ int zgbrd_(char *vect, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *ncc,
3867    __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_doublecomplex *ab, __CLPK_integer *ldab,
    __CLPK_doublereal *d__,
3868    __CLPK_doublereal *e, __CLPK_doublecomplex *q, __CLPK_integer *ldq,
    __CLPK_doublecomplex *pt,
3869    __CLPK_integer *ldpt, __CLPK_doublecomplex *c__, __CLPK_integer *ldc,

```

```

3870     __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *info);
3871
3872 /* Subroutine */ int zgbcon_(char *norm, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku,
3873     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_doublereal *anorm,
3874     __CLPK_doublereal *rcond, __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *
3875     info);
3876
3877 /* Subroutine */ int zgbequ_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
3878     *ku,
3879     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *r__, __CLPK_doublereal *c__,
3880     __CLPK_doublereal *rowcnd, __CLPK_doublereal *colcnd, __CLPK_doublereal *amax, __CLPK_integer *
3881     info);
3882 /* Subroutine */ int zgbrfs_(char *trans, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *
3883     ku, __CLPK_integer *nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *
3884     afb, __CLPK_integer *ldafb, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
3885     __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
3886     __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *info);
3887
3888 /* Subroutine */ int zgbsv_(__CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku, __CLPK_integer *
3889     nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_doublecomplex *
3890     b, __CLPK_integer *ldb, __CLPK_integer *info);
3891
3892 /* Subroutine */ int zgbsvx_(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *kl,
3893     __CLPK_integer *ku, __CLPK_integer *nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab,
3894     __CLPK_doublecomplex *afb, __CLPK_integer *ldafb, __CLPK_integer *ipiv, char *equed,
3895     __CLPK_doublereal *r__, __CLPK_doublereal *c__, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
3896     __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *ferr,
3897     __CLPK_doublereal *berr, __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *
3898     info);
3899
3900 /* Subroutine */ int zgbt2f_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
3901     *ku,
3902     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_integer *info);
3903 /* Subroutine */ int zgbrf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
3904     *ku,
3905     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv, __CLPK_integer *info);
3906 /* Subroutine */ int zgbrts_(char *trans, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *
3907     ku, __CLPK_integer *nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_integer *ipiv,
3908     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_integer *info);
3909
3910 /* Subroutine */ int zgebak_(char *job, char *side, __CLPK_integer *n, __CLPK_integer *ilo,
3911     __CLPK_integer *ihi, __CLPK_doublereal *scale, __CLPK_integer *m, __CLPK_doublecomplex *v,
3912     __CLPK_integer *ldv, __CLPK_integer *info);
3913
3914 /* Subroutine */ int zgebal_(char *job, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer
3915     *lda, __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *scale, __CLPK_integer *info);
3916
3917 /* Subroutine */ int zgeb2d_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
3918     __CLPK_integer *lda, __CLPK_doublereal *d, __CLPK_doublereal *e, __CLPK_doublecomplex *tauq,
3919     __CLPK_doublecomplex *taup, __CLPK_doublecomplex *work, __CLPK_integer *info);
3920
3921 /* Subroutine */ int zgebrd_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
3922     __CLPK_integer *lda, __CLPK_doublereal *d, __CLPK_doublereal *e, __CLPK_doublecomplex *tauq,
3923     __CLPK_doublecomplex *taup, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *
3924     info);
3925
3926 /* Subroutine */ int zgecon_(char *norm, __CLPK_integer *n, __CLPK_doublecomplex *a,
3927     __CLPK_integer *lda, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublecomplex *
3928     work, __CLPK_doublereal *rwork, __CLPK_integer *info);
3929
3930 /* Subroutine */ int zgeequ_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
3931     __CLPK_integer *lda, __CLPK_doublereal *r__, __CLPK_doublereal *c__, __CLPK_doublereal *rowcnd,
3932     __CLPK_doublereal *colcnd, __CLPK_doublereal *amax, __CLPK_integer *info);
3933
3934 /* Subroutine */ int zgees_(char *jobvs, char *sort, __CLPK_L_fp select, __CLPK_integer *n,
3935     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *sdim, __CLPK_doublecomplex *w,
3936     __CLPK_doublecomplex *vs, __CLPK_integer *ldvs, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
3937     __CLPK_doublereal *rwork, __CLPK_logical *bwork, __CLPK_integer *info);
3938
3939 /* Subroutine */ int zgeesx_(char *jobvs, char *sort, __CLPK_L_fp select, char *
3940     sense, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *sdim,
3941     __CLPK_doublecomplex *w, __CLPK_doublecomplex *vs, __CLPK_integer *ldvs, __CLPK_doublereal *
3942     rconde, __CLPK_doublereal *rcondv, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
3943     __CLPK_doublereal *rwork, __CLPK_logical *bwork, __CLPK_integer *info);
3944
3945 /* Subroutine */ int zgeev_(char *jobvl, char *jobvr, __CLPK_integer *n,
3946     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *w, __CLPK_doublecomplex *vl,
3947     __CLPK_integer *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *ldvr, __CLPK_doublecomplex *work,
3948     __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *info);
3949
3950 /* Subroutine */ int zgeevx_(char *balanc, char *jobvl, char *jobvr, char *
3951     sense, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *w,
3952     __CLPK_doublecomplex *vl, __CLPK_integer *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *ldvr,
3953     __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *scale, __CLPK_doublereal *abnrm,

```



```

3954   __CLPK_doublecomplex *rconde, __CLPK_doublecomplex *rcondv, __CLPK_doublecomplex *work, __CLPK_integer *
3955   lwork, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
3956
3957 /* Subroutine */ int zgegs_(char *jobvs1, char *jobvsr, __CLPK_integer *n,
3958   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
3959   __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *beta, __CLPK_doublecomplex *vs1,
3960   __CLPK_integer *ldvs1, __CLPK_doublecomplex *vsr, __CLPK_integer *ldvsr, __CLPK_doublecomplex *
3961   work, __CLPK_integer *lwork, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
3962
3963 /* Subroutine */ int zgegv_(char *jobvl, char *jobvr, __CLPK_integer *n,
3964   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
3965   __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *beta, __CLPK_doublecomplex *v1, __CLPK_integer
3966   *ldv1, __CLPK_doublecomplex *vr, __CLPK_integer *ldvr, __CLPK_doublecomplex *work, __CLPK_integer
3967   *lwork, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
3968
3969 /* Subroutine */ int zgehd2_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi,
3970   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
3971   work, __CLPK_integer *info);
3972
3973 /* Subroutine */ int zgehrd_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi,
3974   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
3975   work, __CLPK_integer *lwork, __CLPK_integer *info);
3976
3977 /* Subroutine */ int zgelq2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
3978   __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *info);
3979
3980 /* Subroutine */ int zgelqf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
3981   __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
3982   __CLPK_integer *info);
3983
3984 /* Subroutine */ int zgels_(char *trans, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *
3985   nrhs, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
3986   __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *info);
3987
3988 /* Subroutine */ int zgelsx_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs,
3989   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
3990   __CLPK_integer *jpvt, __CLPK_doublecomplex *rcond, __CLPK_integer *rank, __CLPK_doublecomplex *work,
3991   __CLPK_doublecomplex *rwork, __CLPK_integer *info);
3992
3993 /* Subroutine */ int zgelsy_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs,
3994   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
3995   __CLPK_integer *jpvt, __CLPK_doublecomplex *rcond, __CLPK_integer *rank, __CLPK_doublecomplex *work,
3996   __CLPK_integer *lwork, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
3997
3998 /* Subroutine */ int zgeql2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
3999   __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *info);
4000
4001 /* Subroutine */ int zgeqlf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4002   __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
4003   __CLPK_integer *info);
4004
4005 /* Subroutine */ int zgeqp3_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4006   __CLPK_integer *lda, __CLPK_integer *jpvt, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work,
4007   __CLPK_integer *lwork, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4008
4009 /* Subroutine */ int zgeqpf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4010   __CLPK_integer *lda, __CLPK_integer *jpvt, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work,
4011   __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4012
4013 /* Subroutine */ int zgeqr2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4014   __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *info);
4015
4016 /* Subroutine */ int zgeqrf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4017   __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
4018   __CLPK_integer *info);
4019
4020 /* Subroutine */ int zgerfs_(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
4021   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *af, __CLPK_integer *ldaf,
4022   __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x,
4023   __CLPK_integer *ldx, __CLPK_doublecomplex *ferr, __CLPK_doublecomplex *berr, __CLPK_doublecomplex *work,
4024   __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4025
4026 /* Subroutine */ int zgerq2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4027   __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *info);
4028
4029 /* Subroutine */ int zgerqf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4030   __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
4031   __CLPK_integer *info);
4032
4033 /* Subroutine */ int zgesc2_(__CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
4034   __CLPK_doublecomplex *rhs, __CLPK_integer *ipiv, __CLPK_integer *jpiv, __CLPK_doublecomplex *scale);
4035
4036 /* Subroutine */ int zgesv_(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublecomplex *a,
4037   __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4038   __CLPK_integer *
4039   info);

```



```

4040 /* Subroutine */ int zgesvx(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *
4041 nrhs, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *af, __CLPK_integer *
4042 ldaf, __CLPK_integer *ipiv, char *equed, __CLPK_doublereal *r__, __CLPK_doublereal *c__,
4043 __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4044 __CLPK_doublereal *rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *
4045 work, __CLPK_doublereal *rwork, __CLPK_integer *info);
4046
4047 /* Subroutine */ int zgetc2(__CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
4048 __CLPK_integer *ipiv, __CLPK_integer *jpiv, __CLPK_integer *info);
4049
4050 /* Subroutine */ int zgetf2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4051 __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_integer *info);
4052
4053 /* Subroutine */ int zgetrf(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4054 __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_integer *info);
4055
4056 /* Subroutine */ int zgetri(__CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
4057 __CLPK_integer *ipiv, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *info);
4058
4059 /* Subroutine */ int zgetrs(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
4060 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *b,
4061 __CLPK_integer *ldb, __CLPK_integer *info);
4062
4063 /* Subroutine */ int zggbak(char *job, char *side, __CLPK_integer *n, __CLPK_integer *ilo,
4064 __CLPK_integer *ihi, __CLPK_doublereal *lscale, __CLPK_doublereal *rscale, __CLPK_integer *m,
4065 __CLPK_doublecomplex *v, __CLPK_integer *ldv, __CLPK_integer *info);
4066
4067 /* Subroutine */ int zggbal(char *job, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer
4068 *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_integer *ilo, __CLPK_integer *ihi,
4069 __CLPK_doublereal *lscale, __CLPK_doublereal *rscale, __CLPK_doublereal *work, __CLPK_integer *
4070 info);
4071
4072 /* Subroutine */ int zgges(char *jobvsl, char *jobvsr, char *sort, __CLPK_L_fp
4073 delctg, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
4074 __CLPK_integer *ldb, __CLPK_integer *sdim, __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *
4075 beta, __CLPK_doublecomplex *vsl, __CLPK_integer *ldvsl, __CLPK_doublecomplex *vsr, __CLPK_integer
4076 *ldvsr, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork,
4077 __CLPK_logical *bwork, __CLPK_integer *info);
4078
4079 /* Subroutine */ int zggesx(char *jobvsl, char *jobvsr, char *sort, __CLPK_L_fp
4080 delctg, char *sense, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
4081 __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_integer *sdim, __CLPK_doublecomplex *alpha,
4082 __CLPK_doublecomplex *beta, __CLPK_doublecomplex *vsl, __CLPK_integer *ldvsl,
4083 __CLPK_doublecomplex *vsr, __CLPK_integer *ldvsr, __CLPK_doublereal *rconde, __CLPK_doublereal *
4084 rcondv, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork,
4085 __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_logical *bwork, __CLPK_integer *info);
4086
4087 /* Subroutine */ int zggev(char *jobvl, char *jobvr, __CLPK_integer *n,
4088 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4089 __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *beta, __CLPK_doublecomplex *vl, __CLPK_integer
4090 *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *ldvr, __CLPK_doublecomplex *work, __CLPK_integer
4091 *lwork, __CLPK_doublereal *rwork, __CLPK_integer *info);
4092
4093 /* Subroutine */ int zggevx(char *balanc, char *jobvl, char *jobvr, char *
4094 sense, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
4095 __CLPK_integer *ldb, __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *beta,
4096 __CLPK_doublecomplex *vl, __CLPK_integer *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *ldvr,
4097 __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublereal *lscale, __CLPK_doublereal *rscale,
4098 __CLPK_doublereal *abnrm, __CLPK_doublereal *bbnrm, __CLPK_doublereal *rconde, __CLPK_doublereal *
4099 rcondv, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork,
4100 __CLPK_integer *iwork, __CLPK_logical *bwork, __CLPK_integer *info);
4101
4102 /* Subroutine */ int zgglm(__CLPK_integer *n, __CLPK_integer *m, __CLPK_integer *p,
4103 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4104 __CLPK_doublecomplex *d__, __CLPK_doublecomplex *x, __CLPK_doublecomplex *y, __CLPK_doublecomplex
4105 *work, __CLPK_integer *lwork, __CLPK_integer *info);
4106
4107 /* Subroutine */ int zgghrd(char *compq, char *compz, __CLPK_integer *n, __CLPK_integer *
4108 ilo, __CLPK_integer *ihi, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
4109 __CLPK_integer *ldb, __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_doublecomplex *z__,
4110 __CLPK_integer *ldz, __CLPK_integer *info);
4111
4112 /* Subroutine */ int zgglse(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *p,
4113 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4114 __CLPK_doublecomplex *c__, __CLPK_doublecomplex *d__, __CLPK_doublecomplex *x,
4115 __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *info);
4116
4117 /* Subroutine */ int zggrf(__CLPK_integer *n, __CLPK_integer *m, __CLPK_integer *p,
4118 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *taua, __CLPK_doublecomplex *b,
4119 __CLPK_integer *ldb, __CLPK_doublecomplex *taub, __CLPK_doublecomplex *work, __CLPK_integer *
4120 lwork, __CLPK_integer *info);
4121
4122 /* Subroutine */ int zggrqf(__CLPK_integer *m, __CLPK_integer *p, __CLPK_integer *n,
4123 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *taua, __CLPK_doublecomplex *b,
4124 __CLPK_integer *ldb, __CLPK_doublecomplex *taub, __CLPK_doublecomplex *work, __CLPK_integer *
4125 lwork, __CLPK_integer *info);
4126

```

```

4127 /* Subroutine */ int zggsvd(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
4128   __CLPK_integer *n, __CLPK_integer *p, __CLPK_integer *k, __CLPK_integer *l, __CLPK_doublecomplex
   *a,
4129   __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *alpha,
4130   __CLPK_doublecomplex *beta, __CLPK_doublecomplex *u, __CLPK_integer *ldu, __CLPK_doublecomplex *v,
4131   __CLPK_integer *ldv, __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_doublecomplex *work,
4132   __CLPK_doublecomplex *rwork, __CLPK_integer *iwork, __CLPK_integer *info);
4133
4134 /* Subroutine */ int zggsvp_(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
4135   __CLPK_integer *p, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
   __CLPK_doublecomplex
4136   *b, __CLPK_integer *ldb, __CLPK_doublecomplex *tol, __CLPK_doublecomplex *tolb, __CLPK_integer *k,
4137   __CLPK_integer *l, __CLPK_doublecomplex *u, __CLPK_integer *ldu, __CLPK_doublecomplex *v,
   __CLPK_integer
4138   *ldv, __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_integer *iwork, __CLPK_doublecomplex *
4139   rwork, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *info);
4140
4141 /* Subroutine */ int zgtcon_(char *norm, __CLPK_integer *n, __CLPK_doublecomplex *dl,
4142   __CLPK_doublecomplex *d, __CLPK_doublecomplex *du, __CLPK_doublecomplex *du2, __CLPK_integer *
4143   ipiv, __CLPK_doublecomplex *anorm, __CLPK_doublecomplex *rcond, __CLPK_doublecomplex *work,
4144   __CLPK_integer *info);
4145
4146 /* Subroutine */ int zgtrfs_(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
4147   __CLPK_doublecomplex *dl, __CLPK_doublecomplex *d, __CLPK_doublecomplex *du,
4148   __CLPK_doublecomplex *dlf, __CLPK_doublecomplex *df, __CLPK_doublecomplex *duf,
4149   __CLPK_doublecomplex *du2, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4150   __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublecomplex *ferr, __CLPK_doublecomplex *berr,
4151   __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4152
4153 /* Subroutine */ int zgtsv_(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublecomplex *dl,
4154   __CLPK_doublecomplex *d, __CLPK_doublecomplex *du, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4155   __CLPK_integer *info);
4156
4157 /* Subroutine */ int zgtsvx_(char *fact, char *trans, __CLPK_integer *n, __CLPK_integer *
4158   nrhs, __CLPK_doublecomplex *dl, __CLPK_doublecomplex *d, __CLPK_doublecomplex *du,
4159   __CLPK_doublecomplex *dlf, __CLPK_doublecomplex *df, __CLPK_doublecomplex *duf,
4160   __CLPK_doublecomplex *du2, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4161   __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublecomplex *rcond, __CLPK_doublecomplex *ferr,
4162   __CLPK_doublecomplex *berr, __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *
4163   info);
4164
4165 /* Subroutine */ int zgtrf_(__CLPK_integer *n, __CLPK_doublecomplex *dl, __CLPK_doublecomplex *
4166   d, __CLPK_doublecomplex *du, __CLPK_doublecomplex *du2, __CLPK_integer *ipiv, __CLPK_integer *
4167   info);
4168
4169 /* Subroutine */ int zgtrrs_(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
4170   __CLPK_doublecomplex *dl, __CLPK_doublecomplex *d, __CLPK_doublecomplex *du,
4171   __CLPK_doublecomplex *du2, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4172   __CLPK_integer *info);
4173
4174 /* Subroutine */ int zgtrs2_(__CLPK_integer *itrans, __CLPK_integer *n, __CLPK_integer *nrhs,
4175   __CLPK_doublecomplex *dl, __CLPK_doublecomplex *d, __CLPK_doublecomplex *du,
4176   __CLPK_doublecomplex *du2, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb);
4177
4178 /* Subroutine */ int zhbev_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4179   __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *w, __CLPK_doublecomplex *z,
4180   __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4181
4182 /* Subroutine */ int zhbev_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4183   __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *w, __CLPK_doublecomplex *z,
4184   __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_integer *iwork, __CLPK_doublecomplex *rwork,
4185   __CLPK_integer *lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
4186
4187 /* Subroutine */ int zhbev_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
4188   __CLPK_integer *kd, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *q,
4189   __CLPK_integer *ldq, __CLPK_doublecomplex *vl, __CLPK_doublecomplex *vu, __CLPK_integer *il,
   __CLPK_integer *
4190   iu, __CLPK_doublecomplex *abstol, __CLPK_integer *m, __CLPK_doublecomplex *w, __CLPK_doublecomplex *z,
4191   __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *iwork,
4192   __CLPK_integer *ifail, __CLPK_integer *info);
4193
4194 /* Subroutine */ int zhbgst_(char *vect, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
4195   __CLPK_integer *kb, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *bb,
4196   __CLPK_integer *ldbb, __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublecomplex *work,
4197   __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4198
4199 /* Subroutine */ int zhbgv_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
4200   __CLPK_integer *kb, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *bb,
4201   __CLPK_integer *ldbb, __CLPK_doublecomplex *w, __CLPK_doublecomplex *z, __CLPK_integer *ldz,
4202   __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4203
4204 /* Subroutine */ int zhbgvx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
4205   __CLPK_integer *ka, __CLPK_integer *kb, __CLPK_doublecomplex *ab, __CLPK_integer *ldab,
4206   __CLPK_doublecomplex *bb, __CLPK_integer *ldbb, __CLPK_doublecomplex *q, __CLPK_integer *ldq,
4207   __CLPK_doublecomplex *vl, __CLPK_doublecomplex *vu, __CLPK_integer *il, __CLPK_integer *iu,
   __CLPK_doublecomplex *
4208   abstol, __CLPK_integer *m, __CLPK_doublecomplex *w, __CLPK_doublecomplex *z, __CLPK_integer *ldz,

```

```

4209     __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *iwork, __CLPK_integer *
4210     ifail, __CLPK_integer *info);
4211
4212 /* Subroutine */ int zhbtrd(char *vect, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4213     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *d, __CLPK_doublereal *e,
4214     __CLPK_doublecomplex *g, __CLPK_integer *ldg, __CLPK_doublecomplex *work, __CLPK_integer *info);
4215
4216 /* Subroutine */ int zhecon_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4217     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond,
4218     __CLPK_doublecomplex *work, __CLPK_integer *info);
4219
4220 /* Subroutine */ int zheev_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_doublecomplex
4221     *a, __CLPK_integer *lda, __CLPK_doublereal *w, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
4222     __CLPK_doublereal *rwork, __CLPK_integer *info);
4223
4224 /* Subroutine */ int zheevd_(char *jobz, char *uplo, __CLPK_integer *n,
4225     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublereal *w, __CLPK_doublecomplex *work,
4226     __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *lrwork, __CLPK_integer *iwork,
4227     __CLPK_integer *liwork, __CLPK_integer *info);
4228
4229 /* Subroutine */ int zheevr_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
4230     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublereal *vl, __CLPK_doublereal *vu,
4231     __CLPK_integer *il, __CLPK_integer *iu, __CLPK_doublereal *abstol, __CLPK_integer *m,
4232     __CLPK_doublereal *
4233     w, __CLPK_doublecomplex *z, __CLPK_integer *ldz, __CLPK_integer *isuppz, __CLPK_doublecomplex *
4234     work, __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *lrwork, __CLPK_integer *
4235     iwork, __CLPK_integer *liwork, __CLPK_integer *info);
4236
4237 /* Subroutine */ int zheevx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
4238     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublereal *vl, __CLPK_doublereal *vu,
4239     __CLPK_integer *il, __CLPK_integer *iu, __CLPK_doublereal *abstol, __CLPK_integer *m,
4240     __CLPK_doublereal *
4241     w, __CLPK_doublecomplex *z, __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_integer *
4242     lwork, __CLPK_doublereal *rwork, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *
4243     info);
4244
4245 /* Subroutine */ int zhegs2_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n,
4246     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4247     __CLPK_integer *info);
4248
4249 /* Subroutine */ int zhegst_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n,
4250     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4251     __CLPK_integer *info);
4252
4253 /* Subroutine */ int zhegv_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
4254     n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4255     __CLPK_doublereal *w, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork,
4256     __CLPK_integer *info);
4257
4258 /* Subroutine */ int zhegvd_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
4259     n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4260     __CLPK_doublereal *w, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork,
4261     __CLPK_integer *lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
4262
4263 /* Subroutine */ int zhegvx_(__CLPK_integer *itype, char *jobz, char *range, char *
4264     uplo, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
4265     __CLPK_integer *ldb, __CLPK_doublereal *vl, __CLPK_doublereal *vu, __CLPK_integer *il,
4266     __CLPK_integer *
4267     iu, __CLPK_doublereal *abstol, __CLPK_integer *m, __CLPK_doublereal *w, __CLPK_doublecomplex *z,
4268     __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork,
4269     __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
4270
4271 /* Subroutine */ int zherfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4272     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *af, __CLPK_integer *ldaf,
4273     __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x,
4274     __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *work,
4275     __CLPK_doublereal *rwork, __CLPK_integer *info);
4276
4277 /* Subroutine */ int zhesv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4278     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *b,
4279     __CLPK_integer *ldb, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *info);
4280
4281 /* Subroutine */ int zhesvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
4282     nrhs, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *af, __CLPK_integer *
4283     ldaf, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x,
4284     __CLPK_integer *ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
4285     __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *info);
4286
4287 /* Subroutine */ int zhetf2_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4288     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_integer *info);
4289
4290 /* Subroutine */ int zhetrd_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4291     __CLPK_integer *lda, __CLPK_doublereal *d, __CLPK_doublereal *e, __CLPK_doublecomplex *tau,
4292     __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *info);
4293
4294 /* Subroutine */ int zhetrf_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4295     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *work, __CLPK_integer *lwork,

```

```

4293     __CLPK_integer *info);
4294
4295 /* Subroutine */ int zhetri_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4296     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *work, __CLPK_integer *info);
4297
4298 /* Subroutine */ int zhetrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4299     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *b,
4300     __CLPK_integer *ldb, __CLPK_integer *info);
4301
4302 /* Subroutine */ int zhgeqz_(char *jobz, char *compq, char *compz, __CLPK_integer *n,
4303     __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublecomplex *a, __CLPK_integer *lda,
4304     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *
4305     beta, __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_doublecomplex *z__, __CLPK_integer *
4306     ldz, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublecomplex *rwork, __CLPK_integer *
4307     info);
4308
4309 /* Subroutine */ int zhpcon_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4310     __CLPK_integer *ipiv, __CLPK_doublecomplex *anorm, __CLPK_doublecomplex *rcond, __CLPK_doublecomplex *
4311     work, __CLPK_integer *info);
4312
4313 /* Subroutine */ int zhpev_(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_doublecomplex
4314     *ap, __CLPK_doublecomplex *w, __CLPK_doublecomplex *z__, __CLPK_integer *ldz, __CLPK_doublecomplex *
4315     work, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4316
4317 /* Subroutine */ int zhpevd_(char *jobz, char *uplo, __CLPK_integer *n,
4318     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *w, __CLPK_doublecomplex *z__, __CLPK_integer *ldz,
4319     __CLPK_doublecomplex *work, __CLPK_doublecomplex *lwork, __CLPK_doublecomplex *rwork, __CLPK_integer *
4320     lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
4321
4322 /* Subroutine */ int zhpevx_(char *jobz, char *range, char *uplo, __CLPK_integer *n,
4323     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *vl, __CLPK_doublecomplex *vu, __CLPK_integer *il,
4324     __CLPK_integer *iu, __CLPK_doublecomplex *abstol, __CLPK_integer *m, __CLPK_doublecomplex *w,
4325     __CLPK_doublecomplex *z__, __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_doublecomplex *
4326     rwork, __CLPK_integer *iwork, __CLPK_integer *ifail, __CLPK_integer *info);
4327
4328 /* Subroutine */ int zhpgst_(__CLPK_integer *itype, char *uplo, __CLPK_integer *n,
4329     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *bp, __CLPK_integer *info);
4330
4331 /* Subroutine */ int zhpgv_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
4332     n, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *bp, __CLPK_doublecomplex *w, __CLPK_doublecomplex
4333     *z__, __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *
4334     info);
4335
4336 /* Subroutine */ int zhpgvd_(__CLPK_integer *itype, char *jobz, char *uplo, __CLPK_integer *
4337     n, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *bp, __CLPK_doublecomplex *w, __CLPK_doublecomplex
4338     *z__, __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublecomplex *
4339     rwork, __CLPK_integer *lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *
4340     info);
4341
4342 /* Subroutine */ int zhpgvx_(__CLPK_integer *itype, char *jobz, char *range, char *
4343     uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *bp, __CLPK_doublecomplex *
4344     vl, __CLPK_doublecomplex *vu, __CLPK_integer *il, __CLPK_integer *iu, __CLPK_doublecomplex *abstol,
4345     __CLPK_integer *m, __CLPK_doublecomplex *w, __CLPK_doublecomplex *z__, __CLPK_integer *ldz,
4346     __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *iwork, __CLPK_integer *
4347     ifail, __CLPK_integer *info);
4348
4349 /* Subroutine */ int zhprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4350     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *afp, __CLPK_integer *ipiv, __CLPK_doublecomplex *
4351     b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublecomplex *ferr,
4352     __CLPK_doublecomplex *berr, __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *
4353     info);
4354
4355 /* Subroutine */ int zhpsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4356     __CLPK_doublecomplex *ap, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4357     __CLPK_integer *info);
4358
4359 /* Subroutine */ int zhpsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
4360     nrhs, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *afp, __CLPK_integer *ipiv,
4361     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4362     __CLPK_doublecomplex *rcond, __CLPK_doublecomplex *ferr, __CLPK_doublecomplex *berr, __CLPK_doublecomplex *
4363     work, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4364
4365 /* Subroutine */ int zhptrd_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4366     __CLPK_doublecomplex *d__, __CLPK_doublecomplex *e, __CLPK_doublecomplex *tau, __CLPK_integer *info);
4367
4368 /* Subroutine */ int zhptrf_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4369     __CLPK_integer *ipiv, __CLPK_integer *info);
4370
4371 /* Subroutine */ int zhptri_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4372     __CLPK_integer *ipiv, __CLPK_doublecomplex *work, __CLPK_integer *info);
4373
4374 /* Subroutine */ int zhptrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4375     __CLPK_doublecomplex *ap, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4376     __CLPK_integer *info);
4377
4378 /* Subroutine */ int zhsein_(char *side, char *eigsrc, char *initv, __CLPK_logical *
4379     select, __CLPK_integer *n, __CLPK_doublecomplex *h__, __CLPK_integer *ldh, __CLPK_doublecomplex *

```

```

4380     w, __CLPK_doublecomplex *vl, __CLPK_integer *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *ldvr,
4381     __CLPK_integer *mm, __CLPK_integer *m, __CLPK_doublecomplex *work, __CLPK_doublereal *rwork,
4382     __CLPK_integer *ifail1, __CLPK_integer *ifailr, __CLPK_integer *info);
4383
4384 /* Subroutine */ int zhseqr_(char *job, char *compz, __CLPK_integer *n, __CLPK_integer *ilo,
4385     __CLPK_integer *ihi, __CLPK_doublecomplex *h, __CLPK_integer *ldh, __CLPK_doublecomplex *w,
4386     __CLPK_doublecomplex *z, __CLPK_integer *ldz, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
4387     __CLPK_integer *info);
4388
4389 /* Subroutine */ int zlabrd__(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nb,
4390     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublereal *d, __CLPK_doublereal *e,
4391     __CLPK_doublecomplex *tauq, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *x, __CLPK_integer *
4392     ldz, __CLPK_doublecomplex *y, __CLPK_integer *ldy);
4393
4394 /* Subroutine */ int zlacgv__(__CLPK_integer *n, __CLPK_doublecomplex *x, __CLPK_integer *incx);
4395
4396 /* Subroutine */ int zlacon__(__CLPK_integer *n, __CLPK_doublecomplex *v, __CLPK_doublecomplex *x,
4397     __CLPK_doublereal *est, __CLPK_integer *kase);
4398
4399 /* Subroutine */ int zlap2__(char *uplo, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *
4400     a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb);
4401
4402 /* Subroutine */ int zlapcy__(char *uplo, __CLPK_integer *m, __CLPK_integer *n,
4403     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb);
4404
4405 /* Subroutine */ int zlacrm__(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4406     __CLPK_integer *lda, __CLPK_doublereal *b, __CLPK_integer *ldb, __CLPK_doublecomplex *c,
4407     __CLPK_integer *ldc, __CLPK_doublereal *rwork);
4408
4409 /* Subroutine */ int zlacrt__(__CLPK_integer *n, __CLPK_doublecomplex *cx, __CLPK_integer *incx,
4410     __CLPK_doublecomplex *cy, __CLPK_integer *incy, __CLPK_doublecomplex *c, __CLPK_doublecomplex *
4411     s);
4412
4413 /* Subroutine */ int zlaed0__(__CLPK_integer *qsiz, __CLPK_integer *n, __CLPK_doublereal *d,
4414     __CLPK_doublereal *e, __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_doublecomplex *qstore,
4415     __CLPK_integer *ldqs, __CLPK_doublereal *rwork, __CLPK_integer *iwork, __CLPK_integer *info);
4416
4417 /* Subroutine */ int zlaed7__(__CLPK_integer *n, __CLPK_integer *cutpnt, __CLPK_integer *qsiz,
4418     __CLPK_integer *tlvl, __CLPK_integer *curlvl, __CLPK_integer *curpbm, __CLPK_doublereal *d,
4419     __CLPK_doublecomplex *g, __CLPK_integer *ldg, __CLPK_doublereal *rho, __CLPK_integer *indxq,
4420     __CLPK_doublereal *qstore, __CLPK_integer *qpnr, __CLPK_integer *prmptr, __CLPK_integer *perm,
4421     __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_doublereal *givnum, __CLPK_doublecomplex *
4422     work, __CLPK_doublereal *rwork, __CLPK_integer *iwork, __CLPK_integer *info);
4423
4424 /* Subroutine */ int zlaed8__(__CLPK_integer *k, __CLPK_integer *n, __CLPK_integer *qsiz,
4425     __CLPK_doublecomplex *g, __CLPK_integer *ldg, __CLPK_doublereal *d, __CLPK_doublereal *rho,
4426     __CLPK_integer *cutpnt, __CLPK_doublereal *z, __CLPK_doublereal *dlamda, __CLPK_doublecomplex *
4427     q2, __CLPK_integer *ldq2, __CLPK_doublereal *w, __CLPK_integer *indxq, __CLPK_integer *indx,
4428     __CLPK_integer *indxq, __CLPK_integer *perm, __CLPK_integer *givptr, __CLPK_integer *givcol,
4429     __CLPK_doublereal *givnum, __CLPK_integer *info);
4430
4431 /* Subroutine */ int zlaein__(__CLPK_logical *rightv, __CLPK_logical *noinit, __CLPK_integer *n,
4432     __CLPK_doublecomplex *h, __CLPK_integer *ldh, __CLPK_doublecomplex *w, __CLPK_doublecomplex *v,
4433     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublereal *rwork, __CLPK_doublereal *eps3,
4434     __CLPK_doublereal *smlnum, __CLPK_integer *info);
4435
4436 /* Subroutine */ int zlaesy__(__CLPK_doublecomplex *a, __CLPK_doublecomplex *b,
4437     __CLPK_doublecomplex *c, __CLPK_doublecomplex *rt1, __CLPK_doublecomplex *rt2,
4438     __CLPK_doublecomplex *evscal, __CLPK_doublecomplex *cs1, __CLPK_doublecomplex *sn1);
4439
4440 /* Subroutine */ int zlaev2__(__CLPK_doublecomplex *a, __CLPK_doublecomplex *b,
4441     __CLPK_doublecomplex *c, __CLPK_doublereal *rt1, __CLPK_doublereal *rt2, __CLPK_doublereal *cs1,
4442     __CLPK_doublecomplex *sn1);
4443
4444 /* Subroutine */ int zlags2__(__CLPK_logical *upper, __CLPK_doublereal *a1, __CLPK_doublecomplex *
4445     a2, __CLPK_doublereal *a3, __CLPK_doublereal *b1, __CLPK_doublecomplex *b2, __CLPK_doublereal *b3,
4446     __CLPK_doublereal *csu, __CLPK_doublecomplex *snu, __CLPK_doublereal *csv, __CLPK_doublecomplex *
4447     snv, __CLPK_doublereal *csq, __CLPK_doublecomplex *snq);
4448
4449 /* Subroutine */ int zlagtm__(char *trans, __CLPK_integer *n, __CLPK_integer *nrhs,
4450     __CLPK_doublereal *alpha, __CLPK_doublecomplex *dl, __CLPK_doublecomplex *d,
4451     __CLPK_doublecomplex *du, __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublereal *beta,
4452     __CLPK_doublecomplex *b, __CLPK_integer *ldb);
4453
4454 /* Subroutine */ int zlahef__(char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_integer *kb,
4455     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *w,
4456     __CLPK_integer *ldw, __CLPK_integer *info);
4457
4458 /* Subroutine */ int zlahqr__(__CLPK_logical *wantt, __CLPK_logical *wantz, __CLPK_integer *n,
4459     __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublecomplex *h, __CLPK_integer *ldh,
4460     __CLPK_doublecomplex *w, __CLPK_integer *iloz, __CLPK_integer *ihiz, __CLPK_doublecomplex *z,
4461     __CLPK_integer *ldz, __CLPK_integer *info);
4462
4463 /* Subroutine */ int zlahrd__(__CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *nb,
4464     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *t,
4465     __CLPK_integer *ldt, __CLPK_doublecomplex *y, __CLPK_integer *ldy);
4466

```



```

4467 /* Subroutine */ int zlaicl(__CLPK_integer *job, __CLPK_integer *j, __CLPK_doublecomplex *x,
4468 __CLPK_doublereal *sest, __CLPK_doublecomplex *w, __CLPK_doublecomplex *gamma, __CLPK_doublereal *
4469 sestpr, __CLPK_doublecomplex *s, __CLPK_doublecomplex *c__);
4470
4471 /* Subroutine */ int zlals0(__CLPK_integer *icompq, __CLPK_integer *nl, __CLPK_integer *nr,
4472 __CLPK_integer *sqre, __CLPK_integer *nrhs, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4473 __CLPK_doublecomplex *bx, __CLPK_integer *ldb, __CLPK_integer *perm, __CLPK_integer *givptr,
4474 __CLPK_integer *givcol, __CLPK_integer *ldgcol, __CLPK_doublereal *givnum, __CLPK_integer *ldgnum,
4475 __CLPK_doublereal *poles, __CLPK_doublereal *difl, __CLPK_doublereal *difr, __CLPK_doublereal *
4476 z__, __CLPK_integer *k, __CLPK_doublereal *c__, __CLPK_doublereal *s, __CLPK_doublereal *rwork,
4477 __CLPK_integer *info);
4478
4479 /* Subroutine */ int zlalsa(__CLPK_integer *icompq, __CLPK_integer *smlsiz, __CLPK_integer *n,
4480 __CLPK_integer *nrhs, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *bx,
4481 __CLPK_integer *ldb, __CLPK_doublereal *u, __CLPK_integer *ldu, __CLPK_doublereal *vt,
4482 __CLPK_integer *
4483 k, __CLPK_doublereal *difl, __CLPK_doublereal *difr, __CLPK_doublereal *z__, __CLPK_doublereal *
4484 poles, __CLPK_integer *givptr, __CLPK_integer *givcol, __CLPK_integer *ldgcol, __CLPK_integer *
4485 perm, __CLPK_doublereal *givnum, __CLPK_doublereal *c__, __CLPK_doublereal *s, __CLPK_doublereal *
4486 rwork, __CLPK_integer *iwork, __CLPK_integer *info);
4487
4488 /* Subroutine */ int zlapll(__CLPK_integer *n, __CLPK_doublecomplex *x, __CLPK_integer *incx,
4489 __CLPK_doublecomplex *y, __CLPK_integer *incy, __CLPK_doublereal *ssmin);
4490
4491 /* Subroutine */ int zlapmt(__CLPK_logical *forwrd, __CLPK_integer *m, __CLPK_integer *n,
4492 __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_integer *k);
4493
4494 /* Subroutine */ int zlaqgb(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer
4495 *ku,
4496 __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *r__, __CLPK_doublereal *c__,
4497 __CLPK_doublereal *rowcnd, __CLPK_doublereal *colcnd, __CLPK_doublereal *amax, char *equad);
4498
4499 /* Subroutine */ int zlaqge(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
4500 __CLPK_integer *lda, __CLPK_doublereal *r__, __CLPK_doublereal *c__, __CLPK_doublereal *rowcnd,
4501 __CLPK_doublereal *colcnd, __CLPK_doublereal *amax, char *equad);
4502
4503 /* Subroutine */ int zlaqhb(char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4504 __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *s, __CLPK_doublereal *scond,
4505 __CLPK_doublereal *amax, char *equad);
4506
4507 /* Subroutine */ int zlaqhe(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4508 __CLPK_integer *lda, __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax,
4509 char *equad);
4510
4511 /* Subroutine */ int zlaqhp(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4512 __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax, char *equad);
4513
4514 /* Subroutine */ int zlaqp2(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *offset,
4515 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *jpvt, __CLPK_doublecomplex *tau,
4516 __CLPK_doublereal *vn1, __CLPK_doublereal *vn2, __CLPK_doublecomplex *work);
4517
4518 /* Subroutine */ int zlaqps(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *offset,
4519 __CLPK_integer
4520 *nb, __CLPK_integer *kb, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *jpvt,
4521 __CLPK_doublecomplex *tau, __CLPK_doublereal *vn1, __CLPK_doublereal *vn2, __CLPK_doublecomplex *
4522 auxv, __CLPK_doublecomplex *f, __CLPK_integer *ldf);
4523
4524 /* Subroutine */ int zlaqsb(char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4525 __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *s, __CLPK_doublereal *scond,
4526 __CLPK_doublereal *amax, char *equad);
4527
4528 /* Subroutine */ int zlaqsp(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4529 __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax, char *equad);
4530
4531 /* Subroutine */ int zlaqsy(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4532 __CLPK_integer *lda, __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax,
4533 char *equad);
4534
4535 /* Subroutine */ int zlar1v(__CLPK_integer *n, __CLPK_integer *b1, __CLPK_integer *bn,
4536 __CLPK_doublereal
4537 *sigma, __CLPK_doublereal *d__, __CLPK_doublereal *l, __CLPK_doublereal *ld, __CLPK_doublereal *
4538 lld, __CLPK_doublereal *gersch, __CLPK_doublecomplex *z__, __CLPK_doublereal *ztz,
4539 __CLPK_doublereal *mingma, __CLPK_integer *r__, __CLPK_integer *isuppz, __CLPK_doublereal *work);
4540
4541 /* Subroutine */ int zlar2v(__CLPK_integer *n, __CLPK_doublecomplex *x, __CLPK_doublecomplex *y,
4542 __CLPK_doublecomplex *z__, __CLPK_integer *incx, __CLPK_doublereal *c__, __CLPK_doublecomplex *s,
4543 __CLPK_integer *inc);
4544
4545 /* Subroutine */ int zlarcm(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer
4546 *
4547 lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *c__, __CLPK_integer *ldc,
4548 __CLPK_doublereal *rwork);
4549
4550 /* Subroutine */ int zlarf(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex
4551 *v, __CLPK_integer *incv, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *c__, __CLPK_integer *
4552 ldc, __CLPK_doublecomplex *work);

```

```

4549 /* Subroutine */ int zlarfb_(char *side, char *trans, char *direct, char *
4550 storev, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublecomplex *v,
    __CLPK_integer
4551 *ldv, __CLPK_doublecomplex *t, __CLPK_integer *ldt, __CLPK_doublecomplex *c__, __CLPK_integer *
4552 ldc, __CLPK_doublecomplex *work, __CLPK_integer *ldwork);
4553
4554 /* Subroutine */ int zlarfg_(__CLPK_integer *n, __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *
4555 x, __CLPK_integer *incx, __CLPK_doublecomplex *tau);
4556
4557 /* Subroutine */ int zlarft_(char *direct, char *storev, __CLPK_integer *n, __CLPK_integer *
4558 k, __CLPK_doublecomplex *v, __CLPK_integer *ldv, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
4559 t, __CLPK_integer *ldt);
4560
4561 /* Subroutine */ int zlarfx_(char *side, __CLPK_integer *m, __CLPK_integer *n,
4562 __CLPK_doublecomplex *v, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *c__, __CLPK_integer *
4563 ldc, __CLPK_doublecomplex *work);
4564
4565 /* Subroutine */ int zlargv_(__CLPK_integer *n, __CLPK_doublecomplex *x, __CLPK_integer *incx,
4566 __CLPK_doublecomplex *y, __CLPK_integer *incy, __CLPK_doublecomplex *c__, __CLPK_integer *incc);
4567
4568 /* Subroutine */ int zlarv_(__CLPK_integer *idist, __CLPK_integer *iseed, __CLPK_integer *n,
4569 __CLPK_doublecomplex *x);
4570
4571 /* Subroutine */ int zlarv_(__CLPK_integer *n, __CLPK_doublecomplex *d__, __CLPK_doublecomplex *l,
4572 __CLPK_integer *split, __CLPK_integer *m, __CLPK_doublecomplex *w, __CLPK_integer *iblock,
4573 __CLPK_doublecomplex *gersch, __CLPK_doublecomplex *tol, __CLPK_doublecomplex *z__, __CLPK_integer *ldz,
4574 __CLPK_integer *isuppz, __CLPK_doublecomplex *work, __CLPK_integer *iwork, __CLPK_integer *info);
4575
4576 /* Subroutine */ int zlartg_(__CLPK_doublecomplex *f, __CLPK_doublecomplex *g, __CLPK_doublecomplex *
4577 cs, __CLPK_doublecomplex *sn, __CLPK_doublecomplex *r__);
4578
4579 /* Subroutine */ int zlartv_(__CLPK_integer *n, __CLPK_doublecomplex *x, __CLPK_integer *incx,
4580 __CLPK_doublecomplex *y, __CLPK_integer *incy, __CLPK_doublecomplex *c__, __CLPK_doublecomplex *s,
4581 __CLPK_integer *incc);
4582
4583 /* Subroutine */ int zlarz_(char *side, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *l,
4584 __CLPK_doublecomplex *v, __CLPK_integer *incv, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
4585 c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work);
4586
4587 /* Subroutine */ int zlarzb_(char *side, char *trans, char *direct, char *
4588 storev, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *l,
    __CLPK_doublecomplex
4589 *v, __CLPK_integer *ldv, __CLPK_doublecomplex *t, __CLPK_integer *ldt, __CLPK_doublecomplex *c__,
4590 __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *ldwork);
4591
4592 /* Subroutine */ int zlarzt_(char *direct, char *storev, __CLPK_integer *n, __CLPK_integer *
4593 k, __CLPK_doublecomplex *v, __CLPK_integer *ldv, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
4594 t, __CLPK_integer *ldt);
4595
4596 /* Subroutine */ int zlascl_(char *type__, __CLPK_integer *kl, __CLPK_integer *ku,
4597 __CLPK_doublecomplex *cfrom, __CLPK_doublecomplex *cto, __CLPK_integer *m, __CLPK_integer *n,
4598 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *info);
4599
4600 /* Subroutine */ int zlaset_(char *uplo, __CLPK_integer *m, __CLPK_integer *n,
4601 __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *beta, __CLPK_doublecomplex *a, __CLPK_integer *
4602 lda);
4603
4604 /* Subroutine */ int zlasr_(char *side, char *pivot, char *direct, __CLPK_integer *m,
4605 __CLPK_integer *n, __CLPK_doublecomplex *c__, __CLPK_doublecomplex *s, __CLPK_doublecomplex *a,
4606 __CLPK_integer *lda);
4607
4608 /* Subroutine */ int zlassq_(__CLPK_integer *n, __CLPK_doublecomplex *x, __CLPK_integer *incx,
4609 __CLPK_doublecomplex *scale, __CLPK_doublecomplex *sumsq);
4610
4611 /* Subroutine */ int zlaswp_(__CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
4612 __CLPK_integer *k1, __CLPK_integer *k2, __CLPK_integer *ipiv, __CLPK_integer *incx);
4613
4614 /* Subroutine */ int zlasyf_(char *uplo, __CLPK_integer *n, __CLPK_integer *nb, __CLPK_integer *kb,
4615 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *w,
4616 __CLPK_integer *ldw, __CLPK_integer *info);
4617
4618 /* Subroutine */ int zlatbs_(char *uplo, char *trans, char *diag, char *
4619 normin, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_doublecomplex *ab, __CLPK_integer *ldab,
4620 __CLPK_doublecomplex *x, __CLPK_doublecomplex *scale, __CLPK_doublecomplex *cnorm, __CLPK_integer *info);
4621
4622 /* Subroutine */ int zlatdf_(__CLPK_integer *ijob, __CLPK_integer *n, __CLPK_doublecomplex *z__,
4623 __CLPK_integer *ldz, __CLPK_doublecomplex *rhs, __CLPK_doublecomplex *rdsum, __CLPK_doublecomplex *
4624 rdscal, __CLPK_integer *ipiv, __CLPK_integer *jpiv);
4625
4626 /* Subroutine */ int zlatps_(char *uplo, char *trans, char *diag, char *
4627 normin, __CLPK_integer *n, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *x, __CLPK_doublecomplex *
4628 scale, __CLPK_doublecomplex *cnorm, __CLPK_integer *info);
4629
4630 /* Subroutine */ int zlatrd_(char *uplo, __CLPK_integer *n, __CLPK_integer *nb,
4631 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *e, __CLPK_doublecomplex *tau,
4632 __CLPK_doublecomplex *w, __CLPK_integer *ldw);
4633

```

```

4634 /* Subroutine */ int zlatrs_(char *uplo, char *trans, char *diag, char *
4635 normin, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *x,
4636 __CLPK_doublereal *scale, __CLPK_doublereal *cnorm, __CLPK_integer *info);
4637
4638 /* Subroutine */ int zlatrz_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *l,
4639 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
4640 work);
4641
4642 /* Subroutine */ int zlatzm_(char *side, __CLPK_integer *m, __CLPK_integer *n,
4643 __CLPK_doublecomplex *v, __CLPK_integer *incv, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
4644 c1, __CLPK_doublecomplex *c2, __CLPK_integer *ldc, __CLPK_doublecomplex *work);
4645
4646 /* Subroutine */ int zlaau2_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4647 __CLPK_integer *lda, __CLPK_integer *info);
4648
4649 /* Subroutine */ int zlaaum_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4650 __CLPK_integer *lda, __CLPK_integer *info);
4651
4652 /* Subroutine */ int zpbcon_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4653 __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *anorm, __CLPK_doublereal *
4654 rcond, __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *info);
4655
4656 /* Subroutine */ int zpbequ_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4657 __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *s, __CLPK_doublereal *scond,
4658 __CLPK_doublereal *amax, __CLPK_integer *info);
4659
4660 /* Subroutine */ int zpbrfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
4661 nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *afb, __CLPK_integer *
4662 ldafb, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4663 __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *work, __CLPK_doublereal *
4664 rwork, __CLPK_integer *info);
4665
4666 /* Subroutine */ int zpbstf_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4667 __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_integer *info);
4668
4669 /* Subroutine */ int zpbsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
4670 nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *b, __CLPK_integer *
4671 ldb, __CLPK_integer *info);
4672
4673 /* Subroutine */ int zpbsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4674 __CLPK_integer *nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *afb,
4675 __CLPK_integer *ldafb, char *equed, __CLPK_doublereal *s, __CLPK_doublecomplex *b, __CLPK_integer
4676 *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *
4677 ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *work, __CLPK_doublereal *rwork,
4678 __CLPK_integer *info);
4679
4680 /* Subroutine */ int zpbt2_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4681 __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_integer *info);
4682
4683 /* Subroutine */ int zpbtfr_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd,
4684 __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_integer *info);
4685
4686 /* Subroutine */ int zpbttrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *kd, __CLPK_integer *
4687 nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *b, __CLPK_integer *
4688 ldb, __CLPK_integer *info);
4689
4690 /* Subroutine */ int zpocon_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4691 __CLPK_integer *lda, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublecomplex *
4692 work, __CLPK_doublereal *rwork, __CLPK_integer *info);
4693
4694 /* Subroutine */ int zpoequ_(__CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
4695 __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax, __CLPK_integer *info);
4696
4697 /* Subroutine */ int zporfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4698 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *af, __CLPK_integer *ldaf,
4699 __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4700 __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *work, __CLPK_doublereal *
4701 rwork, __CLPK_integer *info);
4702
4703 /* Subroutine */ int zposv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4704 __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4705 __CLPK_integer *info);
4706
4707 /* Subroutine */ int zposvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
4708 nrhs, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *af, __CLPK_integer *
4709 ldaf, char *equed, __CLPK_doublereal *s, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4710 __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *ferr,
4711 __CLPK_doublereal *berr, __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *
4712 info);
4713
4714 /* Subroutine */ int zpotf2_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4715 __CLPK_integer *lda, __CLPK_integer *info);
4716
4717 /* Subroutine */ int zpotrf_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4718 __CLPK_integer *lda, __CLPK_integer *info);
4719
4720 /* Subroutine */ int zpotri_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,

```



```

4721     __CLPK_integer *lda, __CLPK_integer *info);
4722
4723 /* Subroutine */ int zpotrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4724     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4725     __CLPK_integer *info);
4726
4727 /* Subroutine */ int zppcon_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4728     __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublecomplex *work, __CLPK_doublereal
4729     *rwork, __CLPK_integer *info);
4730
4731 /* Subroutine */ int zppequ_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4732     __CLPK_doublereal *s, __CLPK_doublereal *scond, __CLPK_doublereal *amax, __CLPK_integer *info);
4733
4734 /* Subroutine */ int zpprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4735     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *afp, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4736     __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
4737     __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *info);
4738
4739 /* Subroutine */ int zppsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4740     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_integer *info);
4741
4742 /* Subroutine */ int zppsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
4743     nrhs, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *afp, char *equed, __CLPK_doublereal *
4744     s, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4745     __CLPK_doublereal *rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *
4746     work, __CLPK_doublereal *rwork, __CLPK_integer *info);
4747
4748 /* Subroutine */ int zptrfs_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4749     __CLPK_integer *info);
4750
4751 /* Subroutine */ int zpptri_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4752     __CLPK_integer *info);
4753
4754 /* Subroutine */ int zpptrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4755     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_integer *info);
4756
4757 /* Subroutine */ int zptcon_(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublecomplex *e,
4758     __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublereal *rwork, __CLPK_integer *
4759     info);
4760
4761 /* Subroutine */ int zptrfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4762     __CLPK_doublereal *d__, __CLPK_doublecomplex *e, __CLPK_doublereal *df, __CLPK_doublecomplex *ef,
4763     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4764     __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *work, __CLPK_doublereal *
4765     rwork, __CLPK_integer *info);
4766
4767 /* Subroutine */ int zpstv_(__CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_doublereal *d__,
4768     __CLPK_doublecomplex *e, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_integer *info);
4769
4770 /* Subroutine */ int zpsvx_(char *fact, __CLPK_integer *n, __CLPK_integer *nrhs,
4771     __CLPK_doublereal *d__, __CLPK_doublecomplex *e, __CLPK_doublereal *df, __CLPK_doublecomplex *ef,
4772     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4773     __CLPK_doublereal *rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *
4774     work, __CLPK_doublereal *rwork, __CLPK_integer *info);
4775
4776 /* Subroutine */ int zptrfs_(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublecomplex *e,
4777     __CLPK_integer *info);
4778
4779 /* Subroutine */ int zptrfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4780     __CLPK_doublereal *d__, __CLPK_doublecomplex *e, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4781     __CLPK_integer *info);
4782
4783 /* Subroutine */ int zppts2_(__CLPK_integer *iuplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4784     __CLPK_doublereal *d__, __CLPK_doublecomplex *e, __CLPK_doublecomplex *b, __CLPK_integer *ldb);
4785
4786 /* Subroutine */ int zrot_(__CLPK_integer *n, __CLPK_doublecomplex *cx, __CLPK_integer *incx,
4787     __CLPK_doublecomplex *cy, __CLPK_integer *incy, __CLPK_doublereal *c__, __CLPK_doublecomplex *s);
4788
4789 /* Subroutine */ int zspcon_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4790     __CLPK_integer *ipiv, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond, __CLPK_doublecomplex *
4791     work, __CLPK_integer *info);
4792
4793 /* Subroutine */ int zspmv_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *alpha,
4794     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *x, __CLPK_integer *incx, __CLPK_doublecomplex *
4795     beta, __CLPK_doublecomplex *y, __CLPK_integer *incy);
4796
4797 /* Subroutine */ int zspr_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *alpha,
4798     __CLPK_doublecomplex *x, __CLPK_integer *incx, __CLPK_doublecomplex *ap);
4799
4800 /* Subroutine */ int zsprfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4801     __CLPK_doublecomplex *ap, __CLPK_doublecomplex *afp, __CLPK_integer *ipiv, __CLPK_doublecomplex *
4802     b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublereal *ferr,
4803     __CLPK_doublereal *berr, __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *
4804     info);
4805
4806 /* Subroutine */ int zspsv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4807     __CLPK_doublecomplex *ap, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,

```

```

4808     __CLPK_integer *info);
4809
4810 /* Subroutine */ int zspsvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
4811     nrhs, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *afp, __CLPK_integer *ipiv,
4812     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4813     __CLPK_doublereal *rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *
4814     work, __CLPK_doublereal *rwork, __CLPK_integer *info);
4815
4816 /* Subroutine */ int zsptrf_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4817     __CLPK_integer *ipiv, __CLPK_integer *info);
4818
4819 /* Subroutine */ int zsptri_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
4820     __CLPK_integer *ipiv, __CLPK_doublecomplex *work, __CLPK_integer *info);
4821
4822 /* Subroutine */ int zsptrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4823     __CLPK_doublecomplex *ap, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4824     __CLPK_integer *info);
4825
4826 /* Subroutine */ int zstedc_(char *compz, __CLPK_integer *n, __CLPK_doublereal *d__,
4827     __CLPK_doublereal *e, __CLPK_doublecomplex *z__, __CLPK_integer *ldz, __CLPK_doublecomplex *work,
4828     __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *lrwork, __CLPK_integer *iwork,
4829     __CLPK_integer *liwork, __CLPK_integer *info);
4830
4831 /* Subroutine */ int zstein_(__CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *e,
4832     __CLPK_integer *m, __CLPK_doublereal *w, __CLPK_integer *iblock, __CLPK_integer *isplit,
4833     __CLPK_doublecomplex *z__, __CLPK_integer *ldz, __CLPK_doublereal *work, __CLPK_integer *iwork,
4834     __CLPK_integer *ifail, __CLPK_integer *info);
4835
4836 /* Subroutine */ int zsteqr_(char *compz, __CLPK_integer *n, __CLPK_doublereal *d__,
4837     __CLPK_doublereal *e, __CLPK_doublecomplex *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
4838     __CLPK_integer *info);
4839
4840 /* Subroutine */ int zsycon_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4841     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublereal *anorm, __CLPK_doublereal *rcond,
4842     __CLPK_doublecomplex *work, __CLPK_integer *info);
4843
4844 /* Subroutine */ int zsymv_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *alpha,
4845     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *x, __CLPK_integer *incx,
4846     __CLPK_doublecomplex *beta, __CLPK_doublecomplex *y, __CLPK_integer *incy);
4847
4848 /* Subroutine */ int zsyr_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *alpha,
4849     __CLPK_doublecomplex *x, __CLPK_integer *incx, __CLPK_doublecomplex *a, __CLPK_integer *lda);
4850
4851 /* Subroutine */ int zsyrfs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4852     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *af, __CLPK_integer *ldaf,
4853     __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x,
4854     __CLPK_integer *ldx, __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *work,
4855     __CLPK_doublereal *rwork, __CLPK_integer *info);
4856
4857 /* Subroutine */ int zsssv_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4858     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *b,
4859     __CLPK_integer *ldb, __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *info);
4860
4861 /* Subroutine */ int zsssvx_(char *fact, char *uplo, __CLPK_integer *n, __CLPK_integer *
4862     nrhs, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *af, __CLPK_integer *
4863     ldaf, __CLPK_integer *ipiv, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x,
4864     __CLPK_integer *ldx, __CLPK_doublereal *rcond, __CLPK_doublereal *ferr, __CLPK_doublereal *berr,
4865     __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *info);
4866
4867 /* Subroutine */ int zsytf2_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4868     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_integer *info);
4869
4870 /* Subroutine */ int zsytrf_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4871     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
4872     __CLPK_integer *info);
4873
4874 /* Subroutine */ int zsytri_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
4875     __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *work, __CLPK_integer *info);
4876
4877 /* Subroutine */ int zsytrs_(char *uplo, __CLPK_integer *n, __CLPK_integer *nrhs,
4878     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *ipiv, __CLPK_doublecomplex *b,
4879     __CLPK_integer *ldb, __CLPK_integer *info);
4880
4881 /* Subroutine */ int ztbcon_(char *norm, char *uplo, char *diag, __CLPK_integer *n,
4882     __CLPK_integer *kd, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *rcond,
4883     __CLPK_doublecomplex *work, __CLPK_doublereal *rwork, __CLPK_integer *info);
4884
4885 /* Subroutine */ int ztbrfs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
4886     __CLPK_integer *kd, __CLPK_integer *nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab,
4887     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx,
4888     __CLPK_doublereal *ferr, __CLPK_doublereal *berr, __CLPK_doublecomplex *work, __CLPK_doublereal *
4889     rwork, __CLPK_integer *info);
4890
4891 /* Subroutine */ int ztbtrs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
4892     __CLPK_integer *kd, __CLPK_integer *nrhs, __CLPK_doublecomplex *ab, __CLPK_integer *ldab,
4893     __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_integer *info);
4894

```

```
4895 /* Subroutine */ int ztgevc_(char *side, char *howmny, __CLPK_logical *select,
4896   __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
   __CLPK_integer
4897   *ldb, __CLPK_doublecomplex *vl, __CLPK_integer *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *
4898   ldvr, __CLPK_integer *mm, __CLPK_integer *m, __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork,
4899   __CLPK_integer *info);
4900
4901 /* Subroutine */ int ztgex2_(__CLPK_logical *wantq, __CLPK_logical *wantz, __CLPK_integer *n,
4902   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4903   __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_doublecomplex *z__, __CLPK_integer *ldz,
4904   __CLPK_integer *j1, __CLPK_integer *info);
4905
4906 /* Subroutine */ int ztgexc_(__CLPK_logical *wantq, __CLPK_logical *wantz, __CLPK_integer *n,
4907   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4908   __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_doublecomplex *z__, __CLPK_integer *ldz,
4909   __CLPK_integer *ifst, __CLPK_integer *ilst, __CLPK_integer *info);
4910
4911 /* Subroutine */ int ztgsen_(__CLPK_integer *ijob, __CLPK_logical *wantq, __CLPK_logical *wantz,
4912   __CLPK_logical *select, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
4913   __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *
4914   beta, __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_doublecomplex *z__, __CLPK_integer *
4915   ldz, __CLPK_integer *m, __CLPK_doublecomplex *pl, __CLPK_doublecomplex *pr, __CLPK_doublecomplex *dif,
4916   __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork,
4917   __CLPK_integer *info);
4918
4919 /* Subroutine */ int ztgsja_(char *jobu, char *jobv, char *jobq, __CLPK_integer *m,
4920   __CLPK_integer *p, __CLPK_integer *n, __CLPK_integer *k, __CLPK_integer *l, __CLPK_doublecomplex
   *a,
4921   __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb, __CLPK_doublecomplex *tola,
4922   __CLPK_doublecomplex *tolb, __CLPK_doublecomplex *alpha, __CLPK_doublecomplex *beta, __CLPK_doublecomplex *
4923   u, __CLPK_integer *ldu, __CLPK_doublecomplex *v, __CLPK_integer *ldv, __CLPK_doublecomplex *q,
4924   __CLPK_integer *ldq, __CLPK_doublecomplex *work, __CLPK_integer *ncycle, __CLPK_integer *info);
4925
4926 /* Subroutine */ int ztgsna_(char *job, char *howmny, __CLPK_logical *select,
4927   __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
   __CLPK_integer
4928   *ldb, __CLPK_doublecomplex *vl, __CLPK_integer *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *
4929   ldvr, __CLPK_doublecomplex *s, __CLPK_doublecomplex *dif, __CLPK_integer *mm, __CLPK_integer *m,
4930   __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *info);
4931
4932 /* Subroutine */ int ztgsy2_(char *trans, __CLPK_integer *ijob, __CLPK_integer *m, __CLPK_integer *
4933   n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4934   __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *d__, __CLPK_integer *ldd,
4935   __CLPK_doublecomplex *e, __CLPK_integer *lde, __CLPK_doublecomplex *f, __CLPK_integer *ldf,
4936   __CLPK_doublecomplex *scale, __CLPK_doublecomplex *rdsum, __CLPK_doublecomplex *rdscal, __CLPK_integer *
4937   info);
4938
4939 /* Subroutine */ int ztgsyl_(char *trans, __CLPK_integer *ijob, __CLPK_integer *m, __CLPK_integer *
4940   n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4941   __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *d__, __CLPK_integer *ldd,
4942   __CLPK_doublecomplex *e, __CLPK_integer *lde, __CLPK_doublecomplex *f, __CLPK_integer *ldf,
4943   __CLPK_doublecomplex *scale, __CLPK_doublecomplex *dif, __CLPK_doublecomplex *work, __CLPK_integer *
4944   lwork, __CLPK_integer *iwork, __CLPK_integer *info);
4945
4946 /* Subroutine */ int ztpcon_(char *norm, char *uplo, char *diag, __CLPK_integer *n,
4947   __CLPK_doublecomplex *ap, __CLPK_doublecomplex *rcond, __CLPK_doublecomplex *work, __CLPK_doublecomplex
   *rwork,
4948   __CLPK_integer *info);
4949
4950 /* Subroutine */ int ztprfs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
4951   __CLPK_integer *nrhs, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4952   __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublecomplex *ferr, __CLPK_doublecomplex *berr,
4953   __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4954
4955 /* Subroutine */ int ztptri_(char *uplo, char *diag, __CLPK_integer *n,
4956   __CLPK_doublecomplex *ap, __CLPK_integer *info);
4957
4958 /* Subroutine */ int ztptrs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
4959   __CLPK_integer *nrhs, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
4960   __CLPK_integer *info);
4961
4962 /* Subroutine */ int ztrcon_(char *norm, char *uplo, char *diag, __CLPK_integer *n,
4963   __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *rcond, __CLPK_doublecomplex *
4964   work, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4965
4966 /* Subroutine */ int ztrevc_(char *side, char *howmny, __CLPK_logical *select,
4967   __CLPK_integer *n, __CLPK_doublecomplex *t, __CLPK_integer *ldt, __CLPK_doublecomplex *vl,
4968   __CLPK_integer *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *ldvr, __CLPK_integer *mm,
   __CLPK_integer
4969   *m, __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4970
4971 /* Subroutine */ int ztrexc_(char *compq, __CLPK_integer *n, __CLPK_doublecomplex *t,
4972   __CLPK_integer *ldt, __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_integer *ifst,
   __CLPK_integer *
4973   ilst, __CLPK_integer *info);
4974
4975 /* Subroutine */ int ztrrfs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
4976   __CLPK_integer *nrhs, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
```

```
4977     __CLPK_integer *ldb, __CLPK_doublecomplex *x, __CLPK_integer *ldx, __CLPK_doublecomplex *ferr,
4978     __CLPK_doublecomplex *berr, __CLPK_doublecomplex *work, __CLPK_doublecomplex *rwork, __CLPK_integer *
4979     info);
4980
4981 /* Subroutine */ int ztrsen_(char *job, char *compq, __CLPK_logical *select, __CLPK_integer
4982     *n, __CLPK_doublecomplex *t, __CLPK_integer *ldt, __CLPK_doublecomplex *q, __CLPK_integer *ldq,
4983     __CLPK_doublecomplex *w, __CLPK_integer *m, __CLPK_doublecomplex *s, __CLPK_doublecomplex *sep,
4984     __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_integer *info);
4985
4986 /* Subroutine */ int ztrsna_(char *job, char *howmny, __CLPK_logical *select,
4987     __CLPK_integer *n, __CLPK_doublecomplex *t, __CLPK_integer *ldt, __CLPK_doublecomplex *vl,
4988     __CLPK_integer *ldvl, __CLPK_doublecomplex *vr, __CLPK_integer *ldvr, __CLPK_doublecomplex *s,
4989     __CLPK_doublecomplex *sep, __CLPK_integer *mm, __CLPK_integer *m, __CLPK_doublecomplex *work,
4990     __CLPK_integer *ldwork, __CLPK_doublecomplex *rwork, __CLPK_integer *info);
4991
4992 /* Subroutine */ int ztrsyl_(char *trana, char *tranb, __CLPK_integer *isgn, __CLPK_integer
4993     *m, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
4994     __CLPK_integer *ldb, __CLPK_doublecomplex *c, __CLPK_integer *ldc, __CLPK_doublecomplex *scale,
4995     __CLPK_integer *info);
4996
4997 /* Subroutine */ int ztrti2_(char *uplo, char *diag, __CLPK_integer *n,
4998     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *info);
4999
5000 /* Subroutine */ int ztrtri_(char *uplo, char *diag, __CLPK_integer *n,
5001     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_integer *info);
5002
5003 /* Subroutine */ int ztrtrs_(char *uplo, char *trans, char *diag, __CLPK_integer *n,
5004     __CLPK_integer *nrhs, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b,
5005     __CLPK_integer *ldb, __CLPK_integer *info);
5006
5007 /* Subroutine */ int ztzrqf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
5008     __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_integer *info);
5009
5010 /* Subroutine */ int ztzrzf_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
5011     __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
5012     __CLPK_integer *info);
5013
5014 /* Subroutine */ int zung2l_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5015     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5016     work, __CLPK_integer *lwork, __CLPK_integer *info);
5017
5018 /* Subroutine */ int zung2r_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5019     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5020     work, __CLPK_integer *lwork, __CLPK_integer *info);
5021
5022 /* Subroutine */ int zungbr_(char *vect, __CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5023     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5024     work, __CLPK_integer *lwork, __CLPK_integer *info);
5025
5026 /* Subroutine */ int zunghr_(__CLPK_integer *n, __CLPK_integer *ilo, __CLPK_integer *ihi,
5027     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5028     work, __CLPK_integer *lwork, __CLPK_integer *info);
5029
5030 /* Subroutine */ int zungl2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5031     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5032     work, __CLPK_integer *lwork, __CLPK_integer *info);
5033
5034 /* Subroutine */ int zunglq_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5035     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5036     work, __CLPK_integer *lwork, __CLPK_integer *info);
5037
5038 /* Subroutine */ int zungql_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5039     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5040     work, __CLPK_integer *lwork, __CLPK_integer *info);
5041
5042 /* Subroutine */ int zungqr_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5043     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5044     work, __CLPK_integer *lwork, __CLPK_integer *info);
5045
5046 /* Subroutine */ int zungr2_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5047     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5048     work, __CLPK_integer *lwork, __CLPK_integer *info);
5049
5050 /* Subroutine */ int zungrq_(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *k,
5051     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *
5052     work, __CLPK_integer *lwork, __CLPK_integer *info);
5053
5054 /* Subroutine */ int zungtr_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
5055     __CLPK_integer *lda, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
5056     __CLPK_integer *info);
5057
5058 /* Subroutine */ int zunm2l_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5059     __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
5060     __CLPK_doublecomplex *c, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *info);
5061
5062 /* Subroutine */ int zunm2r_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5063     __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
```

```

5064     __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *info);
5065
5066 /* Subroutine */ int zunmbr_(char *vect, char *side, char *trans, __CLPK_integer *m,
5067     __CLPK_integer *n, __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda,
5068     __CLPK_doublecomplex
5069     *tau, __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *
5070     lwork, __CLPK_integer *info);
5071
5072 /* Subroutine */ int zunmhr_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5073     __CLPK_integer *ilo, __CLPK_integer *ihi, __CLPK_doublecomplex *a, __CLPK_integer *lda,
5074     __CLPK_doublecomplex *tau, __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *
5075     work, __CLPK_integer *lwork, __CLPK_integer *info);
5076
5077 /* Subroutine */ int zunml2_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5078     __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
5079     __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *info);
5080
5081 /* Subroutine */ int zunmlq_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5082     __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
5083     __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
5084     __CLPK_integer *info);
5085
5086 /* Subroutine */ int zunmql_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5087     __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
5088     __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
5089     __CLPK_integer *info);
5090
5091 /* Subroutine */ int zunmqr_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5092     __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
5093     __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
5094     __CLPK_integer *info);
5095
5096 /* Subroutine */ int zunmr2_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5097     __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
5098     __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *info);
5099
5100 /* Subroutine */ int zunmr3_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5101     __CLPK_integer *k, __CLPK_integer *l, __CLPK_doublecomplex *a, __CLPK_integer *lda,
5102     __CLPK_doublecomplex
5103     *tau, __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *
5104     info);
5105
5106 /* Subroutine */ int zunmrq_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5107     __CLPK_integer *k, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
5108     __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
5109     __CLPK_integer *info);
5110
5111 /* Subroutine */ int zunmrz_(char *side, char *trans, __CLPK_integer *m, __CLPK_integer *n,
5112     __CLPK_integer *k, __CLPK_integer *l, __CLPK_doublecomplex *a, __CLPK_integer *lda,
5113     __CLPK_doublecomplex
5114     *tau, __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *
5115     lwork, __CLPK_integer *info);
5116
5117 /* Subroutine */ int zunmtr_(char *side, char *uplo, char *trans, __CLPK_integer *m,
5118     __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *tau,
5119     __CLPK_doublecomplex *c__, __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *lwork,
5120     __CLPK_integer *info);
5121
5122 /* Subroutine */ int zupgtr_(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
5123     __CLPK_doublecomplex *tau, __CLPK_doublecomplex *q, __CLPK_integer *ldq, __CLPK_doublecomplex *
5124     work, __CLPK_integer *info);
5125
5126 /* Subroutine */ int zupmtr_(char *side, char *uplo, char *trans, __CLPK_integer *m,
5127     __CLPK_integer *n, __CLPK_doublecomplex *ap, __CLPK_doublecomplex *tau, __CLPK_doublecomplex *c__,
5128     __CLPK_integer *ldc, __CLPK_doublecomplex *work, __CLPK_integer *info);
5129
5130 /*
5131 The following prototypes are not present in the reference clapack.h distributed via netlib.
5132 Nevertheless, the supporting code has always been present in the library.
5133 */
5134
5135 /* Subroutine */ int cgelsd__(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex
5136 *
5137 * a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_real *s, __CLPK_real *rcond,
5138 *
5139 * __CLPK_integer *rank, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork,
5140 *
5141 * __CLPK_integer *
5142 * iwork, __CLPK_integer *info);
5143
5144 /* Subroutine */ int cgelss__(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs, __CLPK_complex
5145 *
5146 * a, __CLPK_integer *lda, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_real *s, __CLPK_real *rcond,
5147 *
5148 * __CLPK_integer *rank, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork,
5149 *
5150 * __CLPK_integer *
5151 * info);

```

```

5142 /* Subroutine */ int cgesdd(char *jobz, __CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a,
5143   __CLPK_integer *lda, __CLPK_real *s, __CLPK_complex *u, __CLPK_integer *ldu, __CLPK_complex *vt,
   __CLPK_integer
5144   *ldvt, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork, __CLPK_integer *iwork,
5145   __CLPK_integer *info);
5146
5147 /* Subroutine */ int cgesvd(char *jobu, char *jobvt, __CLPK_integer *m, __CLPK_integer *n,
5148   __CLPK_complex *a, __CLPK_integer *lda, __CLPK_real *s, __CLPK_complex *u, __CLPK_integer *ldu,
   __CLPK_complex *
5149   vt, __CLPK_integer *ldvt, __CLPK_complex *work, __CLPK_integer *lwork, __CLPK_real *rwork,
5150   __CLPK_integer *info);
5151
5152 /* Subroutine */ int chbgvd(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
5153   __CLPK_integer *kb, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_complex *bb, __CLPK_integer
   *ldb,
5154   __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_complex *work, __CLPK_integer
   *lwork,
5155   __CLPK_real *rwork, __CLPK_integer *lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork,
5156   __CLPK_integer *info);
5157
5158 /* Subroutine */ int chetd2(char *uplo, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
5159   __CLPK_real *d__, __CLPK_real *e, __CLPK_complex *tau, __CLPK_integer *info);
5160
5161 /* Complex */ void cladv__(__CLPK_complex *ret_val, __CLPK_complex *x, __CLPK_complex *y);
5162
5163 /* Subroutine */ int clalsd(char *uplo, __CLPK_integer *smlsiz, __CLPK_integer *n, __CLPK_integer
   *nrhs,
5164   __CLPK_real *d__, __CLPK_real *e, __CLPK_complex *b, __CLPK_integer *ldb, __CLPK_real
   *rcond,
5165   __CLPK_integer *rank, __CLPK_complex *work, __CLPK_real *rwork, __CLPK_integer *iwork,
5166   __CLPK_integer *info);
5167
5168 __CLPK_double real clangb(char *norm, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku,
5169   __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *work);
5170
5171 __CLPK_double real clangs(char *norm, __CLPK_integer *m, __CLPK_integer *n, __CLPK_complex *a,
5172   __CLPK_integer *lda, __CLPK_real *work);
5173
5174 __CLPK_double real clangt(char *norm, __CLPK_integer *n, __CLPK_complex *dl, __CLPK_complex *d__,
5175   __CLPK_complex *du);
5176
5177 __CLPK_double real clanhb(char *norm, char *uplo, __CLPK_integer *n, __CLPK_integer *k,
5178   __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *work);
5179
5180 __CLPK_double real clanhe(char *norm, char *uplo, __CLPK_integer *n, __CLPK_complex *a,
5181   __CLPK_integer *lda, __CLPK_real *work);
5182
5183 __CLPK_double real clanhp(char *norm, char *uplo, __CLPK_integer *n, __CLPK_complex *ap,
5184   __CLPK_real *work);
5185
5186 __CLPK_double real clanhs(char *norm, __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda,
5187   __CLPK_real *work);
5188
5189 __CLPK_double real clanht(char *norm, __CLPK_integer *n, __CLPK_real *d__, __CLPK_complex *e);
5190
5191 __CLPK_double real clansb(char *norm, char *uplo, __CLPK_integer *n, __CLPK_integer *k,
5192   __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *work);
5193
5194 __CLPK_double real clansp(char *norm, char *uplo, __CLPK_integer *n, __CLPK_complex *ap,
5195   __CLPK_real *work);
5196
5197 __CLPK_double real clansy(char *norm, char *uplo, __CLPK_integer *n, __CLPK_complex *a,
5198   __CLPK_integer *lda, __CLPK_real *work);
5199
5200 __CLPK_double real clantb(char *norm, char *uplo, char *diag, __CLPK_integer *n,
5201   __CLPK_integer *k, __CLPK_complex *ab, __CLPK_integer *ldab, __CLPK_real *work);
5202
5203 __CLPK_double real clantp(char *norm, char *uplo, char *diag, __CLPK_integer *n,
5204   __CLPK_complex *ap, __CLPK_real *work);
5205
5206 __CLPK_double real clantr(char *norm, char *uplo, char *diag, __CLPK_integer *m,
5207   __CLPK_integer *n, __CLPK_complex *a, __CLPK_integer *lda, __CLPK_real *work);
5208
5209 /* Subroutine */ int cpteqr(char *compz, __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e,
5210   __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_real *work, __CLPK_integer *info);
5211
5212 /* Subroutine */ int cstegr(char *jobz, char *range, __CLPK_integer *n, __CLPK_real *d__,
5213   __CLPK_real *e, __CLPK_real *vl, __CLPK_real *vu, __CLPK_integer *il, __CLPK_integer *iu,
   __CLPK_real *abstol,
5214   __CLPK_integer *m, __CLPK_real *w, __CLPK_complex *z__, __CLPK_integer *ldz, __CLPK_integer
   *isuppz,
5215   __CLPK_real *work, __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork,
5216   __CLPK_integer *info);
5217
5218 __CLPK_double real dlamc3(__CLPK_double real *a, __CLPK_double real *b);
5219
5220 __CLPK_double real dlamch(char *cmach);
5221

```



```
5222 __CLPK_doublereal dlangb_(char *norm, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku,
5223     __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *work);
5224
5225 __CLPK_doublereal dlange_(char *norm, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublereal *a,
5226     __CLPK_integer *lda, __CLPK_doublereal *work);
5227
5228 __CLPK_doublereal dlangt_(char *norm, __CLPK_integer *n, __CLPK_doublereal *dl, __CLPK_doublereal *d__,
5229     __CLPK_doublereal *du);
5230
5231 __CLPK_doublereal dlanhs_(char *norm, __CLPK_integer *n, __CLPK_doublereal *a, __CLPK_integer *lda,
5232     __CLPK_doublereal *work);
5233
5234 __CLPK_doublereal dlansb_(char *norm, char *uplo, __CLPK_integer *n, __CLPK_integer *k,
5235     __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *work);
5236
5237 __CLPK_doublereal dlansp_(char *norm, char *uplo, __CLPK_integer *n, __CLPK_doublereal *ap,
5238     __CLPK_doublereal *work);
5239
5240 __CLPK_doublereal dlanst_(char *norm, __CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublereal *e);
5241
5242 __CLPK_doublereal dlansy_(char *norm, char *uplo, __CLPK_integer *n, __CLPK_doublereal *a,
5243     __CLPK_integer *lda, __CLPK_doublereal *work);
5244
5245 __CLPK_doublereal dlantb_(char *norm, char *uplo, char *diag, __CLPK_integer *n, __CLPK_integer *k,
5246     __CLPK_doublereal *ab, __CLPK_integer *ldab, __CLPK_doublereal *work);
5247
5248 __CLPK_doublereal dlantp_(char *norm, char *uplo, char *diag, __CLPK_integer *n,
5249     __CLPK_doublereal *ap, __CLPK_doublereal *work);
5250
5251 __CLPK_doublereal dlantr_(char *norm, char *uplo, char *diag, __CLPK_integer *m, __CLPK_integer *n,
5252     __CLPK_doublereal *a, __CLPK_integer *lda, __CLPK_doublereal *work);
5253
5254 __CLPK_doublereal dlapy2_(__CLPK_doublereal *x, __CLPK_doublereal *y);
5255
5256 __CLPK_doublereal dlapy3_(__CLPK_doublereal *x, __CLPK_doublereal *y, __CLPK_doublereal *z__);
5257
5258 __CLPK_doublereal dsecnd_();
5259
5260 __CLPK_doublereal dzsum1_(__CLPK_integer *n, __CLPK_doublecomplex *cx, __CLPK_integer *incx);
5261
5262 __CLPK_logical lsame_(char *ca, char *cb);
5263
5264 __CLPK_logical lsamen_(__CLPK_integer *n, char *ca, char *cb);
5265
5266 __CLPK_doublereal ssum1_(__CLPK_integer *n, __CLPK_complex *cx, __CLPK_integer *incx);
5267
5268 __CLPK_doublereal second_();
5269
5270 __CLPK_doublereal slamc3_(__CLPK_real *a, __CLPK_real *b);
5271
5272 __CLPK_doublereal slamch_(char *cmach);
5273
5274 __CLPK_doublereal slangb_(char *norm, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku,
5275     __CLPK_real *ab,
5276     __CLPK_integer *ldab, __CLPK_real *work);
5277
5278 __CLPK_doublereal slange_(char *norm, __CLPK_integer *m, __CLPK_integer *n, __CLPK_real *a,
5279     __CLPK_integer *lda,
5280     __CLPK_real *work);
5281
5282 __CLPK_doublereal slangt_(char *norm, __CLPK_integer *n, __CLPK_real *dl, __CLPK_real *d__, __CLPK_real
5283     *du);
5284
5285 __CLPK_doublereal slanhs_(char *norm, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer *lda,
5286     __CLPK_real *work);
5287
5288 __CLPK_doublereal slansb_(char *norm, char *uplo, __CLPK_integer *n, __CLPK_integer *k, __CLPK_real
5289     *ab,
5290     __CLPK_integer *ldab, __CLPK_real *work);
5291
5292 __CLPK_doublereal slansp_(char *norm, char *uplo, __CLPK_integer *n, __CLPK_real *ap, __CLPK_real
5293     *work);
5294
5295 __CLPK_doublereal slanst_(char *norm, __CLPK_integer *n, __CLPK_real *d__, __CLPK_real *e);
5296
5297 __CLPK_doublereal slansy_(char *norm, char *uplo, __CLPK_integer *n, __CLPK_real *a, __CLPK_integer
5298     *lda,
5299     __CLPK_real *work);
5300
5301 __CLPK_doublereal slantb_(char *norm, char *uplo, char *diag, __CLPK_integer *n, __CLPK_integer *k,
5302     __CLPK_real *ab, __CLPK_integer *ldab, __CLPK_real *work);
5303
5304 __CLPK_doublereal slantp_(char *norm, char *uplo, char *diag, __CLPK_integer *n, __CLPK_real *ap,
5305     __CLPK_real *work);
5306
5307 __CLPK_doublereal slantr_(char *norm, char *uplo, char *diag, __CLPK_integer *m, __CLPK_integer *n,
```

```

5301     __CLPK_real *a, __CLPK_integer *lda, __CLPK_real *work);
5302
5303 __CLPK_doublereal slapy2(__CLPK_real *x, __CLPK_real *y);
5304
5305 __CLPK_doublereal slapy3(__CLPK_real *x, __CLPK_real *y, __CLPK_real *z__);
5306
5307 /* Subroutine */ int zgelsd(__CLPK_integer *m, __CLPK_integer *n, __CLPK_integer *nrhs,
5308     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
5309     __CLPK_doublereal *s, __CLPK_doublereal *rcond, __CLPK_integer *rank, __CLPK_doublecomplex *work,
5310     __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *iwork, __CLPK_integer *info);
5311
5312 /* Subroutine */ int zgesdd(char *jobz, __CLPK_integer *m, __CLPK_integer *n,
5313     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublereal *s, __CLPK_doublecomplex *u,
5314     __CLPK_integer *ldu, __CLPK_doublecomplex *vt, __CLPK_integer *ldvt, __CLPK_doublecomplex *work,
5315     __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *iwork, __CLPK_integer *info);
5316
5317 /* Subroutine */ int zgesvd(char *jobu, char *jobvt, __CLPK_integer *m, __CLPK_integer *n,
5318     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublereal *s, __CLPK_doublecomplex *u,
5319     __CLPK_integer *ldu, __CLPK_doublecomplex *vt, __CLPK_integer *ldvt, __CLPK_doublecomplex *work,
5320     __CLPK_integer *lwork, __CLPK_doublereal *rwork, __CLPK_integer *info);
5321
5322 /* Subroutine */ int zhbgsd(char *jobz, char *uplo, __CLPK_integer *n, __CLPK_integer *ka,
5323     __CLPK_integer *kb, __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublecomplex *bb,
5324     __CLPK_integer *ldb, __CLPK_doublereal *w, __CLPK_doublecomplex *z__, __CLPK_integer *ldz,
5325     __CLPK_doublecomplex *work, __CLPK_integer *lwork, __CLPK_doublereal *rwork,
5326     __CLPK_integer *lrwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
5327
5328 /* Subroutine */ int zhetd2(char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
5329     __CLPK_integer *lda, __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublecomplex *tau,
5330     __CLPK_integer *info);
5331
5332 /* Double Complex */ void zladiv(__CLPK_doublecomplex *ret_val, __CLPK_doublecomplex *x,
5333     __CLPK_doublecomplex *y);
5334
5335 /* Subroutine */ int zlalsd(char *uplo, __CLPK_integer *smlsiz, __CLPK_integer *n, __CLPK_integer
5336     *nrhs, __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublecomplex *b, __CLPK_integer *ldb,
5337     __CLPK_doublereal *rcond, __CLPK_integer *rank, __CLPK_doublecomplex *work,
5338     __CLPK_doublereal *rwork, __CLPK_integer *iwork, __CLPK_integer *info);
5339
5340 __CLPK_doublereal zlangb(char *norm, __CLPK_integer *n, __CLPK_integer *kl, __CLPK_integer *ku,
5341     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *work);
5342
5343 __CLPK_doublereal zlange(char *norm, __CLPK_integer *m, __CLPK_integer *n, __CLPK_doublecomplex *a,
5344     __CLPK_integer *lda, __CLPK_doublereal *work);
5345
5346 __CLPK_doublereal zlangt(char *norm, __CLPK_integer *n, __CLPK_doublecomplex *dl,
5347     __CLPK_doublecomplex *d__, __CLPK_doublecomplex *du);
5348
5349 __CLPK_doublereal zlanhb(char *norm, char *uplo, __CLPK_integer *n, __CLPK_integer *k,
5350     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *work);
5351
5352 __CLPK_doublereal zlanhe(char *norm, char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
5353     __CLPK_integer *lda, __CLPK_doublereal *work);
5354
5355 __CLPK_doublereal zlanhp(char *norm, char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
5356     __CLPK_doublereal *work);
5357
5358 __CLPK_doublereal zlanhs(char *norm, __CLPK_integer *n, __CLPK_doublecomplex *a, __CLPK_integer *lda,
5359     __CLPK_doublereal *work);
5360
5361 __CLPK_doublereal zlanht(char *norm, __CLPK_integer *n, __CLPK_doublereal *d__, __CLPK_doublecomplex
5362     *e);
5363
5364 __CLPK_doublereal zlabsb(char *norm, char *uplo, __CLPK_integer *n, __CLPK_integer *k,
5365     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *work);
5366
5367 __CLPK_doublereal zlansp(char *norm, char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *ap,
5368     __CLPK_doublereal *work);
5369
5370 __CLPK_doublereal zlansy(char *norm, char *uplo, __CLPK_integer *n, __CLPK_doublecomplex *a,
5371     __CLPK_integer *lda, __CLPK_doublereal *work);
5372
5373 __CLPK_doublereal zlantb(char *norm, char *uplo, char *diag, __CLPK_integer *n, __CLPK_integer *k,
5374     __CLPK_doublecomplex *ab, __CLPK_integer *ldab, __CLPK_doublereal *work);
5375
5376 __CLPK_doublereal zlantp(char *norm, char *uplo, char *diag, __CLPK_integer *n,
5377     __CLPK_doublecomplex *ap, __CLPK_doublereal *work);
5378
5379 __CLPK_doublereal zlantr(char *norm, char *uplo, char *diag, __CLPK_integer *m, __CLPK_integer *n,
5380     __CLPK_doublecomplex *a, __CLPK_integer *lda, __CLPK_doublereal *work);
5381
5382 /* Subroutine */ int zpqr(char *compz, __CLPK_integer *n, __CLPK_doublereal *d__,
5383     __CLPK_doublereal *e, __CLPK_doublecomplex *z__, __CLPK_integer *ldz, __CLPK_doublereal *work,
5384     __CLPK_integer *info);
5385
5386 /* Subroutine */ int zstegr(char *jobz, char *range, __CLPK_integer *n,
5387     __CLPK_doublereal *d__, __CLPK_doublereal *e, __CLPK_doublereal *vl, __CLPK_doublereal *vu,

```



```

5387     __CLPK_integer *il, __CLPK_integer *iu, __CLPK_doublereal *abstol, __CLPK_integer *m,
        __CLPK_doublereal *w,
5388     __CLPK_doublecomplex *z__, __CLPK_integer *ldz, __CLPK_integer *isuppz, __CLPK_doublereal *work,
5389     __CLPK_integer *lwork, __CLPK_integer *iwork, __CLPK_integer *liwork, __CLPK_integer *info);
5390
5391 #ifdef __cplusplus
5392 }
5393 #endif
5394 #endif /* __CLAPACK_H */

```

## 7.383 MatLapack.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           28/03/2005
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *   *****
38 *   BUT:            Utilisation de routines lapack dans le cas de matrices
39 *                   rectangulairesn, carrées ou bandes, symétriques ou pas.
40 *                   Les coefficients sont des doubles.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *   ! date ! auteur ! but
46 *   -----
47 *   ! ! !
48 *
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date ! auteur ! but
52 *   -----
53 *
54 *   *****/
55
56 #ifndef MatLapack_H
57 #define MatLapack_H
58
59 // au niveau de la bibliothèque apple
60 // ./System/Library/Frameworks/vecLib.framework/Versions/A/Headers/clapack.h
61 //
62 // ./System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/Headers/clapack.h
63 // #include "Debug.h"
64 #ifdef SYSTEM_MAC_OS_CARBON
65 #include "LAPACK.h"
66 #endif
67 #ifdef ENLINUX_2009
68 // #include <f2c.h> // on supprime l'inclusion, et on la remplace par une introduction dans le debut de
69 // ceci pour la version linux !!
70 // #include "INCLUDE/clapack.h"

```

```

71 #include "Include_lapack/clapack.h"
72 #include "cblas.h"
73 typedef __CLPK_integer ENTIER_POUR_CLAPACK;
74 // typedef long int ENTIER_POUR_CLAPACK;
75 // on n'arrive pas à faire un include en linux ??
76 // mais l'option de mise au point est supprimée à la fin du matlabpack.cc
77 // #define MISE_AU_POINT
78 #else
79 #ifdef SYSTEM_MAC_OS_X_unix
80 // #include </System/Library/Frameworks/vecLib.framework/Headers/clapack.h>
81 // #include "Headers/clapack.h"
82 #include <Accelerate.h>
83
84 // #include
85 // </System/Library/Frameworks/Accelerate.framework/Headers/clapack.h>
86 #endif
87 #endif
88
89 #ifdef SYSTEM_MAC_OS_X
90 #include <Versions/A/Headers/clapack.h>
91 typedef __CLPK_integer ENTIER_POUR_CLAPACK;
92 #endif
93
94 #include "Mat_abstraite.h"
95
96 /// @addtogroup Les_classes_Matrices
97 /// @{
98 ///
99
100
101 class MatLapack : public Mat_abstraite
102 { // surcharge de l'opérateur de lecture typée
103     friend istream & operator » (istream &, MatLapack &);
104     // surcharge de l'opérateur d'écriture typée
105     friend ostream & operator « (ostream &, const MatLapack &);
106
107     public :
108
109     // Constructeur par défaut
110     // il est nécessaire ensuite de définir les données pour l'utiliser
111     MatLapack ( );
112
113     // Constructeur pour une matrice carree quelconque, ou symétrique
114     // se servant d'un nombre de lignes, de = colonnes, et eventuellement
115     // d'une valeur d'initialisation
116     MatLapack (int nb_ligne,int nb_colonne, bool symetrique , double val_init
117         ,Enum_type_resolution_matri type_resol // =RIEN_TYPE_RESOLUTION_MATRI
118         ,Enum_preconditionnement type_precondi // = RIEN_PRECONDITIONNEMENT
119         );
120
121     // Constructeur pour une matrice bande quelconque, ou symétrique
122     // se servant d'un nombre de lignes (dim) , d'une largeur de bande, et eventuellement
123     // d'une valeur d'initialisation et d'un type de résolution
124     MatLapack (Enum_matrice enu_mat, int lb_totale,int dim , bool symetrique, double val_init // =0.0
125         ,Enum_type_resolution_matri type_resol // =GAUSS
126         ,Enum_preconditionnement type_precondi // = RIEN_PRECONDITIONNEMENT
127         );
128
129     // de copie
130     MatLapack (const MatLapack& a);
131
132     // DESTRUCTEUR VIRTUEL :
133
134     ~MatLapack ();
135
136
137     // METHODES DÉCOULANT DE VIRTUELLES PURES :
138
139     // surcharge de l'opérateur d'affectation
140     // IMPORTANT : le fonctionnement de l'implantation dépend de la classe
141     // dérivée : en particulier il peut y avoir redimensionnement automatique
142     // de la matrice en fonction de l'attribut, cas par exemple des matrices
143     // pleines, ou message d'erreur car les tailles ne sont pas identiques
144     // exemple de la classe mat_bande
145     // ici s'il s'agit de même matrices de même place, c'est ok, sinon erreur
146     MatLapack & operator = ( const MatLapack &);
147
148     // transfert des informations de *this dans la matrice passée en paramètre
149     // la matrice paramètre est au préalable, mise à 0.
150     void Transfert_vers_mat ( Mat_abstraite & A );
151
152     Mat_abstraite & operator = ( const Mat_abstraite &);
153
154     //-----
155     // --- plusieurs fonctions virtuelles qui agissent en général sur la matrice -----
156     //-----

```

```

157 // ici l'idée est d'éviter de construire une nouvelle matrice pour des questions
158 // d'encombrement, c'est surtout des méthodes utiles pour le stockage de matrice
159 // de raideur. On met le nombre mini de fonction, pour pouvoir faire des expressions.
160 // Cependant aucune de ces fonctions n'est en désaccord avec les fonctions membres
161 // qui crée une matrice en résultat (cf. Mat_pleine)
162 // pour les 2 premières méthodes : += et -= :
163 // les matrices arguments sont en général du mêmes type que celui de la matrice this
164 // mais cela fonctionne également avec comme argument une matrice diagonale (pour tous les types)
165
166 // Surcharge de l'opérateur += : addition d'une matrice a la matrice courante
167 void operator+= (const MatLapack& mat_pl);
168 void operator+= (const Mat_abstraite& mat_pl);
169
170 // Surcharge de l'opérateur -= : soustraction d'une matrice a la matrice courante
171 void operator-= (const MatLapack& mat_pl);
172 void operator-= (const Mat_abstraite& mat_pl);
173
174 // Surcharge de l'opérateur *= : multiplication de la matrice courante par un scalaire
175 void operator*= (const double r);
176
177 //-----
178 // --- fin de plusieurs fonctions virtuelles qui agissent en général sur la matrice ---
179 //-----
180
181 // Surcharge de l'opérateur == : test d'egalite entre deux matrices
182 int operator== (const Mat_abstraite& mat_pl) const ;
183 int operator== (const MatLapack& mat_pl) const ;
184
185 // fonction permettant de creer une nouvelle instance d'element
186 // dérivé. la nouvelle instance = *this, il y a donc utilisation du constructeur de copie
187 Mat_abstraite * NouvelElement() const;
188
189 // Affichage des valeurs de la matrice
190 void Affiche() const;
191 // Affiche une partie de la matrice (util pour le debug)
192 // min_ et max_ sont les bornes de la sous_matrice
193 // pas_ indique le pas en i et j pour les indices
194 void Affiche1(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j) const ;
195 // Affiche une partie de la matrice idem si dessus
196 // mais avec un nombre de digit (>7) = nd
197 // si < 7 ne fait rien
198 void Affiche2(int min_i,int max_i,int pas_i,int min_j,int max_j,int pas_j,int nd) const ;
199
200 // changement du choix de la méthode de résolution si c'est possible
201 // peut être surchargé par les classes dérivées en fonction des spécificités
202 void Change_Choix_resolution(Enum_type_resolution_matri type_resol,
203                             Enum_preconditionnement type_precondi);
204
205 // Initialisation des valeurs des composantes
206 void Initialise (double val_init);
207
208 // Liberation de la place memoire
209 void Libere ();
210
211 // Retour du nombre de lignes
212 int Nb_ligne () const {return nb_lign;};
213
214 // Retour du nombre de colonnes
215 int Nb_colonne () const {return nb_col;};
216
217 // Symetrie de la matrice
218 int Symetrie () const;
219
220 // Acces aux valeurs de la matrices en écriture
221 // i : indice de ligne, j : indice de colonne
222 double& operator () (int i, int j);
223
224 // Acces aux valeurs de la matrices en lecture
225 // cas ou l'on ne veut pas modifier les valeurs
226 // i : indice de ligne, j : indice de colonne
227 // !!! important, si l'élément n'existe pas du au type de stockage
228 // utilisé, (mais que les indices sont possible) le retour est 0
229 double operator () ( int i, int j) const;
230
231 // ramène true si la place (i,j) existe, false sinon
232 // ici on ne test pas le fait que i et j puissent être négatif ou pas
233 bool Existe(int i, int j) const;
234
235 // Retourne la ieme ligne de la matrice
236 // lorsque implemente ( a verifier )
237 Vecteur& Ligne_set(int i);
238
239 // Retourne la ieme ligne de la matrice
240 Vecteur Ligne(int i) const;
241
242 // Retourne la ieme ligne de la matrice
243 // sous le format de stokage propre a la matrice

```

```

244 // donc a n'utiliser que comme sauvegarde en parralele
245 // avec la fonction RemplaceLigne
246 Vecteur LigneSpe(int i) const;
247 // remplace la ligne de la matrice par la ligne fournie
248 void RemplaceLigneSpe(int i,const Vecteur & v);
249
250 //met une valeur identique sur toute la ligne
251 void MetValLigne(int i,double val);
252
253 // Retourne la jeme colonne de la matrice
254 Vecteur Colonne(int j) const;
255
256 // Retourne la jeme colonne de la matrice
257 // sous le format de stokage propre a la matrice
258 // donc a n'utiliser que comme sauvegarde en parralele
259 // avec la fonction RemplaceColonne
260 Vecteur ColonneSpe(int j) const;
261 // remplace la Colonne de la matrice par la colonne fournie
262 void RemplaceColonneSpe(int j,const Vecteur & v);
263
264 //met une valeur identique sur toute la colonne
265 void MetValColonne(int j,double val);
266
267 // Resolution du systeme Ax=b
268 //1) avec en sortie un new vecteur
269 Vecteur Resol_syst (const Vecteur& b,const double &tol = tol_defaut
270 ,const int maxit = maxit_defaut,const int restart = restart_defaut);
271 //2) avec en sortie le vecteur d'entree
272 Vecteur& Resol_systID (Vecteur& b,const double &tol = tol_defaut
273 ,const int maxit = maxit_defaut,const int restart = restart_defaut);
274 //3) avec en entrée un tableau de vecteur second membre et
275 // en sortie un nouveau tableau de vecteurs
276 Tableau <Vecteur> Resol_syst
277 (const Tableau <Vecteur>& b,const double &tol = tol_defaut
278 ,const int maxit = maxit_defaut,const int restart = restart_defaut);
279 //4) avec en entrée un tableau de vecteur second membre et
280 // en sortie le tableau de vecteurs d'entree
281 Tableau <Vecteur>& Resol_systID
282 (Tableau <Vecteur>& b,const double &tol = tol_defaut
283 ,const int maxit = maxit_defaut,const int restart = restart_defaut);
284 //5) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
285 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
286 // deux étant identiques
287 Vecteur& Resol_systID_2 (const Vecteur& b,Vecteur& vortie
288 ,const double &tol = tol_defaut
289 ,const int maxit = maxit_defaut
290 ,const int restart = restart_defaut);
291 // ===== RÉOLUTION EN DEUX TEMPS ===== :
292 // 1) préparation de la matrice donc modification de la matrice
293 // ici on calcul la factorisation LU de la matrice
294 void Preparation_resol() ;
295 // 2) *** résolution sans modification de la matrice DOIT ÊTRE PRÉCÉDÉ DE L'APPEL DE
296 // Preparation_resol
297 // a) avec en sortie un new vecteur
298 Vecteur Simple_Resol_syst (const Vecteur& b,const double &tol = tol_defaut
299 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
300 // b) avec en sortie le vecteur d'entree
301 Vecteur& Simple_Resol_systID (Vecteur& b,const double &tol = tol_defaut
302 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
303 // c) avec en entrée un tableau de vecteur second membre et
304 // en sortie un nouveau tableau de vecteurs
305 Tableau <Vecteur> Simple_Resol_syst
306 (const Tableau <Vecteur>& b,const double &tol = tol_defaut
307 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
308 // d) avec en entrée un tableau de vecteur second membre et
309 // en sortie le tableau de vecteurs d'entree
310 Tableau <Vecteur>& Simple_Resol_systID
311 (Tableau <Vecteur>& b,const double &tol = tol_defaut
312 ,const int maxit = maxit_defaut,const int restart = restart_defaut) const ;
313 // e) avec en sortie le dernier vecteur d'entree, le premier étant le second membre
314 // et restant inchangé, en sortie c'est donc soit le retour ou soit vortie, les
315 // deux étant identiques
316 Vecteur& Simple_Resol_systID_2 (const Vecteur& b,Vecteur& vortie
317 ,const double &tol = tol_defaut
318 ,const int maxit = maxit_defaut
319 ,const int restart = restart_defaut) const ;
320 // ===== FIN RÉOLUTION EN DEUX TEMPS ===== :
321
322 // Multiplication d'un vecteur par une matrice ( (vec)t * A )
323 Vecteur Prod_vec_mat ( const Vecteur& vec) const;
324 // idem mais on utilise la place du second vecteur pour le résultat
325 Vecteur& Prod_vec_mat ( const Vecteur& vec, Vecteur & resul) const;
326
327 // Multiplication d'une matrice par un vecteur ( A * vec )
328 Vecteur Prod_mat_vec ( const Vecteur& vec) const;
329 // idem mais on utilise la place du second vecteur pour le résultat
330 Vecteur& Prod_mat_vec ( const Vecteur& vec, Vecteur & resul) const;

```

```

331
332 // Multiplication d'une ligne iligne de la matrice avec un vecteur de
333 // dimension = le nombre de colonne de la matrice
334 double Prod_Ligne_vec ( int iligne,const Vecteur& vec) const;
335
336 // Multiplication d'un vecteur avec une colonne icol de la matrice
337 // dimension = le nombre de colonne de la matrice
338 double Prod_vec_col( int icol,const Vecteur& vec) const;
339
340 // calcul du produit : (vec_1)^T * A * (vect_2)
341 double vectT_mat_vec(const Vecteur& vec1, const Vecteur& vec2) const ;
342
343 // retourne la place que prend la matrice en entier
344 int Place() const;
345
346 // ----- méthode non virtuelle pur -> générales -----
347
348
349 //calcul des valeurs propres par la méthode des puissances itérées
350 // inverses.
351 // --> Utilisable que pour des matrices carrées
352 // Intéressant lorsque l'on veut les plus petites
353 // valeurs propres, en nombre restreind car seule les premières valeurs
354 // sont calculées avec une précision convenable. On obtient également
355 // les vecteurs propres correspondant.
356 // résolution de : ((*this) - lambda KG) X = 0
357 // en entrée : KG, VP dont la dimension est défini et fourni le nombre
358 // de valeurs propres que l'on veut calculer
359 // On considère que les conditions limites sont déjà appliquées à (*this) et KG.
360 // c'est à dire des 1 sur la diagonale pour (*this) et des 0 sur la diagonale pour KG
361 // en sortie : VP contiend les valeurs propres et vecteurs propres
362
363 Tableau <VeurPropre>* V_Propres(MatLapack& KG,Tableau <VeurPropre> & VP );
364
365 // ===== Méthodes spécifiques à la classe MatLapack =====
366
367 // changement de taille avec initialisation non nulle éventuelle
368 // si les tailles existantes sont identiques, on ne fait que initialiser
369 void Change_taille(int nb_li,int nb_col, const double& val_init = 0.);
370
371 // Surcharge de l'opérateur + : addition de deux matrices
372 // il y a création d'une matrice de sortie
373 MatLapack operator+ (const MatLapack& mat_pl) const;
374
375 // Surcharge de l'opérateur - : soustraction entre deux matrices
376 // il y a création d'une matrice de sortie
377 MatLapack operator- (const MatLapack& mat_pl) const;
378
379 // Surcharge de l'opérateur - : oppose d'une matrice
380 // il y a création d'une matrice de sortie
381 MatLapack operator- () const;
382
383 // Surcharge de l'opérateur * : multiplication de deux matrices
384 // il y a création d'une matrice de sortie
385 MatLapack operator* (const MatLapack& mat_pl) const;
386
387 // Surcharge de l'opérateur * : multiplication d'une matrice par un scalaire
388 // il y a création d'une matrice de sortie
389 MatLapack operator* (const double& coeff) const ;
390
391 // Retourne la transposee de la matrice mat
392 // il y a création d'une matrice de sortie
393 MatLapack Transpose () const;
394
395 // determine l'inverse d'une matrice carre
396 MatLapack Inverse();
397 // idem mais en retournant l'inverse dans la matrice passée en paramètre
398 // qui doit être de même taille
399 MatLapack& Inverse(MatLapack& mat) ;
400
401 protected :
402
403 // VARIABLES PROTEGEES :
404 //___CLPK_integer: long int en 32bit et int en 64bit, définit dans clapack.h
405 ENTIER_POUR_CLAPACK nb_lign,nb_col; // nombre de ligne, nombre de colonne utilisées par herezh
406 ENTIER_POUR_CLAPACK ldb; // nombre de ligne du stockage effectif
407 // --- dans le cas d'une matrice bande, nb_col est la largeur de bande totale
408 // dans le cas d'une matrice bande symétrique, nb_col=1/2largeur de bande+1
409 // on utilise des variables intermédiaires:
410 ENTIER_POUR_CLAPACK kl,ku,lda; // correspond à la largeur en dessous et au dessus de la diagonale
411 //lda: largeur maxi du premier indice, qu'utilise réellement lapack: = 2*kl+ku+1 en bande non
symétrique
412 // dans le cas d'une matrice rectangulaire = nb_lign
413
414
415 Vecteur mat; // la matrice vue comme un vecteur
416 // le stockage dépend du type de matrice considérée

```

```

417  //-- cas des matrices tridiagonales:
418  Vecteur B,C; // ligne sup et inf. La diagonale est stockée dans mat
419
420  //----- variables utilisées pour la résolution de systèmes avec lapack -----
421  // ces variables sont indépendantes de la matrice donc ne rentre pas en ligne de compte dans les
    opérateurs
422  // par exemple (échange, sauvegarde ...)
423  ENTIER_POUR_CLAPACK * ipiv; // la matrice des pivots d'indices qui définit la matrice de permutation P
424  // utilisé par le prog clapack: dgesv_
425  MatLapack* ptB; // array intermédiaires contenant en entrée les seconds membres et en sortie les
    résultats
426  Vecteur travail; // espace de travail
427
428  // pointeur d'accès aux coordonnées avec i variant de 1 à nb_lign et j variant de 1 à nb_col
429  double& (MatLapack::*Coor) (int i, int j );
430  double (MatLapack::*Coor_const) (int i, int j ) const;
431
432  // ---- declaration des différents accès aux composantes
433  // 1) pour une matrice générale non sym, rectangulaire: stockage de type GE
434  double& Coor_GE(int i, int j)
435  #ifdef MISE_AU_POINT
436  { return mat((j-1)*lda + i); };
437  #else
438  { return mat.v[(j-1)*lda + i - 1]; };
439  #endif
440  double Coor_GE_const(int i, int j) const
441  #ifdef MISE_AU_POINT
442  { return mat((j-1)*lda + i); };
443  #else
444  { return mat.v[(j-1)*lda + i - 1]; };
445  #endif
446  // 2) pour une matrice bande générale: nb_col = la largeur de bande: stockage de type GB
447  double& Coor_GB(int i, int j)
448  #ifdef MISE_AU_POINT
449  { return mat( (j-1)*lda + nb_col + i-j ); };
450  #else
451  { return mat.v[( (j-1)*lda + nb_col + i-j ) - 1]; };
452  #endif
453  double Coor_GB_const(int i, int j) const
454  #ifdef MISE_AU_POINT
455  { return mat( (j-1)*lda + nb_col + i-j ); };
456  #else
457  { return mat.v[( (j-1)*lda + nb_col + i-j ) - 1]; };
458  #endif
459  // 3) pour une matrice bande symétrique: nb_col = la 1/2 largeur de bande: stockage de type PB
460  // c'est à dire symétrique positif, avec un stockage 'U'
461  // AB(ku+1+i-j,j) = A(i,j) pour max(1,j-ku) <= i <= j
462  double& Coor_PB(int i, int j)
463  #ifdef MISE_AU_POINT
464  { if (j >= i) return mat( (j-1)*lda + nb_col + i-j);
465    else return mat( (i-1)*lda + nb_col + j-i);};
466  #else
467  { if (j >= i) return mat.v[( (j-1)*lda + nb_col + i-j ) - 1];
468    else return mat.v[( (i-1)*lda + nb_col + j-i ) - 1];};
469  #endif
470  double Coor_PB_const(int i, int j) const
471  #ifdef MISE_AU_POINT
472  { if (j >= i) return mat( (j-1)*lda + nb_col + i-j);
473    else return mat( (i-1)*lda + nb_col + j-i);};
474  #else
475  { if (j >= i) return mat.v[( (j-1)*lda + nb_col + i-j ) - 1];
476    else return mat.v[( (i-1)*lda + nb_col + j-i ) - 1];};
477  #endif
478
479  // 4) pour une matrice carré symétrique: nb_col = la 1/2 largeur de bande: stockage de type PP
480  // c'est à dire symétrique positif, avec un stockage 'U' et stockage carré compacté
481  // AB(i+(j-1)*j/2) = A(i,j) pour 1 <= i <= j
482  double& Coor_PP(int i, int j)
483  #ifdef MISE_AU_POINT
484  { if (j >= i) return mat( (j-1)*j/2 + i);
485    else return mat( (i-1)*i/2 + j);};
486  #else
487  { if (j >= i) return mat.v[ (j-1)*j/2 + i - 1];
488    else return mat.v[ (i-1)*i/2 + j - 1];};
489  #endif
490  double Coor_PP_const(int i, int j) const
491  #ifdef MISE_AU_POINT
492  { if (j >= i) return mat( (j-1)*j/2 + i);
493    else return mat( (i-1)*i/2 + j);};
494  #else
495  { if (j >= i) return mat.v[ (j-1)*j/2 + i - 1];
496    else return mat.v[ (i-1)*i/2 + j - 1];};
497  #endif
498
499
500
501  // 5) pour une matrice tridiagonale quelconque : mat contient la diagonale

```

```

502 // B la diag sup, C la diag inf
503 //           A(i-1,i) = B(i-1)
504 // A(i,i-1) = C(i-1); A(i,i)   = mat(i);  A(i,i+1) = B(i)
505 //           A(i+1,i) = C(i)
506 // la suite = def de l'accès aux composantes: en chantier
507
508
509 // fonction interne de vérification des composantes
510 // arrêt si pb
511 void Verif_nb_composante(const Vecteur& b,string nom_routine) const ;
512 void Verif_nb_composante(const Tableau <Vecteur>& b,string nom_routine) const ;
513 // gestion erreur dgesv et dgbsv
514 void GestionErreurDg_sv(const int& info) const;
515 // gestion erreur dpbsv
516 void GestionErreurDpbsv(const int& info) const;
517 // gestion erreur Inverse
518 void GestionErreurInverse(const int& info,int cas,const string nom_routine_lapack) const;
519
520 };
521     /// @} // end of group
522
523 // pour l'instant on n'arrive pas à faire des includes en linux ??
524 #ifndef MISE_AU_POINT
525 #undef MatLapack_H_deja_inclus
526 #include "MatLapack.cc"
527 #define MatLapack_H_deja_inclus
528 #endif
529
530 #endif
531
532

```

## 7.384 vecdefs\_GR.h

```

1
2 /*****
3 /*
4 /*
5 /*           MV++ Numerical Matrix/Vector C++ Library
6 /*           MV++ Version 1.5
7 /*
8 /*           R. Pozo
9 /*           National Institute of Standards and Technology
10 /*
11 /*           NOTICE
12 /*
13 /* Permission to use, copy, modify, and distribute this software and
14 /* its documentation for any purpose and without fee is hereby granted
15 /* provided that this permission notice appear in all copies and
16 /* supporting documentation.
17 /*
18 /* Neither the Institution (National Institute of Standards and Technology)
19 /* nor the author makes any representations about the suitability of this
20 /* software for any purpose. This software is provided "as is"without
21 /* expressed or implied warranty.
22 /*
23 /*****
24
25 /*****
26 /*           Which dense vector/matrix classes to build SparseLib++ from
27 /*****
28
29
30 #ifndef vector_defs_H
31 #define vector_defs_H
32
33 // #define VECTOR_H           "mvv.h"
34 #define VECTOR_double       MV_Vector_double
35 #define VECTOR_float        MV_Vector_float
36 #define VECTOR_int          MV_Vector_int
37 #define VECTOR_ref          MV_Vector_::ref
38 // #define MATRIX_H          "mvm.h"
39 #define MATRIX_double       MV_ColMat_double
40 #define MATRIX_float        MV_ColMat_float
41 #define MATRIX_int          MV_ColMat_int
42 #define MATRIX_ref          MV_Matrix_::ref
43
44 #define VECTOR_COMPLEX       MV_Vector_COMPLEX
45 #define MATRIX_COMPLEX       MV_ColMat_COMPLEX
46
47 //===== modif GR =====
48
49 #include "mvvtp_GR.h"
50 #include "mvmtpr_GR.h"
51 // #include <complex.h>

```

```

52 #include <complex>
53
54 typedef MV_Vector <double>           MV_Vector_double;
55 typedef MV_Vector <float>           MV_Vector_float;
56 typedef MV_Vector <int>             MV_Vector_int;
57 typedef MV_Vector <complex <double> > MV_Vector_COMPLEX;
58
59 typedef MV_ColMat <double>           MV_ColMat_double;
60 typedef MV_ColMat <float>           MV_ColMat_float;
61 typedef MV_ColMat <int>             MV_ColMat_int;
62 typedef MV_ColMat <complex <double> > MV_ColMat_COMPLEX;
63
64 //===== fin modif GR =====
65
66 #endif

```

## 7.385 iotext\_GR.h

```

1
2 /*+++++*/
3 /*
4 /*
5 /*           MV++ Numerical Matrix/Vector C++ Library
6 /*           MV++ Version 1.5
7 /*
8 /*           R. Pozo
9 /*           National Institute of Standards and Technology
10 /*
11 /*           NOTICE
12 /*
13 /* Permission to use, copy, modify, and distribute this software and
14 /* its documentation for any purpose and without fee is hereby granted
15 /* provided that this permission notice appear in all copies and
16 /* supporting documentation.
17 /*
18 /* Neither the Institution (National Institute of Standards and Technology)
19 /* nor the author makes any representations about the suitability of this
20 /* software for any purpose. This software is provided "as is"without
21 /* expressed or implied warranty.
22 /*
23 /*+++++*/
24
25 // read and write MV vectors as text files with (at most) one element
26 // per line.
27
28 #ifndef _IOTEXT_H_
29 #define _IOTEXT_H_
30
31 int readtxtfile_vec(const char *filename, MV_Vector_double *Aptr);
32 int writetxtfile_vec(const char *filename, const MV_Vector_double &A);
33 int readtxtfile_vec(const char *filename, MV_Vector_int *Aptr);
34 int writetxtfile_vec(const char *filename, const MV_Vector_int &A);
35
36 #endif
37

```

## 7.386 mvblas\_GR.h

```

1
2 /*+++++*/
3 /*
4 /*
5 /*           MV++ Numerical Matrix/Vector C++ Library
6 /*           MV++ Version 1.5
7 /*
8 /*           R. Pozo
9 /*           National Institute of Standards and Technology
10 /*
11 /*           NOTICE
12 /*
13 /* Permission to use, copy, modify, and distribute this software and
14 /* its documentation for any purpose and without fee is hereby granted
15 /* provided that this permission notice appear in all copies and
16 /* supporting documentation.
17 /*
18 /* Neither the Institution (National Institute of Standards and Technology)
19 /* nor the author makes any representations about the suitability of this
20 /* software for any purpose. This software is provided "as is"without
21 /* expressed or implied warranty.
22 /*
23 /*+++++*/
24

```



```
25 // modification et ajout de g rard rio
26
27 #ifndef _MV_BLAS1_TPL_H_
28 #define _MV_BLAS1_TPL_H_
29
30 #include "Sortie.h"
31 //#include "Debug.h"
32 #include <math.h>
33
34 template <class TYPE>
35 int operator==(const MV_Vector<TYPE> &x, const MV_Vector<TYPE> &y)
36 {
37     int Nx = x.size();
38     if(Nx != y.size())
39         return 0;
40     for (int i=0; i<Nx; i++)
41         if (x(i) != y(i))
42             return 0;
43     return 1;
44 }
45
46 template <class TYPE>
47 int operator!=(const MV_Vector<TYPE> &x, const MV_Vector<TYPE> &y)
48 {
49     if (x == y)
50         return 0;
51     else
52         return 1;
53 }
54
55
56 template <class TYPE>
57 MV_Vector<TYPE>& operator*=(MV_Vector<TYPE> &x, const TYPE &a)
58 {
59     int N = x.size();
60     for (int i=0; i<N; i++)
61         x(i) *= a;
62     return x;
63 }
64
65 template <class TYPE>
66 MV_Vector<TYPE> operator*(const TYPE &a, const MV_Vector<TYPE> &x)
67 {
68     int N = x.size();
69     MV_Vector<TYPE> result(N);
70     for (int i=0; i<N; i++)
71         result(i) = x(i)*a;
72     return result;
73 }
74
75 template <class TYPE>
76 MV_Vector<TYPE> operator*(const MV_Vector<TYPE> &x, const TYPE &a)
77 {
78     // This is the other commutative case of vector*scalar.
79     // It should be just defined to be
80     // "return operator*(a,x);"
81     // but some compilers (e.g. Turbo C++ v.3.0) have trouble
82     // determining the proper template match. For the moment,
83     // we'll just duplicate the code in the scalar * MV_Vector
84     // case above.
85
86     int N = x.size();
87     MV_Vector<TYPE> result(N);
88     for (int i=0; i<N; i++)
89         result(i) = x(i)*a;
90     return result;
91 }
92
93
94 template <class TYPE>
95 MV_Vector<TYPE> operator+(const MV_Vector<TYPE> &x, const MV_Vector<TYPE> &y)
96 {
97     int N = x.size();
98     #ifndef MISE_AU_POINT
99     if (N != y.size())
100     {
101         cout << "Incompatible vector lengths in +." << endl;
102         Sortie(1);
103     }
104     #endif
105
106     MV_Vector<TYPE> result(N);
107     for (int i=0; i<N; i++)
108         result(i) = x(i) + y(i);
109     return result;
110 }
111
```

```

112 template <class TYPE>
113 MV_Vector<TYPE> operator-(const MV_Vector<TYPE> &x, const MV_Vector<TYPE> &y)
114 {
115     int N = x.size();
116     #ifndef MISE_AU_POINT
117     if (N != y.size())
118     {
119         cout << "Incompatible vector lengths in -." << endl;
120         Sortie(1);
121     }
122     #endif
123     MV_Vector<TYPE> result(N);
124     for (int i=0;i<N; i++)
125         result(i) = x(i) - y(i);
126     return result;
127 }
128
129
130 template <class TYPE>
131 MV_Vector<TYPE>& operator+=(MV_Vector<TYPE> &x, const MV_Vector<TYPE> &y)
132 {
133     int N = x.size();
134     #ifndef MISE_AU_POINT
135     if (N != y.size())
136     {
137         cout << "Incompatible vector lengths in +." << endl;
138         Sortie(1);
139     }
140     #endif
141     for (int i=0;i<N; i++)
142         x(i) += y(i);
143     return x;
144 }
145
146
147 template <class TYPE>
148 MV_Vector<TYPE>& operator-=(MV_Vector<TYPE> &x, const MV_Vector<TYPE> &y)
149 {
150     int N = x.size();
151     #ifndef MISE_AU_POINT
152     if (N != y.size())
153     {
154         cout << "Incompatible vector lengths in -." << endl;
155         Sortie(1);
156     }
157     #endif
158     for (int i=0;i<N; i++)
159         x(i) -= y(i);
160     return x;
161 }
162
163
164
165 // norm and dot product functions for the MV_Vector<> class
166
167
168 template <class TYPE>
169 TYPE dot(const MV_Vector<TYPE> &x, const MV_Vector<TYPE> &y)
170 {
171
172     // Check for compatible dimensions:
173     #ifndef MISE_AU_POINT
174     if (x.size() != y.size())
175     {
176         cout << "Incompatible dimensions in dot(). " << endl;
177         Sortie(1);
178     }
179     #endif
180     TYPE temp=0.0;
181     for (int i=0; i<x.size();i++)
182         temp += x(i)*y(i);
183     return temp;
184 }
185
186 template <class TYPE>
187 TYPE norm(const MV_Vector<TYPE> &x)
188 {
189     TYPE temp = dot(x,x);
190     return sqrt(temp);
191 }
192
193 #endif
194 // _MV_BLAS1_TPL_H_

```

## 7.387 mvmtmp\_GR.h

```

1
2 /*****
3 */
4 */
5 */           MV++ Numerical Matrix/Vector C++ Library
6 */           MV++ Version 1.5
7 */
8 */           R. Pozo
9 */           National Institute of Standards and Technology
10 */
11 */           NOTICE
12 */
13 */ Permission to use, copy, modify, and distribute this software and
14 */ its documentation for any purpose and without fee is hereby granted
15 */ provided that this permission notice appear in all copies and
16 */ supporting documentation.
17 */
18 */ Neither the Institution (National Institute of Standards and Technology)
19 */ nor the author makes any representations about the suitability of this
20 */ software for any purpose. This software is provided "as is"without
21 */ expressed or implied warranty.
22 */
23 /*****
24
25 //
26 //     mvmtmp.h : basic templated numerical matrix class, storage
27 //                by columns (Fortran oriented.)
28 //
29 //
30 //
31
32 // modif g rard RIO
33 // - d placement de la surcharge d' criture directement dans la classe (sinon pb
34 //   avec code warrior)
35 // - modification de l'indice du type unsigned int au type int et v rification
36 //   du nombre n gatif
37
38 #ifndef _MV_MATRIX_H_
39 #define _MV_MATRIX_H_
40
41 #include "mvvtp_GR.h"
42
43 struct Matrix_
44 {
45     enum ref_type { ref = 1 };
46 };
47
48
49 #include <iostream>           // for formatted printing of matrices
50 #ifdef MV_MATRIX_BOUNDS_CHECK
51 #include <assert.h>
52 #endif
53
54
55 template <class TYPE>
56 class MV_ColMat
57 {
58     private:
59         MV_Vector<TYPE> v_;
60         int dim0_; // preferred to using dim_[2]. some compilers
61         int dim1_; // refuse to initialize these in the constructor.
62         int lda_;
63         int ref_; // true if this is declared as a reference vector,
64                 // i.e. it does not own the memory space, but
65                 // rather it is a view to another vector or array.
66     public:
67
68         /*::::::::::*/
69         /* Constructors/Destructors */
70         /*::::::::::*/
71
72         MV_ColMat();
73         MV_ColMat( int, int);
74
75         // some compilers have difficulty with inlined 'for' statements.
76         MV_ColMat( int, int, const TYPE&);
77
78         // usual copy by value
79         // (can't use default parameter lda=m, because m is not a constant...)
80         //
81         MV_ColMat( TYPE*, int m, int n);
82         MV_ColMat( TYPE*, int m, int n, int lda);
83
84         // the "reference" versions
85         //

```

```

86 //
87 MV_ColMat(TYPE*, int m, int n, Matrix_::ref_type i);
88 MV_ColMat(TYPE*, int m, int n, int lda,
89           Matrix_::ref_type i);
90
91 MV_ColMat(const MV_ColMat<TYPE>&);
92 ~MV_ColMat();
93
94 /*::::::::::::::::::::::::::*/
95 /* Indices and access operations */
96 /*::::::::::::::::::::::::::*/
97
98 inline TYPE& operator()(int, int);
99 inline const TYPE& operator()(int, int) const;
100 MV_ColMat<TYPE> operator()(const MV_VecIndex &I, const MV_VecIndex &J) ;
101 const MV_ColMat<TYPE> operator()(const MV_VecIndex &I, const MV_VecIndex &J) const;
102 int size(int i) const;
103 MV_ColMat<TYPE>& newsize(int, int);
104 int ref() const { return ref_;}
105
106 /*::::::::::::::::::::::::::*/
107 /* Assignment */
108 /*::::::::::::::::::::::::::*/
109
110 MV_ColMat<TYPE> & operator=(const MV_ColMat<TYPE>&);
111 MV_ColMat<TYPE> & operator=(const TYPE&);
112
113
114 friend ostream& operator<<(ostream &s, const MV_ColMat<TYPE> &V)
115 {
116     int M = V.size(0);
117     int N = V.size(1);
118
119     for (int i=0; i<M; i++)
120     {
121         for (int j=0; j<N; j++)
122             s << V(i,j) << " ";
123         s << endl;
124     }
125
126     return s;
127 }
128 };
129
130
131
132 template<class TYPE>
133 int MV_ColMat<TYPE>::size(int i) const
134 {
135     if (i==0) return dim0_;
136     if (i==1) return dim1_;
137     else
138     {
139         cerr << "Called MV_ColMat::size(" << i << ") must be 0 or 1 " << endl;
140         Sortie(1);
141     }
142
143     // never should be here, but many compilers warn about not
144     // returning a value
145     return 0;
146 }
147
148 // NOTE: null construct have ref_ flag turned OFF, otherwise, we can
149 // never reset the size of matrix....
150 template <class TYPE>
151 MV_ColMat<TYPE>::MV_ColMat() : v_(0), dim0_(0), dim1_(0), lda_(0), ref_(0){}
152
153
154 template <class TYPE>
155 MV_ColMat<TYPE>::MV_ColMat(int m, int n) : v_(m*n),
156     dim0_(m), dim1_(n), lda_(m), ref_(0)
157 {
158     # ifdef MV_MATRIX_BOUNDS_CHECK
159         assert((n >= 0)&& (m>=0)); // modif GR
160     # endif
161 }
162
163 template <class TYPE>
164 MV_ColMat<TYPE>::MV_ColMat(int m, int n, const TYPE &s) : v_(m*n),
165     dim0_(m), dim1_(n), lda_(m), ref_(0)
166 {
167     operator=(s);
168     # ifdef MV_MATRIX_BOUNDS_CHECK
169         assert((n >= 0)&& (m>=0)); // modif GR
170     # endif
171 }
172

```

```

173 // operators and member functions
174
175
176
177 template <class TYPE>
178 inline TYPE& MV_ColMat<TYPE>::operator()( int i, int j)
179 {
180 #ifdef MV_MATRIX_BOUNDS_CHECK
181     assert(0<=i && i<size(0));
182     assert(0<=j && j<size(1));
183 #endif
184     return v_(j*lda_ + i); // could use indirect addressing
185                             // instead...
186 }
187
188 template <class TYPE>
189 inline const TYPE& MV_ColMat<TYPE>::operator()
190     ( int i, int j) const
191 {
192 #ifdef MV_MATRIX_BOUNDS_CHECK
193     assert(0<=i && i<size(0));
194     assert(0<=j && j<size(1));
195 #endif
196     return v_(j*lda_ + i);
197 }
198
199
200 template <class TYPE>
201 MV_ColMat<TYPE>& MV_ColMat<TYPE>::operator=(const TYPE & s)
202 {
203     int M = size(0);
204     int N = size(1);
205
206     if (lda_ == M) // if continuous, then just assign as a ?
207         v_ = s; // single long vector.
208
209     else
210     {
211         // this should run much faster than the just accessing each (i,j)
212         // element individually
213         //
214
215         MV_VecIndex I(0,M-1);
216         for (int j=0; j<N; j++)
217         {
218             v_(I) = s;
219             I += lda_;
220         }
221     }
222
223     return *this;
224 }
225
226 template <class TYPE>
227 MV_ColMat<TYPE>& MV_ColMat<TYPE>::newsize( int M, int N)
228 {
229 #ifdef MV_MATRIX_BOUNDS_CHECK
230     assert((N >= 0)&& (M>=0)); // modif GR
231 #endif
232     v_.newsize(M*N);
233     dim0_ = M;
234     dim1_ = N;
235     lda_ = M;
236
237     return *this;
238 }
239
240 template <class TYPE>
241 MV_ColMat<TYPE>& MV_ColMat<TYPE>::operator=(const MV_ColMat<TYPE> & m)
242 {
243
244     int lM = dim0_; // left hand arg (this)
245     int lN = dim1_;
246
247     int rM = m.dim0_; // right hand arg (m)
248     int rN = m.dim1_;
249
250
251     // if the left-hand side is a matrix reference, then we copy the
252     // elements of m *into* the region specified by the reference.
253     // i.e. inject().
254
255     if (ref_)
256     {
257         // check conformance,
258         if (lM != rM || lN != rN)
259             {

```

```

260         cerr << "MV_ColMatRef::operator= non-conformant assignment.\n";
261         Sortie(1);
262     }
263 }
264 else
265 {
266     newsize(rM,rN);
267 }
268
269 // at this point the left hand and right hand sides are conformant
270
271 // this should run much faster than the just accessing each (i,j)
272 // element individually
273
274 // if both sides are contiguous, then just copy as one vector
275 if ( lM == lda_ && rM == m.lda_)
276 {
277     MV_VecIndex I(0,rM*rN-1);
278     v_(I) = m.v_(I);
279 }
280 else
281 {
282     // slower way...
283
284     MV_VecIndex I(0,rM-1);
285     MV_VecIndex K(0,rM-1);
286     for (int j=0; j<rN; j++)
287     {
288         v_(I) = m.v_(K);
289         I += lda_;
290         K += m.lda_;
291     }
292 }
293
294 return *this;
295 }
296
297 template <class TYPE>
298 MV_ColMat<TYPE>::MV_ColMat(const MV_ColMat<TYPE> & m) :
299     v_(m.dim0_*m.dim1_, dim0_(m.dim0_),
300        dim1_(m.dim1_), ref_(0), lda_(m.dim0_))
301 {
302
303     int M = m.dim0_;
304     int N = m.dim1_;
305
306     // this should run much faster than the just accessing each (i,j)
307     // element individually
308
309     MV_VecIndex I(0,M-1);
310     MV_VecIndex K(0,M-1);
311     for (int j=0; j<N; j++)
312     {
313         v_(I) = m.v_(K);
314         I += lda_;
315         K += m.lda_;
316     }
317 }
318
319
320 template <class TYPE>
321 inline MV_ColMat<TYPE>::MV_ColMat(TYPE* d,   int m,   int n,
322     Matrix_::ref_type i) :
323     v_(d,m*n, MV_Vector_::ref), dim0_(m), dim1_(n), lda_(m), ref_(i)
324     {
325         #   ifdef MV_MATRIX_BOUNDS_CHECK
326             assert((n >= 0)&& (m>=0)); // modif GR
327         #   endif
328     }
329
330 template <class TYPE>
331 inline MV_ColMat<TYPE>::MV_ColMat(TYPE* d,   int m,   int n,
332     int lda, Matrix_::ref_type i) :
333     v_(d, lda*n, MV_Vector_::ref), dim0_(m), dim1_(n), lda_(lda),
334     ref_(i)
335     {
336         #   ifdef MV_MATRIX_BOUNDS_CHECK
337             assert((n >= 0)&& (m>=0)&& (lda>=0)); // modif GR
338         #   endif
339     }
340
341 template <class TYPE>
342 MV_ColMat<TYPE>::MV_ColMat(TYPE* d,   int m,   int n) :
343     v_(m*n), dim0_(m), dim1_(n), lda_(m), ref_(0)
344 {
345     #   ifdef MV_MATRIX_BOUNDS_CHECK
346         assert((n >= 0)&& (m>=0)); // modif GR

```

```

347     #   endif
348     int mn = m*n;
349
350     // d is contiguous, so just copy 1-d vector
351     for (int i=0; i< mn; i++)
352         v_[i] = d[i];
353 }
354
355
356 template <class TYPE>
357 MV_ColMat<TYPE>::MV_ColMat (TYPE* d,   int m,   int n,
358     int lda) :
359     v_(m*n), dim0_(m), dim1_(n), lda_(lda), ref_(0)
360 {
361     #   ifdef MV_MATRIX_BOUNDS_CHECK
362         assert((n >= 0)&& (m>=0)&&(lda>=0)); // modif GR
363     #   endif
364     for (int j=0; j< n; j++)
365         for (int i=0; i<m; i++)
366             operator()(i,j) = d[j*lda + i]; // could be made faster!!
367 }
368
369
370 template <class TYPE>
371 MV_ColMat<TYPE> MV_ColMat<TYPE>::operator() (const MV_VecIndex &I, const MV_VecIndex &J)
372 {
373     // check that index is not out of bounds
374     //
375     if (I.end() >= dim0_ || J.end() >= dim1_)
376     {
377         cerr << "Matrix index: (" << I.start() << ":" << I.end()
378             << "," << J.start() << ":" << J.end()
379             << ") not a subset of (0:" << dim0_ - 1 << ", 0:"
380             << dim1_-1 << ") " << endl;
381         Sortie(1);
382     }
383
384     // this automatically returns a reference
385     //
386     return MV_ColMat<TYPE>(&v_[J.start()*lda_ + I.start()],
387         I.end() - I.start() + 1,
388         J.end() - J.start() + 1, lda_, Matrix_::ref);
389 }
390
391 template <class TYPE>
392 const MV_ColMat<TYPE> MV_ColMat<TYPE>::operator() (const MV_VecIndex &I,
393     const MV_VecIndex &J) const
394 {
395
396     cerr << "Const operator()(MV_VecIndex, MV_VecIndex) called " << endl;
397
398     // check that index is not out of bounds
399     //
400     if (I.end() >= dim0_ || J.end() >= dim1_)
401     {
402         cerr << "Matrix index: (" << I.start() << ":" << I.end()
403             << "," << J.start() << ":" << J.end()
404             << ") not a subset of (0:" << dim0_ - 1 << ", 0:"
405             << dim1_-1 << ") " << endl;
406         Sortie(1);
407     }
408
409     // this automatically returns a reference. we need to
410     // "cast away" constness here, so the &v_[] arg will
411     // not cause a compiler error.
412     //
413     MV_ColMat<TYPE> *t = (MV_ColMat<TYPE>*) this;
414     return MV_ColMat<TYPE>(&(t->v_[J.start()*lda_ + I.start()]),
415         I.end() - I.start() + 1,
416         J.end() - J.start() + 1, lda_, Matrix_::ref);
417 }
418
419 template <class TYPE>
420 MV_ColMat<TYPE>::~MV_ColMat () {}
421
422
423
424 #endif
425 // _MV_MATRIX_H_
426

```

## 7.388 mvvind\_GR.h

```

1
2 /*****

```

```

3 /* */
4 /* */
5 /*          MV++ Numerical Matrix/Vector C++ Library */
6 /*          MV++ Version 1.5 */
7 /* */
8 /*          R. Pozo */
9 /*          National Institute of Standards and Technology */
10 /* */
11 /*          NOTICE */
12 /* */
13 /* Permission to use, copy, modify, and distribute this software and */
14 /* its documentation for any purpose and without fee is hereby granted */
15 /* provided that this permission notice appear in all copies and */
16 /* supporting documentation. */
17 /* */
18 /* Neither the Institution (National Institute of Standards and Technology) */
19 /* nor the author makes any representations about the suitability of this */
20 /* software for any purpose. This software is provided "as is"without */
21 /* expressed or implied warranty. */
22 /* */
23 /*+++++*/
24
25 //
26 //      mvvind.h      MV_Vector Index class
27
28 #ifndef _MV_VEC_INDEX_H_
29 #define _MV_VEC_INDEX_H_
30
31 // A MV_VecIndex is an ordered pair (start,end) denoting a subvector
32 // region, similar to a Fortran 90 or Matlab colon notation. For example,
33 //
34 // MV_Vector_double A(10), B(20);
35 // MV_VecIndex I(2,4);
36 //
37 // A(I) = B(MV_VecIndex(0,2));
38 //
39 // sets the thrid through fifth elements of A to the first two elements
40 // of B. There is no stride argument, only contiguous regions are allowed.
41 //
42
43 // modif GR: passage d'unsigned à int normal
44
45 #include <assert.h>
46 #include "Sortie.h"
47
48 class MV_VecIndex
49 {
50     private:
51         // unsigned int start_;
52         // unsigned int end_;
53         int start_; // modif GR
54         int end_; // modif GR
55         char all_; // true if this index refers to the complete
56                 // vector range. start_ and end_ are ignored.
57     public:
58         MV_VecIndex() : start_(0), end_(0), all_(1) {}
59         // MV_VecIndex(unsigned int i1) :start_(i1), end_(i1), all_(0) {}
60         // MV_VecIndex(unsigned int i1, unsigned int i2): start_(i1), end_(i2),
61         MV_VecIndex(int i1) :start_(i1), end_(i1), all_(0) {}
62         MV_VecIndex(int i1, int i2): start_(i1), end_(i2),
63                 all_(0)
64         {
65             # ifdef MV_VECTOR_BOUNDS_CHECK
66                 assert((i1 >= 0)&&(i2 >=0)); // modif GR
67             # endif
68             assert(i1 <= i2);
69         }
70         MV_VecIndex(const MV_VecIndex &s) : start_(s.start_), end_(s.end_),
71                 all_(s.all_){}
72
73
74         int start() const { return (int) ((all_==1) ? 0 : start_);}
75         int end() const { return (int) ((all_ ==1) ? 0 : end_);}
76         int length() const {
77             return (int)((all_==1) ? 0 : (end_-start_+1));}
78         int all() const { return all_; }
79         MV_VecIndex& operator=(const MV_VecIndex& I)
80         { start_=I.start_; end_ = I.end_; return *this;}
81         MV_VecIndex operator+(int i)
82         { return MV_VecIndex(start_ +i, end_ +i); }
83         MV_VecIndex& operator+=(int i)
84         { start_ += i; end_ += i; return *this; }
85         MV_VecIndex operator-(int i)
86         { return MV_VecIndex(start_ -i, end_ -i); }
87         MV_VecIndex& operator-=(int i)
88         { start_ -= i; end_ -= i; return *this; }
89

```



```

90 };
91
92
93 #endif
94 // _INDEX_H_
95

```

## 7.389 mvvrf\_GR.h

```

1
2 /*+++++*/
3 /*                                           */
4 /*                                           */
5 /*           MV++ Numerical Matrix/Vector C++ Library */
6 /*           MV++ Version 1.5 */
7 /*                                           */
8 /*           R. Pozo */
9 /*           National Institute of Standards and Technology */
10 /*                                           */
11 /*           NOTICE */
12 /*                                           */
13 /* Permission to use, copy, modify, and distribute this software and */
14 /* its documentation for any purpose and without fee is hereby granted */
15 /* provided that this permission notice appear in all copies and */
16 /* supporting documentation. */
17 /*                                           */
18 /* Neither the Institution (National Institute of Standards and Technology) */
19 /* nor the author makes any representations about the suitability of this */
20 /* software for any purpose. This software is provided "as is"without */
21 /* expressed or implied warranty. */
22 /*                                           */
23 /*+++++*/
24
25
26 // this is really used as a sort of global constant. The reason
27 // for creating its own type is that so it can be overloaded to perform
28 // a deep or shallow assignment. (Any variable of type MV_Vector_::ref_type
29 // has only one possible value: one.)
30
31 #ifndef _MV_VECTOR_REF_
32 #define _MV_VECTOR_REF_
33 struct MV_Vector_
34 {
35     enum ref_type { ref = 1};
36 };
37 #endif

```

## 7.390 mvvtp\_GR.h

```

1
2 /*+++++*/
3 /*                                           */
4 /*                                           */
5 /*           MV++ Numerical Matrix/Vector C++ Library */
6 /*           MV++ Version 1.5 */
7 /*                                           */
8 /*           R. Pozo */
9 /*           National Institute of Standards and Technology */
10 /*                                           */
11 /*           NOTICE */
12 /*                                           */
13 /* Permission to use, copy, modify, and distribute this software and */
14 /* its documentation for any purpose and without fee is hereby granted */
15 /* provided that this permission notice appear in all copies and */
16 /* supporting documentation. */
17 /*                                           */
18 /* Neither the Institution (National Institute of Standards and Technology) */
19 /* nor the author makes any representations about the suitability of this */
20 /* software for any purpose. This software is provided "as is"without */
21 /* expressed or implied warranty. */
22 /*                                           */
23 /*+++++*/
24
25 //
26 //     mvvtp.h     Basic templated vector class
27 //
28
29 // modif g rard Rio :
30 // - introduction de la surcharge d' criture dans la classe MVvecteur
31 // - modification de l'indice du type unsigned int au type int et v rification
32 //   du nombre n gatif
33 // - modification de la surcharge d' criture (ajout de l' criture de la taille)

```

```

34 // et ajout de la surcharge de lecture en cohérence avec l'écriture déjà existante
35 // - introduction de la possibilité de créer un vecteur à la place d'un autre
36 // en donnant la référence à l'autre vecteur et un indicateur de non construction
37 // - dans le cas ou les tailles à alouer sont nulles, on met en place une procédure spé
38
39
40 #ifndef _MV_VECTOR_TPL_H_
41 #define _MV_VECTOR_TPL_H_
42
43 #include <iostream> // for formatted printing of vecteur : GR
44
45 #include <stdlib.h>
46 #ifdef MV_VECTOR_BOUNDS_CHECK
47 # include <assert.h>
48 #endif
49 #include "Sortie.h"
50 // #ifdef SYSTEM_MAC_OS_X
51 // #include <stringfwd.h> // a priori ce n'est pas portable
52 // #else
53 #if defined SYSTEM_MAC_OS_CARBON
54 #include <stringfwd.h> // a priori ce n'est pas portable
55 #else
56 #include <string.h> // pour le flot en memoire centrale
57 #endif
58 #include <string>
59
60 #include "mvvind_GR.h"
61 #include "mvvrf_GR.h"
62 using namespace std;
63 template <class TYPE>
64 class MV_Vector
65 {
66     protected:
67         TYPE *p_;
68         // unsigned int dim_;
69         int dim_; // modif GR
70         int ref_; // 0 or 1; does this own its own memory space?
71     public:
72
73         // les informations sont le type puis la taille puis les datas
74         friend istream & operator >> (istream & entree, MV_Vector<TYPE>& vec)
75         { // vérification du type
76             string type;
77             entree >> type;
78             if (type != "MV_Vector<>")
79                 {Sortie (1);
80                  return entree;
81                 }
82             // passage de la chaine donnant la taille puis lecture de la taille
83             int taille;
84             entree >> type >> taille;
85             // vérification de la taille sinon changement
86             if (vec.size() != taille) vec.newsize(taille);
87             // lecture des données
88             for (int i = 0; i < taille; i++)
89                 entree >> vec.p_[i];
90             return entree;
91         };
92         // surcharge de l'operateur d'écriture non formatée
93         // les informations sont le type puis la taille puis les datas séparées par
94         // un espace
95         friend ostream& operator<< (ostream& s, const MV_Vector<TYPE>& V)
96         {
97             int N = V.size();
98             s << "\n MV_Vector<> taille= " << V.size();
99             for (int j=0; j<N; j++)
100                 s << V(j) << " ";
101             s << endl;
102             return s;
103         }
104
105         /*::::::::::::::::::::::::::*/
106         /* Constructors/Destructors */
107         /*::::::::::::::::::::::::::*/
108
109     MV_Vector();
110     // MV_Vector(unsigned int);
111     // MV_Vector(unsigned int, const TYPE&);
112     // MV_Vector(TYPE*, unsigned int);
113     // MV_Vector(const TYPE*, unsigned int);
114
115     MV_Vector( int); // modif GR
116     MV_Vector( int, const TYPE&); // modif GR
117
118     MV_Vector(TYPE*, int); // modif GR
119     MV_Vector(const TYPE*, int); // modif GR
120

```

```

121 // reference of an existing data structure
122 //
123 // MV_Vector(TYPE*, unsigned int, MV_Vector_::ref_type i);
124 MV_Vector(TYPE*, int, MV_Vector_::ref_type i); // modif GR
125 // construction ici réelle
126 MV_Vector(const MV_Vector<TYPE>&);
127 // construction conditionnelle en fonction de l'indicateur
128 // cette construction est moins rapide que la précédente s'il y a vraiment
129 // définition d'un nouveau tableau de data (appel de new moins bon)
130 MV_Vector(MV_Vector<TYPE>&,MV_Vector_::ref_type i);
131 // destructeur
132 ~MV_Vector();
133
134 /*::::::::::::::::::::::::::::*/
135 /* Indices and access operations */
136 /*::::::::::::::::::::::::::::*/
137
138
139 // inline TYPE& operator()(unsigned int i)
140 inline TYPE& operator()(int i) // modif GR
141 {
142 #
143 #ifdef MV_VECTOR_BOUNDS_CHECK
144 // assert(i < dim_);
145 // assert((i < dim_) || (i>=0)); // modif GR
146 #endif
147 // return p_[i];
148 }
149 // inline const TYPE& operator()(unsigned int i) const
150 inline const TYPE& operator()(int i) const // modif GR
151 {
152 #
153 #ifdef MV_VECTOR_BOUNDS_CHECK
154 // assert(i < dim_);
155 // assert((i < dim_) || (i>=0)); // modif GR
156 #endif
157 // return p_[i];
158 }
159 // inline TYPE& operator[](unsigned int i)
160 inline TYPE& operator[](int i) // modif GR
161 {
162 #
163 #ifdef MV_VECTOR_BOUNDS_CHECK
164 // assert(i < dim_);
165 // assert((i < dim_) || (i>=0)); // modif GR
166 #endif
167 // return p_[i];
168 }
169 // inline const TYPE& operator[](unsigned int i) const
170 inline const TYPE& operator[](int i) const // modif GR
171 {
172 #
173 #ifdef MV_VECTOR_BOUNDS_CHECK
174 // assert(i < dim_);
175 // assert((i < dim_) || (i>=0)); // modif GR
176 #endif
177 // return p_[i];
178 }
179
180 inline MV_Vector<TYPE> operator()(const MV_VecIndex &I) ;
181 inline MV_Vector<TYPE> operator()(void);
182 inline const MV_Vector<TYPE> operator()(void) const;
183 inline const MV_Vector<TYPE> operator()(const MV_VecIndex &I) const;
184 // inline unsigned int size() const { return dim_;}
185 inline int size() const { return dim_;} // modif GR
186 inline int ref() const { return ref_;}
187 inline int null() const {return dim_== 0;}
188 //
189 // Create a new *uninitialized* vector of size N
190 MV_Vector<TYPE> & newsize( int );
191
192 /*::::::::::::::::::::*/
193 /* Assignment */
194 /*::::::::::::::::::::*/
195
196 MV_Vector<TYPE> & operator=(const MV_Vector<TYPE>&);
197 MV_Vector<TYPE> & operator=(const TYPE&);
198
199
200
201 };
202
203 template <class TYPE>
204 ostream& operator<<(ostream& s, const MV_Vector<TYPE>& V)
205 {
206 int N = V.size();
207

```

```

208     s << "\n MV_Vector<> taille= " << V.size();
209     for (int j=0; j<N; j++)
210         s << V(j) << " ";
211     s << endl;
212
213     return s;
214 }
215
216 // surcharge de l'operateur de lecture
217 // cohérente avec l'écriture déjà existante
218 template <class TYPE>
219 istream & operator >> (istream & entree, MV_Vector<TYPE>& vec)
220 { // vérification du type
221     string type;
222     entree >> type;
223     if (type != "MV_Vector<>")
224         {Sortie(1);
225          return entree;
226         }
227     // passage de la chaine donnant la taille puis lecture de la taille
228     int taille;
229     entree >> type >> taille;
230     // vérification de la taille sinon changement
231     if (vec.size() != taille) vec.newsize(taille);
232     // lecture des données
233     for (int i = 0; i < vec.taille; i++)
234         entree >> vec.p_[i];
235     return entree;
236 };
237
238 template <class TYPE>
239 MV_Vector<TYPE>::MV_Vector() : p_(NULL), dim_(0) , ref_(0){}
240
241 template <class TYPE>
242 //MV_Vector<TYPE>::MV_Vector(unsigned int n) : p_(new TYPE[n]), dim_(n),
243 // a priori si n est négatif new ramènera une erreur
244 //MV_Vector<TYPE>::MV_Vector( int n) : p_(new TYPE[n]), dim_(n), // modif 1 GR
245 MV_Vector<TYPE>::MV_Vector( int n) : dim_(n), // modif 2 GR
246     ref_(0)
247 { if (n!= 0)
248     {p_ = new TYPE[n];
249     if (p_ == NULL)
250     {
251         cerr << "Error: NULL pointer in MV_Vector(int) constructor " << endl;
252         cerr << "         Most likely out of memory... " << endl;
253         Sortie(1);
254     }
255     }
256     else
257         p_=NULL; // ajout car new ne fait pas bien son boulot !!
258 }
259
260 template <class TYPE>
261 //MV_Vector<TYPE>::MV_Vector(unsigned int n, const TYPE& v) :
262 MV_Vector<TYPE>::MV_Vector( int n, const TYPE& v) : // modif 1 GR
263 //     p_(new TYPE[n]), dim_(n), ref_(0)
264 //     dim_(n), ref_(0) // modif 2 GR
265 {
266     if (n!= 0)
267     { p_ = new TYPE[n];
268     if (p_ == NULL)
269     {
270         cerr << "Error: NULL pointer in MV_Vector(int) constructor " << endl;
271         cerr << "         Most likely out of memory... " << endl;
272         Sortie(1);
273     }
274     }
275     for (int i=0; i<n; i++)
276         p_[i] = v;
277     else
278         p_=NULL; // ajout car new ne fait pas bien son boulot !!
279
280 }
281
282 // operators and member functions
283 //
284
285
286
287
288 template <class TYPE>
289 MV_Vector<TYPE>& MV_Vector<TYPE>::operator=(const TYPE & m)
290 {
291 #ifdef TRACE_VEC
292     cout << "> MV_Vector<TYPE>::operator=(const TYPE & m) " << endl;
293 #endif
294

```

```

295 // unroll loops to depth of length 4
296
297 int N = this->size();
298
299 int Nminus4 = N-4;
300 int i;
301
302 for (i=0; i<Nminus4; )
303 {
304     p_[i++] = m;
305     p_[i++] = m;
306     p_[i++] = m;
307     p_[i++] = m;
308 }
309
310 for (; i<N; p_[i++] = m) ; // finish off last piece...
311
312 #ifdef TRACE_VEC
313     cout << "< MV_Vector<TYPE>::operator=(const TYPE & m) " << endl;
314 #endif
315     return *this;
316 }
317
318 template <class TYPE>
319 //MV_Vector<TYPE>& MV_Vector<TYPE>::newsize(unsigned int n)
320 MV_Vector<TYPE>& MV_Vector<TYPE>::newsize( int n) // modif GR
321 {
322 #ifdef TRACE_VEC
323     // cout << "> MV_Vector<TYPE>::newsize(unsigned int n) " << endl;
324     cout << "> MV_Vector<TYPE>::newsize( int n) " << endl;
325 #endif
326     if (ref_ ) // is this structure just a pointer?
327     {
328         {
329             cerr << "MV_Vector::newsize can't operator on references.\n";
330             Sortie(1);
331         }
332     }
333     else
334     if (dim_ != n) // only delete and new if
335     { // the size of memory is really
336         if (p_) delete [] p_; // changing, otherwise just
337         p_ = new TYPE[n]; // copy in place.
338         if (p_ == NULL)
339         {
340             cerr << "Error : NULL pointer in operator= " << endl;
341             Sortie(1);
342         }
343         dim_ = n;
344     }
345
346 #ifdef TRACE_VEC
347     // cout << "< MV_Vector<TYPE>::newsize(unsigned int n) " << endl;
348     cout << "< MV_Vector<TYPE>::newsize( int n) " << endl;
349 #endif
350
351     return *this;
352 }
353
354
355
356
357 template <class TYPE>
358 MV_Vector<TYPE>& MV_Vector<TYPE>::operator=(const MV_Vector<TYPE> & m)
359 {
360
361     int N = m.dim_;
362     int i;
363
364     if (ref_ ) // is this structure just a pointer?
365     {
366         if (dim_ != m.dim_) // check conformance,
367         {
368             cerr << "MV_VectorRef::operator= non-conformant assignment.\n";
369             Sortie(1);
370         }
371
372         // handle overlapping matrix references
373         if ((m.p_ + m.dim_) >= p_)
374         {
375             // overlap case, copy backwards to avoid overwriting results
376             for (i= N-1; i>=0; i--)
377                 p_[i] = m.p_[i];
378         }
379         else
380         {
381             for (i=0; i<N; i++)

```

```

382         p_[i] = m.p_[i];
383     }
384
385 }
386 else
387 {
388     newsize(N);
389
390     // no need to test for overlap, since this region is new
391     for (i =0; i< N; i++) // careful not to use bcopy()
392         p_[i] = m.p_[i]; // here, but TYPE::operator= TYPE.
393 }
394 return *this;
395 }
396
397 template <class TYPE>
398 //MV_Vector<TYPE>::MV_Vector(const MV_Vector<TYPE> & m) : p_(new TYPE[m.dim_]),
399 MV_Vector<TYPE>::MV_Vector(const MV_Vector<TYPE> & m) : // modif 2 GR
400     dim_(m.dim_) , ref_(0) , p_(NULL) // gerard: ajout de p_(NULL)
401 { if (m.dim_ != 0)
402     {p_ = new TYPE[m.dim_];
403     if (p_ == NULL)
404     {
405         cerr << "Error: Null pointer in MV_Vector(const MV_Vector&); " << endl;
406         Sortie(1);
407     }
408
409     int N = m.dim_;
410
411     for (int i=0; i<N; i++)
412         p_[i] = m.p_[i];
413 };
414 }
415
416
417 // note that ref() is initialized with i rather than 1.
418 // this is so compilers will not generate a warning that i was
419 // not used in the construction. (MV_Vector::ref_type is an enum that
420 // can *only* have the value of 1.
421 //
422 template <class TYPE>
423 //MV_Vector<TYPE>::MV_Vector(TYPE* d, unsigned int n, MV_Vector_::ref_type i) :
424 MV_Vector<TYPE>::MV_Vector(TYPE* d, int n, MV_Vector_::ref_type i) : // modif GR
425     p_(d) , dim_(n) , ref_(i)
426     {
427         # ifdef MV_VECTOR_BOUNDS_CHECK
428             assert(n < 0); // modif GR
429         # endif
430     }
431
432 template <class TYPE>
433 //MV_Vector<TYPE>::MV_Vector(TYPE* d, unsigned int n) : p_(new TYPE[n]),
434 //MV_Vector<TYPE>::MV_Vector(TYPE* d, int n) : p_(new TYPE[n]), // modif 1 GR
435 MV_Vector<TYPE>::MV_Vector(TYPE* d, int n) : // modif 2 GR
436 // a priori si n est négatif l'opérateur new renverra une erreur
437     dim_(n) , ref_(0)
438 { if (n != 0)
439     {p_ = new TYPE[n];
440     if (p_ == NULL)
441     {
442         cerr << "Error: Null pointer in MV_Vector(TYPE*, int) " << endl;
443         Sortie(1);
444     }
445     for (int i=0; i<n; i++)
446         p_[i] = d[i];
447 }
448 else
449     p_=NULL; // ajout car new ne fait pas bien son boulot !!
450 }
451 }
452
453 // construction conditionnelle en fonction de l'indicateur
454 // cette construction est moins rapide que la précédente s'il y a vraiment
455 // définition d'un nouveau tableau de data (appel de new moins bon)
456 template <class TYPE>
457 MV_Vector<TYPE>::MV_Vector(MV_Vector<TYPE>& B ,MV_Vector_::ref_type i) :
458     p_(B.p_) , dim_(B.dim_) , ref_(i)
459     { // le vecteur est construit en fonction du cas donné par ref_
460     if (ref_ != 1)
461         // cas avec construction
462         { p_ = new TYPE[dim_];
463         for (int i=0; i<dim_; i++)
464             p_[i] = B.p_[i];
465         }
466     // sinon le pointeur p_ est déjà bien positionné
467
468     # ifdef MV_VECTOR_BOUNDS_CHECK

```

```

469         assert(n < 0); // modif GR
470     # endif
471     }
472
473 template <class TYPE>
474 //MV_Vector<TYPE>::MV_Vector(const TYPE* d, unsigned int n) : p_(new TYPE[n]),
475 //MV_Vector<TYPE>::MV_Vector(const TYPE* d, int n) : p_(new TYPE[n]), // modif 1 GR
476 MV_Vector<TYPE>::MV_Vector(const TYPE* d, int n) : // modif 2 GR
477 // a priori si n est négatif l'opérateur new renvera une erreur
478     dim_(n) , ref_(0)
479 {if (n != 0)
480     {p_=new TYPE[n];
481     if (p_ == NULL)
482     {
483         cerr << "Error: Null pointer in MV_Vector(TYPE*, int) " << endl;
484         Sortie(1);
485     }
486     for (int i=0; i<n; i++)
487         p_[i] = d[i];
488     }
489     else
490     p_=NULL; // ajout car new ne fait pas bien son boulot !!
491 }
492 }
493
494 template <class TYPE>
495 MV_Vector<TYPE> MV_Vector<TYPE>::operator () (void)
496 {
497     return MV_Vector<TYPE>(p_, dim_, MV_Vector_::ref);
498 }
499
500 template <class TYPE>
501 const MV_Vector<TYPE> MV_Vector<TYPE>::operator () (void) const
502 {
503     return MV_Vector<TYPE>(p_, dim_, MV_Vector_::ref);
504 }
505
506 template <class TYPE>
507 MV_Vector<TYPE> MV_Vector<TYPE>::operator () (const MV_VecIndex &I)
508 {
509     // default parameters
510     if (I.all())
511         return MV_Vector<TYPE>(p_, dim_, MV_Vector_::ref);
512     else
513     {
514         // check that index is not out of bounds
515         //
516         if ( I.end() >= dim_)
517         {
518             cerr << "MV_VecIndex: (" << I.start() << ":" << I.end() <<
519             " ) too big for matrix (0:" << dim_ - 1 << ") " << endl;
520             Sortie(1);
521         }
522         return MV_Vector<TYPE>(p_+ I.start(), I.end() - I.start() + 1,
523             MV_Vector_::ref);
524     }
525 }
526
527 template <class TYPE>
528 const MV_Vector<TYPE> MV_Vector<TYPE>::operator () (const MV_VecIndex &I) const
529 {
530     // check that index is not out of bounds
531     //
532     if ( I.end() >= dim_)
533     {
534         cerr << "MV_VecIndex: (" << I.start() << ":" << I.end() <<
535         " ) too big for matrix (0:" << dim_ - 1 << ") " << endl;
536         Sortie(1);
537     }
538     return MV_Vector<TYPE>(p_+ I.start(), I.end() - I.start() + 1,
539         MV_Vector_::ref);
540 }
541
542 template <class TYPE>
543 MV_Vector<TYPE>::~MV_Vector ()
544 {
545     if (p_ && !ref_) delete [] p_;
546 }
547
548
549 template <class TYPE>
550 class FMV_Vector : public MV_Vector<TYPE>
551 {
552     public:
553     // FMV_Vector(unsigned int n) : MV_Vector<TYPE>(n) {}
554     FMV_Vector( int n) : MV_Vector<TYPE>(n) {}
555     FMV_Vector<TYPE>& operator=(const FMV_Vector<TYPE>& m);

```

```

556         FMV_Vector<TYPE>& operator=(const TYPE& m);
557 };
558
559 template <class TYPE>
560 FMV_Vector<TYPE>& FMV_Vector<TYPE>::operator=( const FMV_Vector<TYPE>& m)
561 {
562
563 #ifdef TRACE_VEC
564     cout << "> FMV_Vector<TYPE>::operator=( const FMV_Vector<TYPE>& m)" << endl;
565 #endif
566
567     int N = m.dim_;
568
569
570
571     if (this->ref_ )                // is this structure just a pointer?
572     {
573         if (this->dim_ != m.dim_)    // check conformance,
574         {
575             cerr << "MV_VectorRef::operator= non-conformant assignment.\n";
576             Sortie(1);
577         }
578     }
579     else if ( this->dim_ != m.dim_ ) // resize only if necessary
580         this->newsize(N);
581
582     memmove(this->p_, m.p_, N * sizeof(TYPE));
583
584 #ifdef TRACE_VEC
585     cout << "< FMV_Vector<TYPE>::operator=( const FMV_Vector<TYPE>& m)" << endl;
586 #endif
587
588     return *this;
589 }
590
591 template <class TYPE>
592 FMV_Vector<TYPE>& FMV_Vector<TYPE>::operator=(const TYPE & m)
593 {
594 #ifdef TRACE_VEC
595     cout << "> FMV_Vector<TYPE>::operator=(const TYPE & m) " << endl;
596 #endif
597
598     // unroll loops to depth of length 4
599
600     int N = this->size();
601
602     int Nminus4 = N-4;
603     int i;
604
605     for (i=0; i<Nminus4; )
606     {
607         this->p_[i++] = m;
608         this->p_[i++] = m;
609         this->p_[i++] = m;
610         this->p_[i++] = m;
611     }
612
613     for (; i<N; this->p_[i++] = m); // finish off last piece...
614
615 #ifdef TRACE_VEC
616     cout << "< FMV_Vector<TYPE>::operator=(const TYPE & m) " << endl;
617 #endif
618     return *this;
619 }
620
621
622 #include "mvblas_GR.h"
623
624 #endif
625 // _MV_VECTOR_TPL_H_
626

```

## 7.391 diagpre\_double\_GR.h

```

1 /+++++
2 /*          *****   ***                               SparseLib++   */
3 /*          *****  **  ***          ***          ***                               v. 1.5c   */
4 /*          *****   ***          ***** *****                               */
5 /*          *****   ***          ***** *****                               R. Pozo     */
6 /*          **  *****   ***  **   ***          ***                               K. Remington */
7 /*          *****   *****                               A. Lumsdaine */
8 /+++++
9 /*
10 /*
11 /*          SparseLib++ : Sparse Matrix Library   */

```



```

12 /*
13 /*           National Institute of Standards and Technology
14 /*           University of Notre Dame
15 /*           Authors: R. Pozo, K. Remington, A. Lumsdaine
16 /*
17 /*           NOTICE
18 /*
19 /* Permission to use, copy, modify, and distribute this software and
20 /* its documentation for any purpose and without fee is hereby granted
21 /* provided that the above notice appear in all copies and supporting
22 /* documentation.
23 /*
24 /* Neither the Institutions (National Institute of Standards and Technology,
25 /* University of Notre Dame) nor the Authors make any representations about
26 /* the suitability of this software for any purpose. This software is
27 /* provided "as is" without expressed or implied warranty.
28 /*
29 /*+++++*/
30
31 // modification GR
32 // 1) on utilise une interface unique pour tous les préconditionneurs
33 // d'où l'héritage d'une classe virtuelle
34 // 2) on utilise systématiquement les classes templates MV++
35 // 3) introduction des matrices carrées et des matrices bandes
36
37 #ifndef DIAGPRE_GR_H
38 #define DIAGPRE_GR_H
39
40 // #include "vecdefs.h"
41 #include "vecdefs_GR.h" // modif GR
42 // #include VECTOR_H
43
44 #include "comprow_double.h"
45 #include "compcol_double.h"
46
47 #include "pre_cond_double.h"
48 #include "Mat_abstraite.h"
49
50 class DiagPreconditioner_double : public Pre_cond_double
51 {
52
53 private:
54     VECTOR_double diag_;
55
56 public:
57     DiagPreconditioner_double (const CompCol_Mat_double &);
58     DiagPreconditioner_double (const CompRow_Mat_double &);
59     // cas d'une matrice abstraite
60     DiagPreconditioner_double(const Mat_abstraite &A);
61     ~DiagPreconditioner_double (void) { };
62
63     // méthodes-----
64
65     VECTOR_double solve (const VECTOR_double &x) const;
66     VECTOR_double trans_solve (const VECTOR_double &x) const;
67
68 protected : // modif GR : les deux fonctions suivantes ne sont pas génériques
69             // d'où le status protected
70     const double& diag(int i) const { return diag_(i); }
71     double& diag(int i) { return diag_(i); }
72
73     // modif GR : cette fonction est interne, elle n'avait pas de prototype !!
74     int CopyInvDiagonals(int n, const int *pntr, const int *indx,
75         const double *sa, double *diag);
76     // surcharge de la fonction pour les matrices autres que celle sparse
77     int CopyInvDiagonals(const Mat_abstraite& A);
78 };
79
80 #endif

```

## 7.392 icpre\_double\_GR.h

```

1 /*+++++*/
2 /*           *****   ***                               SparseLib++   */
3 /*           ***** **  ***                               v. 1.5c       */
4 /*           *****   **  ***** *****              */
5 /*           *****   ***   ***** *****              R. Pozo       */
6 /*           **  *****   *** **   ***   ***           K. Remington  */
7 /*           *****   *****              A. Lumsdaine  */
8 /*+++++*/
9 /*
10 /*
11 /*           SparseLib++ : Sparse Matrix Library
12 /*
13 /*           National Institute of Standards and Technology

```

```

14 /*          University of Notre Dame          */
15 /*          Authors: R. Pozo, K. Remington, A. Lumsdaine          */
16 /*          */
17 /*          NOTICE          */
18 /*          */
19 /* Permission to use, copy, modify, and distribute this software and
20 /* its documentation for any purpose and without fee is hereby granted
21 /* provided that the above notice appear in all copies and supporting
22 /* documentation.
23 /*
24 /* Neither the Institutions (National Institute of Standards and Technology,
25 /* University of Notre Dame) nor the Authors make any representations about
26 /* the suitability of this software for any purpose. This software is
27 /* provided "as is" without expressed or implied warranty.          */
28 /*          */
29 /*+++++++*/
30
31 // modification GR
32 // 1) on utilise une interface unique pour tous les préconditionneurs
33 //    d'où l'héritage d'une classe virtuelle
34 // 2) on utilise systématiquement les classes templates MV++
35
36 #ifndef ICPRE_GR_H
37 #define ICPRE_GR_H
38
39 // #include "vecdefs.h"
40 #include "vecdefs_GR.h" // modif GR
41 // #include VECTOR_H // modif GR
42 #include "compcol_double.h"
43 #include "comprow_double.h"
44
45 #include "pre_cond_double.h"
46 #include "Mat_abstraite.h"
47
48 class ICPreconditioner_double : public Pre_cond_double
49 {
50
51 private:
52     VECTOR_double val_;
53     VECTOR_int   pntr_;
54     VECTOR_int   indx_;
55     int nz_;
56     int dim_[2];
57
58 public:
59     ICPreconditioner_double(const CompCol_Mat_double &A);
60     ICPreconditioner_double(const CompRow_Mat_double &A);
61     // cas d'une matrice abstraite
62     ICPreconditioner_double(const Mat_abstraite &A);
63
64     // méthodes-----
65
66     ~ICPreconditioner_double(void){};
67
68     VECTOR_double solve(const VECTOR_double &x) const;
69     VECTOR_double trans_solve(const VECTOR_double &x) const;
70
71     //===== protégée =====
72
73 protected :
74     // fonction interne d'initialisation pour éviter la recopie
75     void Init_CompCol_Mat_double(const CompCol_Mat_double &A);
76     void Init_CompRow_Mat_double(const CompRow_Mat_double &A);
77     void Init_Mat_creuse_CompCol(const Mat_abstraite &A);
78 };
79
80 #endif

```

## 7.393 ilupre\_double\_GR.h

```

1 /*+++++++*/
2 /*          *****          SparseLib++          */
3 /*          ***** ** ***          v. 1.5c          */
4 /*          *****          *****          */
5 /*          *****          *****          R. Pozo          */
6 /*          ** ***** ** **          ** ***          K. Remington          */
7 /*          *****          *****          A. Lumsdaine          */
8 /*+++++++*/
9 /*          */
10 /*          */
11 /*          SparseLib++ : Sparse Matrix Library          */
12 /*          */
13 /*          National Institute of Standards and Technology          */
14 /*          University of Notre Dame          */
15 /*          Authors: R. Pozo, K. Remington, A. Lumsdaine          */

```

```

16 /*                                                                                               */
17 /*                                                                                               */
18 /*                                                                                               */
19 /* Permission to use, copy, modify, and distribute this software and                               */
20 /* its documentation for any purpose and without fee is hereby granted                               */
21 /* provided that the above notice appear in all copies and supporting                               */
22 /* documentation.                                                                                   */
23 /*                                                                                               */
24 /* Neither the Institutions (National Institute of Standards and Technology,                               */
25 /* University of Notre Dame) nor the Authors make any representations about                               */
26 /* the suitability of this software for any purpose. This software is                               */
27 /* provided "as is" without expressed or implied warranty.                                       */
28 /*                                                                                               */
29 /*+++++*/
30
31 // modification GR
32 // 1) on utilise une interface unique pour tous les préconditionneurs
33 //    d'où l'héritage d'une classe virtuelle
34 // 2) on utilise systématiquement les classes templates MV++
35
36 #ifndef ILUPRE_GR_H
37 #define ILUPRE_GR_H
38
39 // #include "vecdefs.h"
40 #include "vecdefs_GR.h" // modif GR
41 // #include VECTOR_H // modif GR
42 #include "comprow_double.h"
43 #include "compcol_double.h"
44
45 #include "pre_cond_double.h"
46 #include "Mat_abstraite.h"
47
48 class CompCol_ILUPreconditioner_double : public Pre_cond_double
49 {
50
51 private:
52     VECTOR_double l_val_;
53     VECTOR_int l_colptr_;
54     VECTOR_int l_rowind_;
55     int l_nz_;
56
57     VECTOR_double u_val_;
58     VECTOR_int u_colptr_;
59     VECTOR_int u_rowind_;
60     int u_nz_;
61
62     int dim_[2];
63
64 public:
65     CompCol_ILUPreconditioner_double(const CompCol_Mat_double &A);
66     // cas d'une matrice abstraite
67     CompCol_ILUPreconditioner_double(const Mat_abstraite &A);
68     ~CompCol_ILUPreconditioner_double(void){};
69
70     // méthodes-----
71
72     VECTOR_double solve(const VECTOR_double &x) const;
73     VECTOR_double trans_solve(const VECTOR_double &x) const;
74
75     //===== protégée =====
76
77 protected :
78     // fonction interne d'initialisation pour éviter la recopie
79     void Init_CompCol_Mat_double(const CompCol_Mat_double &A);
80     void Init_Mat_creuse_CompCol(const Mat_abstraite &A);
81
82 };
83
84
85 class CompRow_ILUPreconditioner_double : public Pre_cond_double
86 {
87
88 private:
89     VECTOR_double l_val_;
90     VECTOR_int l_rowptr_;
91     VECTOR_int l_colind_;
92     int l_nz_;
93
94     VECTOR_double u_val_;
95     VECTOR_int u_rowptr_;
96     VECTOR_int u_colind_;
97     int u_nz_;
98
99     int dim_[2];
100
101 public:
102     CompRow_ILUPreconditioner_double(const CompRow_Mat_double &A);

```

```

103 // cas d'une matrice abstraite
104 CompRow_ILUPreconditioner_double(const Mat_abstraite &A);
105 ~CompRow_ILUPreconditioner_double(void){};
106
107 // méthodes-----
108
109 VECTOR_double solve(const VECTOR_double &x) const;
110 VECTOR_double trans_solve(const VECTOR_double &x) const;
111
112 //===== protégée =====
113
114 protected :
115 // fonction interne d'initialisation pour éviter la recopie
116 void Init_CompRow_Mat_double(const CompRow_Mat_double &A);
117
118 };
119
120 // introduction d'une classe qui regroupe les différentes possibilités de
121 // stockage matricielle
122
123 class ILUPreconditioner_double : public Pre_cond_double
124 {
125 public:
126 // CONSTRUCTEUR
127
128 // cas d'une matrice abstraite
129 ILUPreconditioner_double(const Mat_abstraite &A);
130 // destructeur
131 ~ILUPreconditioner_double(void);
132
133 // méthodes-----
134
135 VECTOR_double solve(const VECTOR_double &x) const
136 { return preCond_mat->solve(x);};
137 VECTOR_double trans_solve(const VECTOR_double &x) const
138 { return preCond_mat->trans_solve(x);};
139
140 //===== protégée =====
141
142 private:
143 // pour l'instant deux types de matrices possibles
144 Pre_cond_double * preCond_mat;
145 };
146
147 #endif

```

## 7.394 pre\_cond\_double.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 * DATE: 03/01/01 *
32 * * $ *
33 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
34 * * $ *
35 * PROJET: Herezh++ *
36 * * $ *
37 *****/

```

```

38 *      BUT: Faire une interface générique pour les préconditionnements. *
39 *                                          $ *
40 *      ***** *
41 *      VERIFICATION: *
42 * *
43 *      ! date ! auteur ! but ! *
44 *      ----- *
45 *      ! ! ! ! *
46 *                                          $ *
47 *      ***** *
48 *      MODIFICATIONS: *
49 *      ! date ! auteur ! but ! *
50 *      ----- *
51 *                                          $ *
52 *****/
53 #ifndef PRE_COND_DOUBLE_H
54 #define PRE_COND_DOUBLE_H
55
56 #include "vecdefs_GR.h"
57
58 #include "comprow_double.h"
59 #include "compcol_double.h"
60
61 /// @addtogroup Les_classes_Matrices
62 /// @{
63 ///
64
65
66 class Pre_cond_double {
67
68 public:
69     virtual ~Pre_cond_double(){}; //destructeur virtuel pour l'appel des vrais destructeur
70     // résolution du système AX=b
71     virtual VECTOR_double solve (const VECTOR_double &x) const = 0;
72     // résolution du système Xt A = b
73     virtual VECTOR_double trans_solve (const VECTOR_double &x) const = 0;
74
75 };
76 /// @} // end of group
77
78 #endif

```

## 7.395 Assemblage.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97 *
32 * * *
33 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr) *
34 * * *
35 *      PROJET:    Herezh++ *
36 * * *
37 *****
38 *      BUT:      Assemblage des grandeurs locales dans les grandeurs *
39 *      globales. *
40 * * *

```

```

41 *      *
42 *      VERIFICATION:
43 *
44 *      ! date ! auteur ! but !
45 *      -----
46 *      ! ! ! ! $
47 *
48 *      *
49 *      MODIFICATIONS:
50 *      ! date ! auteur ! but !
51 *      -----
52 *      $
53 *****/
54 #ifndef ASSEMBLAGE_H
55 #define ASSEMBLAGE_H
56
57 #include "Mat_abstraite.h"
58 #include "DdlElement.h"
59 #include "Noeud.h"
60 #include "Tableau_T.h"
61 #include "Nb_assemb.h"
62
63 /// @addtogroup Les_classes_Matrices
64 /// @{
65 ///
66
67
68 class Assemblage
69 {
70 public :
71 // CONSTRUCTEURS :
72 // par défaut
73 Assemblage ();
74 // fonction d'un cas d'assemblage
75 Assemblage (Nb_assemb nb_cas) : nb_casAssemb(nb_cas) {};
76 // de copie
77 Assemblage (const Assemblage& a) :
78 nb_casAssemb(a.nb_casAssemb)
79 {};
80
81 // DESTRUCTEUR :
82
83 // METHODES PUBLIQUES :
84
85 // assemblage du second membre
86 // vecglob : second membre global
87 // vecloc : contribution de l'element au second membre
88 // tab_ddl : ddl de l'element, tab_noeud : tableau de noeud de l'element
89 // a l'aide du tab_ddl et de tab_noeud on recupere le pointeur
90 // d'assemblage de chaque noeud
91 void AssemSM (Vecteur& vecglob, const Vecteur& vecloc, const DdlElement& tab_ddl,
92 const Tableau<Noeud *>&tab_noeud);
93
94 // assemblage de plusieurs second membre en parallèle
95 // vecglob : les seconds membres globaux
96 // vecloc : contributions de l'element aux seconds membres
97 // tab_ddl : ddl de l'element, tab_noeud : tableau de noeud de l'element
98 // a l'aide du tab_ddl et de tab_noeud on recupere le pointeur
99 // d'assemblage de chaque noeud
100 void AssemSM (Tableau<Vecteur>& vecglob, const Tableau<Vecteur*>& vecloc,
101 const DdlElement& tab_ddl, const Tableau<Noeud *>&tab_noeud);
102
103 // assemblage des matrices symetriques
104 // seul la moitiee supérieure de la matrice est assemblée
105 // matglob : matrice globale
106 // matloc : matrice locale ,
107 // tab_ddl : ddl de l'element, tab_noeud : tqbleau de noeud de l'element
108 void AssembMatSym (Mat_abstraite & matglob, const Mat_abstraite & matloc,
109 const DdlElement& tab_ddl, const Tableau<Noeud *>&tab_noeud);
110 // assemblage des matrices non symetriques
111 // toute la matrice est assemblée
112 // matglob : matrice globale
113 // matloc : matrice locale ,
114 // tab_ddl : ddl de l'element, tab_noeud : tqbleau de noeud de l'element
115 void AssembMatnonSym (Mat_abstraite & matglob, const Mat_abstraite & matloc,
116 const DdlElement& tab_ddl, const Tableau<Noeud *>&tab_noeud);
117
118 // assemblage uniquement de la diagonale de matloc dans vecglob
119 // vecglob : second membre global
120 // matloc : matrice locale ,
121 // tab_ddl : ddl de l'element, tab_noeud : tqbleau de noeud de l'element
122 void AssembDiagonale (Vecteur& vecglob, const Mat_abstraite & matloc,
123 const DdlElement& tab_ddl, const Tableau<Noeud *>&tab_noeud);
124
125 // assemblage diagonale, = une majoration de la matrice des valeurs propre des matloc
126 // vecglob : second membre global

```

```

127 // matloc : matrice locale ,
128 // tab_ddl : ddl de l'element,tab_noeud : tqbleau de noeud de l'element
129 void AssembDiagoMajorValPropre (Vecteur& vecglob,const Mat_abstraite & matloc,
130                               const DdlElement& tab_ddl,const Tableau<Noeud *>&tab_noeud);
131
132
133 // récup en lecture le numéro d'assemblage auquel l'instance se rapporte
134 Nb_assemb Nb_cas_assemb() {return nb_casAssemb;};
135
136 // change le cas d'assemblage
137 void Change_cas_assemblage(Nb_assemb nb_cas) { nb_casAssemb = nb_cas;};
138
139 protected :
140 // VARIABLES PROTEGEES :
141 // définition du numéro de cas d'assemblage auquel l'instance
142 // se rapporte
143 Nb_assemb nb_casAssemb;
144
145 // CONSTRUCTEURS :
146
147 // DESTRUCTEUR :
148
149 // METHODES PROTEGEES :
150
151 };
152 /// @} // end of group
153
154 #endif

```

## 7.396 Condilinaire.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *      *****
38 *      BUT:      Definition d'un container pour une condition limite
39 *               lineaire.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date !   auteur !           but
45 *      -----
46 *      !       !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !           but
51 *      -----
52 *
53 *****/

```

```

54 #ifndef CONDILINEAIRE_H
55 #define CONDILINEAIRE_H
56
57 #include "Noeud.h"
58 #include "Nb_assemb.h"
59
60
61 /// @addtogroup Les_classes_Matrices
62 /// @{
63 ///
64
65
66 class Condilinaire
67 { // surcharge de l'opérateur de lecture typée
68   // en fait ces fonctions ne doivent pas être utilisées, elles existent uniquement
69   // pour que Tableau_T puisse exister
70   friend istream & operator » (istream & ent, Condilinaire &)
71   { Sortie(1); return ent; }; // erreur
72   // surcharge de l'opérateur d'écriture typée
73   friend ostream & operator « (ostream & sort, const Condilinaire &)
74   { Sortie(1); return sort; }; // erreur
75
76 public :
77   // CONSTRUCTEURS :
78   // par défaut
79   Condilinaire () :
80     pt(), val(), beta(0.), t_noeud(), t_enu(), casAssemb()
81     , Uk_impose(ConstMath::tresgrand)
82   {} ;
83   // cas ou l'on connait toutes les infos, sauf Uk_impose, qui est une variable de stockage gérée
   // indépendamment
84   Condilinaire (Tableau <Enum_ddl>& t_enuu, const Tableau<int> & ptt, const Vecteur& vall
85     , double betar, int posiddl, const Tableau < Noeud *>& t_n) :
86     pt(ptt), val(vall), beta(betar), iddl(posiddl), t_noeud(t_n)
87     , t_enu(t_enuu), casAssemb()
88     , Uk_impose (ConstMath::tresgrand)
89   {} ;
90   // cas ou l'on connait les infos relatives uniquement aux noeuds, aux enum ddl
91   Condilinaire (Tableau <Enum_ddl>& t_enuu, const Tableau < Noeud *>& t_n) :
92     t_noeud(t_n), t_enu(t_enuu), casAssemb()
93   {} ;
94   // de copie
95   Condilinaire (const Condilinaire& a) :
96     pt(a.pt), val(a.val), beta(a.beta), iddl(a.iddl), t_noeud(a.t_noeud)
97     , t_enu(a.t_enu), casAssemb(a.casAssemb), Uk_impose(a.Uk_impose)
98   {} ;
99   // DESTRUCTEUR :
100  ~Condilinaire () {} ;
101   // METHODES PUBLIQUES :
102   // surcharge de l'opérateur =
103   Condilinaire& operator = (const Condilinaire& cond);
104
105   inline const Tableau<int>& Pt_t() const { return pt; };
106   Tableau<int>& ChangePt() { return pt; }; // acces lecture / écriture
107   inline const Vecteur& Val() const { return val; };
108   Vecteur& Valchange() { return val; }; // acces lecture / écriture
109   void ChangeCoeff(const Vecteur& v) { val=v; };
110   inline double Beta() const { return beta; };
111   double& BetaChange() { return beta; }; // acces lecture / écriture
112   void ChangeBeta(const double& x) { beta = x; };
113
114   // dans le cas de la mise en place de la CL à partir d'un changement de repère, on peut stocker la
   // valeur imposée avant chg de repère
115   // cette valeur a imposer sur le ddl avant changement de repère, correspondra à beta après chg de
   // repère
116   // Uk_impose sert uniquement de stockage, mais n'est pas forcément cohérent avec la CL, sa
   // manipulation est faite en dehors de la classe
117   // via les deux méthodes qui suivent
118   void ChangeUk_impose(double Uk_new) {Uk_impose=Uk_new;};
119   // lorsque la valeur retourné par Val_Uk_impose() == ConstMath::tresgrand, cela signifie qu'elle
   // n'est pas a considérer
120   const double & Val_Uk_impose() const {return Uk_impose;};
121
122   Noeud* Noe() {return t_noeud(1);}; // acces lecture / écriture
123   void Change_tab_enum(const Tableau <Enum_ddl> & t_enuu) {t_enu = t_enuu;};
124   // le tableau de noeuds associé
125   const Tableau < Noeud *>& TabNoeud() const { return t_noeud;};
126   void ChangeTabNoeud(const Tableau < Noeud *>& t_n) {t_noeud = t_n;};
127
128   // Iddl() -> le numéro d'ordre dans sa famille, du ddl bloqué
129   // NB: ce n'est pas la position du ddl dans le noeud !!, cette dernière est: Tab_Enum()(1)
130   inline const int& Iddl() const { return iddl;};
131   int& ChangeIddl() { return iddl;}; // acces lecture / écriture
132
133   // changement de la taille des tableaux de pointeur et valeurs
134   void Change_taille(int taille) {pt.Change_taille(taille); val.Change_taille(taille);};
135

```



```

136 // mise en place des pointeurs de ddl d'assemblage
137 const Condilinaire& ConditionPourPointeursAssemblage(const Nb_assemb& nb_casAssemb);
138 const Nb_assemb& CasAssemb() const {return casAssemb;};
139
140 // retour du tableau d'énuméré correspondant aux coefficients de la CLL
141 const Tableau <Enum_ddl >& Tab_Enum() const {return t_enu;};
142
143 // ramène la différence maxi qui existe entre les numéros de noeuds de la condition linéaire
144 // peut-être utilisé pour calculer une largeur de bande par exemple
145 // important: cette méthode n'est valide que si les numéros de noeuds sont tous différents
146 // donc que la numérotation interne des noeuds a été changé pour cela
147 // NB: en fonctionnement normal, ce n'est pas le cas ! sauf dans le cas où un seul maillage existe
148 // voir LesMaillages::Renumerotation( pour un exemple d'utilisation
149 int DiffMaxiNumeroNoeud()const;
150
151 // ramène la largeur de bande en ddl
152 // à cause de la condition linéaire
153 // casAssemb : donne le cas d'assemblage a prendre en compte
154 // les condi linéaires ne donnent pas des largeurs de bande sup et inf égales !!!
155 // I/O : demi = la demi largeur de bande maxi ,
156 // total = le maxi = la largeur sup + la largeur inf +1
157 void Largeur_Bande(int& demi,int& total,const Nb_assemb& casAssemb);
158
159 // affichage à l'écran des infos de la CL
160 void Affiche() const ;
161
162 //----- lecture écriture de restart -----
163 // la lecture ramène en retour le numéro de maillage et le numéro de
164 // noeud sur lequel s'applique la condition limite
165 // il est nécessaire ensuite d'utiliser la fonction Change_noeud pour `
166 // attribuer le noeud
167 void Lecture_base_info(Tableau <int>& numMaillage, ifstream& ent,Tableau <int>& numNoeud) ;
168 void Ecriture_base_info(ofstream& sort) ;
169
170 protected :
171 // VARIABLES PROTEGEES :
172 Tableau<int> pt; //tableau des pointeurs de ddl concerne, pt(i) = la position du ddl i
173 // dans la matrice globale
174 Vecteur val; // tableau des coefficients de la condition lineaire
175 double beta; // valeur beta a laquelle est egale la condition lineaire
176 // dans le cas de la mise en place de la CL à partir d'un changement de repère, on peut stocker la
177 // valeur imposée avant chg de repère
178 double Uk_impose; // valeur a imposer sur le ddl avant changement de repère, qui correspondra à beta
179 // après chg de repère
180 // Uk_impose sert uniquement de stockage, mais n'est pas forcément cohérent avec la CL, sa
181 // manipulation est faite en dehors de la classe
182 // via : ChangeUk_impose et Val_Uk_impose
183 Nb_assemb casAssemb; // le cas d'assemblage associé
184
185 // le blocage de condition est appliquee sur le ddl numero "iddl" du noeud "noe"
186 Tableau <Enum_ddl > t_enu; // tableau des identificateur de ddl de la CLL
187 // t_enu(1) est l'identificateur du ddl qui est bloqué pour la CLL
188 // lorsque seul t_enu(1) existe, cela signifie qu'il faut construire
189 // les indices,
190
191 // iddl -> le numéro d'ordre dans sa famille, du ddl bloqué
192 // NB: ce n'est pas la position du ddl dans le noeud !, cette dernière est: Tab_Enum()(1)
193 int iddl;
194 // le tableau des noeuds de la CLL, le premier contient la condition
195 Tableau < Noeud * > t_noeud;
196
197 // METHODES PROTEGEES :
198 };
199 /// @} // end of group
200 #endif

```

## 7.397 CondLim.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //

```

```

15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *   ****
38 *   BUT:      Def et application de differentes conditions limites.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date !   auteur !           but
44 *   -----
45 *   !           !           !
46 *
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date !   auteur !           but
50 *   -----
51 *
52 *   *****/
53 #ifndef CONDLIM_H
54 #define CONDLIM_H
55
56 #include "Mat_abstraite.h"
57 #include "Vecteur.h"
58 #include <list>
59 #include "MathUtil.h"
60 #include "Mat_pleine.h"
61
62
63 /// @addtogroup Les_classes_Matrices
64 /// @{
65 ///
66
67
68 class CondLim
69 { // pour pouvoir faire des tableaux de condlim on introduit les surcharges de lecture
70 // écriture, mais en fait elles ne doivent pas être utilisé d'où une sortie d'erreur
71 // surcharge de l'operator de lecture typée
72 friend istream & operator » (istream & ent, CondLim &)
73 { cout << "\n erreur, la surcharge de lecture n'est pas implanté "
74   << "\n operator » (istream & ent, CondLim &) " ;
75   Sortie(1); return ent;
76 };
77 // surcharge de l'operator d'écriture typée
78 friend ostream & operator « (ostream & sort, const CondLim &)
79 { cout << "\n erreur, la surcharge d'écriture n'est pas implanté "
80   << "\n operator « (ostream & sort, const CondLim &)";
81   Sortie(1); return sort;
82 };
83
84 public :
85 // CONSTRUCTEURS :
86 CondLim();
87 // DESTRUCTEUR :
88 ~CondLim();
89 // METHODES PUBLIQUES :
90 //=====
91 // IMPORTANT !! = lorsque l'on a des conditions limites a imposer a un second =
92 // membre et a une matrice , il faut d'abord impose a la matrice puis =
93 // apres au second membre =
94 //=====
95 // valeur imposee au second membre
96 // vecglob : le second membre, i : la position globale du ddl impose
97 // val : la valeur a imposer
98 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que vecglob
99 // mais sans sauvegarde (correspond par exemple à une partie de vecglob)
100 void Val_imposee_Sm(Vecteur& vecglob,int i,double val,Vecteur* vec2);

```

```

101 // valeur imposee a la matrice et au second membre si la valeur est
102 // differente de zero
103 // matglob : la matrice, i : la position globale du ddl impose
104 // val : la valeur a imposer
105 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que
vecglob
106 //      mais sans sauvegarde (correspond par exemple à une partie de vecglob)
107 void Val_imposee_Mat(Mat_abstraite & matglob,Vecteur& vecglob,
108                    int i,double val,Vecteur* vec2);
109 // cas particulier de valeur imposee a une matrice
110 // c'a-dire val sur la diagonale
111 // et des zéros sur le reste de la ligne et colonne correspondantes
112 // matglob : la matrice, i : la position globale du ddl impose
113 // dans ce cas-ci il n'y a aucune information sauvegardée, des valeurs modifiées
114 // cela signifie qu'après cette fonction, les appels aux routines pour la
115 // remontée aux réactions, n'ont aucun sens
116 void Val_imposSimple_Mat(Mat_abstraite & matglob,int i,double val);
117 // remontee aux efforts apres resolution
118 // ceci pour la ligne i dans le cas de ddl bloque
119 // la matrice est utilisee pour restorer les lignes supprimee
120 // par les conditions limites, elle est donc modifiee
121 // par cette operation, parcontre ses valeurs initiales ne sont pas utilisees
122 double RemonteDdlBloqueMat(Mat_abstraite & matglob,Vecteur& solution, int i);
123
124 // retourne la valeur absolu du maxi des efforts extérieurs
125 // et le numero d'assemblage correspondant
126 double MaxEffort(int & ili);
127
128 // retourne la valeur initiale au second membre avant condition limite
129 // en fonction du pointeur d'assemblage
130 double ValReact(int & ili);
131
132 //-----
133 //      cas de conditions limites lineaires entre plusieurs ddl
134 //-----
135 // la condition est imposee sur la matrice et sur le second membre
136 // la mise en place des condition lineaire doit ce faire en deux temps
137 // premier temps : preparation des conditions, c'est a dire expression
138 // de la raideur et du second membre dans les reperes locaux ceci pour
139 // TOUS LES CONDITIONS
140 // second temps : seulement une fois que TOUTES LES changements de reperes
141 // sont effectuee, on impose les valeurs de second memebres apres ou avant
142 // les valeurs fixe.
143
144 // pt : tableau des pointeurs de ddl concerne, pt(i) = la position du ddl i
145 // dans la matrice globale
146 // val : tableau des coefficients de la condition lineaire
147 // valeur : valeur a laquelle est egale la condition lineaire
148 // cond lineaire -> somme des val(i) * ddl(pt(i)) = valeur
149 // la procedure modifie les reperes d'expression des ddl, mais sauvegarde
150 // les infos permettant de reconstruire les reperes initiaux
151 // !!! par principe, c'est la direction du premier indice qui est bloquee
152 // il faut donc ecrire la condition en consequence
153
154 // vec2 : est un second vecteur éventuel (si != NULL) sur lequel on impose les mêmes CL que
vecglob
155 //      mais sans sauvegarde (correspond par exemple à une partie de vecglob)
156
157 // premier temps: changement de repère
158
159 void CondlineaireCHRepere(Mat_abstraite & matglob,Vecteur& vecglob,
160                          const Tableau<int> & pt,const Vecteur& val, double valeur,Vecteur* vec2);
161
162 // second temps : imposition des blocages correspondant
163 //      aux conditions lineaires
164
165 void CondlineaireImpose (Mat_abstraite & matglob,Vecteur& vecglob,Vecteur* vec2);
166
167
168 //expression du vecteur resultat dans les reperes initiaux
169 // sol : la solution, est modifiee et retournee dans les reperes initiaux
170 void RepInitiaux( Vecteur& sol);
171
172 // application d'une condition linéaire seule, avec en retour, la situation de la condition
linéaire
173 // imposée, ramené dans le repère initial
174 void CondiLineaireImposeComplet(Mat_abstraite & matglob,Vecteur& vecglob,
175                                const Tableau<int> & pt,const Vecteur& val, double valeur,Vecteur* vec2);
176
177
178 // remise a zero des sauvegardes de second membre et de matrice
179 void EffaceSauvegarde();
180
181 // remise a zero des sauvegardes de condition lineaire
182 void EffaceCoLin();
183
184

```

```

185
186 private :
187 // VARIABLES PROTEGEES :
188 // concernant une valeur imposee pour le second membre, on defini une
189 // class qui permet d'enregistrer les ddl impose
190 class ImpSM
191 { public :
192     ImpSM () {}; // par default
193     ImpSM (int i, double val) : iligne(i), valeur(val) {};
194     ImpSM (const ImpSM & a) : iligne(a.iligne), valeur(a.valeur) {};
195     ~ImpSM () {};
196     ImpSM& operator=( const ImpSM& a) ; // assignment
197
198     int iligne; // ligne ou le ddl est impose
199     double valeur; // sauvegarde de la valeur existant avant le ddl impose
200 };
201
202 // on definit une liste STL comme container
203 list <ImpSM> VImpSM;
204
205 //concernant une valeur imposee pour la raideur, on defini une
206 // class qui permette d'enregistrer les modifications effectuees
207 // sur la raideur
208 class ImpRaid
209 { public :
210     ImpRaid (); // par default
211     ImpRaid (int i,const Vecteur& vL,const Vecteur& vC,double val);
212     ImpRaid (const ImpRaid & a) ; // constructeur de copie
213     ~ImpRaid() {}; // destructeur
214     ImpRaid& operator=( const ImpRaid& a) ; // assignment
215
216     int ilicol; // ligne et colonne ou le ddl est impose
217     Vecteur ligne; // ligne du ddl impose
218     Vecteur colonne; // colonne du ddl impose
219     double valeur; // valeur du ddl impose
220 };
221 // on definit une liste STL comme container
222 list <ImpRaid> VImpRaid;
223
224 //concernant les conditions lineaires, on defini une
225 // class qui permette d'enregistrer le changement de repere
226 class lineaires
227 { public :
228     lineaires (); // par default
229     lineaires (int i,const Vecteur& vL,const Tableau<int> & ptt,double vale);
230     lineaires (const lineaires & a) ; // constructeur de copie
231     ~lineaires() {}; // destructeur
232     lineaires& operator=( const lineaires& a) ; // assignment
233
234     int indice; // numero dans le tableau pt qui permet d'obtenir les
235     //ligne et colonne du ddl modifie en condition lineaire
236     Vecteur val; // vecteur des coefficients de la condition lineaire
237     Tableau<int> pt; // pt : tableau des pointeurs de ddl concerne, pt(i) = la position du ddl i
238     // dans la matrice globale
239     double valeur; // valeur de la condition lineaire
240 };
241 // on definit une liste STL comme container
242 list <lineaires> Vlineaires;
243
244 // méthode pour faire une rotation de la matrice et du second membre suivant un vecteur
245 // vec2 : est un second vecteur éventuel (si != NULL)
246 void Rotation(Mat_abstraite & matglob,Vecteur& vecglob,
247             const Tableau<int> & pt,const Vecteur& val,Vecteur* vec2);
248
249 // déclaration d'une variable de travail, utilisée par CondlinaireCHRepere
250 static Mat_pleine matinter; // mise en commun pour éviter des constructions inutiles
251
252
253 // METHODES PROTEGEES :
254
255 };
256 /// @} // end of group
257
258 #endif

```

## 7.398 Frontiere\_initiale.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //

```

```

9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           09/02/2003
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *   BUT:            Visualisation de la frontiere initiale: partie générale
38 *                  c'est à dire indépendante du type de visualisation.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date ! auteur ! but
45 *   -----
46 *   ! ! !
47 *   $
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date ! auteur ! but
51 *   -----
52 *   $
53 *****/
54 #ifndef FRONTIERE_INITIALE_T
55 #define FRONTIERE_INITIALE_T
56
57 #include "OrdreVisu.h"
58
59 /// @addtogroup Les_sorties_generiques
60 /// @{
61 ///
62
63
64 class Frontiere_initiale : public OrdreVisu
65 {
66 public :
67     // CONSTRUCTEURS :
68     // par défaut
69     Frontiere_initiale () ;
70
71     // constructeur de copie
72     Frontiere_initiale (const Frontiere_initiale& algo);
73
74     // DESTRUCTEUR :
75     ~Frontiere_initiale () ;
76
77     // METHODES PUBLIQUES :
78     // execution de l'ordre
79     // tab_mail : donne les numéros de maillage concerné
80     // incre : numéro d'incrément qui en cours
81     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
82     // animation : indique si l'on est en animation ou pas
83     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
84     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail,LesMaillages * ,bool
unseul_incre,LesReferences*
85 ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
86 ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
87 ,bool animation,const map < string, const double * , std::less <string> >&
88
89     listeVarGlob
90 ,const List_io < TypeQuelconque >& listeVecGlob);
91 // choix de l'ordre, cet méthode peut entraîner la demande d'informations
92 // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
93 void ChoixOrdre();
94

```

```

93 protected :
94     // VARIABLES PROTEGEES :
95     bool filaire;
96     bool surface;
97     bool numero;
98     double Rcoull,Gcoull,Bcoull; // couleur en RGB du tracé filaire
99     double Rcoulf,Gcoulf,Bcoulf; // couleur en RGB du tracé des faces
100    double Rcouln,Gcouln,Bcouln; // couleur en RGB du tracé des numéros de noeud
101    double taille_numero; // taille des numéros
102
103    // METHODES PROTEGEES :
104
105 };
106 /// @} // end of group
107
108 #endif

```

## 7.399 color.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 #ifdef __cplusplus
31 extern "C" {
32 #endif
33
34 void x_open_display(void);
35 void x_close_display(void);
36 void x_get_palette(const char*, int*, int*, int*);
37
38 #ifdef __cplusplus
39     }
40 #endif

```

## 7.400 rgb.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,

```

```

21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 #ifndef RGB_H
31 #define RGB_H
32
33 #include <iostream>
34
35     using namespace std;
36
37 class Rgb
38 {
39     float r, g, b;
40
41     public :
42         Rgb();
43         Rgb(float, float, float);
44
45     friend istream& operator>>(istream& i, Rgb& c);
46     friend ostream& operator<<(ostream& o, const Rgb& c);
47     friend int operator != (Rgb c1, Rgb c2);
48     friend Rgb operator + (Rgb c1, Rgb c2);
49     friend Rgb operator * (float f, Rgb c);
50     friend Rgb hexaToRgb(const char*);
51 };
52
53 ////////////////////////////////////////////////////
54 inline int operator != (Rgb c1, Rgb c2)
55 {
56     return (c1.r!=c2.r || c1.g!=c2.g || c1.b!=c2.b);
57 }
58
59 ////////////////////////////////////////////////////
60 // couleurs utilisees pour les spectres :
61 const Rgb Noir(.1, .1, .1);
62 const Rgb Bleu(.0, .0, 1.);
63 const Rgb Bleu2(.07, .28, .8);
64 const Rgb Bleu3(.0, .5, 1.);
65 const Rgb Cyan(.0, 1., 1.);
66 const Rgb Cyan2(.0, .7, .9);
67 const Rgb Vert(.0, 1., .0);
68 const Rgb Vert2(.0, .85, .2);
69 const Rgb Jaune(1., 1., .0);
70 const Rgb Rouge(1., .0, .0);
71 const Rgb Violet(1., .0, 1.);
72 const Rgb Magenta(.425, .0, .425);
73 const Rgb Mag(.3, .0, .5);
74 const Rgb Blanc(1., 1., 1.);
75
76 ////////////////////////////////////////////////////
77 // couleur par defaut pour les maillages:
78 // il faut esperer ne pas retrouver dans des legendes
79 // ou autre, car autrement ca ne sort pas
80 // (Cf sources de ostream& operator<<(ostream& o, const Rgb& c)
81 const Rgb DefautRgb(.0123, .0213, .0312);
82
83 #endif

```

## 7.401 spectre.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.

```

```

19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      31/12/2002
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *
38 *   BUT:   définition est manipulation du spectre de couleur,
39 *          util pour la visualisation d'isovaleurs.
40 *          Le programme utilise les spectres définis par
41 *          Bertrand orvoine dans livan++
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *
47 *   ! date !   auteur !           but
48 *   -----
49 *   !           !           !
50 *   *****
51 *
52 *   MODIFICATIONS:
53 *
54 *   ! date !   auteur !           but
55 *   -----
56 *
57 *   *****/
58
59 #ifndef SPECTRE_H
60 #define SPECTRE_H
61 #include "rgb.h"
62 #include "Tableau_T.h"
63 #include "Vecteur.h"
64
65 class Spectre
66 {
67 public :
68     enum EnumType_spectre {SPECTRE_STANDART=1,SPECTRE_ARCENCIEL,SPECTRE_THERMIQUE
69                             ,SPECTRE_MAXIMUM,SPECTRE_NOIRBLANCN,SPECTRE_GEO,SPECTRE_PHASER};
70     // Retourne le nom d'un type de spectre a partir de son identificateur de
71     // type enumere id_Type_spectre correspondant
72     char* Nom_Type_Spectre (const Spectre::EnumType_spectre id_TypeSpectre);
73     // Retourne l'identificateur de type enumere associe au nom du type de spectre
74     EnumType_spectre Id_nom_TypeSpectre (const char* nom_TypeSpectre);
75
76     // CONSTRUCTEUR
77     // constructeur par défaut
78     Spectre(); // initialisations par défaut au spectre standart, min et max = 0 1
79     // constructeur fonction d'un énuméré, et d'un mini et maxi équivalent en réel
80     Spectre(EnumType_spectre type_spect,double val_min=0.,double val_max=1.);
81     // constructeur de copie
82     Spectre(const Spectre& spect);
83
84     // DESTRUCTEUR
85     ~Spectre() {};
86
87     // Méthodes
88
89     // changement des valeurs min et max
90     void Change_min_max(const double& val_min,const double& val_max) ;
91     // changement du spectre courant
92     void Change_spectre_courant(EnumType_spectre nv_spectre);
93
94     // récup des maxi et mini du spectre
95     double Valeur_maxi() const { return val_des_coul(val_des_coul.Taille());};
96     double Valeur_mini() const { return val_des_coul(1);};
97     // récup du spectre en cours
98     EnumType_spectre TypeSpectreEnCours() const { return type_spectre;};
99     // nombre de couleurs de base du spectre en cours
100     int NbCouleurBaseSpectre() const {return spectre_courant.Taille();};
101
102     // définition d'une couleur en fonction d'une valeur numérique
103     // si la valeur est inférieure à la valeur mini du spectre, on retourne la valeur mini
104     // si la valeur est supérieure à la valeur maxi du spectre, on retourne la valeur maxi
105     Rgb CouleurVal(const double& val)const ;

```



```

105
106 // récup d'un tableau donnant la description de chaque spectre disponible
107 static const Tableau <string> & Description_spectres_disponibles() {return description_spectre;};
108
109 protected :
110 // données protégées
111 static Rgb spectre[7][8];
112 static Tableau <string> description_spectre; // description des différents spectres
113 class Initialisation_description_spectre
114 { public: Initialisation_description_spectre(); };
115 friend class Initialisation_description_spectre;
116 static Initialisation_description_spectre inti_spectre;
117
118 // ----- def du spectre courant -----
119 Tableau <Rgb> spectre_courant;
120 EnumType_spectre type_spectre;
121 // def du min max correspondant au spectre et des valeurs intermédiaires
122 Vecteur val_des_coul; // valeurs des couleurs du spectre
123 double delta_val; // incrément de valeur le long du spectre de base
124
125 };
126
127
128 #endif

```

## 7.402 Animation\_geomview.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *      *****
38 *      BUT:      Sortie de l'animation dans le cas de geomview.
39 *               Celle-ci est automatique lorsqu'il y a plus d'un increment
40 *               différent de 0.
41 *
42 *      *****
43 *
44 *      VERIFICATION:
45 *      ! date !   auteur !           but
46 *      -----
47 *      !           !           !
48 *
49 *      *****
50 *      MODIFICATIONS:
51 *      ! date !   auteur !           but
52 *      -----
53 *
54 *      *****/
55 #ifndef ANIMATION_GEOMVIEW_T
56 #define ANIMATION_GEOMVIEW_T
57

```

```

58
59 #include "OrdreVisu.h"
60 #include "Isovaleurs_geomview.h"
61 #include "ChoixDesMaillages_vrml.h"
62 #include "Deformees_geomview.h"
63 #include "Animation_vrml.h"
64
65 /// @addtogroup Les_sorties_geomview
66 /// @{
67 ///
68
69
70 class Animation_geomview : public OrdreVisu
71 {
72 public :
73 // CONSTRUCTEURS :
74 // par défaut
75 Animation_geomview () ;
76
77 // constructeur de copie
78 Animation_geomview (const Animation_geomview& ord);
79
80 // DESTRUCTEUR :
81 ~Animation_geomview () ;
82
83 // METHODES PUBLIQUES :
84
85 // execution de l'ordre
86 // tab_mail : donne les numéros de maillage concerné
87 // incre : numéro d'incrément qui en cours
88 // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
89 // animation : indique si l'on est en animation ou pas
90 // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
91 void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
unseul_incre,LesReferences*
92 ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
93 ,Resultats*,UtilLecture & entreePrinc,EnumTypeIncre type_incre,int incre
94 ,bool animation,const map < string, const double * , std::less <string> &
listeVarGlob
95 ,const List_io < TypeQuelconque >& listeVecGlob);
96
97 // choix de l'ordre, cet méthode peut entraîner la demande d'informations
98 // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
99 void ChoixOrdre();
100
101 // --- méthodes particulière ----
102 // initialisation d'une liaison avec une instance de classe d'isovaleur
103 void Jonction_isovaleur(const Isovaleurs_geomview * iso) {isovaleurs_geomview = iso;};
104
105 // initialisation d'une liaison avec une instance de classe de choix des maillages
106 void Jonction_ChoixDesMaillages(const ChoixDesMaillages_vrml* choix_m) {choix_mail = choix_m;};
107
108 // initialisation d'une liaison avec une instance de classe de choix de la déformée
109 void Jonction_Deformees_geomview(const Deformees_geomview* choix_def) {choix_deformee = choix_def;};
110
111 // initialisation des listes de coordonnées
112 void Init_liste_coordonnee();
113
114 // lecture des paramètres de l'ordre dans un flux
115 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
116 // écriture des paramètres de l'ordre dans un flux
117 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
118
119 protected :
120 // VARIABLES PROTEGEES :
121 double cycleInterval; // durée de l'animation
122 bool loop; // indique si on boucle ou pas
123 double startTime; // temps de démarrage en absolu depuis 1970
124 double stopTime; // temps de fin en absolu depuis 1970,
125 // si < a starttime on n'en tiend pas compte
126
127 const Isovaleurs_geomview * isovaleurs_geomview; // contient lorsqu'il est actif les couleurs des
noeuds
128 const ChoixDesMaillages_vrml* choix_mail; // contient lorsqu'il est actif le choix des maillages
129 const Deformees_geomview* choix_deformee; // contient lorsqu'il est actif les choix relatifs à la
déformée
130
131 // dans le cas de l'animation : définition des listes de positions et de nom si rattachant
132
133 // METHODES PROTEGEES :
134 };
135 /// @} // end of group
136
137 #endif

```

138

## 7.403 Deformees\_geomview.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           03/02/2003
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *****/
38 *   BUT:  Visualisation de la déformée en geomview.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date ! auteur ! but
44 *   -----
45 *   ! ! !
46 *   $
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date ! auteur ! but
50 *   -----
51 *   $
52 *****/
53 #ifndef DEFORMEES_GEOMVIEW_T
54 #define DEFORMEES_GEOMVIEW_T
55
56 #include "OrdreVisu.h"
57 #include "Isovaleurs_geomview.h"
58 #include "Deformees_vrml.h"
59 #include "ChoixDesMaillages_vrml.h"
60
61 /// @addtogroup Les_sorties_geomview
62 /// @{
63 ///
64
65
66 class Deformees_geomview : public Deformees_vrml
67 {
68 public :
69     // CONSTRUCTEURS :
70     // par défaut
71     Deformees_geomview () ;
72     // constructeur à utiliser
73     Deformees_geomview(const string& comment_som, const string& explication, const string& ordre) :
74         Deformees_vrml(comment_som,explication,ordre)
75     {};
76
77     // constructeur de copie
78     Deformees_geomview (const Deformees_geomview& algo);
79
80     // DESTRUCTEUR :

```

```

81     ~Deformees_geomview () ;
82
83     // METHODES PUBLIQUES :
84     // execution de l'ordre
85     // tab_mail : donne les numéros de maillage concerné
86     // incre : numéro d'incrément qui en cours
87     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
88     // animation : indique si l'on est en animation ou pas
89     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
90     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
91     unseul_incre,LesReferences*
92     ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
93     ,Resultats*,Utilecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
94     ,bool animation,const map < string, const double * , std::less <string> >&
95     listeVarGlob
96     ,const List_io < TypeQuelconque >& listeVecGlob);
97
98     protected :
99     // VARIABLES PROTEGEES :
100
101     // METHODES PROTEGEES :
102
103 };
104 /// @} // end of group
105
106 #endif

```

## 7.404 Fin\_geomview.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:     Herezh++
36 *
37 *      *****
38 *      BUT:  Fin de la visualisation en geomview
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date !   auteur !           but
44 *      -----
45 *      !           !           !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date !   auteur !           but
50 *      -----
51 *
52 *      *****/
53 #ifndef FIN_GEOMVIEW_T

```

```

54 #define FIN_GEOMVIEW_T
55
56 #include "OrdreVisu.h"
57 #include "Fin_vrml.h"
58
59 /// @addtogroup Les_sorties_geomview
60 /// @{
61 ///
62
63
64 class Fin_geomview : public Fin_vrml
65 {
66 public :
67     // CONSTRUCTEURS :
68     // par défaut
69     Fin_geomview () ;
70
71     // constructeur de copie
72     Fin_geomview (const Fin_geomview& algo);
73
74     // DESTRUCTEUR :
75     ~Fin_geomview () ;
76
77     // METHODES PUBLIQUES :
78     // execution de l'ordre
79     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
80                 ,LesLoisDeComp* ,DiversStockage*
81                 ,Charge* ,LesCondLim*
82                 ,LesContacts* ,Resultats* ,UtilLecture & ,OrdreVisu::EnumTypeIncre ,int
83                 ,bool ,const map < string, const double * , std::less <string> >&
84                 ,const List_io < TypeQuelconque >& listeVecGlob)
85     {};
86
87     // lecture des paramètres de l'ordre dans un flux
88     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
89     // écriture des paramètres de l'ordre dans un flux
90     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
91
92 private :
93     // VARIABLES PROTEGEES :
94
95     // METHODES PROTEGEES :
96
97 };
98 /// @} // end of group
99
100 #endif

```

## 7.405 Frontiere\_initiale\_geomview.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           09/02/2003
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++

```

```

36 *                                                                 $ *
37 *****
38 *   BUT:  Visualisation de la frontiere initiale: partie spécifique *
39 *       à géomview. *
40 *                                                                 $ *
41 *   //***** *
42 *   *
43 *   VERIFICATION: *
44 *   ! date ! auteur ! but ! *
45 *   ----- *
46 *   ! ! ! ! *
47 *   $ *
48 *   //***** *
49 *   MODIFICATIONS: *
50 *   ! date ! auteur ! but ! *
51 *   ----- *
52 *   $ *
53 *****/
54 #ifndef FRONTIERE_INITIALE_GEOMVIEW_T
55 #define FRONTIERE_INITIALE_GEOMVIEW_T
56
57 #include "OrdreVisu.h"
58 #include "Frontiere_initiale.h"
59
60 /// @addtogroup Les_sorties_geomview
61 /// @{
62 ///
63
64
65 class Frontiere_initiale_geomview : public Frontiere_initiale
66 {
67 public :
68 // CONSTRUCTEURS :
69 // par défaut
70 Frontiere_initiale_geomview () ;
71
72 // constructeur de copie
73 Frontiere_initiale_geomview (const Frontiere_initiale_geomview& algo);
74
75 // DESTRUCTEUR :
76 ~Frontiere_initiale_geomview () ;
77
78 // METHODES PUBLIQUES :
79 // execution de l'ordre
80 // tab_mail : donne les numéros de maillage concerné
81 // incre : numéro d'incrément qui en cours
82 // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
83 // animation : indique si l'on est en animation ou pas
84 // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
85 void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
unseul_incre,LesReferences*
86             ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
87             ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
88             ,bool animation,const map < string, const double * , std::less <string> >&
89             listeVarGlob
90             ,const List_io < TypeQuelconque >& listeVecGlob);
91
92 // lecture des paramètres de l'ordre dans un flux
93 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
94 // écriture des paramètres de l'ordre dans un flux
95 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
96
97 protected :
98 // VARIABLES PROTEGEES :
99 // METHODES PROTEGEES :
100
101 };
102 /// @} // end of group
103
104 #endif

```

## 7.406 Isovaleurs\_geomview.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //

```

```

11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29 //
30 /*****
31 *   DATE:      03/02/2003
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:  Visualisation des isovaleurs en geomview.
39 *
40 *   *****
41 *
42 *   REMARQUE:  Par rapport aux isovaleurs vrml, c'est
43 *              uniquement la sortie de la légende qui change.
44 *
45 *   *****
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !       but
50 *   -----
51 *   !       !       !
52 *   *****
53 *
54 *   MODIFICATIONS:
55 *
56 *   ! date !   auteur !       but
57 *   -----
58 *   !       !       !
59 *   *****/
60 #ifndef ISOVALEURS_GEOMVIEW_T
61 #define ISOVALEURS_GEOMVIEW_T
62 #include "OrdreVisu.h"
63 #include "spectre.h"
64 #include "Isovaleurs_vrml.h"
65
66 /// @addtogroup Les_sorties_geomview
67 /// @{
68
69 class Isovaleurs_geomview : public Isovaleurs_vrml
70 {
71 public :
72     // CONSTRUCTEURS :
73     // par défaut
74     Isovaleurs_geomview () ;
75
76     // constructeur de copie
77     Isovaleurs_geomview (const Isovaleurs_geomview& algo);
78
79     // DESTRUCTEUR :
80     ~Isovaleurs_geomview () ;
81
82     // METHODES PUBLIQUES :
83
84 protected :
85     // VARIABLES PROTEGEES :
86     // METHODES PROTEGEES :
87     // sortie du dessin de la légende
88     void Sortie_dessin_legende(UtilLecture & entreePrinc);
89
90 };
91 /// @} // end of group
92
93 #endif

```

## 7.407 Mail\_initiale\_geomview.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 * *****/
38 *   BUT:  Visualisation du maillage initiale en geomview.
39 *   Uniquement l'ordre d'exécution est différent de la classe
40 *   vrml.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *   ! date !   auteur !       but
46 *   -----
47 *   !       !       !
48 *
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date !   auteur !       but
52 *   -----
53 *
54 * *****/
55 #ifndef MAIL_INITIALE_GEOMVIEW_T
56 #define MAIL_INITIALE_GEOMVIEW_T
57
58 #include "OrdreVisu.h"
59 #include "Mail_initiale_vrml.h"
60
61 /// @addtogroup Les_sorties_geomview
62 /// @{
63 ///
64
65
66 class Mail_initiale_geomview : public Mail_initiale_vrml
67 {
68 public :
69     // CONSTRUCTEURS :
70     // par défaut
71     Mail_initiale_geomview () ;
72
73     // constructeur de copie
74     Mail_initiale_geomview (const Mail_initiale_geomview& algo);
75
76     // DESTRUCTEUR :
77     ~Mail_initiale_geomview () ;
78
79     // METHODES PUBLIQUES :
80     // execution de l'ordre
81     // tab_mail : donne les numéros de maillage concerné
82     // incre : numéro d'incrément qui en cours
83     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
84     // animation : indique si l'on est en animation ou pas

```



```

85 // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
86 void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
unseul_incre,LesReferences*
87 ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
88 ,Resultats*,Utillecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
89 ,bool animation,const map < string, const double * , std::less <string> >&
listeVarGlob
90 ,const List_io < TypeQuelconque >& listeVecGlob);
91
92 private :
93 // VARIABLES PROTEGEES :
94
95 // METHODES PROTEGEES :
96
97 };
98 /// @} // end of group
99
100 #endif

```

## 7.408 Visuali\_geomview.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 * DATE: 03/02/2003 *
32 * * $ *
33 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
34 * * $ *
35 * PROJET: Herezh++ *
36 * * $ *
37 *****/
38 * BUT: Arrêt des question Mise en route de la visualisation *
39 * en geomview. *
40 * * $ *
41 * ***** *
42 * VERIFICATION: *
43 * * *
44 * ! date ! auteur ! but ! *
45 * ----- *
46 * ! ! ! ! *
47 * * $ *
48 * ***** *
49 * MODIFICATIONS: *
50 * ! date ! auteur ! but ! *
51 * ----- *
52 * * $ *
53 *****/
54 #ifndef VISUALI_GEOMVIEW_T
55 #define VISUALI_GEOMVIEW_T
56
57 #include "OrdreVisu.h"
58
59 /** @defgroup Les_sorties_geomview
60 *
61 * BUT: groupe concernant le posttraitement avec geomview
62 *
63 *

```

```

64 * \author      Gérard Rio
65 * \version    1.0
66 * \date      03/02/2003
67 * \brief     groupe concernant le posttraitement avec geomview
68 *
69 */
70
71 /// @addtogroup Les_sorties_geomview
72 /// @{
73 ///
74
75 class Visuali_geomview : public OrdreVisu
76 {
77     public :
78         // CONSTRUCTEURS :
79         // par défaut
80         Visuali_geomview () ;
81
82         // constructeur de copie
83         Visuali_geomview (const Visuali_geomview& algo);
84
85         // DESTRUCTEUR :
86         ~Visuali_geomview () ;
87
88         // METHODES PUBLIQUES :
89         // execution de l'ordre
90         void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
91                     ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*
92                     ,LesContacts*,Resultats*,UtilLecture & ,OrdreVisu::EnumTypeIncre,int
93                     ,bool,const map < string, const double * , std::less <string> >&
94                     ,const List_io < TypeQuelconque >& listeVecGlob)
95             {};
96
97         // lecture des paramètres de l'ordre dans un flux
98         void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
99         // écriture des paramètres de l'ordre dans un flux
100        void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
101
102     private :
103         // VARIABLES PROTEGEES :
104
105         // METHODES PROTEGEES :
106
107 };
108 /// @} // end of group
109
110 #endif

```

## 7.409 Visualisation\_geomview.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      03/02/2003
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++

```

```

36 *
37 ***** $ *
38 * BUT: Gérer et définir les fichiers de visualisation geomview. *
39 * $ *
40 * ***** *
41 * VERIFICATION: *
42 * *
43 * ! date ! auteur ! but ! *
44 * ----- *
45 * ! ! ! ! *
46 * $ *
47 * ***** *
48 * MODIFICATIONS: *
49 * ! date ! auteur ! but ! *
50 * ----- *
51 * $ *
52 *****/
53 #ifndef VISUALISATION_GEOMVIEW_H
54 #define VISUALISATION_GEOMVIEW_H
55
56 #include "UtilLecture.h"
57 // #include "bool.h"
58 #include "MotCle.h"
59 #include "string.h"
60 #include "Tableau_T.h"
61 #include "LesMaillages.h"
62 #include "LesReferences.h"
63 #include "LesCondLim.h"
64 #include <list>
65 #include "LesContacts.h"
66 #include "LesValVecPropres.h"
67 #include "LesLoisDeComp.h"
68 #include "DiversStockage.h"
69 #include "Charge.h"
70 #include "LesContacts.h"
71 #include "Resultats.h"
72 #include "OrdreVisu.h"
73
74 /// @addtogroup Les_sorties_geomview
75 /// @{
76 ///
77
78
79 class Visualisation_geomview
80 {
81 public :
82 // CONSTRUCTEURS :
83 // le constructeur par défaut ne doit pas être utilisé
84 // il y a dans ce cas un message d'erreur et arrêt
85 Visualisation_geomview ();
86 // le bon constructeur
87 Visualisation_geomview (UtilLecture* ent);
88
89 // DESTRUCTEUR :
90 ~Visualisation_geomview ();
91
92 // METHODES PUBLIQUES :
93 // affichage des différentes possibilités (ordres possibles)
94 // ramène un numéro qui renseigne le programme appelant
95 // =-1 : signifie que l'on veut stopper la visualisation
96 // = 0 : signifie que l'on demande la visualisation effective
97 int OrdresPossible();
98
99 // information de l'instance de la liste d'incrément disponible pour la visualisation
100 void List_increment_disponible(list <int> & list_incr) ;
101 // indique le choix de la liste d'incrément à utiliser pour l'initialisation
102 // des fonctions d'initialisation
103 const list<int> & List_balaie_init();
104 // impose une liste d'incrément à utiliser
105 void List_balaie_init(const list<int> & list_init);
106 // indique le choix de la liste d'incrément à visualiser
107 const list<int> & List_balaie() {return *list_balaie;};
108 // information de l'instance du nombre de maillages disponibles pour la visualisation
109 // par défaut tous les maillages seront visualisés, ceci est descidé aussi ici
110 void List_maillage_disponible(int nombre_maillage_dispo) ;
111
112 // initialisation des ordres disponibles
113 // par exemple pour les isovaleurs on définit la liste des isovaleurs disponibles et les extrémas
114 void Initialisation(ParaGlob * paraGlob, LesMaillages * lesMaillages, LesReferences* lesReferences
115 , LesLoisDeComp* lesLoisDeComp, DiversStockage* diversStockage
116 , Charge* charge, LesCondLim* lesCondLim, LesContacts* lesContacts
117 , Resultats* resultats, OrdreVisu::EnumTypeIncre type_incre, int incre
118 , const map < string, const double * , std::less <string> >& listeVarGlob
119 , const List_io < TypeQuelconque >& listeVecGlob
120 , bool fil_calcul);
121

```

```

122 // méthode principale pour activer la visualisation
123 // sort : le flux de visualisation
124 void Visu(ParaGlob * paraGlob, LesMaillages * lesMaillages, LesReferences* lesReferences
125 , LesLoisDeComp* lesLoisDeComp, DiversStockage* diversStockage
126 , Charge* charge, LesCondLim* lesCondLim, LesContacts* lesContacts
127 , Resultats* resultats, OrdreVisu::EnumTypeIncre type_incre, int incre
128 , const map < string, const double * , std::less <string> >& listeVarGlob
129 , const List_io < TypeQuelconque >& listeVecGlob);
130
131 // == définition des paramètres de visualisation
132 // titre, navigation, éclairage
133 // et initialisation des paramètres de la classe
134 void Contexte_debut_visualisation();
135 // (points de vue) et enchaînement si nécessaire
136 void Contexte_fin_visualisation();
137
138 // lecture des paramètres de l'ordre dans un flux
139 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
140 // écriture des paramètres de l'ordre dans un flux
141 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
142 // demande si la visualisation geomview est validé ou pas
143 bool Visu_geomview_valide() {return activ_sort_geomview;};
144 // inactive la visualisation geomview
145 void Inactiv_Visu_geomview() {activ_sort_geomview=false;};
146
147 private :
148 // VARIABLES PROTEGEES :
149 UtilLecture* entreePrinc; // gestion des entrees/sorties
150 list <OrdreVisu> ordre_possible; // l'ensemble des ordres possibles
151 // la sauvegarde de valeurs propres et vecteurs propres éventuelles
152 LesValVecPropres lesValVecPropres;
153 OrdreVisu* fin_o; // ordre de fin de la visualisation
154 OrdreVisu* visuali; // ordre de visualiser
155 OrdreVisu* choix_incre; // ordre de choix des incréments
156 OrdreVisu* ptdeformee; // déformée
157 OrdreVisu* anim; // ordre d'animation
158 list <int> list_incre; // liste des incréments possibles
159 const list <int>* list_balaie; // liste des incréments à visualiser
160 int nb_maillage_dispo; // le nombre de maillage disponible
161 OrdreVisu* choix_mail; // ordre de choix des maillages
162 OrdreVisu* choix_isevaleur; // ordre de visualisation d'isevaleur
163 // définition de la boîte d'encombrement maxi des # maillages
164 Tableau <Vecteur> boite;
165 // indication si l'on est en animation ou pas
166 bool animation;
167 // indication si la visualisation de type geomview est active ou pas, par défaut = faux
168 bool activ_sort_geomview;
169
170 // METHODES PROTEGEES :
171 // test si la réponse fait partie des ordres possibles
172 // si l'on trouve un ordre ok on ramène un pointeur sur l'ordre
173 // sinon on ramène un pointeur null
174 OrdreVisu* Existe(string& reponse);
175 // affichage des options possibles
176 void Affiche_options();
177 // calcul de la plus grande boîte
178 void Calcul_maxi_boite (Tableau <Vecteur>& boite_inter);
179 // définition des points de vue avec l'angle adoc
180 void DefinitionViewPoint();
181
182
183 };
184 /// @} // end of group
185
186 #endif

```

## 7.410 Deformees\_Gid.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,

```

```

18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      28/09/2004
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *   ****
38 *   BUT:      Visualisation de la déformée avec Gid.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date !   auteur !           but
44 *   -----
45 *   !           !           !
46 *
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date !   auteur !           but
50 *   -----
51 *
52 *****/
53 #ifndef DEFORMEES_GID_T
54 #define DEFORMEES_GID_T
55
56 #include "OrdreVisu.h"
57 #include "Isovaleurs_Gid.h"
58 #include "OrdreVisu.h"
59 #include "ChoixDesMaillages_vrml.h"
60 #include "Basiques.h"
61
62 /// @addtogroup Les_sorties_Gid
63 /// @{
64 ///
65
66
67 class Deformees_Gid : public OrdreVisu
68 {
69 public :
70     // CONSTRUCTEURS :
71     // par défaut
72     Deformees_Gid () ;
73     // constructeur à utiliser
74     Deformees_Gid(const string& comment_som, const string& explication, const string& ordre) :
75         OrdreVisu(comment_som,explication,ordre)
76     {} ;
77
78     // constructeur de copie
79     Deformees_Gid (const Deformees_Gid& algo);
80
81     // DESTRUCTEUR :
82     ~Deformees_Gid () ;
83
84     // METHODES PUBLIQUES :
85     // execution de l'ordre
86     // tab_mail : donne les numéros de maillage concerné
87     // incre : numéro d'incrément qui en cours
88     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
89     // animation : indique si l'on est en animation ou pas
90     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
91     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
unseul_incre,LesReferences*
92                 ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
93                 ,Resultats*,Utillecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
94                 ,bool animation,const map < string, const double * , std::less <string> >&
listeVarGlob
95                 ,const List_io < TypeQuelconque >& listeVecGlob);
96     // choix de l'ordre, cet méthode peut entraîner la demande d'informations
97     // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
98     void ChoixOrdre();
99
100     // --- méthodes particulière ----
101     // initialisation d'une liaison avec une instance de classe d'isovaleur

```

```

102 void Jonction_isevaleur(const Isevaleurs_Gid * iso) {isevaleurs_Gid = iso;};
103 // initialisation d'une liaison avec une instance de classe de choix des maillages
104 void Jonction_ChoixDesMaillages(const ChoixDesMaillages_vrml* choix_m) {choix_mail = choix_m;};

105 // initialisation d'une liaison avec une instance de classe de maillage initiale
106 void Jonction_MaillageInitiale(const Mail_initiale_Gid * mailIni) {mailInitial = mailIni;};
107
108 // récup de la liste des noms de matières utilisés dans les shapes
109 const list <string> & Noms_matiere() const { return noms_matiere;};
110 // récup de la liste des noms de couleurs utilisés dans les shapes
111 const list <string> & Noms_couleurs() const { return noms_couleurs;};
112
113 // lecture des paramètres de l'ordre dans un flux
114 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
115 // écriture des paramètres de l'ordre dans un flux
116 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
117
118
119 protected :
120 // VARIABLES PROTEGEES :
121 // définition des alertes sur les min max
122 list <DeuxDoubles> li_bornes; // min max des alertes
123 list <string> li_nom_bornes; // nom des alertes
124 const Mail_initiale_Gid * mailInitial; // pour une jonction avec le maillage initial
125
126
127 const Isevaleurs_Gid * isevaleurs_Gid; // contient lorsqu'il est actif les couleurs des noeuds
128 list < DeuxDoubles > num_mail_incr; // numéro de maillage et d'incrément correspondant
129 // à couleurs noeud
130 list <string> noms_matiere; // liste des noms de matériaux utilisée

131 list <string> noms_couleurs; // liste des noms de couleurs utilisée

132
133 const ChoixDesMaillages_vrml* choix_mail; // contient lorsqu'il est actif le choix des maillages
134
135
136 // METHODES PROTEGEES :
137
138 };
139 /// @} // end of group
140
141 #endif

```

## 7.411 Fin\_Gid.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      28/09/2004
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *      *****
38 *      BUT:      Fin de la visualisation en Gid
39 *

```

```

40 *      *
41 *      VERIFICATION: *
42 *      *
43 *      ! date ! auteur ! but ! *
44 *      ----- *
45 *      ! ! ! ! *
46 *      $ *
47 *      *
48 *      MODIFICATIONS: *
49 *      ! date ! auteur ! but ! *
50 *      ----- *
51 *      $ *
52 *****/
53 #ifndef FIN_GID_T
54 #define FIN_GID_T
55
56 #include "OrdreVisu.h"
57 #include "Fin_vrml.h"
58
59 /// @addtogroup Les_sorties_Gid
60 /// @{
61 ///
62
63
64 class Fin_Gid : public Fin_vrml
65 {
66 public :
67     // CONSTRUCTEURS :
68     // par default
69     Fin_Gid () ;
70
71     // constructeur de copie
72     Fin_Gid (const Fin_Gid& algo);
73
74     // DESTRUCTEUR :
75     ~Fin_Gid () ;
76
77     // METHODES PUBLIQUES :
78     // execution de l'ordre
79     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaitrages * ,bool ,LesReferences*
80                 ,LesLoisDeComp* ,DiversStockage*
81                 ,Charge* ,LesCondLim*
82                 ,LesContacts* ,Resultats* ,UtilLecture & ,OrdreVisu::EnumTypeIncre ,int
83                 ,bool,const map < string, const double * , std::less <string> >&
84                 ,const List_io < TypeQuelconque >& listeVecGlob)
85     {};
86
87     // lecture des paramètres de l'ordre dans un flux
88     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
89     // écriture des paramètres de l'ordre dans un flux
90     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
91
92 private :
93     // VARIABLES PROTEGEES :
94
95     // METHODES PROTEGEES :
96
97 };
98 /// @} // end of group
99
100 #endif

```

## 7.412 Isovaleurs\_Gid.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,

```

```

21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          24/09/2004
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *
38 *      BUT:  Visualisation des isovaleurs en Gid.
39 *
40 *      *****
41 *
42 *      REMARQUE:
43 *
44 *      VERIFICATION:
45 *
46 *      ! date !   auteur !           but
47 *      -----
48 *      !           !           !
49 *
50 *      *****
51 *      MODIFICATIONS:
52 *      ! date !   auteur !           but
53 *      -----
54 *
55 *      *****/
56 #ifndef ISOVALEURS_GID_T
57 #define ISOVALEURS_GID_T
58
59 #include "OrdreVisu.h"
60 #include "TypeQuelconque.h"
61 #include "Mail_initiale_Gid.h"
62 #include <map>
63
64 /// @addtogroup Les_sorties_Gid
65 /// @{
66 ///
67
68
69 class Isovaleurs_Gid : public OrdreVisu
70 {
71 public :
72     // CONSTRUCTEURS :
73     // par défaut
74     Isovaleurs_Gid () ;
75
76     // constructeur de copie
77     Isovaleurs_Gid (const Isovaleurs_Gid& algo);
78
79     // DESTRUCTEUR :
80     ~Isovaleurs_Gid () ;
81
82     // METHODES PUBLIQUES :
83     // initialisation : permet d'initialiser les différents paramètres de l'ordre
84     // lors d'un premier passage des différents incréments
85     // en virtuelle, a priori est défini si nécessaire dans les classes dérivées
86     // incre : numéro d'incrément qui en cours
87     void Initialisation(ParaGlob * ,LesMaillages * ,LesReferences*
88         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
89         ,Resultats*,EnumTypeIncre type_incre,int incre
90         ,const map < string, const double * , std::less <string> >& listeVarGlob
91         ,const List_io < TypeQuelconque >& listeVecGlob
92         ,bool fil_calcul) ;
93     // execution de l'ordre
94     // tab_mail : donne les numéros de maillage concerné
95     // incre : numéro d'incrément qui en cours
96     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
97     // animation : indique si l'on est en animation ou pas
98     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
99     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
100 unseul_incre,LesReferences*
101     ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
102     ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int
103     incre
104     ,bool animation,const map < string, const double * , std::less <string> >&
105     listeVarGlob

```



```

103         ,const List_io < TypeQuelconque >& listeVecGlob);
104 // choix de l'ordre, cet méthode peut entraîner la demande d'informations
105 // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
106 void ChoixOrdre();
107
108 // initialisation de la liste des différentes isovaleurs possibles
109 void Init_liste_isovaleur(LesMaillages * lesMail,LesContacts* lesContacts,bool fil_calcul);
110 // initialisation d'une liaison avec une instance de classe de maillage initiale
111 void Jonction_MaillageInitiale(const Mail_initiale_Gid * mailIni) {mailInitial = mailIni};
112
113 // lecture des paramètres de l'ordre dans un flux
114 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
115 // écriture des paramètres de l'ordre dans un flux
116 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
117
118 // def d'une classe à usage interne
119 class P_gauss
120 { public :
121     friend ostream & operator << (ostream & sort, const P_gauss & a)
122     { cout << a.elemgeom->TypeGeometrie() << a.elemgeom->TypeInterpolation() << a.elemgeom->Nbi()
123       << " ptinteg "
124       << a.nom_groupe_pt_integ << a.enu << a.num_maill << ", " ; return sort;};
125     P_gauss(): elemgeom(NULL),nom_groupe_pt_integ(""),enu(NU_DDL),num_maill(1) {};
126     P_gauss(const ElemGeomC0* eg,string nom,Enum_ddl en,const int& num_m):
127     elemgeom(eg),nom_groupe_pt_integ(nom),enu(en),num_maill(num_m) {};
128     P_gauss(const P_gauss& p):
129     elemgeom(p.elemgeom),nom_groupe_pt_integ(p.nom_groupe_pt_integ)
130     ,enu(p.enu),num_maill(p.num_maill) {};
131
132     bool operator==(const P_gauss& pg) const // égalité uniquement si même elem
133     {return ((pg.elemgeom==this->elemgeom)&&(num_maill==pg.num_maill));}; // géom et
même maillage
134     bool operator!=(const P_gauss& pg) const // inégalité sur l'élément géom ou
135     {return (!(pg==(*this)));}; // sur le maillage
136
137     // les données
138     const ElemGeomC0* elemgeom;
139     string nom_groupe_pt_integ;
140     Enum_ddl enu;
141     // numéro de maillage associé
142     int num_maill;
143 };
144 protected :
145     // VARIABLES PROTEGEES :
146     bool absolue; // par défaut on sort en absolue les tenseurs
147     // pour pouvoir correctement les visualiser, même pour les elem 2D
148
149     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddl; // ddl aux noeuds possibles à visualiser
150     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddl_retenu; // ddl à visualiser
151     Tableau <List_io <bool> > choix_var_ddl; // indique si c'est une variation ou le ddl
152
153     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddlEtendu; // ddl étendu aux noeuds possibles à
visualiser
154     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddlEtendu_retenu; // ddl étendu aux noeuds à
visualiser
155     // tab de travail: une sous partie de tabnoeud_type_ddlEtendu_retenu,
156     // qui ne contient que les grandeurs à accumuler aux noeuds venant des éléments, idem pour les ddl
initiaux
157     Tableau <List_io < Ddl_enum_etendu > > a_accumuler_tabnoeud_type_ddlEtendu;
158     Tableau <List_io < Ddl_enum_etendu > > a_accumuler_tabnoeud_type_ddlEtendu_retenu;
159
160     Tableau <List_io < TypeQuelconque > > tabnoeud_evoluee; // types évolués aux noeuds possibles à
visualiser
161     Tableau <List_io < TypeQuelconque > > tabnoeud_evoluee_retenu; // types évolués aux noeuds à
visualiser
162     // tab de travail: une sous partie de tabnoeud_evoluee_retenu,
163     // qui ne contient que les grandeurs à accumuler aux noeuds venant des éléments
164     Tableau <List_io < TypeQuelconque > > a_accumuler_tabnoeud_evoluee;
165     Tableau <List_io < TypeQuelconque > > a_accumuler_tabnoeud_evoluee_retenu;
166
167     // cas des vecteurs globaux, transférable directement aux noeuds
168     List_io < TypeQuelconque > list_vect_globalPourNoeud; // les vecteurs globaux que l'on peut visualiser
169     List_io < TypeQuelconque > list_vect_globalPourNoeud_retenu; // les vecteurs globaux à visualiser
170
171
172
173
174
175     Tableau <List_io < Ddl_enum_etendu > > tabellement_type_ddl; // ddl aux elements possibles à
visualiser
176     Tableau <List_io < Ddl_enum_etendu > > tabellement_type_ddl_retenu; // ddl aux elements à visualiser
177
178     Tableau <List_io < TypeQuelconque > > tabellement_evoluee; // types evoluee aux elements possibles à
visualiser
179     Tableau <List_io < TypeQuelconque > > tabellement_evoluee_retenu; // type evoluee aux elements à
visualiser
180

```

```

181   Tableau <List_io < TypeQuelconque > > tabellement_typeParti; // types particuliers aux elements
    possibles à visualiser
182   Tableau <List_io < TypeQuelconque > > tabellement_typeParti_retenue; // types particuliers aux
    elements à visualiser
183
184   bool transfert_au_noeud; // indique si oui ou non on transfert aux noeuds
185   int cas_transfert; // indique la méthode utilisée pour le transfert aux noeuds
186
187   const Mail_initiale_Gid * mailInitial; // pour une jonction avec le maillage initial
188   LesMaillages * lesMail; // les maillages
189
190   bool ddlSurY_1D ; // indique pour chaque maillage dans le cas 1D si l'on veut les ddl selon y plutôt
    que x
191
192 //--- variables internes pour la sortie Ddl_enum_etendu, de grandeurs quelconque évoluées ou non aux pt
    d'integ -----
193 // dans gid il n'y a pas de notion de maillage, il y a une seule liste de noeud et une seule liste
    d'éléments
194 // (par contre je défini un groupe d'élément particulier pour chaque maillage mais ça n'intervient
    pas ici en post-traitement)
195 // donc on a donc a définir
196 // 1) des groupes de pt_integ : qui indique combien et quel type de répartition ont un groupe de pt
    d'integ.
197 //
    // Le groupe est identifié par un nom = "nom de ptgauss", mais au
198 // moment de l'écriture (donc de la
    // définition du groupe pour gid) on n'indique pas quels sont les
199 // grandeurs (ddl) que l'on va écrire
    // pour ce groupe, on peut donc ainsi utiliser un même groupe pour
200 // plusieurs grandeurs.
    // Le nom de groupe contient le numéro du maillage, donc les noms de
201 // groupe sont différents pour
    // chaque maillage. Le groupe de pt d'intég contient le ddl de
202 // référence, auquel peuvent se rattacher
    // plusieurs ddl secondaires (provenant des types évolués, ou
203 // particulier) mais qui sont
    // calculé aux mêmes pt d'integ que le ddl de base.
204
205 // 2) des listes de grandeurs à sortir pour un groupes de pt_integ donné, ces listes peuvent être
    différentes pour chaque maillage
206 //
207 // --- au bilan on a en stockage globale: ---
208 // li_P_gauss_total: contient tous les noms de groupe de pt integ différent et l'élément géométrique
    associé (nb pt etc),
209 //
    // et le ddl associé
210 // tab_point_enum: permet a partir du ddl de retrouver un élément de li_P_gauss_total associé
211 // map_gauss_base: permet de connaitre pour un nom_groupe_pt_integ donné, la liste des
    Ddl_enum_etendu à sortir
212 // map_gauss_baseEVol: idem pour les grandeurs évoluées
213 // map_gauss_basePQ: idem pour les grandeurs quelconques
214 // map_gauss_tab_baseEVol et map_gauss_tab_basePQ : définissent des tableaux de npt conteneurs
    associés
215 //
    // à map_gauss_tab_baseEVol et
216 // map_gauss_tab_basePQ, ceci pout une manipulation
    // globale de toutes les grandeurs à tous les pt
    d'integ d'un groupe
217 // En fait quand on calcul les grandeurs aux pt d'integ, on les stocke dans le tableau (donc
    l'élément de
218 // map_gauss_tab_baseEVol ou map_gauss_tab_basePQ) au lieu de la liste map_gauss_baseEVol ou
    map_gauss_basePQ. Donc le véritable
219 // conteneur c'est le tableau (les 2), les listes initiales (élément de map_gauss_baseEVol ou
    map_gauss_basePQ) ne servent que
220 // 1) pour les entêtes, 2) à savoir s'il y a des grandeurs à considérer ou non, et 3) à
    construire les tableaux.
221 //
222 // --- maintenant les grandeurs associées à chaque maillage ---
223 // tp_tp_tp_gauss_base : tp_tp_tp_gauss_base(imail)(isous_mail) contient tous les groupe de pt
    d'integ différent pour un
224 //
    // sous maillage donné
225
226
227 // on crée une liste de P_gauss contenant en autre une
228 // référence d'élément géométrique associés aux ddl
229 // pour ne pas créer de groupe de pt_integ inutile car identique à un groupe déjà existant.
230 // li_P_gauss contient "tous" les associations "nom de ptgauss" élément géom et un ddl de référence
231 list < P_gauss > li_P_gauss_total;
232 // on utilise un même "nom de ptgauss" pour plusieurs ddl, le rassemblement s'effectue dans
    l'écriture
233
234 // un tableau de travail que l'on dimentionne qu'en locale dans ExeOrdrePremierIncrement
235 // Tableau < P_gauss * > tab_point_enum; // permet de retrouver le P_gauss dans la liste
    li_P_gauss
236 //associé à un enum de base
237 // en fait est construit à partir de li_P_gauss_total, et est indicer par l'énum du P_gauss,
238
239 // le tableau des P_gauss représente les groupes de pt de gauss existants (l'enum ici n'a pas
    d'importance)
240 // pointe sur des éléments de li_P_gauss_total

```

```

241 // donc tp_gauss_base(imail) (ims) (i) = pour le maillage imail, pour le sous_maillage ims, donne la
liste des pt_gauss
242 // de base qui permettent de générer tous les ddl (de base ou dérivés)
243 Tableau < Tableau < Tableau < P_gauss > > > tp_tp_tp_gauss_base;
244 // permet de récolter tous les "nom de ptgauss" utilisé par le maillage imail
245
246 // une map pour faire l'association entre le nom du groupe de pt de gauss et la liste
247 // de ddl étendue associée
248 map < string, List_io < Ddl_enum_etendu > , std::less <string> > map_gauss_base;
249 // une map pour faire l'association entre le nom du groupe de pt de gauss et la liste
250 // de type quelconque associée contenant les types évolués
251 map < string, List_io < TypeQuelconque > , std::less <string> > map_gauss_baseEVol;
252 // idem mais pour un tableau de liste, indicé par le nombre de ptinteg
253 map < string, Tableau < List_io < TypeQuelconque > > , std::less <string> > map_gauss_tab_baseEVol;
254
255 // une map pour faire l'association entre le nom du groupe de pt de gauss et la liste
256 // de type quelconque associée contenant les grandeurs particulières
257 map < string, List_io < TypeQuelconque > , std::less <string> > map_gauss_basePQ;
258 // idem mais pour un tableau de liste, indicé par le nombre de ptinteg
259 map < string, Tableau < List_io < TypeQuelconque > > , std::less <string> > map_gauss_tab_basePQ;
260
261 //--- fin variables internes pour la sortie Ddl_enum_etendu et type quelconque aux pt d'integ -----
262
263 //--- variables internes pour la sortie aux noeuds -----
264 // pour les sorties aux noeuds because Gid ne sait pas gérer plusieurs maillages, on globalise
toutes les sorties
265 // et on définit un tableau de booléen qui indique si oui ou non la grandeur est à sortir en
fonction des tableaux
266 // tabelement_type_ddl_retenu, tabelement_evoluee_retenu, tabelement_typeParti_retenu
267
268 // .. pour les ddl aux noeuds
269 List_io < Ddl_enum_etendu > glob_noe_ddl_retenu; // le global
270 Tableau <List_io < bool > > t_g_noeud_ddl_asortir;
271
272 // .. pour les ddl étendues définis aux noeuds
273 List_io < Ddl_enum_etendu > glob_noeud_ddl_etendu_retenu; // le global
274 Tableau <List_io < bool > > t_g_noeud_ddl_etendu_asortir;
275 // .. pour les types évoluées définis aux noeuds
276 List_io < TypeQuelconque > glob_noeud_evol_retenu; // le global
277 Tableau <List_io < bool > > t_g_noeud_evoluee_asortir;
278
279 // .. pour les ddl venant d'un transfert aux noeuds des grandeurs aux pt d'integ
280 List_io < Ddl_enum_etendu > glob_elem_ddl_retenu; // le global
281 Tableau <List_io < bool > > t_g_elem_ddl_asortir;
282 // .. pour les types évoluées venant d'un transfert aux noeuds des grandeurs aux pt d'integ
283 List_io < TypeQuelconque > glob_elem_evol_retenu; // le global
284 Tableau <List_io < bool > > t_g_elem_evoluee_asortir;
285 // .. pour les types particuliers aux elements venant d'un transfert aux noeuds
286 List_io < TypeQuelconque > glob_elem_Part_i_retenu; // le global
287 Tableau <List_io < bool > > t_g_elem_typeParti_asortir;
288 // .. globalise tous les types quelconques ...
289 Tableau <List_io < TypeQuelconque > * > tab_quelconque;
290
291 // -- particularités liées aux contacts: dont les grandeurs sont stockées aux noeuds
292 // la liste suivante contient les infos choisies restreintes aux contacts
293 List_io < TypeQuelconque > li_glob_restreinte_TQ;
294
295
296 //-- fin variables internes pour la sortie aux noeuds -----
297
298
299 // METHODES PROTEGEES :
300
301 // constructeur utilisé par les classes dérivées
302 // en fait il s'agit de transmettre directement les infos à la classe mère Visu
303 Isovaleurs_Gid(const string& comment_som, const string& explication
304 , const string& ordre) :
305     OrdreVisu(comment_som,explication,ordre)
306     {};
307
308 // définition interactives des paramètres généraux des isovaleurs
309 void ParametresGeneraux(const string& choix);
310 // choix de l'isovaleur à visualiser calculée à partir des noeuds
311 void ChoixIsovaleur_noeud(const string& choix);
312 // choix de l'isovaleur à visualiser calculée à partir des ddl etendu aux noeuds
313 void ChoixIsovaleur_ddl_etendu_noeud(const string& choix);
314 // choix de l'isovaleur à visualiser calculée à partir de grandeurs évoluées aux noeuds
315 void ChoixIsovaleur_evoluee_noeud(const string& choix);
316 // choix de l'isovaleur à visualiser calculée à partir des ddl aux points d'intégrations
317 void ChoixIsovaleur_ddl_ptinteg(const string& choix);
318 // choix de l'isovaleur à visualiser calculée à partir des grandeurs évoluées aux points
d'intégrations
319 void ChoixIsovaleur_tensorielle_ptinteg(const string& choix);
320 // choix de l'isovaleur à visualiser calculée à partir des grandeurs quelconques aux points
d'intégrations
321 void ChoixIsovaleur_quelc_ptinteg(const string& choix);
322 // choix de grandeur existantes aux points d'intégrations et à ramener aux noeuds

```

```

323 // pour préparer une visualisation d'isovaleurs
324 void TransfertGrandeursPtIntegAuNoeud();
325 // initialisation de l'exécution du transferts
326 void InitPremierIncrExecutionTransfertAuxNoeuds();
327 // exécution du transfert
328 void ExecutionTransfert();
329 // vérification que le transfert peut se faire (pas de doublon de grandeurs)
330 void VerificationTransfertPossible();
331 // sortie de l'entête des isovaleurs de grandeurs definis aux pt integ pour des grandeurs
    quelconques
332 // et également tout simplement aux noeuds
333 // sert pour les sorties aux pt integ et aux noeuds
334 // -- 1) le programme général quelque soit la structure (simple, tableau, list ... ) de la grandeur
    quelconque
335 // drap_noeud = false ---> sortie aux pt integ, =true ---> sortie aux noeuds
336 void EnteteSortieIsovaleurs_G_Quelconque(ostream & sort, TypeQuelconque& typ, bool drap_noeud);
337 // -- 2) le programme spécifique pour "une" grandeur de base (utilisé en 1))
338 void EnteteSortieIsovaleurs_G_Quelconque_TYPE_SIMPLE(ostream & sort, const TypeQuelconque::Grandeur&
    grandeur
339
    , const string nom_typeQuelconque, int
    dima);
340
341 // -- 3) le programme spécifique pour "une" grandeur de base à un noeud (utilisé en 1))
342 void EnteteSortieIsovaleursNoeud_G_Quelconque_TYPE_SIMPLE
343     (ostream & sort, const TypeQuelconque::Grandeur& grandeur
344     , const string nom_typeQuelconque, int dima);
345
346
347 // sortie d'une grandeurs quelconque aux pt integ ou aux noeuds
348 // -- a) le programme général quelque soit la structure (simple, tableau, list ... ) de la grandeur
    quelconque
349 void SortieGrandeursQuelconque(ostream & sort, const TypeQuelconque& typ);
350 // -- b) le programme spécifique pour "une" grandeur de base (utilisé en a))
351 void SortieGrandeursQuelconque_TYPE_SIMPLE(ostream & sort, const TypeQuelconque::Grandeur&
    grandeur, int dima);
352 // initialisation les différentes listes internes qui globalisent tous les maillages
353 void GlobalisationListSurTousLesMaillagesPourLesNoeuds();
354 // exeoOrdre: cas du premier increments
355 void ExeOrdrePremierIncrement(const Tableau<int>& tab_mail, LesMaillages * lesMail, ostream &sort);
356 // écriture des grandeurs aux points d'intégration
357 void EcritureAuxPtInteg(const Tableau<int>& tab_mail, LesMaillages * lesMail
358     , ostream &sort, int incre);
359 // écriture des grandeurs aux noeuds
360 void EcritureAuxNoeuds(const Tableau<int>& tab_mail, LesMaillages * lesMail
361     , ostream &sort, int incre);
362
363
364 };
365 /// @} // end of group
366
367 #endif

```

## 7.413 Mail\_initiale\_Gid.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      24/09/2004      *

```

```

32 *                                                                 $ *
33 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)           *
34 *                                                                 $ *
35 *   PROJET:      Herezh++                                       *
36 *                                                                 $ *
37 * *****
38 *   BUT:   Visualisation du maillage initiale en Gid.           *
39 *   Uniquement l'ordre d'exécution est différent de la classe *
40 *   vrml.                                                       *
41 *                                                                 $ *
42 *   *****
43 *   VERIFICATION:                                               *
44 *   ! date ! auteur ! but !                                     *
45 *   ----- ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
46 *   ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
47 *   ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
48 *   ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
49 *   *****
50 *   MODIFICATIONS:                                             *
51 *   ! date ! auteur ! but ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
52 *   ----- ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
53 *   ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
54 * *****/
55 #ifndef MAIL_INITIALE_GID_T
56 #define MAIL_INITIALE_GID_T
57
58 #include "OrdreVisu.h"
59
60 /// @addtogroup Les_sorties_Gid
61 /// @
62 ///
63
64
65 class Mail_initiale_Gid : public OrdreVisu
66 {
67 public :
68     // CONSTRUCTEURS :
69     // par défaut
70     Mail_initiale_Gid () ;
71
72     // constructeur de copie
73     Mail_initiale_Gid (const Mail_initiale_Gid& algo);
74
75     // DESTRUCTEUR :
76     ~Mail_initiale_Gid () ;
77
78     // METHODES PUBLIQUES :
79     // initialisation : permet d'initialiser les différents paramètres de l'ordre
80     // lors d'un premier passage des différents incréments
81     // en virtuelle, a priori est défini si nécessaire dans les classes dérivées
82     // incre : numéro d'incrément qui en cours
83     void Initialisation(ParaGlob * ,LesMaillages * ,LesReferences*
84         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
85         ,Resultats*,EnumTypeIncre type_incre,int incre
86         ,const map < string, const double * , std::less <string> >& listeVarGlob
87         ,const List_io < TypeQuelconque >& listeVecGlob
88         ,bool fil_calcul) ;
89     // execution de l'ordre
90     // tab_mail : donne les numéros de maillage concerné
91     // incre : numéro d'incrément qui en cours
92     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
93     // animation : indique si l'on est en animation ou pas
94     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
95     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
96         unseul_incre,LesReferences*
97         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
98         ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
99         ,bool animation,const map < string, const double * , std::less <string> >&
100         listeVarGlob
101         ,const List_io < TypeQuelconque >& listeVecGlob);
102
103     // choix de l'ordre, cet méthode peut entraîner la demande d'informations
104     // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
105     void ChoixOrdre();
106
107     // lecture des paramètres de l'ordre dans un flux
108     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
109     // écriture des paramètres de l'ordre dans un flux
110     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
111
112     // ramène le tableau de listes de sous maillage
113     const Tableau < Tableau < List_io < Element* > > > &
114         Tableau_de_sous_maillage() const {return tabtab_sous_mesh;};
115     // ramène le tableau de le listes de types d'éléments différents
116     const Tableau < List_io <Element::Signature> >&
117         Tab_liste_type_element() const { return tabli_sig_elem;};

```

```

116 // ramène le tableau de listes de noms des sous maillages
117 const Tableau < List_io <string> >&
118     Tab_listes_nom_sousMaillage() const { return tabli_sous_mesh;};
119 // ramène le décalages pour les numéros de noeuds du maillage "nmail"
120 int DecalNumNoeud(int nmail) const {return decal_noeud(nmail);};
121 // ramène le décalages pour les numéros d'éléments du maillage "nmail"
122 int DecalNumElement(int nmail) const {return decal_element(nmail);};
123
124 private :
125 // VARIABLES PROTEGEES :
126 // tableau de liste de sous maillages
127 // chaque sous maillage représente un type d'élément fini particulier
128 // l'indice du tableau= le numéro de maillage
129 Tableau < Tableau < List_io < Element* > > > tabtab_sous_mesh;
130 // tableau de listes de type d'éléments différents
131 Tableau < List_io <Element::Signature> > tabli_sig_elem;
132 // tableau de listes des nom des sous_maillages créés
133 Tableau < List_io <string> > tabli_sous_mesh;
134 // tableau des décalages pour les numéros de noeuds et d'éléments
135 Tableau <int > decal_noeud;
136 Tableau <int > decal_element;
137
138
139 // METHODES PROTEGEES :
140
141 };
142 /// @} // end of group
143
144 #endif

```

## 7.414 Visuali\_Gid.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      24/09/2004
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: Arrêt des question Mise en route de la visualisation
39 *         en Gid.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *
51 *   ! date !   auteur !           but
52 *

```

```

53  *****/
54 #ifndef VISUALI_GID_T
55 #define VISUALI_GID_T
56
57 #include "OrdreVisu.h"
58
59 /** @defgroup Les_sorties_Gid
60 *
61 *     BUT:   groupe concernant le postraitement avec Gid
62 *
63 *
64 * \author   Gérard Rio
65 * \version  1.0
66 * \date    24/09/2004
67 * \brief   groupe concernant le postraitement avec Gid
68 *
69 */
70
71 /// @addtogroup Les_sorties_Gid
72 /// @{
73 ///
74
75 class Visuali_Gid : public OrdreVisu
76 {
77 public :
78     // CONSTRUCTEURS :
79     // par default
80     Visuali_Gid () ;
81
82     // constructeur de copie
83     Visuali_Gid (const Visuali_Gid& algo);
84
85     // DESTRUCTEUR :
86     ~Visuali_Gid () ;
87
88     // METHODES PUBLIQUES :
89     // execution de l'ordre
90     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
91                 ,LesLoisDeComp* ,DiversStockage* ,Charge* ,LesCondLim*
92                 ,LesContacts* ,Resultats* ,UtilLecture & ,OrdreVisu::EnumTypeIncr,int
93                 ,bool,const map < string, const double * , std::less <string> >&
94                 ,const List_io < TypeQuelconque >& listeVecGlob)
95         {};
96
97     // lecture des paramètres de l'ordre dans un flux
98     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
99     // écriture des paramètres de l'ordre dans un flux
100    void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
101
102 private :
103     // VARIABLES PROTEGEES :
104
105     // METHODES PROTEGEES :
106
107 };
108 /// @} // end of group
109
110 #endif

```

## 7.415 Visualisation\_Gid.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //

```

```

25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      24/09/2004
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:  Gérer et définir les fichiers de visualisation Gid.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date ! auteur ! but
44 *   -----
45 *   ! ! !
46 *   $
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date ! auteur ! but
50 *   -----
51 *   $
52 *****/
53 #ifndef VISUALISATION_GID_H
54 #define VISUALISATION_GID_H
55
56 #include "UtilLecture.h"
57 // #include "bool.h"
58 #include "MotCle.h"
59 #include "string.h"
60 #include "Tableau_T.h"
61 #include "LesMaillages.h"
62 #include "LesReferences.h"
63 #include "LesCondLim.h"
64 #include <list>
65 #include "LesContacts.h"
66 #include "LesValVecPropres.h"
67 #include "LesLoisDeComp.h"
68 #include "DiversStockage.h"
69 #include "Charge.h"
70 #include "LesContacts.h"
71 #include "Resultats.h"
72 #include "OrdreVisu.h"
73
74 /// @addtogroup Les_sorties_Gid
75 /// @{
76 ///
77
78
79 class Visualisation_Gid
80 {
81 public :
82 // CONSTRUCTEURS :
83 // le constructeur par défaut ne doit pas être utilisé
84 // il y a dans ce cas un message d'erreur et arrêt
85 Visualisation_Gid ();
86 // le bon constructeur
87 Visualisation_Gid (UtilLecture* ent);
88
89 // DESTRUCTEUR :
90 ~Visualisation_Gid ();
91
92 // METHODES PUBLIQUES :
93 // affichage des différentes possibilités (ordres possibles)
94 // ramène un numéro qui renseigne le programme appelant
95 // ==-1 : signifie que l'on veut stopper la visualisation
96 // = 0 : signifie que l'on demande la visualisation effective
97 int OrdresPossible();
98
99 // information de l'instance de la liste d'incrément disponible pour la visualisation
100 void List_increment_disponible(list <int> & list_incr) ;
101 // indique le choix de la liste d'incrément à utiliser pour l'initialisation
102 // des fonctions d'initialisation
103 const list<int> & List_balaie_init();
104 // impose une liste d'incrément à utiliser
105 void List_balaie_init(const list<int> & list_init);
106 // indique le choix de la liste d'incrément à visualiser
107 const list<int> & List_balaie() {return *list_balaie;};
108 // information de l'instance du nombre de maillages disponibles pour la visualisation
109 // par défaut tous les maillages seront visualisés, ceci est descidé aussi ici
110 void List_maillage_disponible(int nombre_maillage_dispo) ;

```



```

111
112 // initialisation des ordres disponibles
113 // par exemple pour les isovaleurs on définit la liste des isovaleurs disponibles et les extrêmes
114 void Initialisation(ParaGlob * paraGlob, LesMaillages * lesMaillages, LesReferences* lesReferences
115 , LesLoisDeComp* lesLoisDeComp, DiversStockage* diversStockage
116 , Charge* charge, LesCondLim* lesCondLim, LesContacts* lesContacts
117 , Resultats* resultats, OrdreVisu::EnumTypeIncre type_incre, int incre
118 , const map < string, const double * , std::less <string> >& listeVarGlob
119 , const List_io < TypeQuelconque >& listeVecGlob
120 , bool fil_calcul);
121
122 // méthode principale pour activer la visualisation
123 // sort : le flux de visualisation
124 void Visu(ParaGlob * paraGlob, LesMaillages * lesMaillages, LesReferences* lesReferences
125 , LesLoisDeComp* lesLoisDeComp, DiversStockage* diversStockage
126 , Charge* charge, LesCondLim* lesCondLim, LesContacts* lesContacts
127 , Resultats* resultats, OrdreVisu::EnumTypeIncre type_incre, int incre
128 , const map < string, const double * , std::less <string> >& listeVarGlob
129 , const List_io < TypeQuelconque >& listeVecGlob);
130
131 // == définition des paramètres de visualisation
132 // titre, navigation, éclairage
133 // et initialisation des paramètres de la classe
134 void Contexte_debut_visualisation();
135 // (points de vue) et enchaînement si nécessaire
136 void Contexte_fin_visualisation();
137
138 // lecture des paramètres de l'ordre dans un flux
139 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
140 // écriture des paramètres de l'ordre dans un flux
141 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
142 // demande si la visualisation Gid est validé ou pas
143 bool Visu_Gid_valide() {return activ_sort_Gid;};
144 // inactive la visualisation Gid
145 void Inactiv_Visu_Gid() {activ_sort_Gid=false;};
146
147 private :
148 // VARIABLES PROTEGEES :
149 UtilLecture* entreePrinc; // gestion des entrees/sorties
150 list <OrdreVisu> ordre_possible; // l'ensemble des ordres possibles
151 // la sauvegarde de valeurs propres et vecteurs propres éventuelles
152 LesValVecPropres lesValVecPropres;
153 OrdreVisu* fin_o; // ordre de fin de la visualisation
154 OrdreVisu* mailInit; // Maillage initial
155 OrdreVisu* visuali; // ordre de visualiser
156 OrdreVisu* choix_inc; // ordre de choix des incréments
157 OrdreVisu* ptdeformee; // déformée
158 list <int> list_incre; // liste des incréments possibles
159 const list <int>* list_balaie; // liste des incréments à visualiser
160 int nb_maillage_dispo; // le nombre de maillage disponible
161 OrdreVisu* choix_mail; // ordre de choix des maillages
162 OrdreVisu* choix_isovaleur; // ordre de visualisation d'isovaleurs
163 // indication si l'on est en animation ou pas
164 bool animation;
165 // indication si la visualisation de type Gid est active ou pas, par défaut = faux
166 bool activ_sort_Gid;
167
168 // METHODES PROTEGEES :
169 // test si la réponse fait partie des ordres possibles
170 // si l'on trouve un ordre ok on ramène un pointeur sur l'ordre
171 // sinon on ramène un pointeur null
172 OrdreVisu* Existe(string& reponse);
173 // affichage des options possibles
174 void Affiche_options();
175
176
177 };
178 /// @} // end of group
179
180 #endif

```

## 7.416 Deformees\_Gmsh.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio

```

```

13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29 //
30 /*****
31 *      DATE:      07/01/2008
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Visualisation de la déformée avec Gmsh.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *
44 *      ! date !   auteur !           but
45 *      -----
46 *      !       !           !           !
47 *      *****
48 *      MODIFICATIONS:
49 *
50 *      ! date !   auteur !           but
51 *      -----
52 *      $
53 *****/
54 #ifndef DEFORMEES_GMSH_T
55 #define DEFORMEES_GMSH_T
56 #include "OrdreVisu.h"
57 #include "Isovaleurs_Gmsh.h"
58 #include "OrdreVisu.h"
59 #include "ChoixDesMaillages_vrml.h"
60 #include "Basiques.h"
61
62 /// @addtogroup Les_sorties_gmsh
63 /// @{
64 ///
65
66
67 class Deformees_Gmsh : public OrdreVisu
68 {
69 public :
70     // CONSTRUCTEURS :
71     // par défaut
72     Deformees_Gmsh () ;
73     // constructeur à utiliser
74     Deformees_Gmsh(const string& comment_som, const string& explication, const string& ordre) :
75         OrdreVisu(comment_som,explication,ordre)
76     {} ;
77
78     // constructeur de copie
79     Deformees_Gmsh (const Deformees_Gmsh& algo);
80
81     // DESTRUCTEUR :
82     ~Deformees_Gmsh () ;
83
84     // METHODES PUBLIQUES :
85     // execution de l'ordre
86     // tab_mail : donne les numéros de maillage concerné
87     // incre : numéro d'incrément qui en cours
88     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
89     // animation : indique si l'on est en animation ou pas
90     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
91     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
unseul_incre,LesReferences*
92         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
93         ,Resultats*,Utilecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
94         ,bool animation,const map < string, const double * , std::less <string> &
95         listeVarGlob
96         ,const List_io < TypeQuelconque >& listeVecGlob);
97     // choix de l'ordre, cet méthode peut entraîner la demande d'informations

```

```

97 // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
98 void ChoixOrdre();
99
100 // --- méthodes particulière ----
101 // initialisation d'une liaison avec une instance de classe d'isovaleur
102 void Jonction_isovaleur(const Isovaleurs_Gmsh * iso) {isovaleurs_Gmsh = iso;};
103 // initialisation d'une liaison avec une instance de classe de choix des maillages
104 void Jonction_ChoixDesMaillages(const ChoixDesMaillages_vrml* choix_m) {choix_mail = choix_m;};
105
106 // initialisation d'une liaison avec une instance de classe de maillage initiale
107 void Jonction_MaillageInitiale(Mail_initiale_Gmsh * mailIni) {mailInitial = mailIni;};
108
109 // récup de la liste des noms de matières utilisés dans les shapes
110 const list <string> & Noms_matiere() const { return noms_matiere;};
111 // récup de la liste des noms de couleurs utilisés dans les shapes
112 const list <string> & Noms_couleurs() const { return noms_couleurs;};
113
114 // lecture des paramètres de l'ordre dans un flux
115 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
116 // écriture des paramètres de l'ordre dans un flux
117 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
118 // récupération du nom pour la sortie gmsh en particulier
119 const string & Nom_deplace() const {return nom_deplace;};
120 // récupération du nom pour la sortie gmsh en particulier= NULL si ce n'est pas nécessaire
121 const string * Nom_Vitesse() const
122 {if (avec_vitesse) return &nom_vitesse;else return NULL;};
123 // récupération du nom pour la sortie gmsh en particulier= NULL si ce n'est pas nécessaire
124 const string * Nom_Acceleration() const
125 {if (avec_acceleration) return &nom_acceleration;else return NULL;};
126 // sortie de la déformée dans le cas du nouveau format
127 void SortieDeformee(const Tableau <int>& tab_mail,LesMaillages * lesMail
128 ,ostream & sort,int incre);
129 // idem pour le champ de vitesse dans le cas du nouveau format
130 void SortieVitesse(const Tableau <int>& tab_mail,LesMaillages * lesMail
131 ,ostream & sort,int incre);
132 // idem pour le champ d'accélération dans le cas du nouveau format
133 void SortieAcceleration(const Tableau <int>& tab_mail,LesMaillages * lesMail
134 ,ostream & sort,int incre);
135
136 protected :
137 // VARIABLES PROTEGEES :
138 // définition des alertes sur les min max
139 list <DeuxDoubles> li_bornes; // min max des alertes
140 list <string> li_nom_bornes; // nom des alertes
141 Mail_initiale_Gmsh * mailInitial; // pour une jonction avec le maillage initial
142 int avec_vitesse; // indique si l'on veut également les vitesses si elles sont disponibles
143 int avec_acceleration; // indique si l'on veut également les accélérations si elles sont disponibles
144
145 string nom_deplace; // le nom de sortie qui sert pour le fichier de sortie gmsh
146 string nom_vitesse; // idem pour la vitesse
147 string nom_acceleration; // idem pour l'acceleration
148
149 const Isovaleurs_Gmsh * isovaleurs_Gmsh; // contient lorsqu'il est actif les couleurs des noeuds
150 // et d'autres utilitaires
151 list < Deuxentiers > num_mail_incr; // numéro de maillage et d'incrément correspondant
152 // à couleurs noeud
153 list <string> noms_matiere; // liste des noms de matériaux utilisée
154
155 list <string> noms_couleurs; // liste des noms de couleurs utilisée
156
157 const ChoixDesMaillages_vrml* choix_mail; // contient lorsqu'il est actif le choix des maillages
158
159 // METHODES PROTEGEES :
160 // sortie de la déformée dans le cas de l'ancien format
161 void SortieDeformee_ancien_format(const Tableau <int>& tab_mail,LesMaillages * lesMail
162 ,ostream & sort,int incre);
163
164 };
165 /// @} // end of group
166
167 #endif

```

## 7.417 Fin\_Gmsh.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //

```

```

9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           07/01/2008
32 *
33 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:        Herezh++
36 *
37 *   ****
38 *   BUT:   Fin de la visualisation en Gmsh
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date !   auteur !           but
44 *   -----
45 *   !           !           !
46 *   *****
47 *
48 *   MODIFICATIONS:
49 *   ! date !   auteur !           but
50 *   -----
51 *
52 *   *****/
53 #ifndef FIN_GMSH_T
54 #define FIN_GMSH_T
55
56 #include "OrdreVisu.h"
57 #include "Fin_vrml.h"
58
59 /// @addtogroup Les_sorties_gmsh
60 /// @{
61 ///
62
63
64 class Fin_Gmsh : public Fin_vrml
65 {
66 public :
67     // CONSTRUCTEURS :
68     // par défaut
69     Fin_Gmsh () ;
70
71     // constructeur de copie
72     Fin_Gmsh (const Fin_Gmsh& algo);
73
74     // DESTRUCTEUR :
75     ~Fin_Gmsh () ;
76
77     // METHODES PUBLIQUES :
78     // execution de l'ordre
79     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
80                 ,LesLoisDeComp* ,DiversStockage*
81                 ,Charge*,LesCondlim*
82                 ,LesContacts*,Resultats*,UtilLecture & ,OrdreVisu::EnumTypeIncre ,int
83                 ,bool,const map < string, const double * , std::less <string> >&
84                 ,const List_io < TypeQuelconque >& listeVecGlob)
85     {};
86
87     // lecture des paramètres de l'ordre dans un flux
88     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
89     // écriture des paramètres de l'ordre dans un flux
90     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
91
92 private :
93     // VARIABLES PROTEGEES :
94

```

```

95     // METHODES PROTEGEES :
96
97 };
98 /// @} // end of group
99
100 #endif

```

## 7.418 Isovaleurs\_Gmsh.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           07/01/2008
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *
38 *   BUT:             Visualisation des isovaleurs en Gmsh.
39 *
40 *   *****
41 *
42 *   REMARQUE:
43 *
44 *   VERIFICATION:
45 *
46 *   ! date ! auteur ! but
47 *   -----
48 *   ! ! !
49 *
50 *   *****
51 *   MODIFICATIONS:
52 *   ! date ! auteur ! but
53 *   -----
54 *
55 *   *****/
56 #ifndef ISOVALEURS_GMSH_T
57 #define ISOVALEURS_GMSH_T
58
59 #include "OrdreVisu.h"
60 #include "TypeQuelconque.h"
61 #include "Mail_initiale_Gmsh.h"
62 #include <map>
63 #include <vector>
64
65 /// @addtogroup Les_sorties_gmsh
66 /// @{
67 ///
68
69
70 class Isovaleurs_Gmsh : public OrdreVisu
71 {
72 public :
73     // CONSTRUCTEURS :
74     // par défaut

```

```

75     Isovaleurs_Gmsh () ;
76
77     // constructeur de copie
78     Isovaleurs_Gmsh (const Isovaleurs_Gmsh& algo);
79
80     // DESTRUCTEUR :
81     ~Isovaleurs_Gmsh () ;
82
83     // METHODES PUBLIQUES :
84     // initialisation : permet d'initialiser les différents paramètres de l'ordre
85     // lors d'un premier passage des différents incréments
86     // en virtuelle, a priori est défini si nécessaire dans les classes dérivées
87     // incre : numéro d'incrément qui en cours
88     void Initialisation(ParaGlob * ,LesMaillages * ,LesReferences*
89         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
90         ,Resultats*,EnumTypeIncre type_incre,int incre
91         ,const map < string, const double * , std::less <string> >& listeVarGlob
92         ,const List_io < TypeQuelconque >& listeVecGlob
93         ,bool fil_calcul) ;
94     // execution de l'ordre
95     // tab_mail : donne les numéros de maillage concerné
96     // incre : numéro d'incrément qui en cours
97     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
98     // animation : indique si l'on est en animation ou pas
99     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
100    void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
101        unseul_incre,LesReferences*
102        ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
103        ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int
104        incre
105        ,bool animation,const map < string, const double * , std::less <string> >&
106        listeVarGlob
107        ,const List_io < TypeQuelconque >& listeVecGlob);
108
109    // choix de l'ordre, cet méthode peut entraîner la demande d'informations
110    // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
111    void ChoixOrdre();
112
113    // initialisation de la liste des différentes isovaleurs possibles
114    void Init_liste_isovaleur(LesMaillages * lesMail,LesContacts* lesContacts,bool fil_calcul);
115    // initialisation d'une liaison avec une instance de classe de maillage initiale
116    void Jonction_MaillageInitiale(Mail_initiale_Gmsh * mailIni) {mailInitial = mailIni;};
117    // indique si oui ou non on utilise l'ancien format gmsh
118    bool Use_hold_gmsh_format() const {return use_hold_gmsh_format;};
119
120    // lecture des paramètres de l'ordre dans un flux
121    void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
122    // écriture des paramètres de l'ordre dans un flux
123    void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
124
125    // récupération de la liste des noms des différentes grandeurs à sortir
126    // à l'écriture, chaque grandeurs s'écrit sur un fichier propre
127    // cette fonction sert donc à préparer l'ouverture de ces fichiers
128    const list <string> & NomsGrandeurSortie() const {return nomsGrandeursSortie;};
129
130    //----- méthodes utilitaires qui sont également utilisées par d'autres classes-----
131    // sortie de la def des coordonnées pour une view
132    void SortieDefCoordonnees_old_format(ostream & sort,const int & incre
133        ,int nbmail, const Tableau <int>& tab_mail,LesMaillages * lesMail) const;
134    // sortie de l'utilisation d'une coordonnée
135    void SortieUseUneCoordonnee(int incre, ostream & sort,bool fin,int num_N) const;
136
137    // utilitaire pour la sortie de l'entête d'une view
138    // numero: dans le cas où c'est une view pour un élément de tableau, donc de numéro numero
139    void EnteteView(ostream & sort,const int & incre, const string& nom,const int & numero ) const;
140
141    // utilitaire pour la sortie de l'entête d'un bloc NodeData utilisé avec le nouveau format gmsh
142    // numero: dans le cas où c'est une view pour un élément de tableau, donc de numéro numero
143    void EnteteNodeData(ostream & sort,const int & incre, const double& temps
144        ,const string& nom,const int & numero ) const;
145    // ddl aux noeuds possibles à visualiser
146    const Tableau <List_io < Ddl_enum_etendu > >& Tabnoeud_type_ddl() const {return
147        tabnoeud_type_ddl;};
148
149    // def d'une classe à usage interne
150    class P_gauss
151    { public :
152        friend ostream & operator << (ostream & sort, const P_gauss & a)
153        { cout << a.elemgeom->TypeGeometrie() << a.elemgeom->TypeInterpolation() << a.elemgeom->Nbi()
154            << " ptinteg "
155            << a.nom_groupe_pt_integ << a.enu << a.num_maill << ", " ; return sort;};
156        P_gauss(): elemgeom(NULL),nom_groupe_pt_integ(""),enu(NU_DDL),num_maill(1) {};
157        P_gauss(const ElemGeomC0* eg,string nom,Enum_ddl en,const int& num_m):
158            elemgeom(eg),nom_groupe_pt_integ(nom),enu(en),num_maill(num_m) {};
159        P_gauss(const P_gauss& p):
160            elemgeom(p.elemgeom),nom_groupe_pt_integ(p.nom_groupe_pt_integ)
161            ,enu(p.enu),num_maill(p.num_maill) {};
162    }
163

```

```

158         bool operator== (const P_gauss& pg) const // égalité uniquement si même elem
159             {return ((pg.elemgeom==this->elemgeom)&&(num_maill==pg.num_maill));}; // géom et
même maillage
160         bool operator!= (const P_gauss& pg) const // inégalité sur l'élément géom ou
161             {return (!(pg==(*this)));}; // sur le maillage
162         // les données
163         const ElemGeomCO* elemgeom;
164         string nom_groupe_pt_integ;
165         Enum_ddl enu;
166         // numéro de maillage associé
167         int num_maill;
168     };
169
170     // ---tableaux pour optimiser et faciliter les sorties gmsh
171     // on utilise une class static pour la construction soit unique
172     class ConstructTabScalVecTensGmsh // pour construire les tableaux
173     { public : ConstructTabScalVecTensGmsh (); // constructeur par défaut qui permet de construire
les tableaux
174         Tableau <string> scalar_pourView; // tableau de 2 caractères, indicé par les numéros de type
d'élément gmsh
175             // scalar_pourView(i) donne pour l'élément gmsh i, le type scalaire pour View
176         Tableau <string> vector_pourView; //idem pour les vecteurs: 3 coordonnées / vecteur
177         Tableau <string> tensor_pourView; //idem pour les tenseurs: 9 coordonnées / tenseur
178         Tableau <string> text_pourView; //idem pour du texte en 2D ou 3D
179     };
180     static ConstructTabScalVecTensGmsh tabScalVecTensGmsh;
181
182     // récupération de l'instanceConstructTabScalVecTensGmsh
183     static const ConstructTabScalVecTensGmsh& TabScalVecTensGmsh(){return tabScalVecTensGmsh;};
184
185 protected :
186     // VARIABLES PROTEGEES :
187     bool use_hold_gmsh_format; // indique si oui ou non on utilise l'ancien format gmsh
188     bool absolue; // par défaut on sort en absolue les tenseurs
189     // pour pouvoir correctement les visualiser, même pour les elem 2D
190
191     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddl; // ddl aux noeuds possibles à visualiser
192     Tableau < List_io < Ddl_enum_etendu > > tabnoeud_type_ddl_retenu; // ddl à visualiser
193     Tableau <List_io <bool> > choix_var_ddl; // indique si c'est une variation ou le ddl
194
195     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddlEtendu; // ddl étendu aux noeuds possibles à
visualiser
196     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddlEtendu_retenu; // ddl étendu aux noeuds à
visualiser
197     // tab de travail: une sous partie de tabnoeud_type_ddlEtendu_retenu,
198     // qui ne contient que les grandeurs à accumuler aux noeuds venant des éléments, idem pour les ddl
initiauw
199     Tableau <List_io < Ddl_enum_etendu > > a_accumuler_tabnoeud_type_ddlEtendu;
200     Tableau <List_io < Ddl_enum_etendu > > a_accumuler_tabnoeud_type_ddlEtendu_retenu;
201
202     Tableau <List_io < TypeQuelconque > > tabnoeud_evoluee; // types évolués aux noeuds possibles à
visualiser
203     Tableau <List_io < TypeQuelconque > > tabnoeud_evoluee_retenu; // types évolués aux noeuds à
visualiser
204     // tab de travail: une sous partie de tabnoeud_evoluee_retenu,
205     // qui ne contient que les grandeurs à accumuler aux noeuds venant des éléments
206     Tableau <List_io < TypeQuelconque > > a_accumuler_tabnoeud_evoluee;
207     Tableau <List_io < TypeQuelconque > > a_accumuler_tabnoeud_evoluee_retenu;
208
209     // cas des vecteurs globaux, transférable directement aux noeuds
210     List_io < TypeQuelconque > list_vect_globalPourNoeud; // les vecteurs globaux que l'on peut
visualiser
211     List_io < TypeQuelconque > list_vect_globalPourNoeud_retenu; // les vecteurs globaux à visualiser
212
213     Tableau <List_io < Ddl_enum_etendu > > tabelement_type_ddl; // ddl aux elements possibles à
visualiser
214     Tableau <List_io < Ddl_enum_etendu > > tabelement_type_ddl_retenu; // ddl aux elements à visualiser
215
216     Tableau <List_io < TypeQuelconque > > tabelement_evoluee; // types evoluee aux elements possibles à
visualiser
217     Tableau <List_io < TypeQuelconque > > tabelement_evoluee_retenu; // type evoluee aux elements à
visualiser
218
219     Tableau <List_io < TypeQuelconque > > tabelement_typeParti; // types particuliers aux elements
possibles à visualiser
220     Tableau <List_io < TypeQuelconque > > tabelement_typeParti_retenu; // types particuliers aux
elements à visualiser
221
222     int cas_transfert; // indique la méthode utilisée pour le transfert aux noeuds
223
224     Mail_initiale_Gmsh * mailInitial; // pour une jonction avec le maillage initial
225     LesMaillages * lesMail; // les maillages
226
227     //--- variables internes pour la sortie Ddl_enum_etendu, de grandeurs quelconque évoluées ou non aux pt
d'integ -----
228     // dans Gmsh il n'y a pas de notion de maillage, il y a une seule liste de noeud et une seule liste
d'éléments

```

```

229 // (par contre je défini un groupe d'élément particulier pour chaque maillage mais ça n'intervient
pas ici en post-traitement)
230 // donc on a donc a définir
231 // 1) des groupes de pt_integ : qui indique combien et quel type de répartition ont un groupe de pt
d'integ.
232 // Le groupe est identifié par un nom = "nom de ptgauss", mais au
moment de l'écriture (donc de la
233 // définition du groupe pour Gmsh) on n'indique pas quels sont les
grandeurs (ddl) que l'on va écrire
234 // pour ce groupe, on peut donc ainsi utiliser un même groupe pour
plusieurs grandeurs.
235 // Le nom de groupe contient le numéro du maillage, donc les noms de
groupe sont différents pour
236 // chaque maillage. Le groupe de pt d'intég contient le ddl de
référence, auquel peuvent se rattacher
237 // plusieurs ddl secondaires (provenant des types évolués, ou
particulier) mais qui sont
238 // calculé aux mêmes pt d'integ que le ddl de base.
239
240 // 2) des listes de grandeurs à sortir pour un groupes de pt_integ donné, ces listes peuvent être
différentes pour chaque maillage
241 //
242 // --- au bilan on a en stockage globale: ---
243 // li_P_gauss_total: contient tous les noms de groupe de pt integ différent et l'élément géométrique
associé (nb pt etc),
244 // et le ddl associé
245 // tab_point_enum: permet a partir du ddl de retrouver un élément de li_P_gauss_total associé
246 // map_gauss_base: permet de connaitre pour un nom_groupe_pt_integ donné, la liste des
Ddl_enum_etendu à sortir
247 // map_gauss_baseEVol: idem pour les grandeurs évoluées
248 // map_gauss_basePQ: idem pour les grandeurs quelconques
249 // map_gauss_tab_baseEVol et map_gauss_tab_basePQ : définissent des tableaux de npt conteneurs
associés
250 // à map_gauss_tab_baseEVol et
251 // map_gauss_tab_basePQ, ceci pout une manipulation
globale de toutes les grandeurs à tous les pt
d'integ d'un groupe
252 // En fait quand on calcul les grandeurs aux pt d'integ, on les stocke dans le tableau (donc
l'élément de
253 // map_gauss_tab_baseEVol ou map_gauss_tab_basePQ) au lieu de la liste map_gauss_baseEVol ou
map_gauss_basePQ. Donc le véritable
254 // conteneur c'est le tableau (les 2), les listes initiales (élément de map_gauss_baseEVol ou
map_gauss_basePQ) ne servent que
255 // 1) pour les entêtes, 2) à savoir s'il y a des grandeurs à considérer ou non, et 3) à
construire les tableaux.
256 //
257 // --- maintenant les grandeurs associées à chaque maillage ---
258 // tp_tp_tp_gauss_base : tp_tp_tp_gauss_base(imail)(isous_mail) contient tous les groupe de pt
d'integ différent pour un
259 // sous maillage donné
260
261
262 // on crée une liste de P_gauss contenant en autre une
263 // référence d'élément géométrique associés aux ddl
264 // pour ne pas créer de groupe de pt_integ inutile car identique à un groupe déjà existant.
265 // li_P_gauss contient "tous" les associations "nom de ptgauss" élément géom et un ddl de référence
266 list < P_gauss > li_P_gauss_total;
267 // on utilise un même "nom de ptgauss" pour plusieurs ddl, le rassemblement s'effectue dans
l'écriture
268
269 // un tableau de travail que l'on dimentionne qu'en locale dans ExeOrdrePremierIncrement
270 // Tableau < P_gauss * > tab_point_enum; // permet de retrouver le P_gauss dans la liste
li_P_gauss
271 //associé à un enum de base
272 // en fait est construit à partir de li_P_gauss_total, et est indicer par l'énum du P_gauss,
273
274 // le tableau des P_gauss représente les groupes de pt de gauss existants (l'énum ici n'a pas
d'importance)
275 // pointe sur des éléments de li_P_gauss_total
276 // donc tp_gauss_base(imail) (ims) (i) = pour le maillage imail, pour le sous_maillage ims, donne la
liste des pt_gauss
277 // de base qui permettent de générer tous les ddl (de base ou dérivés)
278 Tableau < Tableau < Tableau < P_gauss > > > tp_tp_tp_gauss_base;
279 // permet de récolter tous les "nom de ptgauss" utilisé par le maillage imail
280
281 // une map pour faire l'association entre le nom du groupe de pt de gauss et la liste
282 // de ddl étendue associée
283 map < string, List_io < Ddl_enum_etendu > , std::less <string> > map_gauss_base;
284 // une map pour faire l'association entre le nom du groupe de pt de gauss et la liste
285 // de type quelconque associée contenant les types évolués
286 map < string, List_io < TypeQuelconque > , std::less <string> > map_gauss_baseEVol;
287 // idem mais pour un tableau de liste, indicé par le nombre de ptinteg
288 map < string, Tableau < List_io < TypeQuelconque > > , std::less <string> > map_gauss_tab_baseEVol;
289
290 // une map pour faire l'association entre le nom du groupe de pt de gauss et la liste
291 // de type quelconque associée contenant les grandeurs particulières
292 map < string, List_io < TypeQuelconque > , std::less <string> > map_gauss_basePQ;

```



```

293 // idem mais pour un tableau de liste, indiqué par le nombre de ptinteg
294 map < string, Tableau < List_io < TypeQuelconque > > , std::less <string> > map_gauss_tab_basePQ;
295
296 //--- fin variables internes pour la sortie Ddl_enum_etendu et type quelconque aux pt d'integ -----
297
298 //--- variables internes pour la sortie aux noeuds -----
299 // pour les sorties aux noeuds on globalise toutes les sorties
300 // et on definit un tableau de booléen qui indique si oui ou non la grandeur est à sortir en
fonction des tableaux
301 // tabelement_type_ddl_retenu, tabelement_evoluee_retenu, tabelement_typeParti_retenu etc
302
303 // .. pour les ddl aux noeuds
304 List_io < Ddl_enum_etendu > glob_noe_ddl_retenu; // le global
305 Tableau <List_io < bool > > t_g_noeud_ddl_asortir;
306 // .. pour les ddl étendues définis aux noeuds
307 List_io < Ddl_enum_etendu > glob_noeud_ddl_etendu_retenu; // le global
308 Tableau <List_io < bool > > t_g_noeud_ddl_etendu_asortir;
309 // .. pour les types évoluées définis aux noeuds
310 List_io < TypeQuelconque > glob_noeud_evol_retenu; // le global
311 Tableau <List_io < bool > > t_g_noeud_evoluee_asortir;
312 // .. pour les ddl venant d'un transfert aux noeuds des grandeurs aux pt d'integ
313 List_io < Ddl_enum_etendu > glob_elem_ddl_retenu; // le global
314 Tableau <List_io < bool > > t_g_elem_ddl_asortir;
315 // .. pour les types évoluées venant d'un transfert aux noeuds des grandeurs aux pt d'integ
316 List_io < TypeQuelconque > glob_elem_evol_retenu; // le global
317 Tableau <List_io < bool > > t_g_elem_evoluee_asortir;
318 // .. pour les types particuliers aux elements venant d'un transfert aux noeuds
319 List_io < TypeQuelconque > glob_elem_Partir_retenu; // le global
320 Tableau <List_io < bool > > t_g_elem_typeParti_asortir;
321 // .. globalise tous les types quelconques ...
322 Tableau <List_io < TypeQuelconque > * > tab_quelconque;
323
324 // -- particularités liées aux contacts: dont les grandeurs sont stockées aux noeuds
325 // la liste suivante contient les infos choisies restreintes aux contacts
326 List_io < TypeQuelconque > li_glob_restreinte_TQ;
327
328
329 //--- fin variables internes pour la sortie aux noeuds -----
330
331 list <string > nomsGrandeursSortie; // un tableau de nom qui globalise l'ensemble des noms
332 // des grandeurs à sortir. On utilise list au lieu de Tableau because pb avec utillecture
333 // qui ne peut pas utiliser de Tableau. Mais ça n'a pas de conséquences
334
335
336
337 // METHODES PROTEGEES :
338
339 // constructeur utilisé par les classes dérivées
340 // en fait il s'agit de transmettre directement les infos à la classe mère Visu
341 Isovaleurs_Gmsh(const string& comment_som, const string& explication
342 , const string& ordre) :
343     OrdreVisu(comment_som,explication,ordre)
344     {};
345
346 // définition interactives des paramètres généraux des isovaleurs
347 void ParametresGeneraux(const string& choix);
348 // choix de l'isovaleur à visualiser calculée à partir des ddl naturels aux noeuds
349 void ChoixIsovaleur_noeud(const string& choix);
350 // choix de l'isovaleur à visualiser calculée à partir des ddl etendu aux noeuds
351 void ChoixIsovaleur_ddl_etendu_noeud(const string& choix);
352 // choix de l'isovaleur à visualiser calculée à partir de grandeurs évoluées aux noeuds
353 void ChoixIsovaleur_evoluee_noeud(const string& choix);
354 // choix de l'isovaleur à visualiser calculée à partir des ddl aux points d'intégrations
355 void ChoixIsovaleur_ddl_ptinteg(const string& choix);
356 // choix de l'isovaleur à visualiser calculée à partir des grandeurs évoluées aux points
d'intégrations
357 void ChoixIsovaleur_tensorielle_ptinteg(const string& choix);
358 // choix de l'isovaleur à visualiser calculée à partir des grandeurs quelconques aux points
d'intégrations
359 void ChoixIsovaleur_quelc_ptinteg(const string& choix);
360 // initialisation de l'exécution du transferts
361 void InitPremierIncrExecutionTransfertAuxNoeuds();
362 // exécution du transfert
363 void ExecutionTransfert();
364 // vérification que le transfert peut se faire et pas de doublon de grandeurs
365 void VerificationTransfertPossible_et_doublon();
366 // passage des grandeurs choisies venant des éléments, aux noeuds, lorsque ces grandeurs sont
367 // directement disponibles aux noeuds
368 void Passage_grandeursRetenuesAuxNoeuds_elementsVersNoeuds();
369 // passage des grandeurs disponibles venant des éléments, aux noeuds, lorsque ces grandeurs sont
370 // directement disponibles aux noeuds
371 void Passage_grandeursDisponiblesAuxNoeuds_elementsVersNoeuds();
372 // initialisation les différentes listes internes qui globalisent tous les maillages
373 void GlobalisationListSurTousLesMaillagesPourLesNoeuds();
374 // exeoOrdre: cas du premier increments
375 void ExeOrdrePremierIncrement(const Tableau <int>& tab_mail, LesMaillages * lesMail);
376 // calcul des grandeurs aux points d'intégration

```

```

377 void CalculAuxPtInteg(const Tableau <int>& tab_mail,LesMaillages * lesMail);
378 // calcul des grandeurs aux noeuds, directement accessible des éléments
379 void CalculAuxElemAuxNoeuds(const Tableau <int>& tab_mail,LesMaillages * lesMail);
380 // écriture des grandeurs aux noeuds à l'ancien format
381 void EcritureAuxNoeuds_ancien_format(const Tableau <int>& tab_mail,LesMaillages * lesMail
382 ,int incre,UtilLecture & entreePrinc);
383 // écriture des grandeurs aux noeuds au nouveau format
384 void EcritureAuxNoeuds(const Tableau <int>& tab_mail,LesMaillages * lesMail
385 ,int incre,UtilLecture & entreePrinc);
386 // -- cas de l'ancien format ---
387 // sortie d'une view pour un type quelconque simple !!, c'est-à-dire pas un tableau par exemple
388 // en fait ne concerne que les types scalaires, vectoriels, ou tensoriels
389 // dans le cas où le type simple provient d'un tableau
390 // numero donne un numéro de l'élément à sortir dans l'entête de la view si diff de 0
391 void SortirUneViewTypeQuelconque_old_format(ostream & sort,TypeQuelconque_enum_etendu enu
392 ,EnumType2Niveau enuStructure, int numero,Tableau <List_io < bool >::iterator >&
t_indic_sortie,int nbmail
393 ,const Tableau <int>& tab_mail,EnumTypeGrandeur enugrandeur,LesMaillages *
lesMail,const int& incre);
394 // sortie d'une view pour un type quelconque simple !!, c'est-à-dire pas un tableau par exemple
395 // en fait ne concerne que les types scalaires, vectoriels, ou tensoriels
396 // dans le cas où le type simple provient d'un tableau
397 // numero donne un numéro de l'élément à sortir dans l'entête de la view si diff de 0
398 void SortirUneViewTypeQuelconque(ostream & sort,TypeQuelconque& typ
399 ,EnumType2Niveau enuStructure, int numero,Tableau <List_io < bool >::iterator >&
t_indic_sortie,int nbmail
400 ,const Tableau <int>& tab_mail,EnumTypeGrandeur enugrandeur,LesMaillages *
lesMail,const int& incre);
401
402 ///-- ancien format // utilitaire pour la sortie d'un tenseur HH
403 void SortieTenseurHH_ancien_format(ostream & sort,const TenseurHH& t_HH);
404 ///-- ancien format // utilitaire pour la sortie d'un tenseur BB
405 void SortieTenseurBB_ancien_format(ostream & sort,const TenseurBB& t_BB);
406 // utilitaire pour la sortie d'un tenseur HH
407 void SortieTenseurHH(ostream & sort,const TenseurHH& t_HH);
408 // utilitaire pour la sortie d'un tenseur BB
409 void SortieTenseurBB(ostream & sort,const TenseurBB& t_BB);
410 // utilitaire pour une sortie de type Vecteur
411 void SortieVecteur(ostream & sort,const Vecteur& vec);
412 // création de la liste des noms correspondants à tous les grandeurs à sortir
413 void CreatListNomsTousLesGrandeurs();
414
415 };
416 /// @} // end of group
417
418 #endif

```

## 7.419 Mail\_initiale\_Gmsh.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 * DATE: 07/01/2008 *
32 * * $ *
33 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
34 * * $ *
35 * PROJET: Herezh++ *

```

```

36 *                                                                 $ *
37 *****
38 *      BUT:  Visualisation du maillage initiale en Gmsh.
39 *      Uniquement l'ordre d'exécution est différent de la classe *
40 *      vrml.
41 *                                                                 $ *
42 *      *****
43 *      VERIFICATION:
44 *
45 *      ! date ! auteur ! but !
46 *      -----
47 *      ! ! ! !
48 *                                                                 $ *
49 *      *****
50 *      MODIFICATIONS:
51 *      ! date ! auteur ! but !
52 *      -----
53 *                                                                 $ *
54 *****/
55 #ifndef MAIL_INITIALE_GMSH_T
56 #define MAIL_INITIALE_GMSH_T
57
58 #include "OrdreVisu.h"
59
60 /// @addtogroup Les_sorties_gmsh
61 /// @{
62 ///
63
64
65 class Mail_initiale_Gmsh : public OrdreVisu
66 {
67 public :
68     // CONSTRUCTEURS :
69     // par défaut
70     Mail_initiale_Gmsh () ;
71
72     // constructeur de copie
73     Mail_initiale_Gmsh (const Mail_initiale_Gmsh& algo);
74
75     // DESTRUCTEUR :
76     ~Mail_initiale_Gmsh () ;
77
78     // METHODES PUBLIQUES :
79     // initialisation : permet d'initialiser les différents paramètres de l'ordre
80     // lors d'un premier passage des différents incréments
81     // en virtuelle, a priori est défini si nécessaire dans les classes dérivées
82     // incre : numéro d'incrément qui en cours
83     void Initialisation(ParaGlob * ,LesMaillages * ,LesReferences*
84         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
85         ,Resultats*,EnumTypeIncre type_incre,int incre
86         ,const map < string, const double * , std::less <string> >& listeVarGlob
87         ,const List_io < TypeQuelconque >& listeVecGlob
88         ,bool fil_calcul) ;
89
90     // execution de l'ordre
91     // tab_mail : donne les numéros de maillage concerné
92     // incre : numéro d'incrément qui en cours
93     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
94     // animation : indique si l'on est en animation ou pas
95     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
96     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
97         unseul_incre,LesReferences*
98         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
99         ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
100         ,bool animation,const map < string, const double * , std::less <string> >&
101         listeVarGlob
102         ,const List_io < TypeQuelconque >& listeVecGlob);
103
104     // choix de l'ordre, cet méthode peut entraîner la demande d'informations
105     // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
106     void ChoixOrdre();
107
108     // lecture des paramètres de l'ordre dans un flux
109     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
110     // écriture des paramètres de l'ordre dans un flux
111     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
112
113     // récupération éventuelle de la liste des noms des différentes référence à sortir
114     // à l'écriture, chaque grandeurs s'écrit sur un fichier propre
115     // cette fonction sert donc à préparer l'ouverture de ces fichiers
116     // dans le cas où le tableau est vide cela signifie que les ref seront dans le fichier principal
117     const Tableau <string>* NomsGrandeurSortie() const
118         {if (sortie_des_references==2) {return &tab_nom_tag_ref;} else {return NULL;}};
119
120     // ramène le tableau de listes de sous maillage
121     const Tableau < Tableau < List_io < Element* > > > &
122         Tableau_de_sous_maillage() const {return tabtab_sous_mesh;};

```

```

120 // ramène le tableau de le listes de types d'éléments différents
121 const Tableau < List_io <Element::Signature> >&
122     Tab_liste_type_element() const { return tabli_sig_elem;};
123 // ramène le tableau de listes de noms des sous maillages
124 const Tableau < List_io <string> >&
125     Tab_listes_nom_sousMaillage() const { return tabli_sous_mesh;};
126 // ramène le décalages pour les numéros de noeuds du maillage "nmail"
127 int DecalNumNoeud(int nmail) const {return decal_noeud(nmail);};
128 // ramène le décalages pour les numéros d'éléments du maillage "nmail"
129 int DecalNumElement(int nmail) const {return decal_element(nmail);};
130
131 // ramène le tableau t_Egmsh
132 // t_Egmsh(im)(ie): donne pour le maillage im, et pour l'élément ie, le numéro gmsh de l'élément
133 const Tableau < Tableau < int > >& TabEgmsh() const {return t_Egmsh;};
134
135 // indique s'il y a quelque chose à regarder du coté des homothétie sur le maillage initiale
136 bool Considerer_homothetie() const {return considerer_homothetie;};
137 // ramène un tableau de bool qui indique si oui ou non on fait une pseudo-homothétie sur les
// coordonnées initiales de chaque maillage
138 const Tableau < bool >& T_homothetie() const{return t_homothetie ;};
139 // ramène l'origine et les facteurs multiplicatifs pour une pseudo-homothétie pour chaque maillage
140 const Tableau < Coordonnee >& T_orig()const {return t_orig;};
141 const Tableau < Coordonnee >& T_fact_mult() const {return t_fact_mult;};
142
143 // sortie maillage initiale (sans références ni tag) avec le nouveau format
144 void sortie_maillage_initial(const Tableau <int>& tab_mail, LesMaillages * lesMail, ostream &sort);
145 // ramène le nombre total de noeuds
146 int Nb_total_noeud()const {return nb_total_noeud;};
147 // ramène le nombre total d'éléments
148 int Nb_total_element() const {return nb_total_element;};
149
150 private :
151 // VARIABLES PROTEGEES :
152 // tableau de liste de sous maillages
153 // chaque sous maillage représente un type d'élément fini particulier
154 // l'indice du tableau= le numéro de maillage
155 Tableau < Tableau < List_io < Element* > > > tabtab_sous_mesh;
156 // tableau de listes de types d'éléments différents
157 Tableau < List_io <Element::Signature> > tabli_sig_elem;
158 // tableau de listes des nom des sous_maillages créés
159 Tableau < List_io <string> > tabli_sous_mesh;
160 // tableau des décalages pour les numéros de noeuds et d'éléments
161 Tableau <int > decal_noeud;
162 Tableau <int > decal_element;
163 int nb_total_noeud; // nombre total de noeuds
164 int nb_total_element; // nombre total d'éléments
165 int nb_total_ref; // nombre total de références
166 int sortie_des_references; // = 0, 1 ou 2: indique si oui (diff de 0) ou non
167 // on veut une sortie des références
168 // = 1: par défaut sortie des ref, dans un fichier unique
169 // = 2: sortie des refs dans des fichiers séparés
170 Tableau <string> tab_nom_tag_ref; //le tableau des noms des tag correspondant aux ref
171 // tab_nom_tag_ref(i) = le nom du tag associé à la ref i, i étant le numéro d'apparition
172 // de la référence, quand on parcourt toutes les références
173 // tableau construit dans la méthode: Initialisation(..
174 Tableau <const Reference* > lesRefInitiales; // le tableau des ref initiales, construit avec
Initialisation(..
175
176 Tableau <int > tab_num_tag; // le tableau des numéros des tag correspondant aux ref
177 // tab_num_tag(i) = le numéro du tag associé à la ref i, i étant le numéro d'apparition
178 // de la référence, quand on parcourt toutes les références
179 // tableau construit dans la méthode:
SortieTagPhysicalNames
180 // concernant les éléments points qu'il faut rajouter pour voir sous gmsh les références de noeud !!
181 // un tableau d'adressage indirecte qui permet d'utiliser ces noeuds éléments
182 Tableau <Tableau <int > > t_jonction_N_Nelem;
183 // t_jonction_N_Nelem(i)(j) donne le numéro de noeud_élément correspondant à l'ancien numéro j pour
le maillage i
184 // n'est rempli que pour les noeuds ayant conduit à la création d'un noeud_élément
185
186 // concernant les ref de pti: on stocke le nombre de noeud additionnel et de noeud élément
additionnel:
187 int nombre_noeud_ref_pti; // c'est le même nombre: à chaque pti on ajoute un noeud et un noeud
élément
188 int nombre_ref_pti; // nombre de référence de point d'intégration
189 // un tableau d'adressage indirecte qui permet d'utiliser ces noeuds éléments pour les pti
190 Tableau <Tableau <int > > t_jonction_N_NelemPti;
191 // t_jonction_N_NelemPti(i)(j) donne le numéro de noeud_élément correspondant
192 // au numéro j dans la ref, pour la référence de pti i
193 // n'est rempli que pour les noeuds ayant conduit à la création d'un noeud_élément
194
195 Tableau <int > tab_dim_de_ref; // tab_dim_de_ref(i) donne la dimension des éléments
196 // de la ref de nom tab_nom_tag_ref(i)
197 // 1 pour ligne 2 pour surface, 3 pour volume, 0 pour autre
198 // un tableau de liaison entre les éléments herezh des maillages et les numéros associés de gmsh
199 // t_Egmsh(im)(ie): donne pour le maillage im, et pour l'élément ie, le numéro gmsh de l'élément
200 Tableau < Tableau < int > > t_Egmsh;

```

```

201
202 bool considerer_homothetie; // indique s'il y a quelque chose à regarder du coté des homotheties
203     // (on pourrait faire avec t_homothetie mais c'est plus facile avec un paramètre globale)
204 Tableau < bool > t_homothetie ; // oui ou non on fait une pseudo-homothétie sur les coordonnées
205     // initiales de chaque maillage
206 // on considère un rapport d'homothétie qui peut-être différent sur les 3 axes d'où le nom de pseudo
207 Tableau < Coordonnee > t_orig,t_fact_mult; // origine et facteurs multiplicatifs pour
208     // une pseudo-homothétie pour chaque maillage
209
210 // METHODES PROTEGEES :
211 // sortie de la définition des tag de PhysicalNames (sert pour les références)
212 void SortieTagPhysicalNames(const Tableau <int>& tab_mail, LesMaillages * lesMail, ostream &sort);
213 // sortie uniquement des noeuds avec éventuellement, def de noeud aux points d'intégration
214 void SortieDesNoeuds(const Tableau <int>& tab_mail, LesMaillages * lesMail, ostream &sort);
215 // sortie uniquement des éléments
216 void SortieDesElements(const Tableau <int>& tab_mail, LesMaillages * lesMail, ostream &sort);
217 // sortie des références
218 void SortieReferences(const Tableau <int>& tab_mail, LesMaillages * lesMail, UtilLecture &
entreePrinc);
219 // sortie éventuelle d'un seul élément au format gmsh
220 // et modification éventuelle du tableau t_Egmsh
221 // num_elem : numero initial de l'élément
222 // decal_ele : décalage de numéro pour tenir compte de la présence de plusieurs maillages
223 //     (ou sous maillages)
224 // decal_noe : décalage de numéro de noeud, pour tenir compte de la présence de plusieurs maillages
225 // tab_noeud : le tableau des noeuds de la connection
226 // id_geom et id_interpol : permettent de repérer un type d'élément fini associé
227 //     (donc pas seulement géométrique)
228 //     : il s'agit ici de la géométrie et interpolation "élément fini"
229 // sort : flux de sortie
230 // modifier_t_Egmsh : indique si pour l'élément considéré, on abonde le tableau t_Egmsh
231 // im : num du maillage ou sous maillage
232 // elem_a_sortir : indique si pour l'élémen considéré, on le sort dans le fichier gmsh
233 void SortieDunElements(bool elem_a_sortir, const int& num_elem
234     , ostream &sort
235     , const int& decal_ele, Tableau<Noeud *>& tab_noeud
236     , const int& decal_noe, Enum_geom id_geom, Enum_interpol id_interpol
237     , bool modifier_t_Egmsh , int im);
238 // constitution des sous maillages, utilisé par les isovaleurs par exemple
239 // on suit la même logique que dans gid, un sous maillage par type d'élément
240 // ce n'est pas nécessaire, c'est pas simplicité, à modifier si nécessaire par la suite
241 void CreaSousMaillages(const Tableau <int>& tab_mail, LesMaillages * lesMail);
242 // sortie sur le flot passé en paramètre, de la partie noeud du maillage sans entête ni fin
243 void Sortie_noeuds_initiaux(const Tableau <int>& tab_mail, LesMaillages * lesMail, ostream &sort);
244 // sortie sur le flot passé en paramètre, de la partie élément du maillage sans entête ni fin
245 void Sortie_element_initiaux(const Tableau <int>& tab_mail, LesMaillages * lesMail, ostream &sort);
246 };
247 /// @} // end of group
248
249 #endif

```

## 7.420 Visuali\_Gmsh.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      07/01/2008
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 *
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 *
49 *
50 *
51 *
52 *
53 *
54 *
55 *
56 *
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```

```

33 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)           *
34 *                                                     $           *
35 *   PROJET:      Herezh++                                     *
36 *                                                     $           *
37 * *****
38 *   BUT: Arrêt des question Mise en route de la visualisation *
39 *         en Gmsh.                                           *
40 *                                                     $           *
41 *   ////////////////////////////////////////////////// *
42 *   *
43 *   VERIFICATION:                                           *
44 *   ! date ! auteur ! but ! *
45 *   ----- *
46 *   ! ! ! ! *
47 *   $ *
48 *   ////////////////////////////////////////////////// *
49 *   MODIFICATIONS:                                           *
50 *   ! date ! auteur ! but ! *
51 *   ----- *
52 *   $ *
53 * *****/
54 #ifndef VISUALI_GMSH_T
55 #define VISUALI_GMSH_T
56
57 #include "OrdreVisu.h"
58
59 /** @defgroup Les_sorties_gmsh
60 *
61 *   BUT: groupe concernant le postraitement avec gmsh
62 *
63 *
64 *   \author Gérard Rio
65 *   \version 1.0
66 *   \date 07/01/2008
67 *   \brief groupe concernant le postraitement avec gmsh
68 *
69 */
70
71 /// @addtogroup Les_sorties_gmsh
72 /// @{
73 ///
74
75
76 class Visuali_Gmsh : public OrdreVisu
77 {
78 public :
79     // CONSTRUCTEURS :
80     // par défaut
81     Visuali_Gmsh () ;
82
83     // constructeur de copie
84     Visuali_Gmsh (const Visuali_Gmsh& algo);
85
86     // DESTRUCTEUR :
87     ~Visuali_Gmsh () ;
88
89     // METHODES PUBLIQUES :
90     // execution de l'ordre
91     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
92                 ,LesLoisDeComp* ,DiversStockage* ,Charge* ,LesCondLim*
93                 ,LesContacts* ,Resultats* ,UtilLecture & ,OrdreVisu::EnumTypeIncre,int
94                 ,bool,const map < string, const double * , std::less <string> >&
95                 ,const List_io < TypeQuelconque >& listeVecGlob)
96     {};
97
98     // lecture des paramètres de l'ordre dans un flux
99     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
100    // écriture des paramètres de l'ordre dans un flux
101    void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
102
103 private :
104     // VARIABLES PROTEGEES :
105
106     // METHODES PROTEGEES :
107
108 };
109 /// @} // end of group
110
111 #endif

```

## 7.421 Visualisation\_Gmsh.h

```

1
2 // This file is part of the Herezh++ application.

```

```

3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           07/01/2008
32 *
33 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:        Herezh++
36 *
37 *   ****
38 *   BUT: Gérer et définir les fichiers de visualisation Gmsh.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !           !
47 *   *****
48 *   MODIFICATIONS:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   $
53 *****/
54 #ifndef VISUALISATION_GMSH_H
55 #define VISUALISATION_GMSH_H
56
57 #include "UtilLecture.h"
58 // #include "bool.h"
59 #include "MotCle.h"
60 #include "string.h"
61 #include "Tableau_T.h"
62 #include "LesMaillages.h"
63 #include "LesReferences.h"
64 #include "LesCondLim.h"
65 #include <list>
66 #include "LesContacts.h"
67 #include "LesValVecPropres.h"
68 #include "LesLoisDeComp.h"
69 #include "DiversStockage.h"
70 #include "Charge.h"
71 #include "LesContacts.h"
72 #include "Resultats.h"
73 #include "OrdreVisu.h"
74
75 /// @addtogroup Les_sorties_gmsh
76 /// @{
77
78
79 class Visualisation_Gmsh
80 {
81 public :
82     // CONSTRUCTEURS :
83     // le constructeur par défaut ne doit pas être utilisé
84     // il y a dans ce cas un message d'erreur et arrêt
85     Visualisation_Gmsh ();
86     // le bon constructeur
87     Visualisation_Gmsh (UtilLecture* ent);
88

```

```

89 // DESTRUCTEUR :
90 ~Visualisation_Gmsh ();
91
92 // METHODES PUBLIQUES :
93 // affichage des différentes possibilités (ordres possibles)
94 // ramène un numéro qui renseigne le programme appelant
95 // == -1 : signifie que l'on veut stopper la visualisation
96 // = 0 : signifie que l'on demande la visualisation effective
97 int OrdresPossible();
98
99 // information de l'instance de la liste d'incrément disponible pour la visualisation
100 void List_increment_disponible(list <int> & list_incr) ;
101 // indique le choix de la liste d'incrément à utiliser pour l'initialisation
102 // des fonctions d'initialisation
103 const list<int> & List_balaie_init();
104 // impose une liste d'incrément à utiliser
105 void List_balaie_init(const list<int> & list_init);
106 // indique le choix de la liste d'incrément à visualiser
107 const list<int> & List_balaie() {return *list_balaie;};
108 // information de l'instance du nombre de maillages disponibles pour la visualisation
109 // par défaut tous les maillages seront visualisés, ceci est descidé aussi ici
110 void List_maillage_disponible(int nombre_maillage_dispo) ;
111
112 // initialisation des ordres disponibles
113 // par exemple pour les isovaleurs on définit la liste des isovaleurs disponibles et les extrémas
114 void Initialisation(ParaGlob * paraGlob,LesMaillages * lesMaillages,LesReferences* lesReferences
115 ,LesLoisDeComp* lesLoisDeComp,DiversStockage* diversStockage
116 ,Charge* charge,LesCondLim* lesCondLim,LesContacts* lesContacts
117 ,Resultats* resultats,OrdreVisu::EnumTypeIncre type_incre,int incre
118 ,const map < string, const double * , std::less <string> >& listeVarGlob
119 ,const List_io < TypeQuelconque >& listeVecGlob
120 ,bool fil_calcul);
121
122 // méthode principale pour activer la visualisation
123 // sort : le flux de visualisation
124 void Visu(ParaGlob * paraGlob,LesMaillages * lesMaillages,LesReferences* lesReferences
125 ,LesLoisDeComp* lesLoisDeComp,DiversStockage* diversStockage
126 ,Charge* charge,LesCondLim* lesCondLim,LesContacts* lesContacts
127 ,Resultats* resultats,OrdreVisu::EnumTypeIncre type_incre,int incre
128 ,const map < string, const double * , std::less <string> >& listeVarGlob
129 ,const List_io < TypeQuelconque >& listeVecGlob);
130
131 // == définition des paramètres de visualisation
132 // titre, navigation, éclairage
133 // et initialisation des paramètres de la classe
134 void Contexte_debut_visualisation();
135 // (points de vue) et enchainement si nécessaire
136 void Contexte_fin_visualisation();
137
138 // lecture des paramètres de l'ordre dans un flux
139 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
140 // écriture des paramètres de l'ordre dans un flux
141 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
142 // demande si la visualisation Gmsh est validé ou pas
143 bool Visu_Gmsh_valide() {return activ_sort_Gmsh;};
144 // inactive la visualisation Gmsh
145 void Inactiv_Visu_Gmsh() {activ_sort_Gmsh=false;};
146
147 // récupération de la liste des noms des différentes grandeurs à sortir en résultat
148 const list <string> & NomsTousLesGrandeursSortie(const {return list_tousLesNomsASortir;};
149
150 // retour de la connection herezh vers gmsh
151 // t_HerGmsh(i)(j) : donne pour le noeud j de l'élément i d'herezh, le numéro gmsh
152 static const Tableau < Tableau < int > > & Tab_HerGmsh() {return t_HerGmsh;};
153 // retour de la connection gmsh vers herezh
154 // t_GmshHer(i)(j) : donne pour le noeud j de l'élément i de gmsh, le numéro d'herezh
155 static const Tableau < Tableau < int > > & Tab_GmshHer() {return t_GmshHer;};
156
157 private :
158 // VARIABLES PROTEGEES :
159 UtilLecture* entreePrinc; // gestion des entrees/sorties
160 list <OrdreVisu> ordre_possible; // l'ensemble des ordres possibles
161 // la sauvegarde de valeurs propres et vecteurs propres éventuelles
162 LesValVecPropres lesValVecPropres;
163 OrdreVisu* fin_o; // ordre de fin de la visualisation
164 OrdreVisu* mailInit; // Maillage initial
165 OrdreVisu* visuali; // ordre de visualiser
166 OrdreVisu* choix_inc; // ordre de choix des incréments
167 OrdreVisu* ptdeformee; // déformée
168 list <int> list_incre; // liste des incréments possibles
169 const list <int>* list_balaie; // liste des incréments à visualiser
170 int nb_maillage_dispo; // le nombre de maillage disponible
171 OrdreVisu* choix_mail; // ordre de choix des maillages
172 OrdreVisu* choix_isovaleur; // ordre de visualisation d'isovaleurs
173 // indication si l'on est en animation ou pas
174 bool animation;
175 // indication si la visualisation de type Gmsh est active ou pas, par défaut = faux

```



```

176     bool activ_sort_Gmsh;
177     // liste de tous les noms de grandeurs à sortir
178     list <string> list_tousLesNomsASortir;
179
180 protected :
181     // variable static servant pour toutes les classes travaillant avec gmsh
182     // on définit un tableau de connection entre la numérotation gmsh et celle d'herezh++
183     // t_HerGmsh(i)(j) : donne pour le noeud j de l'élément i d'herezh, le numéro gmsh
184     static Tableau < Tableau < int > > t_HerGmsh;
185     // idem mais l'inverse
186     static Tableau < Tableau < int > > t_GmshHer;
187
188     // METHODES PROTEGEES :
189     // test si la réponse fait partie des ordres possibles
190     // si l'on trouve un ordre ok on ramène un pointeur sur l'ordre
191     // sinon on ramène un pointeur null
192     OrdreVisu* Existe(string& reponse);
193     // affichage des options possibles
194     void Affiche_options();
195
196
197 };
198 /// @} // end of group
199
200 #endif

```

## 7.422 Animation\_maple.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          23/01/97
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *
38 *      BUT:           Sortie de l'animation pour le format gnuplot et autre.
39 *                   Celle-ci est automatique lorsqu'il y a plus d'un increment
40 *                   différent de 0.
41 *
42 *                   *****
43 *
44 *      VERIFICATION:
45 *
46 *      ! date !   auteur !           but
47 *      !           !           !
48 *
49 *                   *****
50 *      MODIFICATIONS:
51 *      ! date !   auteur !           but
52 *      !           !           !
53 *
54 *                   *****/
55 #ifndef ANIMATION_MAPLE_T
56 #define ANIMATION_MAPLE_T

```

```

57
58
59 #include "OrdreVisu.h"
60 #include "ChoixDesMaillages_vrml.h"
61 #include "Basiques.h"
62 #include "Choix_grandeurs_maple.h"
63
64 /// @addtogroup Les_sorties_au_format_maple
65 /// @{
66 ///
67
68
69 class Animation_maple : public OrdreVisu
70 {
71 public :
72 // CONSTRUCTEURS :
73 // par défaut
74 Animation_maple () ;
75
76 // constructeur de copie
77 Animation_maple (const Animation_maple& ord);
78
79 // DESTRUCTEUR :
80 ~Animation_maple () ;
81
82 // METHODES PUBLIQUES :
83
84 // initialisation : permet d'initialiser les différents paramètres de l'ordre
85 // lors d'un premier passage des différents incréments
86 // en virtuelle, a priori est défini si nécessaire dans les classes dérivées
87 void Initialisation(ParaGlob * paraGlob,LesMaillages * lesmail,LesReferences* lesRefer
88 ,LesLoisDeComp* lesLoisDeComp,DiversStockage* diversStockage,Charge* charge
89 ,LesCondLim* lesCondLim,LesContacts* lesContacts
90 ,Resultats* resultats,EnumTypeIncre type_incre,int incre
91 ,const map < string, const double * , std::less <string> >& listeVarGlob
92 ,const List_io < TypeQuelconque >& listeVecGlob
93 ,bool fil_calcul);
94
95 // execution de l'ordre
96 // tab_mail : donne les numéros de maillage concerné
97 // incre : numéro d'incrément qui en cours
98 // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
99 // animation : indique si l'on est en animation ou pas
100 // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
101 void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
unseul_incre,LesReferences*
102 ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
103 ,Resultats*,UtilLecture & entreePrinc,EnumTypeIncre type_incre,int incre
104 ,bool animation,const map < string, const double * , std::less <string> >&
listeVarGlob
105 ,const List_io < TypeQuelconque >& listeVecGlob);
106
107 // choix de l'ordre, cet méthode peut entraîner la demande d'informations
108 // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
109 void ChoixOrdre();
110
111 // --- méthodes particulière ----
112 // initialisation d'une liaison avec une instance de classe choix_grandeurs_maple
113 void Jonction_choix_grandeurs_maple( Choix_grandeurs_maple * choix) {choix_grandeurs_maple =
choix;};
114 // initialisation d'une liaison avec une instance de classe de choix des maillages
115 void Jonction_ChoixDesMaillages(const ChoixDesMaillages_vrml* choix_m) {choix_mail = choix_m;};
116 // ajout d'une courbe
117 void Ajout_courbe(List_io <DeuxDoubles>& courbe) { xyanim.push_back(courbe);};
118 // écriture des informations d'entête, renseigne sur les infos du fichier, ceci
119 // pour permettre l'exploitation par les programmes graphiques
120 // en entrée : list_mail = la liste des maillages à visualiser
121 void Entete_fichier_maple(const list<int>& list_mail,ostream & sort);
122
123 // lecture des paramètres de l'ordre dans un flux
124 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
125 // écriture des paramètres de l'ordre dans un flux
126 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
127
128 private :
129 // VARIABLES PROTEGEES :
130 double cycleInterval; // durée de l'animation
131 // bool loop; // indique si on boucle ou pas
132 // double startTime; // temps de démarrage en absolu depuis 1970
133 // double stopTime; // temps de fin en absolu depuis 1970,
134 // // si < a starttime on n'en tiend pas compte
135 // bool debut_auto; // pour le début automatique
136 // double inter_2pas; // interval entre deux dessins dans le cas d'un début automatique
137
138 const ChoixDesMaillages_vrml* choix_mail; // contient lorsqu'il est actif le choix des maillages
139 Choix_grandeurs_maple* choix_grandeurs_maple; // choix des grandeurs à visualiser

```

```

140 // dans le cas de l'animation : définition des listes de grandeurs s'y rattachant
141 List_io <List_io <DeuxDoubles> > xyanim; // x et y(x) : courbes à visualiser
142
143 // METHODES PROTEGEES :
144
145
146 };
147 /// @} // end of group
148
149 #endif
150

```

## 7.423 Choix\_grandeurs\_maple.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *
38 *      BUT:      choix des grandeurs à visualiser.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date ! auteur ! but
44 *      -----
45 *      ! ! !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date ! auteur ! but
50 *      -----
51 *
52 *      *****/
53 #ifndef CHOIX_Grandeurs_MAPLE_T
54 #define CHOIX_Grandeurs_MAPLE_T
55
56 #include "OrdreVisu.h"
57 #include "Ddl_enum_etendu.h"
58 #include "Bloc.h"
59 #include "BlocDdlLim.h"
60 #include "ChoixDesMaillages_vrml.h"
61 #include "TypeQuelconque.h"
62 #include "Basiques.h"
63
64 class Animation_maple;
65
66 /// @addtogroup Les_sorties_au_format_maple
67 /// @{
68 ///
69
70

```

```

71 class Choix_grandeurs_maple : public OrdreVisu
72 {
73 public :
74     // CONSTRUCTEURS :
75     // par défaut
76     Choix_grandeurs_maple () ;
77
78     // constructeur de copie
79     Choix_grandeurs_maple (const Choix_grandeurs_maple& algo);
80
81     // DESTRUCTEUR :
82     ~Choix_grandeurs_maple () ;
83
84     // METHODES PUBLIQUES :
85     // initialisation : permet d'initialiser les différents paramètres de l'ordre
86     // lors d'un premier passage des différents incréments
87     // en virtuelle, a priori est défini si nécessaire dans les classes dérivées
88     // incre : numéro d'incrément qui en cours
89     void Initialisation(ParaGlob * ,LesMaillages * ,LesReferences*
90         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
91         ,Resultats*,EnumTypeIncre type_incre,int incre
92         ,const map < string, const double * , std::less <string> >& listeVarGlob
93         ,const List_io < TypeQuelconque >& listeVecGlob
94         ,bool fil_calcul) ;
95     // execution de l'ordre
96     // tab_mail : donne les numéros de maillage concerné
97     // incre : numéro d'incrément qui en cours
98     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
99     // animation : indique si l'on est en animation ou pas
100    // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
101    void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
102        unseul_incre,LesReferences*
103        ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
104        ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int
105        incre
106        ,bool animation,const map < string, const double * , std::less <string> >&
107        listeVarGlob
108        ,const List_io < TypeQuelconque >& listeVecGlob);
109    // choix de l'ordre, cet méthode peut entraîner la demande d'informations
110    // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
111    void ChoixOrdre();
112
113    // initialisation de la liste des différentes grandeurs possibles à visualiser
114    void Init_liste_grandeurs(LesMaillages * lesMail,LesCondLim* lesCondLim
115        ,LesContacts* lesContacts,bool fil_calcul);
116
117    // écriture des informations d'entête, renseigne sur les infos du fichier, ceci
118    // pour permettre l'exploitation par le programme en maple
119    // en entrée : list_mail = la liste des maillages à visualiser
120    void Entete_fichier_maple(const list<int>& list_mail,ostream & sort);
121
122    // demande de choix d'uniquement 2 grandeurs à visualiser: aux noeuds ou (exclusif) aux éléments
123    // retourne un booléen indiquant si l'opération est un succes ou pas
124    // num_mail : indique le maillage sur lequel on récupère l'info
125    bool Choix_deux_grandeurs(int num_mail);
126    // initialisation d'une liaison avec une instance de classe Animation_maple
127    void Jonction_Animation_maple( Animation_maple * choix) {animation_maple = choix;};
128
129    // initialisation d'une liaison avec une instance de classe de choix des maillages
130    void Jonction_ChoixDesMaillages(const ChoixDesMaillages_vrml* choix_m) {choix_mail = choix_m;};
131
132    // lecture des paramètres de l'ordre dans un flux
133    void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
134    // écriture des paramètres de l'ordre dans un flux
135    void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
136
137 private :
138     // VARIABLES PROTEGEES :
139     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddl; // ddl principaux aux noeuds possibles à
140     visualiser
141     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddl_retenu; // ddl principaux aux noeuds à
142     visualiser
143     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddl_retenu_pourSM; // idem pour somme, moy etc.
144     sur une ref
145     // il s'agit ici des ddl principaux directement gérés par le noeuds
146     int type_sortie_ddl_retendue; // indique un type de sortie parmi les types suivants
147     // =0 sortie des grandeurs à t
148     // =1 sortie des grandeurs à t-0
149     // =2 sortie des grandeurs à 0 et à t
150     bool absolue; // par défaut on sort en absolue les tenseurs
151     // pour pouvoir correctement les visualiser, même pour les elem 2D
152
153     // -> noeuds
154     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddlEtendu; // ddl étendu aux noeuds possibles à
155     visualiser
156     Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddlEtendu_retenu; // ddl étendu aux noeuds à

```

```

visualiser
149 Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddlEtendu_retenu_pourSM; // idem pour somme,
    moy etc. sur une ref
150
151 Tableau <List_io < TypeQuelconque > > tabnoeud_TypeQuelconque; // TypeQuelconque aux noeuds
    possibles à visualiser
152 Tableau <List_io < TypeQuelconque > > tabnoeud_TypeQuelconque_retenu; // TypeQuelconque aux noeuds
    à visualiser
153 Tableau <List_io < TypeQuelconque > > tabnoeud_TypeQuelconque_retenu_pourSM; // idem pour somme, moy
    etc. sur une ref
154
155 // -> éléments
156 Tableau <List_io < Ddl_enum_etendu > > tabelement_type_ddl; // ddl aux elements possibles à
    visualiser
157 Tableau <List_io < Ddl_enum_etendu > > tabelement_type_ddl_retenu; // ddl aux elements à visualiser
158 Tableau <List_io < Ddl_enum_etendu > > tabelement_type_ddl_retenu_pourSM; // idem pour somme, moy
    etc. sur une ref
159
160 Tableau <List_io < TypeQuelconque > > tabelement_typeParti; // types particuliers aux elements
    possibles à visualiser
161 Tableau <List_io < TypeQuelconque > > tabelement_typeParti_retenu; // ddl aux elements à visualiser
162 Tableau <List_io < TypeQuelconque > > tabelement_typeParti_retenu_pourSM; // idem pour somme, moy
    etc. sur une ref
163
164 Tableau <List_io < TypeQuelconque > > tabelement_evoluee; // types evoluee aux elements possibles à
    visualiser
165 Tableau <List_io < TypeQuelconque > > tabelement_evoluee_retenu; // type evoluee aux elements à
    visualiser
166 Tableau <List_io < TypeQuelconque > > tabelement_evoluee_retenu_pourSM; // idem pour somme, moy etc.
    sur une ref
167
168 // -> face d'éléments
169 Tableau <List_io < TypeQuelconque > > tab_F_element_TypeQuelconque; // TypeQuelconque aux faces
    d'elements possibles à visualiser
170 Tableau <List_io < TypeQuelconque > > tab_F_element_TypeQuelconque_retenu; // TypeQuelconque aux
    faces d'elements à visualiser
171 Tableau <List_io < TypeQuelconque > > tab_F_element_TypeQuelconque_retenu_pourSM; // idem pour
    somme, moy etc. sur une ref
172
173 // -> arête d'éléments
174 Tableau <List_io < TypeQuelconque > > tab_A_element_TypeQuelconque; // TypeQuelconque aux arêtes
    d'elements possibles à visualiser
175 Tableau <List_io < TypeQuelconque > > tab_A_element_TypeQuelconque_retenu; // TypeQuelconque aux
    arête d'elements à visualiser
176 Tableau <List_io < TypeQuelconque > > tab_A_element_TypeQuelconque_retenu_pourSM; // idem pour
    somme, moy etc. sur une ref
177
178 List_io <string > list_grandeur_global; // les grandeurs globales que l'on peut visualiser
179 List_io <string > list_grand_global_retenu; // les grandeurs globales à visualiser
180
181 List_io <TypeQuelconque > listeVecGlobbal; // les vecteurs globaux que l'on peut visualiser
182 List_io <TypeQuelconque > listeVecGlobbal_retenu; // les vecteurs globaux à visualiser
183
184 // --- concerne les torseurs de réaction liées aux conditions limites ---
185 // le string contient le nom de la référence qui conduit au torseur de réaction, l'entier est un
    indice utilisé par CondLim
186 // pour optimiser
187 Tableau < List_io <String_et_entier > > tab_list_torseur_condLim; // les torseurs associées aux
    conditions limites, que l'on peut visualiser
188 Tableau < List_io <String_et_entier > > tab_list_torseur_condLim_retenu; // les torseurs associées
    aux conditions limite, à visualiser
189
190 LesMaillages * lesMail; // les maillages
191 LesReferences * lesRef; // les références
192
193 // --- pour les sorties concernant les sommes et moyennes etc. relativement à des références
194 Tableau <List_io <string > > tab_nomDeRef_SM; // pour somme, moy etc. sur une ref
195 Tableau <List_io <BlocScal > > tab_nomDeRef_E_SM; // list de nom de références d'éléments et de point
    d'intégration associé
196 Tableau <List_io <string > > tab_nomDeRef_ptinteg_SM; // list de nom de références de pt d'integ
    d'éléments
197
198 Tableau <List_io <BlocScal > > tab_nomDeRef_F_E_SM; // list de nom de références de faces d'élément
    et de point d'intégration associé
199 Tableau <List_io <string > > tab_nomDeRef_F_ptinteg_SM; // list de nom de références de pt d'integ de
    faces d'éléments
200
201 Tableau <List_io <BlocScal > > tab_nomDeRef_A_E_SM; // list de nom de références d'arêtes d'élément
    et de point d'intégration associé
202 Tableau <List_io <string > > tab_nomDeRef_A_ptinteg_SM; // list de nom de références de pt d'integ
    d'arêtes d'élément
203
204 // --- pour les sorties individuelles: référencement des ref, noeuds, éléments, npti
205
206 // -> les noeuds
207 Tableau <List_io <string > > nomDeRef; // list de nom de références de noeuds
208 Tableau <List_io <int > > tab_num_noeud; // liste des noeuds à visualiser

```

```

209
210 // -> les pti d'éléments
211 Tableau <List_io <int> > tab_num_element; // liste des éléments à visualiser
212 Tableau <List_io <int> > tab_num_integ; // liste des numéros de point d'integration à visualiser
    associé à tab_num_element
213 Tableau <List_io <BlocScal> > nomDeRef_E; // list de nom de références d'éléments et de point
    d'intégration associé
214 Tableau <List_io <string> > nomDeRef_ptinteg; // list de nom de références de pt d'integ d'éléments
215
216 // -> les pti de face d'éléments
217 Tableau <List_io <int> > tab_num_F_element; // liste des éléments à visualiser
218 Tableau <List_io <int> > tab_num_F; // liste des num de face ou arêtes d'éléments à visualiser
219 Tableau <List_io <int> > tab_num_F_integ; // liste des numéros de point d'integration à visualiser
    associé à tab_num_F_element
220 Tableau <List_io <BlocScal> > nomDeRef_F_E; // list de nom de références face d'éléments associés
221 Tableau <List_io <string> > nomDeRef_F_ptinteg; // list de nom de références de pt d'integ de face
    d'éléments
222
223 // -> les pti d'arête d'éléments
224 Tableau <List_io <int> > tab_num_A_element; // liste des éléments à visualiser
225 Tableau <List_io <int> > tab_num_A; // liste des num d'arêtes d'éléments à visualiser
226 Tableau <List_io <int> > tab_num_A_integ; // liste des numéros de point d'integration à visualiser
    associé à tab_num_A_element
227 Tableau <List_io <BlocScal> > nomDeRef_A_E; // list de nom de références d'arête d'éléments associé
228 Tableau <List_io <string> > nomDeRef_A_ptinteg; // list de nom de références de pt d'integ d'arête
    d'éléments
229
230
231 // info de stockage utiliser dans le cas d'une animation
232 Ddl_enum_etendu x1; Ddl_enum_etendu x2;
233 TypeQuelconque xx1; TypeQuelconque xx2;
234 int ddl_etpas_TQ_1, ddl_etpas_TQ_2; // indique si pour l'animation on utilise des ddl (1),
235 // ddl étendues de noeud (2), ddl étendues d'élément (3), ou un type quelconque (0)
236 int nb_ordre_1, nb_ordre_2; // numéros d'ordre éventuelle des 2 types quelconques
237 bool accroi_x1, accroi_x2; // indique si oui on non la visualisation concerne l'accroissement d'une
    grandeur entre t=0 et t
238 bool type_xi; // indique si oui ou non x1 et x2 sont des grandeurs aux noeuds (sinon c'est aux
    éléments)
239 Animation_maple* animation_maple; // pour la liaison avec l'animation
240 const ChoixDesMaillages_vrml* choix_mail; // contient lorsqu'il est actif le choix des maillages
241
242 // info pour le style de sortie
243 int style_de_sortie;
244
245 int der_numero_mail; // variable de travail : dernier numéro de maillage à visualiser
246
247 // METHODES PROTEGEES :
248 // entrée du choix de ddl principaux aux noeuds
249 void Entree_grandeur_principale_aux_noeuds(string& rep);
250 // entrée du choix de ddl étendus secondaires aux noeuds
251 void Entree_grandeur_etendue_secondaire_aux_noeuds(string& rep);
252 // entrée du choix de cas de grandeurs particulière aux noeuds
253 void Entree_grandeur_TypeQuelconque_secondaire_aux_noeuds(string& rep);
254 // entrée du choix d'une grandeur aux éléments
255 void Entree_grandeur_aux_elements(string& rep);
256 // entrée du choix d'une grandeur particulières (diff des ddl génériques) aux éléments
257 void Entree_grandeur_particuliere_aux_elements(string& rep);
258 // entrée du choix d'une grandeur tensorielle aux éléments
259 void Entree_grandeur_tensorielle_aux_elements(string& rep);
260
261 // entrée du choix d'une grandeur quelconque aux faces d'éléments
262 void Entree_grandeur_quelconque_aux_faces_element(string& rep);
263 // entrée du choix d'une grandeur quelconque aux arêtes d'éléments
264 void Entree_grandeur_quelconque_aux_arettes_element(string& rep);
265
266 // choix des éléments
267 void Choix_element_a_retenir(int n_mail, List_io < Ddl_enum_etendu >& li_ddl);
268 // choix des faces d'éléments
269 void Choix_faces_element_a_retenir(int n_mail, List_io < EnumTypeQuelconque >& li_enu_quelc);
270 // choix des arête d'éléments
271 void Choix_arettes_element_a_retenir(int n_mail, List_io < EnumTypeQuelconque >& li_enu_quelc);
272 // entrée du choix de grandeurs globales
273 void Entree_grandeur_gobal(string& rep);
274 // entrée du choix des torseurs de réaction
275 void Entree_torseurs_reaction(string& rep);
276 // entrée du choix pour les moyennes sommes etc. sur ref N
277 void Entree_somme_moy_N(string& rep);
278 // entrée du choix pour les moyennes sommes etc. sur ref E
279 void Entree_somme_moy_E(string& rep);
280 // entrée du choix pour les moyennes sommes etc. sur ref de face d'E
281 void Entree_somme_moy_F_E(string& rep);
282 // entrée du choix pour les moyennes sommes etc. sur ref d'arête d'E
283 void Entree_somme_moy_A_E(string& rep);
284 // choix des noeud
285 void Choix_noeud_a_retenir(int n_mail, List_io < Ddl_enum_etendu >& li_ddl);
286 // dans le cas d'une animation, il n'y a pas de sortie directe mais la création de listes qui sont
    transmises

```

```

287 // à animation_maple
288 void Construction_liste_pour_animation(int numMail ,LesMaillages * lesMail,Charge* charge);
289 // entrée du choix de l'utilisation sur le style de sortie
290 void Style_de_sortie(string& rep);
291 // vérification des listes de pt d'integ et d'éléments relativement au grandeur à sortir
292 void VerifListes();
293 // utilitaire de sortie d'entête pour les noeuds:
294 void Sortie_entete_noeud_unitaire(list<int>::const_iterator imail,ostream & sort, int& num_col);
295 // utilitaire de sortie d'entête pour les éléments:
296 void Sortie_entete_element_unitaire(list<int>::const_iterator imail,ostream & sort, int& num_col,
bool & temps_deja_affiche);
297 // utilitaire de sortie d'entête pour les faces d'élément:
298 void Sortie_entete_face_element_unitaire(list<int>::const_iterator imail,ostream & sort, int&
num_col, bool & temps_deja_affiche);
299 // utilitaire de sortie d'entête pour les arêtes d'élément:
300 void Sortie_entete_arete_element_unitaire(list<int>::const_iterator imail,ostream & sort, int&
num_col, bool & temps_deja_affiche);
301
302 // sortie pour les moyennes sommes etc. sur ref N
303 void Sortie_somme_moy_N(ostream &sort,Charge* charge,bool unseul_incre);
304 // sortie pour les moyennes sommes etc. sur ref E
305 void Sortie_somme_moy_E(ostream &sort,Charge* charge,bool unseul_incre);
306
307 // sortie pour les moyennes sommes etc. sur ref de face E
308 void Sortie_somme_moy_face_E(ostream &sort,Charge* charge,bool unseul_incre);
309 // sortie pour les moyennes sommes etc. sur ref d'arête E
310 void Sortie_somme_moy_arete_E(ostream &sort,Charge* charge,bool unseul_incre);
311
312 // sortie de l'entete pour les moyennes sommes etc. sur ref N
313 void Sortie_entete_somme_moy_N(ostream &sort,int & num_col);
314 // sortie de l'entete pour les moyennes sommes etc. sur ref E
315 void Sortie_entete_somme_moy_E(ostream &sort,int & num_col);
316
317 // sortie de l'entete pour les moyennes sommes etc. sur ref de face E
318 void Sortie_entete_somme_moy_face_E(ostream &sort,int & num_col);
319 // sortie de l'entete pour les moyennes sommes etc. sur ref d'arête E
320 void Sortie_entete_somme_moy_arete_E(ostream &sort,int & num_col);
321
322
323 };
324 /// @} // end of group
325
326 #include "Animation_maple.h"
327
328 #endif

```

## 7.424 Deformees\_maple.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 * *****/

```

```

38 *      BUT: paramétrisatio de la déformée pour le format maple.      *
39 *                                                                 $   *
40 *      ***** *
41 *      VERIFICATION: *
42 * * *
43 *      ! date ! auteur ! but ! *
44 *      ----- *
45 *      ! ! ! ! *
46 *      $ *
47 *      ***** *
48 *      MODIFICATIONS: *
49 *      ! date ! auteur ! but ! *
50 *      ----- *
51 *                                                                 $   *
52 *****/
53 #ifndef DEFORMEES_MAPLE_T
54 #define DEFORMEES_MAPLE_T
55
56 #include "Deformees_vrml.h"
57
58 /// @addtogroup Les_sorties_au_format_maple
59 /// @{
60 ///
61
62
63 class Deformees_maple : public Deformees_vrml
64 {
65     public :
66         // CONSTRUCTEURS :
67         // par défaut
68         Deformees_maple () ;
69
70         // constructeur de copie
71         Deformees_maple (const Deformees_maple& algo);
72
73         // DESTRUCTEUR :
74         ~Deformees_maple () ;
75
76         // METHODES PUBLIQUES :
77         // execution de l'ordre
78         // tab_mail : donne les numéros de maillage concerné
79         // incre : numéro d'incrément qui en cours
80         // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
81         // animation : indique si l'on est en animation ou pas
82         void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool unseul_incre,
83             LesReferences*
84             ,LesLoisDeComp* ,DiversStockage* ,Charge* ,LesCondLim* ,LesContacts*
85             ,Resultats* ,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
86             ,bool animation,const map < string, const double * , std::less <string> >&
87             listeVarGlob
88             ,const List_io < TypeQuelconque >& listeVecGlob);
89         // choix de l'ordre, cet méthode peut entraîner la demande d'informations
90         // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
91         void ChoixOrdre();
92
93     private :
94         // VARIABLES PROTEGEES :
95         double amplification; // amplification de la déformée
96
97         // METHODES PROTEGEES :
98
99 };
100 /// @} // end of group
101 #endif

```

## 7.425 Fin\_maple.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by

```



```

17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT:  Fin de la visualisation en format maple.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !       but
45 *   -----
46 *   !       !       !
47 *   *****
48 *
49 *   MODIFICATIONS:
50 *
51 *   ! date !   auteur !       but
52 *   -----
53 *   $
54 */
55
56 #ifndef FIN_MAPLE_T
57 #define FIN_MAPLE_T
58
59 #include "OrdreVisu.h"
60
61 /// @addtogroup Les_sorties_au_format_maple
62 /// @{
63
64 class Fin_maple : public OrdreVisu
65 {
66 public :
67     // CONSTRUCTEURS :
68     // par default
69     Fin_maple () ;
70
71     // constructeur de copie
72     Fin_maple (const Fin_maple& algo);
73
74     // DESTRUCTEUR :
75     ~Fin_maple () ;
76
77     // METHODES PUBLIQUES :
78     // execution de l'ordre
79     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
80                 ,LesLoisDeComp* ,DiversStockage*
81                 ,Charge*,LesCondLim*
82                 ,LesContacts*,Resultats*,UtilLecture & ,OrdreVisu::EnumTypeIncre ,int
83                 ,bool,const map < string, const double * , std::less <string> >&
84                 ,const List_io < TypeQuelconque >& listeVecGlob)
85     {} ;
86
87     // lecture des paramètres de l'ordre dans un flux
88     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
89     // écriture des paramètres de l'ordre dans un flux
90     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
91
92 private :
93     // VARIABLES PROTEGEES :
94
95     // METHODES PROTEGEES :
96
97 };
98 /// @} // end of group
99
100 #endif

```

## 7.426 Visuali\_maple.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *   ****
38 *   BUT: Arrêt des question Mise en route de la visualisation
39 *        au format maple.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   -----
47 *   !           !           !
48 *   *****
49 *
50 *   MODIFICATIONS:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !
55 *   *****/
54 #ifndef VISUALI_MAPLE_T
55 #define VISUALI_MAPLE_T
56
57 #include "OrdreVisu.h"
58
59 /** @defgroup Les_sorties_au_format_maple
60 *
61 *   BUT: groupe concernant le postraitement au format maple: format ascii auto-documenté, type
62 *        tableau bidimensionnel
63 *
64 *   \author Gérard Rio
65 *   \version 1.0
66 *   \date 23/01/97
67 *   \brief groupe concernant le postraitement au format maple: format ascii auto-documenté, type
68 *          tableau bidimensionnel
69 */
70
71 /// @addtogroup Les_sorties_au_format_maple
72 /// @{
73 ///
74
75
76 class Visuali_maple : public OrdreVisu
77 {
78 public :
79     // CONSTRUCTEURS :
80     // par défaut
81     Visuali_maple () ;
82

```

```

83     // constructeur de copie
84     Visuali_maple (const Visuali_maple& algo);
85
86     // DESTRUCTEUR :
87     ~Visuali_maple () ;
88
89     // METHODES PUBLIQUES :
90     // execution de l'ordre
91     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
92                 ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*
93                 ,LesContacts*,Resultats*,UtilLecture & ,OrdreVisu::EnumTypeIncre ,int
94                 ,bool ,const map < string, const double * , std::less <string> >&
95                 ,const List_io < TypeQuelconque >& listeVecGlob)
96         {};
97
98     // lecture des paramètres de l'ordre dans un flux
99     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
100    // écriture des paramètres de l'ordre dans un flux
101    void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
102
103    private :
104        // VARIABLES PROTEGEES :
105
106        // METHODES PROTEGEES :
107
108    };
109    /// @} // end of group
110
111 #endif

```

## 7.427 Visualisation\_maple.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 * *****/
38 *   BUT: Gérer et définir les fichiers de visualisation par points
39 *   au format lisible par maple.
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   !           !           !
47 *
48 *   *****
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   !           !           !
52 *

```

```

53  *****/
54 #ifndef VISUALISATION_MAPLE_H
55 #define VISUALISATION_MAPLE_H
56
57 #include "UtilLecture.h"
58 // #include "bool.h"
59 #include "MotCle.h"
60 #include "string.h"
61 #include "Tableau_T.h"
62 #include "LesMaillages.h"
63 #include "LesReferences.h"
64 #include "LesCondLim.h"
65 #include <list>
66 #include "LesContacts.h"
67 #include "LesValVecPropres.h"
68 #include "LesLoisDeComp.h"
69 #include "DiversStockage.h"
70 #include "Charge.h"
71 #include "LesContacts.h"
72 #include "Resultats.h"
73 #include "OrdreVisu.h"
74
75 /// @addtogroup Les_sorties_au_format_maple
76 /// @{
77 ///
78
79
80 class Visualisation_maple
81 {
82 public :
83 // CONSTRUCTEURS :
84 // le constructeur par défaut ne doit pas être utilisé
85 // il y a dans ce cas un message d'erreur et arrêt
86 Visualisation_maple ();
87 // le bon constructeur
88 Visualisation_maple (UtilLecture* ent);
89
90 // DESTRUCTEUR :
91 ~Visualisation_maple ();
92
93 // METHODES PUBLIQUES :
94 // affichage des différentes possibilités (ordres possibles)
95 // ramène un numéro qui renseigne le programme appelant
96 // == -1 : signifie que l'on veut stopper la visualisation
97 // == 0 : signifie que l'on demande la visualisation effective
98 int OrdresPossible();
99
100 // information de l'instance de la liste d'incrément disponible pour la visualisation
101 void List_increment_disponible(list <int> & list_incr) ;
102 // indique le choix de la liste d'incrément à utiliser pour l'initialisation
103 // des fonctions d'initialisation
104 const list<int> & List_balaie_init();
105 // impose une liste d'incrément à utiliser
106 void List_balaie_init(const list<int> & list_init);
107 // indique le choix d'une liste d'incrément à visualiser
108 const list<int> & List_balaie() {return *list_balaie;};
109 // information de l'instance du nombre de maillages disponibles pour la visualisation
110 // par défaut tous les maillages seront visualisés, ceci est descidé aussi ici
111 void List_maillage_disponible(int nombre_maillage_dispo) ;
112
113 // initialisation des ordres disponibles
114 // par exemple pour les isovaleurs on définit la liste des isovaleurs disponibles et les extrémas
115 void Initialisation(ParaGlob * paraGlob,LesMaillages * lesMaillages,LesReferences* lesReferences
116 ,LesLoisDeComp* lesLoisDeComp,DiversStockage* diversStockage
117 ,Charge* charge,LesCondLim* lesCondLim,LesContacts* lesContacts
118 ,Resultats* resultats,OrdreVisu::EnumTypeIncr type_incr,int incr
119 ,const map < string, const double * , std::less <string> >& listeVarGlob
120 ,const List_io < TypeQuelconque >& listeVecGlob
121 ,bool fil_calcul);
122
123 // méthode principale pour activer la visualisation
124 // sort : le flux de visualisation
125 void Visu(ParaGlob * paraGlob,LesMaillages * lesMaillages,LesReferences* lesReferences
126 ,LesLoisDeComp* lesLoisDeComp,DiversStockage* diversStockage
127 ,Charge* charge,LesCondLim* lesCondLim,LesContacts* lesContacts
128 ,Resultats* resultats,OrdreVisu::EnumTypeIncr type_incr,int incr
129 ,const map < string, const double * , std::less <string> >& listeVarGlob
130 ,const List_io < TypeQuelconque >& listeVecGlob);
131
132 // == définition des paramètres de visualisation
133 // titre, navigation, éclairage
134 // et initialisation des paramètres de la classe
135 void Contexte_debut_visualisation();
136 // (points de vue) et enchainement si nécessaire
137 void Contexte_fin_visualisation();
138
139 // lecture des paramètres de l'ordre dans un flux

```

```

140 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
141 // écriture des paramètres de l'ordre dans un flux
142 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
143 // demande si la visualisation vrml est validé ou pas
144 bool Visu_maple_valide() {return activ_sort_maple;};
145 // inactive la visualisation maple
146 void Inactiv_Visu_maple() {activ_sort_maple=false;};
147
148 private :
149 // VARIABLES PROTEGEES :
150 UtilLecture* entreePrinc; // gestion des entrees/sorties
151 list <OrdreVisu> ordre_possible; // l'ensemble des ordres possibles
152 // la sauvegarde de valeurs propres et vecteurs propres éventuelles
153 LesValVecPropres lesValVecPropres;
154 OrdreVisu* fin_o; // ordre de fin de la visualisation
155 OrdreVisu* visuali; // ordre de visualiser
156 OrdreVisu* choix_inc; // ordre de choix des incréments
157 OrdreVisu* anim; // ordre d'animation
158 list <int> list_incre; // liste des incréments possibles
159 const list <int>* list_balaie; // liste des incréments à visualiser
160 int nb_maillage_dispo; // le nombre de maillage disponible
161 OrdreVisu* choix_mail; // ordre de choix des maillages
162 OrdreVisu* choix_grandeurs; // choix des grandeurs à visualiser
163 // indication si l'on est en animation ou pas
164 bool animation;
165 // indication si la visualisation de type maple est active ou pas, par défaut = faux
166 bool activ_sort_maple;
167
168 // METHODES PROTEGEES :
169 // test si la réponse fait partie des ordres possibles
170 // si l'on trouve un ordre ok on ramène un pointeur sur l'ordre
171 // sinon on ramène un pointeur null
172 OrdreVisu* Existe(string& reponse);
173 // affichage des options possibles
174 void Affiche_options();
175
176
177 };
178 /// @} // end of group
179
180 #endif

```

## 7.428 OrdreVisu.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 * DATE: 23/01/97 *
32 * * $ *
33 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
34 * * $ *
35 * PROJET: Herezh++ *
36 * * $ *
37 *****/
38 * BUT: Maillon de base d'un ordre de visualisation interactive. *
39 * * $ *
40 * ***** *

```

```

41 * VERIFICATION: *
42 * * *
43 * ! date ! auteur ! but ! *
44 * ----- *
45 * ! ! ! ! *
46 * $ *
47 * ***** *
48 * MODIFICATIONS: *
49 * ! date ! auteur ! but ! *
50 * ----- *
51 * $ *
52 * *****/
53 #ifndef ORDREVISU_T
54 #define ORDREVISU_T
55
56 #include <iostream>
57 #include "string"
58
59 #include "ParaGlob.h"
60 #include "LesMaillages.h"
61 #include "LesReferences.h"
62 #include "LesLoisDeComp.h"
63 #include "DiversStockage.h"
64 #include "Charge.h"
65 #include "LesCondLim.h"
66 #include "LesContacts.h"
67 #include "Resultats.h"
68
69 /// @addtogroup Les_sorties_generiques
70 /// @{
71 ///
72
73
74 class OrdreVisu
75 {
76 public :
77     enum EnumTypeIncre {INCRE_0=0,PREMIER_INCRE,INCRE_COURANT,DERNIER_INCRE};
78     // Retourne le nom d'un type d'increment a partir de son identificateur de
79     // type enumere id_TypeIncre correspondant
80     string Nom_TypeIncre (const OrdreVisu::EnumTypeIncre id_TypeIncre);
81     // Retourne l'identificateur de type enumere associe au nom du type d'increment
82     EnumTypeIncre Id_nom_TypeIncre (const string nom_TypeIncre);
83
84     class Deuxentiers {public: int mail;int incr;}; // un conteneur de travail
85
86     // CONSTRUCTEURS :
87     // par default
88     OrdreVisu () ;
89     // permettant la définition de l'affichage
90     // l'ordre doit être en minuscule
91     OrdreVisu(const string& comment_som, const string& explication, const string& ordre);
92
93     // constructeur de copie
94     OrdreVisu (const OrdreVisu& ord);
95
96     // DESTRUCTEUR :
97     virtual ~OrdreVisu () ;
98
99     // METHODES PUBLIQUES :
100    // affichage de l'ordre
101    void Affiche_ordre();
102    // affichage de l'explication détaillée
103    void Explication_detaillée();
104
105    // initialisation : permet d'initialiser les différents paramètres de l'ordre
106    // lors d'un premier passage des différents incréments
107    // en virtuelle, a priori est défini si nécessaire dans les classes dérivées
108    // incre : numéro d'incrément qui en cours
109    virtual void Initialisation(ParaGlob * ,LesMaillages * ,LesReferences*
110        ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
111        ,Resultats*,EnumTypeIncre type_incre,int incre
112        ,const map < string, const double * , std::less <string> >& listeVarGlob
113        ,const List_io < TypeQuelconque >& listeVecGlob
114        ,bool fil_calcul) ;
115
116    // execution de l'ordre
117    // en virtuelle, a priori est défini dans les classes dérivées
118    // tab_mail : donne les numéros de maillage concerné
119    // incre : numéro d'incrément qui en cours
120    // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
121    // animation : indique si l'on est en animation ou pas
122    // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
123    virtual void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool unseul_incre,
124        LesReferences*
125        ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
126        ,Resultats*,Utillecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int
127        incre

```

```

126         ,bool animation,const map < string, const double * , std::less <string> >&
listeVarGlob
127         ,const List_io < TypeQuelconque >& listeVecGlob);
128 // retour vrai si l'ordre proposée est celui de l'instance
129 bool OrdreVrai(const string& ord);
130 // choix de l'ordre, cet méthode peut entraîner la demande d'informations
131 // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
132 virtual void ChoixOrdre();
133 // renseigne si l'ordre est activé ou pas
134 bool Actif() const {return actif;};
135 // inactive l'ordre
136 void Inactive_ordre() {actif = false;};
137
138 // lecture des paramètres de l'ordre dans un flux
139 virtual void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
140 // écriture des paramètres de l'ordre dans un flux
141 virtual void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
142
143
144 protected :
145 // VARIABLES PROTEGEES :
146 string commentaire_sommaire;
147 string explication_detail;
148 string ordre; // doit être en minuscule
149 bool actif; // ordre actif ou pas
150
151
152 // *** les déclarations suivantes pourraient être modifiée en créant une liaison entre
153 // *** animation et déformation, de la même manière qu'avec les isovaleurs !!!
154 // *** dans le cas de l'animation : définition des listes de positions et de nom si rattachant
155 static list <string> list_nom_coordinate;
156 static list < list <Coordonnee> > list_coordinate;
157 static string nom_base_couleur; // utilisé dans déformée
158
159
160 // METHODES PROTEGEES :
161 // ordonne une liste, supprime les doublons et finalement l'affiche à l'écran
162 void Propre_liste(list<int>& lis);
163 // mise a zéro des lists de coordonnées
164 void Mise_zero_coordo();
165 // -- les deux méthodes suivantes sont utilisées par les ordres dérivées
166 // lecture des paramètres de l'ordre général dans un flux
167 void Lect_para_OrdreVisu_general(UtilLecture & entreePrinc);
168 // écriture des paramètres de l'ordre dans un flux
169 void Ecrit_para_OrdreVisu_general(UtilLecture & entreePrinc);
170 // changement des strings (cas d'un changement de langue par exemple)
171 void changeLibelle(const string& comment_som, const string& explication, const string& ord)
172     {commentaire_sommaire=comment_som;explication_detail=explication;ordre = ord;};
173
174
175 };
176 /// @} // end of group
177
178 #endif

```

## 7.429 Resultats.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.

```

```

29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Gestion de la sortie des differents resultats.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date !   auteur !       but
44 *      -----
45 *      !       !       !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date !   auteur !       but
50 *      -----
51 *
52 *****/
53 #ifndef RESULTATS_H
54 #define RESULTATS_H
55
56 #include "UtilLecture.h"
57 // #include "bool.h"
58 #include "MotCle.h"
59 #include "string.h"
60 #include "Tableau_T.h"
61 #include "LesMaillages.h"
62 #include "LesReferences.h"
63 #include "LesCondLim.h"
64 #include <list>
65 #include "LesContacts.h"
66 #include "LesValVecPropres.h"
67
68 /** @defgroup Les_sorties_generiques
69 *
70 *      BUT:      groupe concernant le postraitement de manière générique (classe virtuelle et sorties avec un
71 *                  format spécifique d'Herezh )
72 *
73 * \author      Gérard Rio
74 * \version    1.0
75 * \date      23/01/97
76 * \brief     groupe concernant le postraitement de manière générique
77 *
78 */
79
80 /// @addtogroup Les_sorties_generiques
81 /// @{
82 ///
83
84
85 class Resultats
86 {
87 public :
88     // CONSTRUCTEURS :
89     // entreePrinc gere les entrees sorties
90     Resultats(UtilLecture* entreePrinc);
91     // DESTRUCTEUR :
92     ~Resultats();
93
94     // METHODES PUBLIQUES :
95     // lecture des parametres de gestions de la sortie des resultats
96     void Lecture(LesReferences* lesRef);
97     // affichage des parametres de gestions de la sortie des resultats
98     void Affiche() const ;
99     // retourne la variable copie
100    bool Copie() { return copie;};
101
102    // def interactive des parametres de gestion de la sortie des resultats
103    void Info_commande_Resultats();
104
105    // sortie des resultats suivant les infos stockee durant la lecture du fichier d'entree
106    void SortieInfo(ParaGlob * paraGlob, LesMaillages* lesMail, LesReferences* lesRef,
107                  LesCondLim * lesCondLim, LesContacts* lescontacts);
108    // ramène le container relatif aux valeurs et vecteur propres
109    inline LesValVecPropres & Veapropre() { return lesValVecPropres;};
110
111    //----- lecture écriture de restart -----
112    // cas donne le niveau de la récupération
113    // = 1 : on récupère tout

```



```

114     // = 2 : on récupère uniquement les données variables (supposées comme telles)
115     void Lect_result_base_info(ifstream& ent,const int cas);
116     // cas donne le niveau de sauvegarde
117     // = 1 : on sauvegarde tout
118     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
119     void Ecri_result_base_info(ofstream& sort,const int cas);
120
121
122 private :
123     // VARIABLES PROTEGEES :
124     UtilLecture* entreePrinc; // gestion des entrees/sorties
125
126     list < BlocDdlLim < BlocGen> > tabRegion; // tableau des regions ou l'on veut sortir
127         // des resultats des variables duales aux pt d'integ
128
129     // parametres de gestion
130     bool copie; // copie de la lecture des donnees ?
131     bool pas_de_sortie; // indique éventuellement que l'on ne veut pas de sortie
132
133     // la sauvegarde de valeurs propres et vecteurs propres éventuelles
134     LesValVecPropres lesValVecPropres;
135
136     // METHODES PROTEGEES :
137     // lecture concernant les sorties aux points d'integration
138     void ReadPtInteg(LesReferences* lesRef,Tableau<string>& TsousMot);
139     // Lectures concernant les sorties aux noeuds
140     void ReadNoeuds(LesReferences* lesRef);
141     // lectures concernant les reactions
142     void ReadReac(LesReferences* lesRef);
143     // lectures concernant les deformeés
144     void ReadDeformee(LesReferences* );
145
146     // sortie des grandeurs duales aux points d'integration
147     void PointsDintegration(LesMaillages* lesMail,LesReferences* lesRef);
148     // sortie des ddls et grandeurs aux noeuds
149     void SortieNoeuds(LesMaillages* lesMail,LesReferences* lesRef);
150     // sortie des reactions
151     void SortieReactions(LesMaillages* lesMail,LesReferences* lesRef,LesCondLim * lesCondLim);
152     // sortie du contact
153     void SortieContacts(LesContacts* lescontacts);
154     // sortie pour le flambement
155     void SortieFlamb();
156     // sortie des deformeés
157     void SortieDeformee(LesMaillages* lesMail,LesReferences* lesRef,ParaGlob * paraGlob);
158     // dans le cas ou l'on a une remontée aux contraintes, sortie d'isovaleurs
159     // de contraintes.
160     void SortieNoeudContaintes
161         (LesMaillages* lesMail,LesReferences* lesRef,ParaGlob * paraGlob);
162     // dans le cas ou l'on a une remontée aux erreurs, sortie d'isovaleurs
163     // d'erreur aux noeuds.
164     void SortieNoeudErreur
165         (LesMaillages* lesMail,LesReferences* lesRef,ParaGlob * paraGlob);
166     // dans le cas ou l'on a une remontée aux erreurs, sortie des valeurs d'erreurs
167     // aux éléments.
168     void SortieElemErreur
169         (LesMaillages* lesMail,LesReferences* lesRef,ParaGlob * paraGlob);
170     // sortie des contraintes aux éléments en texte peu lisible mais condensé
171     void SortieElemenContrainte
172         (LesMaillages* lesMail,ParaGlob * paraGlob);
173     // sortie des frontières
174     void SortieFrontieres (LesMaillages* lesMail);
175
176
177     // sortie de la deformee du maillage pour l'article
178     void DeformeeArticle(LesMaillages* lesMail);
179
180     // pour un cube, creation de noeuds intermediaires et de faces pour la visualisation
181     void ZoomElem
182         (Tableau <Noeud*>& TabNo, int n, Tableau <Coordonnee>& TabC, Tableau < Tableau <int> > & Tabnn);
183     // dessin d'un cube initial
184     void InitialC
185         (Tableau <Noeud*>& TabNo, int n, Tableau <Coordonnee>& TabC, Tableau < Tableau <int> > & Tabnn);
186
187 };
188 /// @} // end of group
189
190 #endif

```

## 7.430 Visualisation.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.

```

```

6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           23/01/97
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *
38 *   BUT:  Gérer et définir les fichiers de visualisation vmrl.
39 *   Sert aussi à gérer toutes les fonctions communes à toutes les
40 *   différents types de visualisation.
41 *
42 *   *****
43 *
44 *   VERIFICATION:
45 *
46 *   ! date ! auteur ! but
47 *   -----
48 *   ! ! !
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date ! auteur ! but
52 *   -----
53 *   $
54 *****/
55 #ifndef VISUALISATION_H
56 #define VISUALISATION_H
57
58 #include "UtilLecture.h"
59 // #include "bool.h"
60 #include "MotCle.h"
61 #include "string.h"
62 #include "Tableau_T.h"
63 #include "LesMaillages.h"
64 #include "LesReferences.h"
65 #include "LesCondLim.h"
66 #include <list>
67 #include "LesContacts.h"
68 #include "LesValVecPropres.h"
69 #include "LesLoisDeComp.h"
70 #include "DiversStockage.h"
71 #include "Charge.h"
72 #include "LesContacts.h"
73 #include "Resultats.h"
74 #include "OrdreVisu.h"
75
76 /// @addtogroup Les_sorties_generiques
77 /// @{
78 ///
79
80
81 class Visualisation
82 {
83 public :
84     // CONSTRUCTEURS :
85     // le constructeur par défaut ne doit pas être utilisé
86     // il y a dans ce cas un message d'erreur et arrêt
87     Visualisation ();
88     // le bon constructeur
89     Visualisation (UtilLecture* ent);
90
91     // DESTRUCTEUR :

```

```

92 ~Visualisation ();
93
94 // METHODES PUBLIQUES :
95 // affichage des différentes possibilités (ordres possibles)
96 // ramène un numéro qui renseigne le programme appelant
97 // ==-1 : signifie que l'on veut stopper la visualisation
98 // = 0 : signifie que l'on demande la visualisation effective
99 int OrdresPossible();
100
101 // information de l'instance de la liste d'incrément disponible pour la visualisation
102 void List_increment_disponible(list <int> & list_incr) ;
103 // indique le choix de la liste d'incrément à utiliser pour l'initialisation
104 // des fonctions d'initialisation
105 // si interactif = -1 : initialisation à 0 et au dernier incrément par défaut
106 // si interactif = 0 : le choix de la liste s'effectue via une lecture dans le fichier
107 // de commande
108 // si interactif = 1 : initialisation interactive
109 const list<int> & List_balaie_init(int interactif = 1);
110 // impose une liste d'incrément à utiliser
111 void List_balaie_init(const list<int> & list_init);
112 // indique le choix de la liste d'incrément à visualiser
113 const list<int> & List_balaie() {return *list_balaie;};
114 // information de l'instance du nombre de maillages disponibles pour la visualisation
115 // par défaut tous les maillages seront visualisés, ceci est descidé aussi ici
116 void List_maillage_disponible(int nombre_maillage_dispo) ;
117 // initialisation des ordres disponibles
118 // par exemple pour les isovaleurs on définit la liste des isovaleurs disponibles et les extrémas
119 void Initialisation(ParaGlob * paraGlob,LesMaillages * lesMaillages,LesReferences* lesReferences
120 ,LesLoisDeComp* lesLoisDeComp,DiversStockage* diversStockage
121 ,Charge* charge,LesCondLim* lesCondLim,LesContacts* lesContacts
122 ,Resultats* resultats,OrdreVisu::EnumTypeIncre type_incre,int incre
123 ,const map < string, const double * , std::less <string> >& listeVarGlob
124 ,const List_io < TypeQuelconque >& listeVecGlob
125 ,bool fil_calcul);
126
127 // méthode principale pour activer la visualisation
128 // sort : le flux de visualisation
129 void Visu(ParaGlob * paraGlob,LesMaillages * lesMaillages,LesReferences* lesReferences
130 ,LesLoisDeComp* lesLoisDeComp,DiversStockage* diversStockage
131 ,Charge* charge,LesCondLim* lesCondLim,LesContacts* lesContacts
132 ,Resultats* resultats,OrdreVisu::EnumTypeIncre type_incre,int incre
133 ,const map < string, const double * , std::less <string> >& listeVarGlob
134 ,const List_io < TypeQuelconque >& listeVecGlob
135 );
136
137 // == définition des paramètres de visualisation
138 // titre, navigation, éclairage
139 // et initialisation des paramètres de la classe
140 void Contexte_debut_visualisation();
141 // (points de vue) et enchainement si nécessaire
142 void Contexte_fin_visualisation();
143
144 // lecture des paramètres de l'ordre dans un flux
145 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
146 // écriture des paramètres de l'ordre dans un flux
147 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
148 // demande si la visualisation vrml est validé ou pas
149 bool Visu_vrml_valide() {return activ_sort_vrml;};
150 // inactive la visualisation vrml
151 void Inactiv_Visu_vrml() {activ_sort_vrml=false;};
152 // écriture de l'entête du fichier de commande de visualisation
153 void EcritDebut_fichier_OrdreVisu(UtilLecture & entreePrinc);
154 // écriture de la fin du fichier de commande de visualisation
155 void EcritFin_fichier_OrdreVisu(UtilLecture & entreePrinc);
156
157 private :
158 // VARIABLES PROTEGEES :
159 UtilLecture* entreePrinc; // gestion des entrees/sorties
160 list <OrdreVisu> ordre_possible; // l'ensemble des ordres possibles
161 // la sauvegarde de valeurs propres et vecteurs propres éventuelles
162 LesValVecPropres lesValVecPropres;
163 OrdreVisu* fin_o; // ordre de fin de la visualisation
164 OrdreVisu* visuali; // ordre de visualiser
165 OrdreVisu* choix_inc; // ordre de choix des incréments
166 OrdreVisu* ptdeformee; // déformée
167 OrdreVisu* anim; // ordre d'animation
168 list <int> list_incre; // liste des incréments possibles
169 const list <int>* list_balaie; // liste des incréments à visualiser
170 int nb_maillage_dispo; // le nombre de maillage disponible
171 OrdreVisu* choix_mail; // ordre de choix des maillages
172 OrdreVisu* choix_isovaleur; // ordre de visualisation d'isovaleurs
173 // définition de la boite d'encombrement maxi des # maillages
174 Tableau <Vecteur> boite;
175 // indication si l'on est en animation ou pas
176 bool animation;
177 // indication si la visualisation de type vrml est active ou pas, par défaut = faux
178 bool activ_sort_vrml;

```

```

179
180 // METHODES PROTEGEES :
181 // test si la réponse fait partie des ordres possibles
182 // si l'on trouve un ordre ok on ramène un pointeur sur l'ordre
183 // sinon on ramène un pointeur null
184 OrdreVisu* Existe(string& reponse);
185 // affichage des options possibles
186 void Affiche_options();
187 // calcul de la plus grande boite
188 void Calcul_maxi_boite (Tableau <Vecteur>& boite_inter);
189 // définition des points de vue avec l'angle adoc
190 void DefinitionViewPoint();
191
192
193 };
194 /// @} // end of group
195
196 #endif

```

## 7.431 Animation\_vrml.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:          23/01/97
32 *
33 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:        Herezh++
36 *
37 *
38 *      BUT:           Sortie de l'animation.
39 *                   Celle-ci est automatique lorsqu'il y a plus d'un increment
40 *                   différent de 0.
41 *
42 *      *****
43 *
44 *      VERIFICATION:
45 *      ! date ! auteur ! but
46 *      -----
47 *      ! ! !
48 *
49 *      *****
50 *      MODIFICATIONS:
51 *      ! date ! auteur ! but
52 *      -----
53 *
54 *      *****/
55 #ifndef ANIMATION_T
56 #define ANIMATION_T
57
58
59 #include "OrdreVisu.h"
60 #include "Isovaleurs_vrml.h"
61 #include "ChoixDesMaillages_vrml.h"
62 #include "Deformees_vrml.h"
63

```

```

64 /// @addtogroup Les_sorties_vrml
65 /// @{
66 ///
67
68
69 class Animation_vrml : public OrdreVisu
70 {
71     public :
72         // CONSTRUCTEURS :
73         // par défaut
74         Animation_vrml () ;
75
76         // constructeur de copie
77         Animation_vrml (const Animation_vrml& ord);
78
79         // DESTRUCTEUR :
80         ~Animation_vrml () ;
81
82         // METHODES PUBLIQUES :
83
84         // execution de l'ordre
85         // tab_mail : donne les numéros de maillage concerné
86         // incre : numéro d'incrément qui en cours
87         // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
88         // animation : indique si l'on est en animation ou pas
89         // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
90         void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
unseul_incre,LesReferences*
91
92         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
93         ,Resultats*,UtilLecture & entreePrinc,EnumTypeIncre type_incre,int incre
, bool animation,const map < string, const double * , std::less <string> >&
94
95         listeVarGlob
96         ,const List_io < TypeQuelconque >& listeVecGlob);
97
98         // choix de l'ordre, cet méthode peut entraîner la demande d'informations
99         // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
100        void ChoixOrdre();
101
102        // --- méthodes particulière ----
103        // initialisation d'une liaison avec une instance de classe d'isovaleur
104        void Jonction_isovaleur(const Isovaleurs_vrml * iso) {isovaleurs_vrml = iso;};
105
106        // initialisation d'une liaison avec une instance de classe de choix des maillages
107        void Jonction_ChoixDesMaillages(const ChoixDesMaillages_vrml* choix_m) {choix_mail = choix_m;};
108
109        // initialisation d'une liaison avec une instance de classe de choix de la déformée
110        void Jonction_Deformees_vrml(const Deformees_vrml* choix_def) {choix_deformee = choix_def;};
111
112
113        // initialisation des listes de coordonnées
114        void Init_liste_coordonnee();
115
116        // lecture des paramètres de l'ordre dans un flux
117        void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
118        // écriture des paramètres de l'ordre dans un flux
119        void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
120
121     private :
122         // VARIABLES PROTEGEES :
123         double cycleInterval; // durée de l'animation
124         bool loop; // indique si on boucle ou pas
125         double startTime; // temps de démarrage en absolu depuis 1970
126         double stopTime; // temps de fin en absolu depuis 1970,
127         // si < a starttime on n'en tiend pas compte
128         bool debut_auto; // pour le début automatique
129         double inter_2pas; // interval entre deux dessins dans le cas d'un début automatique
130
131         const Isovaleurs_vrml * isovaleurs_vrml; // contient lorsqu'il est actif les couleurs des noeuds
132         const ChoixDesMaillages_vrml* choix_mail; // contient lorsqu'il est actif le choix des maillages
133         const Deformees_vrml* choix_deformee; // contient lorsqu'il est actif les choix relatifs à la
134         déformée
135
136         // dans le cas de l'animation : définition des listes de positions et de nom si rattachant
137
138         // METHODES PROTEGEES :
139
140 };
141 /// @} // end of group
142
143 #endif
144

```

## 7.432 ChoixDesMaillages\_vrml.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 * *****/
38 *   BUT:  Choix du ou des maillages à visualiser.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !           but
45 *   !           !           !
46 *
47 *   *****
48 *   MODIFICATIONS:
49 *
50 *   ! date !   auteur !           but
51 *
52 * *****/
53 #ifndef CHOIXDESMAILLAGES_VRML_T
54 #define CHOIXDESMAILLAGES_VRML_T
55
56 #include "OrdreVisu.h"
57
58 /// @addtogroup Les_sorties_vrml
59 /// @{
60 ///
61
62
63 class ChoixDesMaillages_vrml : public OrdreVisu
64 {
65 public :
66     // CONSTRUCTEURS :
67     // par défaut
68     ChoixDesMaillages_vrml () ;
69
70     // constructeur de copie
71     ChoixDesMaillages_vrml (const ChoixDesMaillages_vrml& ord);
72
73     // DESTRUCTEUR :
74     ~ChoixDesMaillages_vrml () ;
75
76     // METHODES PUBLIQUES :
77     // execution de l'ordre
78     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
79     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
80                 ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*
81                 ,LesContacts*,Resultats*,Utilecture & ,OrdreVisu::EnumTypeIncre ,int
82                 ,bool,const map < string, const double * , std::less <string> >&
83                 ,const List_io < TypeQuelconque >& listeVecGlob)
84     {};
```

```

85 // choix de l'ordre, cet méthode peut entraîner la demande d'informations
86 // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
87 void ChoixOrdre();
88 // initialisation du nombre de maillages possibles
89 // et définition de tous les maillages par défaut dans la liste des maillages
90 // a visualiser
91 void Init_nb_maill(int& nb_maillage_poss);
92 // ramène la liste des maillages choisit
93 // dans le cas où les numéros sont négatifs, indique que c'est uniquement
94 // les frontières où les données relatives aux frontières qui sont choisit
95 const list <int>& List_choisit() const
96 { return lis_sor;};
97
98 // lecture des paramètres de l'ordre dans un flux
99 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
100 // écriture des paramètres de l'ordre dans un flux
101 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
102
103 private :
104 // VARIABLES PROTEGEES :
105 int* nb_maillage; // nombre de maillage possible
106 list <int> lis_sor; // liste de maillage choisit
107 // METHODES PROTEGEES :
108
109 };
110 /// @} // end of group
111
112 #endif

```

## 7.433 Deformees\_vrml.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *      *****
38 *      BUT:      Visualisation de la déformée en vrml.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date !   auteur !           but
44 *      -----
45 *      !           !           !
46 *      *****
47 *
48 *      MODIFICATIONS:
49 *      ! date !   auteur !           but
50 *      -----
51 *
52 *      *****
53 #ifndef DEFORMEES_VRML_T

```

```

54 #define DEFORMEES_VRML_T
55
56 #include "OrdreVisu.h"
57 #include "Isovaleurs_vrml.h"
58 #include "ChoixDesMaillages_vrml.h"
59
60 /// @addtogroup Les_sorties_vrml
61 /// @{
62 ///
63
64
65 class Deformees_vrml : public OrdreVisu
66 {
67     public :
68         // CONSTRUCTEURS :
69         // par défaut
70         Deformees_vrml () ;
71         // constructeur à utiliser
72         Deformees_vrml(const string& comment_som, const string& explication, const string& ordre) :
73             OrdreVisu(comment_som,explication,ordre)
74             {};
75
76
77         // constructeur de copie
78         Deformees_vrml (const Deformees_vrml& algo);
79
80         // DESTRUCTEUR :
81         ~Deformees_vrml () ;
82
83         // METHODES PUBLIQUES :
84         // execution de l'ordre
85         // tab_mail : donne les numéros de maillage concerné
86         // incre : numéro d'incrément qui en cours
87         // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
88         // animation : indique si l'on est en animation ou pas
89         // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
90         virtual void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail,LesMaillages *,bool
unseul_incre,LesReferences*
91             ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
92             ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
93             ,bool animation,const map < string, const double * , std::less <string> >&
listeVarGlob
94             ,const List_io < TypeQuelconque >& listeVecGlob);
95         // choix de l'ordre, cet méthode peut entraîner la demande d'informations
96         // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
97         void ChoixOrdre();
98
99         // --- méthodes particulière ----
100         // initialisation d'une liaison avec une instance de classe d'isovaleur
101         void Jonction_isovaleur(const Isovaleurs_vrml * iso) {isovaleurs_vrml = iso;};
102         // initialisation d'une liaison avec une instance de classe de choix des maillages
103         void Jonction_ChoixDesMaillages(const ChoixDesMaillages_vrml* choix_m) {choix_mail = choix_m;};
104
105         // récup de la liste des noms de matières utilisés dans les shapes
106         const list <string> & Noms_matiere() const { return noms_matiere;};
107         // récup de la liste des noms de couleurs utilisés dans les shapes
108         const list <string> & Noms_couleurs() const { return noms_couleurs;};
109
110         // lecture des paramètres de l'ordre dans un flux
111         void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
112         // écriture des paramètres de l'ordre dans un flux
113         void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
114
115     protected :
116         // VARIABLES PROTEGEES :
117         bool filaire;
118         bool surface;
119         bool numero;
120         double Rcoull,Gcoull,BCoull; // couleur en RGB du tracé filaire
121         double Rcoulf,Gcoulf,BCoulf; // couleur en RGB du tracé des faces
122         double RcoulN,GcoulN,BCoulN; // couleur en RGB du tracé des numéros de noeud
123
124         double amplification; // amplification de la déformée
125
126         const Isovaleurs_vrml * isovaleurs_vrml; // contient lorsqu'il est actif les couleurs des noeuds
127
128         // on définit des tableaux intermédiaires pour le tracé des facettes et des lignes
129         list < Tableau < int > > tab_facette;
130         list < Tableau < int > > tab_arrete;
131
132         class Deuxentiers {public: int mail;int incr;}; // un conteneur de travail
133         list < Deuxentiers > num_mail_incr; // numéro de maillage et d'incrément correspondant
134         // à couleurs noeud
135         list <string> noms_matiere; // liste des noms de matériaux utilisée
136
137         list <string> noms_couleurs; // liste des noms de couleurs utilisée

```



```

137
138     const ChoixDesMaillages_vrml* choix_mail; // contient lorsqu'il est actif le choix des maillages
139
140
141     // METHODES PROTEGEES :
142
143 };
144 /// @} // end of group
145
146 #endif

```

## 7.434 Fin\_vrml.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *
38 *      BUT:      Fin de la visualisation en vrml.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *      ! date !   auteur !           but
44 *      -----
45 *      !           !           !
46 *
47 *      *****
48 *      MODIFICATIONS:
49 *      ! date !   auteur !           but
50 *      -----
51 *
52 *      *****/
53 #ifndef FIN_VRML_T
54 #define FIN_VRML_T
55
56 #include "OrdreVisu.h"
57
58 /// @addtogroup Les_sorties_vrml
59 /// @{
60 ///
61
62
63 class Fin_vrml : public OrdreVisu
64 {
65     public :
66         // CONSTRUCTEURS :
67         // par défaut
68         Fin_vrml () ;
69
70         // constructeur de copie

```

```

71     Fin_vrml (const Fin_vrml& algo);
72
73     // DESTRUCTEUR :
74     ~Fin_vrml () ;
75
76     // METHODES PUBLIQUES :
77     // execution de l'ordre
78     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
79                 ,LesLoisDeComp* ,DiversStockage*
80                 ,Charge*,LesCondLim*
81                 ,LesContacts*,Resultats*,UtilLecture & ,OrdreVisu::EnumTypeIncre ,int
82                 ,bool,const map < string, const double * , std::less <string> >&
83                 ,const List_io < TypeQuelconque >& listeVecGlob)
84
85     {};
86
87     // lecture des paramètres de l'ordre dans un flux
88     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
89     // écriture des paramètres de l'ordre dans un flux
90     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
91 private :
92     // VARIABLES PROTEGEES :
93
94     // METHODES PROTEGEES :
95
96 };
97 /// @} // end of group
98
99 #endif

```

## 7.435 Increment\_vrml.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:      23/01/97
32 *
33 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:    Herezh++
36 *
37 *****/
38 *   BUT: Demande d'un numéro ou d'une plage d'incrément.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date ! auteur ! but
44 *   -----
45 *   ! ! !
46 *   $
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date ! auteur ! but
50 *   -----
51 *   $
52 *****/

```

```

53 #ifndef INCREMENT_VRML_T
54 #define INCREMENT_VRML_T
55
56 #include "OrdreVisu.h"
57
58 /// @addtogroup Les_sorties_vrml
59 /// @{
60 ///
61
62
63 class Increment_vrml : public OrdreVisu
64 {
65 public :
66     // CONSTRUCTEURS :
67     // par défaut
68     Increment_vrml () ;
69
70     // constructeur de copie
71     Increment_vrml (const Increment_vrml& algo);
72
73     // DESTRUCTEUR :
74     ~Increment_vrml () ;
75
76     // METHODES PUBLIQUES :
77     // execution de l'ordre
78     // tab_mail : donne les numéros de maillage concerné
79     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
80     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
81                 ,LesLoisDeComp* ,DiversStockage* ,Charge* ,LesCondLim*
82                 ,LesContacts* ,Resultats* ,UtilLecture & ,OrdreVisu::EnumTypeIncre ,int
83                 ,bool ,const map < string , const double * , std::less <string> >&
84                 ,const List_io < TypeQuelconque >& listeVecGlob)
85         {};
86     // choix de l'ordre, cet méthode peut entraîner la demande d'informations
87     // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
88     void ChoixOrdre();
89     // initialisation de la liste d'incrément possible
90     void Init_list_inc(list<int>& li_inc)
91         {lis_inc = & li_inc;};
92     // initialisation de la liste à l'incrément 0 et au dernier increment
93     // s'il est différent
94     void Init_list_inc_DebuEtFin();
95     // ramène la liste des incréments choisit
96     const list <int>& List_choisit()
97         { return lis_sor;};
98     // imposition d'une liste d'incrément donné
99     void Impos_list(const list<int>& li_inc) {lis_sor = li_inc;};
100
101     // lecture des paramètres de l'ordre dans un flux
102     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
103     // écriture des paramètres de l'ordre dans un flux
104     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
105
106 private :
107     // VARIABLES PROTEGEES :
108     list <int>* lis_inc; // liste d'incrément possible
109     // c'est uniquement le pointeur qui est sauvegardé, il n'y a pas de new
110     // associé dans les méthodes
111     list <int> lis_sor; // liste d'incrément lue
112     // METHODES PROTEGEES :
113
114 };
115 /// @} // end of group
116
117 #endif

```

## 7.436 Isovaleurs\_vrml.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,

```

```

18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           23/01/97
32 *
33 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:        Herezh++
36 *
37 *
38 *   BUT:  Visualisation des isovaleurs en vrml.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date !   auteur !           but
44 *   -----
45 *   !           !           !
46 *
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date !   auteur !           but
50 *   -----
51 *
52 *   *****/
53 #ifndef ISOVALEURS_VRML_T
54 #define ISOVALEURS_VRML_T
55
56 #include "OrdreVisu.h"
57 #include "spectre.h"
58
59 /// @addtogroup Les_sorties_vrml
60 /// @{
61 ///
62
63
64 class Isovaleurs_vrml : public OrdreVisu
65 {
66 public :
67     // CONSTRUCTEURS :
68     // par défaut
69     Isovaleurs_vrml () ;
70
71     // constructeur de copie
72     Isovaleurs_vrml (const Isovaleurs_vrml& algo);
73
74     // DESTRUCTEUR :
75     ~Isovaleurs_vrml () ;
76
77     // METHODES PUBLIQUES :
78     // initialisation : permet d'initialiser les différents paramètres de l'ordre
79     // lors d'un premier passage des différents incréments
80     // en virtuelle, a priori est défini si nécessaire dans les classes dérivées
81     // incre : numéro d'incrément qui en cours
82     void Initialisation(ParaGlob * ,LesMaillages * ,LesReferences*
83         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
84         ,Resultats*,EnumTypeIncre type_incre,int incre
85         ,const map < string, const double * , std::less <string> >& listeVarGlob
86         ,const List_io < TypeQuelconque >& listeVecGlob
87         ,bool fil_calcul) ;
88     // execution de l'ordre
89     // tab_mail : donne les numéros de maillage concerné
90     // incre : numéro d'incrément qui en cours
91     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
92     // animation : indique si l'on est en animation ou pas
93     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
94     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
95         unseul_incre,LesReferences*
96         ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
97         ,Resultats*,Utilecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
98         ,bool animation,const map < string, const double * , std::less <string> >&
99         listeVarGlob
100         ,const List_io < TypeQuelconque >& listeVecGlob);
101     // choix de l'ordre, cet méthode peut entraîner la demande d'informations
102     // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
103     void ChoixOrdre();

```

```

102
103 // initialisation de la liste des différentes isovaleurs possibles
104 void Init_liste_isovaleur(LesMaillages * lesMail,LesContacts* lesContacts);
105
106 // ramène une référence constante des couleurs des noeuds des incréments
107 const list < Tableau < Rgb > > & List_couleur() const {return couleurs_noeud;};
108 // ramène une référence constante des numéros de maillage et d'incrément correspondants aux couleurs
109 const list < Deuxentiers > & List_num_mail_incr() const {return num_mail_incr;};
110
111 // lecture des paramètres de l'ordre dans un flux
112 void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
113 // écriture des paramètres de l'ordre dans un flux
114 void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
115
116 protected :
117 // VARIABLES PROTEGEES :
118 bool choix_peau; // si oui visualisation uniquement de la peau
119 bool choix_legende; // si oui affichage de la légende
120 int choix_min_max; // si 1 min max calculé automatiquement et globalement
121 // si 2 min max calculé en % à chaque incr / à un pourcentage
122 // si 3 min max calculé auto en % globalement
123 // si 0 min max fixé
124 double valMini,valMaxi; // les miniMax totaux ou pour chaque incrément
125 double pour_legende_min,pour_legende_max; // min et max en % : set éventuellement, dépend de
choix_min_max
126 int nb_iso_val; // nombre d'isovaleurs dans la légende
127 bool absolue; // par défaut on sort en absolue les tenseurs
128 // pour pouvoir correctement les visualiser, même pour les elem 2D
129
130 Tableau <List_io < Ddl_enum_etendu > > tabnoeud_type_ddl; // ddl aux noeuds possibles à visualiser
131 Tableau < Ddl_enum_etendu > type_ddl_retenu; // ddl à visualiser, un par maillage
132 Tableau <bool> choix_var_ddl; // indique si c'est une variation ou le ddl
133 Tableau <List_io < Ddl_enum_etendu > > tabelement_type_ddl; // ddl aux elements possibles à
visualiser
134
135 // on définit les valeurs des couleurs aux noeuds
136 // chaque élément de la liste correspond à un incrément
137 // (i) : correspond au noeud i
138 list < Tableau < Rgb > > couleurs_noeud;
139 list < Tableau < double > > val_noeud; // tableau de travail
140
141 list < Deuxentiers > num_mail_incr; // numéro de maillage et d'incrément correspondant
142 // à couleurs noeud
143
144 Spectre spectre_coul; // le spectre de couleur utilisé pour les isovaleurs
145
146 // METHODES PROTEGEES :
147
148 // sortie du dessin de la légende
149 virtual void Sortie_dessin_legende(UtilLecture & entreePrinc);
150
151 // constructeur utilisé par les classes dérivées
152 // en fait il s'agit de transmettre directement les infos à la classe mère Visu
153 Isovaleurs_vrml(const string& comment_som, const string& explication
154 , const string& ordre) :
155 OrdreVisu(comment_som,explication,ordre)
156 {};
157
158 // définition interactives des paramètres généraux des isovaleurs
159 void ParametresGeneraux();
160 // choix de l'isovaleur à visualiser
161 void ChoixIsovaleur();
162 // choix de grandeur existantes aux points d'intégrations et à ramener aux noeuds
163 // pour préparer une visualisation d'isovaleurs
164 void TransfertGrandeursPtIntegAuNoeud();
165
166 };
167 /// @} // end of group
168
169 #endif

```

## 7.437 Mail\_initiale\_vrml.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio

```

```

13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29 //
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *****/
38 *      BUT:      Visualisation du maillage initiale en vrml.
39 *
40 *      *****
41 *
42 *      VERIFICATION:
43 *
44 *      ! date !   auteur !           but
45 *      -----
46 *      !           !           !           !
47 *      *****
48 *      MODIFICATIONS:
49 *
50 *      ! date !   auteur !           but
51 *      -----
52 *      $
53 *****/
54 #ifndef MAIL_INITIALE_VRML_T
55 #define MAIL_INITIALE_VRML_T
56 #include "OrdreVisu.h"
57
58 /// @addtogroup Les_sorties_vrml
59 /// @{
60 ///
61
62 class Mail_initiale_vrml : public OrdreVisu
63 {
64 public :
65     // CONSTRUCTEURS :
66     // par defaut
67     Mail_initiale_vrml () ;
68
69     // constructeur de copie
70     Mail_initiale_vrml (const Mail_initiale_vrml& algo);
71
72     // DESTRUCTEUR :
73     ~Mail_initiale_vrml () ;
74
75     // METHODES PUBLIQUES :
76     // execution de l'ordre
77     // tab_mail : donne les numéros de maillage concerné
78     // incre : numéro d'incrément qui en cours
79     // type_incre : indique si c'est le premier le dernier ou l'incrément courant a visualiser ou pas
80     // animation : indique si l'on est en animation ou pas
81     // unseul_incre : indique si oui ou non il y a un seul increment à visualiser
82     void ExeOrdre(ParaGlob * ,const Tableau <int>& tab_mail ,LesMaillages * ,bool
unseul_incre,LesReferences*
83     ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*,LesContacts*
84     ,Resultats*,UtilLecture & entreePrinc,OrdreVisu::EnumTypeIncre type_incre,int incre
85     ,bool animation,const map < string, const double * , std::less <string> >&
86     listeVarGlob
87     ,const List_io < TypeQuelconque >& listeVecGlob);
88     // choix de l'ordre, cet méthode peut entraîner la demande d'informations
89     // supplémentaires si nécessaire. qui sont ensuite gérer par la classe elle même
90     void ChoixOrdre();
91
92     // lecture des paramètres de l'ordre dans un flux
93     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
94     // écriture des paramètres de l'ordre dans un flux
95     void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
96     protected :

```

```

97 // VARIABLES PROTEGEES :
98 bool filaire;
99 bool surface;
100 bool numero;
101 double Rcoull,Gcoull,Bcoull; // couleur en RGB du tracé filaire
102 double Rcoulf,Gcoulf,Bcoulf; // couleur en RGB du tracé des faces
103 double Rcouln,Gcouln,Bcouln; // couleur en RGB du tracé des numéros de noeud
104 double taille_numero; // taille des numéros
105
106 // METHODES PROTEGEES :
107
108 };
109 ///< @} // end of group
110
111 #endif

```

## 7.438 Visuali\_vrml.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *      DATE:      23/01/97
32 *
33 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *      PROJET:    Herezh++
36 *
37 *
38 *      BUT: Arrêt des question Mise en route de la visualisation
39 *          en vrml.
40 *
41 *      *****
42 *
43 *      VERIFICATION:
44 *      ! date !   auteur !   but
45 *      -----
46 *      !           !           !
47 *
48 *      *****
49 *      MODIFICATIONS:
50 *      ! date !   auteur !   but
51 *      -----
52 *
53 *
54 #ifndef VISUALI_VRML_T
55 #define VISUALI_VRML_T
56
57 #include "OrdreVisu.h"
58
59 /** @defgroup Les_sorties_vrml
60 *
61 *      BUT: groupe concernant le postraitement au format vrml
62 *
63 *
64 * \author Gérard Rio
65 * \version 1.0
66 * \date 03/02/2003

```

```

67 * \brief      groupe concernant le postraitement au format vrml
68 *
69 */
70
71 /// @addtogroup Les_sorties_vrml
72 /// @{
73 ///
74
75 class Visuali_vrml : public OrdreVisu
76 {
77 public :
78     // CONSTRUCTEURS :
79     // par défaut
80     Visuali_vrml () ;
81
82     // constructeur de copie
83     Visuali_vrml (const Visuali_vrml& algo);
84
85     // DESTRUCTEUR :
86     ~Visuali_vrml () ;
87
88     // METHODES PUBLIQUES :
89     // execution de l'ordre
90     void ExeOrdre(ParaGlob * ,const Tableau <int>& ,LesMaillages * ,bool ,LesReferences*
91                 ,LesLoisDeComp* ,DiversStockage*,Charge*,LesCondLim*
92                 ,LesContacts*,Resultats*,UtilLecture & ,OrdreVisu::EnumTypeIncre,int
93                 ,bool ,const map < string, const double * , std::less <string> >&
94                 ,const List_io < TypeQuelconque >& listeVecGlob)
95         {};
96
97     // lecture des paramètres de l'ordre dans un flux
98     void Lecture_parametres_OrdreVisu(UtilLecture & entreePrinc);
99     // écriture des paramètres de l'ordre dans un flux
100    void Ecriture_parametres_OrdreVisu(UtilLecture & entreePrinc);
101
102 private :
103     // VARIABLES PROTEGEES :
104
105     // METHODES PROTEGEES :
106
107 };
108 /// @} // end of group
109
110 #endif

```

## 7.439 Tableau2\_T.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           23/01/97
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *****/
38 *   BUT: définir un tableau template de dimension 2.

```



```

39 *                                     $ *
40 *      ***** *
41 *      VERIFICATION: *
42 *      * *
43 *      ! date ! auteur ! but ! *
44 *      ----- *
45 *      ! ! ! ! *
46 *      * *
47 *      ***** *
48 *      MODIFICATIONS: *
49 *      ! date ! auteur ! but ! *
50 *      ----- *
51 *      * *
52 *      *****/
53 #ifndef TABLEAU2_T_H
54 #define TABLEAU2_T_H
55
56 #include "Tableau_T.h"
57
58 /// @addtogroup Les_Tableaux_generiques
59 /// @{
60 ///
61
62
63 template <class T>
64 class Tableau2 : public Tableau <T>
65 {
66     public :
67         // CONSTRUCTEURS :
68
69         // Constructeur par défaut
70         Tableau2 ();
71
72         // Constructeur fonction de la taille du tableau
73         // dans le cas d'une taille il s'agit d'un tableau carré
74         Tableau2 (const int nb);
75         // cas de deux dimensions différentes
76         Tableau2 (const int nbl,const int nb2);
77         // Constructeur fonction de la taille du tableau et d'une
78         // valeur d'initialisation pour les composantes
79         //(1) tableau carré
80         Tableau2 (const int nb,const T& val);
81         //(2) tableau rectangulaire
82         Tableau2 (const int nbl,const int nb2,const T& val);
83         // Constructeur de copie
84         Tableau2 (const Tableau2<T> & tab);
85
86
87         // DESTRUCTEUR :
88         ~Tableau2 ();
89
90
91         // METHODES :
92         inline int Taille1 () const
93         // Retourne la première taille du tableau
94         { return taille1; };
95         inline int Taille2 () const
96         // Retourne la seconde taille du tableau
97         { return taille2; };
98         // gestion d'erreur
99         // cette fonction n'est plus valable d'où la surcharge pour filtrer les
100        // appels éventuels
101        inline int Taille () const
102        {
103            cout << "\nErreur : fonction non disponible \n" ;
104            cout << "TABLEAU2_T::Taille () \n";
105            Sortie(1);
106            return 0; // pour éviter le warning
107        };
108
109        inline T& operator() (int i, int j)
110        // Retourne la (ieme,jeme) composante du tableau : acces en lecture et ecriture
111        {
112            #ifdef MISE_AU_POINT
113            if ( (i<1) || (i>taille1) || (j<1) || (j>taille2) )
114            { cout << "\nErreur : composante inexistante !, (i,j) demandé = (" <i> i
115            << ',' << j << ")" << '\n';
116            cout << "TABLEAU2_T::OPERATOR() (int, int ) \n";
117            Sortie(1);
118            };
119            #endif
120            return this->t[(i-1)*taille2 + j-1];
121        };
122        inline T operator() (int i, int j) const
123        // Retourne la (ieme,jeme) composante du tableau : acces en lecture uniquement
124        {

```

```

125     #ifndef MISE_AU_POINT
126     if ( (i<1) || (i>taille1) || (j<1) || (j>taille2) )
127     { cout << "\nErreur : composante inexistante !, (i,j) demandé = (" << i
128       << ',' << j << ")" << '\n';
129       cout << "TABLEAU2_T::OPERATOR() (int, int ) \n";
130       Sortie(1);
131     };
132     #endif
133     return this->t[(i-1)*taille2 + j-1];
134 };
135
136 inline int operator!=(const Tableau2<T>& tab) const
137 // Surcharge de l'operateur !=
138 // Renvoie 1 si les deux tableaux ne sont pas egaux
139 // Renvoie 0 sinon
140 { if ( (*this)==tab ) // test de l'egalite des deux tableaux a l'aide
141     // de l'operateur surcharge ==
142     return 0;
143     else
144     return 1;
145 };
146
147 // Surcharge de l'operateur ==
148 int operator==(const Tableau2<T>& tab) const ;
149 // const {return ((*this)==tab); };
150
151 // Surcharge de l'operateur d'affectation =
152 Tableau2<T>& operator= (const Tableau2<T>& tab);
153
154 // Change la taille du tableau (la nouvelle taille est n)
155 // uniquement valable pour les tableaux carrés
156 // N.B. : Si la nouvelle taille est superieure a l'ancienne alors le tableau est
157 // complete par default
158 void Change_taille (int n);
159 // idem et initialisation de toutes les valeurs, anciennes et nouvelles à tb
160 void Change_taille (int n,const T& tb);
161 // Change la taille du tableau (les nouvelles tailles sont n1 et n2)
162 void Change_taille (int n1, int n2);
163 // Change la taille du tableau (les nouvelles tailles sont n1 et n2)
164 // et initialisation de toutes les valeurs, anciennes et nouvelles à tb
165 void Change_taille (int n1, int n2,const T& tb);
166 // Permet de desallouer l'ensemble des elements du tableau
167 void Libere ();
168
169
170 protected :
171     int taille1; // la première taille du tableau
172     int taille2; // seconde taille du tableau
173
174 };
175 /// @} // end of group
176
177 //===== def des differents elements =====
178
179 // Constructeur par default
180 template <class T>
181 inline Tableau2<T>::Tableau2 () :
182     Tableau<T> ()
183 {   taille1=0;taille2=0;
184 }
185
186 // Constructeur fonction de la taille du tableau
187 // dans le cas d'une taille il s'agit d'un tableau carré
188 template <class T>
189 inline Tableau2<T>::Tableau2 (const int nb) :
190     Tableau<T> (nb*nb)
191     {   taille1=nb;taille2=nb;
192     }
193 // cas de deux dimensions différentes
194 template <class T>
195 inline Tableau2<T>::Tableau2 (const int nb1,const int nb2) :
196     Tableau<T> (nb1*nb2)
197     {   taille1=nb1;taille2=nb2;
198     }
199 // Constructeur fonction de la taille du tableau et d'une
200 // valeur d'initialisation pour les composantes
201 //1) tableau carré
202 template <class T>
203 inline Tableau2<T>::Tableau2 (const int nb,const T& val) :
204     Tableau<T> (nb*nb,val)
205     {   taille1=nb;taille2=nb;
206     }
207 //2) tableau rectangulaire
208 template <class T>
209 inline Tableau2<T>::Tableau2 (const int nb1,const int nb2,const T& val) :
210     Tableau<T> (nb1*nb2,val)
211     {   taille1=nb1;taille2=nb2;

```

```

212     }
213 // Constructeur de copie
214 template <class T>
215 inline Tableau2<T>::Tableau2 (const Tableau2<T> & tab) :
216     Tableau<T> (tab)
217     { taille1 = tab.taille1;
218       taille2 = tab.taille2;
219     }
220 // DESTRUCTEUR :
221 template <class T>
222 inline Tableau2<T>::~~Tableau2 ()
223 {   Libere(); }
224
225 // METHODES :
226 template <class T>
227 inline Tableau2<T> & Tableau2<T>::operator= (const Tableau2<T> & tab)
228 // Surcharge de l'operateur = : realise l'egalite entre deux tableaux de pointeurs
229 {   Tableau<T> & T1 = *this;
230     //this->Tableau<T>::operator= (T1,T2);
231     T1 = (Tableau<T>) tab;
232     taille1 = tab.taille1;
233     taille2 = tab.taille2;
234     return (*this);
235 }
236
237 // Change la taille du tableau (la nouvelle taille est n)
238 // uniquement valable pour les tableaux carrés
239 // N.B. : Si la nouvelle taille est superieure a l'ancienne alors le tableau est
240 // complete par default
241 template <class T>
242 inline void Tableau2<T>::Change_taille (int n)
243 {
244     #ifdef MISE_AU_POINT
245     if (taille1 != taille2)
246         { cout << "\nErreur : le changement de taille demandé n'est valide que pour des tableaux carrés,"
247           << " ici taille 1= " << taille1 << " et taille2 = " << taille2 << '\n';
248           cout << "Tableau2<T>::Change_taille (int n) \n";
249           Sortie(1);
250         };
251     if ( n < 0)
252         { cout << "\nErreur : la nouvelle taille demandée est négative , n= " << n << '\n';
253           cout << "Tableau2<T>::Change_taille (int n) \n";
254           Sortie(1);
255         };
256     #endif
257     this->Tableau<T>::Change_taille (n*n);
258     taille1 = n; taille2 = n;
259 }
260 // idem et initialisation de toutes les valeurs, anciennes et nouvelles à tb
261 template <class T>
262 inline void Tableau2<T>::Change_taille (int n,const T& tb)
263 {
264     #ifdef MISE_AU_POINT
265     if (taille1 != taille2)
266         { cout << "\nErreur : le changement de taille demandé n'est valide que pour des tableaux carrés,"
267           << " ici taille 1= " << taille1 << " et taille2 = " << taille2 << '\n';
268           cout << "Tableau2<T>::Change_taille (int n) \n";
269           Sortie(1);
270         };
271     if ( n < 0)
272         { cout << "\nErreur : la nouvelle taille demandée est négative , n= " << n << '\n';
273           cout << "Tableau2<T>::Change_taille (int n) \n";
274           Sortie(1);
275         };
276     #endif
277     this->Tableau<T>::Change_taille (n*n,tb);
278     taille1 = n; taille2 = n;
279 }
280
281 // Change la taille du tableau (les nouvelles tailles sont n1 et n2)
282 template <class T>
283 inline void Tableau2<T>::Change_taille (int n1, int n2)
284 {
285     #ifdef MISE_AU_POINT
286     if (( n1 < 0) || (n2 < 0))
287         { cout << "\nErreur : une des nouvelles tailles demandée est négative , n1= " << n1 << " n2 = " <<
288           n2 << '\n';
289           cout << "Tableau2<T>::Change_taille (int n1,int n2) \n";
290           Sortie(1);
291         };
292     #endif
293     this->Tableau<T>::Change_taille (n1*n2);
294     taille1 = n1; taille2 = n2;
295 }
296 // Permet de desallouer l'ensemble des elements du tableau
297 template <class T>
298 inline void Tableau2<T>::Libere ()

```

```

298 { this->Tableau<T>::Libere ();
299     taille1 = 0; taille2 = 0;
300 }
301 // Surcharge de l'operateur ==
302 template <class T>
303 inline int Tableau2<T>::operator==(const Tableau2<T>& tab) const
304 { if ((tab.taille1 != taille1) || (tab.taille2 != taille2))
305     return 0;
306     else
307     return this->Tableau<T>::operator==(tab);
308 }
309 // Change la taille du tableau (les nouvelles tailles sont n1 et n2)
310 // et initialisation de toutes les valeurs, anciennes et nouvelles à tb
311 template <class T>
312 inline void Tableau2<T>::Change_taille (int n1, int n2,const T& tb)
313 {
314     #ifdef MISE_AU_POINT
315     if ((n1 < 0) || (n2 < 0))
316     { cout << "\nErreur : une des nouvelles tailles demandée est négative , n1= " << n1 << " n2 = " <<
n2 << "\n";
317         cout << "Tableau2<T>::Change_taille (int n1,int n2) \n";
318         Sortie(1);
319     };
320     #endif
321     this->Tableau<T>::Change_taille (n1*n2,tb);
322     taille1 = n1; taille2 = n2;
323 }
324
325 #endif

```

## 7.440 Tableau4\_T.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *   DATE:           8/6/2003
32 *
33 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
34 *
35 *   PROJET:         Herezh++
36 *
37 *****/
38 *   BUT: définir un tableau template de dimension 4.
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *   ! date ! auteur ! but
44 *   -----
45 *   ! ! !
46 *
47 *   *****
48 *   MODIFICATIONS:
49 *   ! date ! auteur ! but
50 *   -----
51 *
52 *****/

```

```

53 #ifndef TABLEAU4_T_H
54 #define TABLEAU4_T_H
55
56 #include "Tableau_T.h"
57
58 /// @addtogroup Les_Tableaux_generiques
59 /// @{
60 ///
61
62
63 template <class T>
64 class Tableau4 : public Tableau <T>
65 {
66 public :
67     // CONSTRUCTEURS :
68
69     // Constructeur par défaut
70     Tableau4 ();
71
72     // Constructeur fonction de la taille du tableau
73     // dans le cas d'une taille il s'agit d'un tableau 4 fois la même dim
74     Tableau4 (const int nb);
75     // cas de 4 dimensions différentes
76     Tableau4 (const int nb1,const int nb2,const int nb3,const int nb4);
77     // Constructeur fonction de la taille du tableau et d'une
78     // valeur d'initialisation pour les composantes
79     //1) tableau 4 dim identique
80     Tableau4 (const int nb,const T& val);
81     //2) tableau avec des dimensions différentes
82     Tableau4 (const int nb1,const int nb2,const int nb3,const int nb4,const T& val);
83     // Constructeur de copie
84     Tableau4 (const Tableau4<T> & tab);
85
86
87     // DESTRUCTEUR :
88     ~Tableau4 ();
89
90
91     // METHODES :
92     // Retourne la première taille du tableau
93     inline int Taille1 () const { return taille1; };
94     // Retourne la deuxième taille du tableau
95     inline int Taille2 () const{ return taille2; };
96     // Retourne la troisième taille du tableau
97     inline int Taille3 () const{ return taille3; };
98     // Retourne la quatrième taille du tableau
99     inline int Taille4 () const{ return taille4; };
100    // gestion d'erreur
101    // cette fonction n'est plus valable d'où la surcharge pour filtrer les
102    // appels éventuels
103    inline int Taille () const
104    {
105        cout << "\nErreur : fonction non disponible \n" ;
106        cout << "Tableau4_T::Taille () \n";
107        Sortie(1);
108        return 0; // pour éviter le warning
109    };
110
111    inline T& operator() (int i, int j, int k, int l)
112    // Retourne la (ieme,jeme) composante du tableau : acces en lecture et ecriture
113    {
114        #ifdef MISE_AU_POINT
115        if ( (i<1) || (i>taille1) || (j<1) || (j>taille2)
116            || (k<1) || (k>taille3) || (l<1) || (l>taille4))
117            { cout << "\nErreur : composante inexistante !, (i,j) demandé = (" << i
118              << ',' << j << ", " << k << ", " << l << ") " << '\n';
119              cout << "Tableau4_T::OPERATOR() (int, int, int, int ) \n";
120              Sortie(1);
121            };
122        #endif
123        return this->t[taille4*((k-1) + taille3*((j-1)+taille2*(i-1))) + l-1];
124    };
125    inline T operator() (int i, int j, int k, int l) const
126    // Retourne la (ieme,jeme) composante du tableau : acces en lecture uniquement
127    {
128        #ifdef MISE_AU_POINT
129        if ( (i<1) || (i>taille1) || (j<1) || (j>taille2)
130            || (k<1) || (k>taille3) || (l<1) || (l>taille4))
131            { cout << "\nErreur : composante inexistante !, (i,j) demandé = (" << i
132              << ',' << j << ", " << k << ", " << l << ") " << '\n';
133              cout << "Tableau4_T::OPERATOR() (int, int ) \n";
134              Sortie(1);
135            };
136        #endif
137        return this->t[taille4*((k-1) + taille3*((j-1)+taille2*(i-1))) + l-1];
138    };
139

```

```

140     inline int operator!=(const Tableau4<T>& tab) const
141     // Surcharge de l'operateur !=
142     // Renvoie 1 si les deux tableaux ne sont pas egaux
143     // Renvoie 0 sinon
144     {   if ( (*this)==tab ) // test de l'egalite des deux tableaux a l'aide
145         // de l'operateur surcharge ==
146         return 0;
147     else
148         return 1;
149     };
150
151     // Surcharge de l'operateur ==
152     int operator==(const Tableau4<T>& tab) const ;
153     // const {return ((*this)==tab); };
154
155     // Surcharge de l'operateur d'affectation =
156     Tableau4<T>& operator=(const Tableau4<T>& tab);
157
158     // Change la taille du tableau (la nouvelle taille est n)
159     // uniquement valable pour les tableaux de dimension identique
160     // N.B. : Si la nouvelle taille est superieure a l'ancienne alors le tableau est
161     // complete par default, mais les valeurs anciennes et nouvelles sont mises a la valeur par
    défaut de T
162     // contrairement au Tableau_T
163     void Change_taille (int n);
164     // Change la taille du tableau (les nouvelles tailles sont n1 et n2,n3,n4)
165     // et toutes les valeurs sont initialisées a la valeur par défaut de T
166     void Change_taille (int n1, int n2,int n3, int n4);
167     // Change la taille du tableau (la nouvelle taille est n)
168     // et initialisation de toutes les valeurs, anciennes et nouvelles a tb
169     void Change_taille (int n,T& tb)
170     { cout << "\n fonction non disponible pour des tableaux a quatre dim"
171       << "\n Tableau4<T>::Change_taille (int n,T& tb)"; Sortie(1); };
172     // Permet de desallouer l'ensemble des elements du tableau
173     void Libere ();
174
175
176     protected :
177         int taille1; // la première taille du tableau
178         int taille2; // seconde taille du tableau
179         int taille3; // troisième taille du tableau
180         int taille4; // quatrième taille du tableau
181
182     };
183     /// @} // end of group
184
185     //===== def des differents elements =====
186
187     // Constructeur par default
188     template <class T>
189     inline Tableau4<T>::Tableau4 () :
190     Tableau<T> (),taille1(0),taille2(0),taille3(0),taille4(0)
191     { }
192
193     // Constructeur fonction de la taille du tableau
194     // dans le cas d'une taille identique
195     template <class T>
196     inline Tableau4<T>::Tableau4 (const int nb) :
197     Tableau<T> (nb*nb*nb*nb),taille1(nb),taille2(nb),taille3(nb),taille4(nb)
198     { }
199     // cas de deux dimensions différentes
200     template <class T>
201     inline Tableau4<T>::Tableau4 (const int nb1,const int nb2,const int nb3,const int nb4) :
202     Tableau<T> (nb1*nb2*nb3*nb4),taille1(nb1),taille2(nb2),taille3(nb3),taille4(nb4)
203     { }
204     // Constructeur fonction de la taille du tableau et d'une
205     // valeur d'initialisation pour les composantes
206     // //1) tableau de dim identique
207     template <class T>
208     inline Tableau4<T>::Tableau4 (const int nb,const T& val) :
209     Tableau<T> (nb*nb*nb*nb,val),taille1(nb),taille2(nb),taille3(nb),taille4(nb)
210     { }
211     //2) tableau rectangulaire
212     template <class T>
213     inline Tableau4<T>::Tableau4 (const int nb1,const int nb2,const int nb3,const int nb4,const T& val) :
214     Tableau<T> (nb1*nb2*nb3*nb4,val),taille1(nb1),taille2(nb2),taille3(nb3),taille4(nb4)
215     { }
216     // Constructeur de copie
217     template <class T>
218     inline Tableau4<T>::Tableau4 (const Tableau4<T> & tab) :
219     Tableau<T> (tab,taille1(tab.taille1),taille2(tab.taille2)
220     ,taille3(tab.taille3),taille4(tab.taille4)
221     { }
222     // DESTRUCTEUR :
223     template <class T>
224     inline Tableau4<T>::~Tableau4 ()
225     { Libere(); }

```

```

226
227 // METHODES :
228 template <class T>
229 inline Tableau4<T> & Tableau4<T>::operator= (const Tableau4<T> & tab)
230 // Surcharge de l'operateur = : realise l'egalite entre deux tableaux de pointeurs
231 {
232     Tableau4<T> & T1 = *this;
233     //this->Tableau4<T>::operator= (T1,T2);
234     T1 = (Tableau4<T>) tab;
235     taille1 = tab.taille1;taille2 = tab.taille2;
236     taille3 = tab.taille3;taille4 = tab.taille4;
237     return (*this);
238 }
239 // Change la taille du tableau (la nouvelle taille est n)
240 // uniquement valable pour les tableaux carrés
241 // N.B. : Si la nouvelle taille est superieure a l'ancienne alors le tableau est
242 // complete par default
243 template <class T>
244 inline void Tableau4<T>::Change_taille (int n)
245 {
246     #ifdef MISE_AU_POINT
247     if ((taille1 != taille2) || (taille1 != taille3) || (taille1 != taille4)
248         || (taille2 != taille3) || (taille2 != taille4))
249         { cout << "\nErreur : le changement de taille demande n'est valide que pour des tableaux de meme 4
250         dimensions,"
251           << " ici taille 1= " << taille1 << " et taille2 = " << taille2
252           << " et taille3 = " << taille3 << " et taille4 = " << taille4 << '\n';
253           cout << "Tableau4<T>::Change_taille (int n) \n";
254           Sortie(1);
255         };
256     if ( n < 0)
257         { cout << "\nErreur : la nouvelle taille demandée est négative , n= " << n << '\n';
258           cout << "Tableau4<T>::Change_taille (int n) \n";
259           Sortie(1);
260         };
261     #endif
262     if (n != taille1)
263     { // on intervient que si la nouvelle dimension est différente
264       T tb; // def d'une valeur par défaut
265       // déf du tableau avec initialisation à tb
266       this->Tableau4<T>::Change_taille (n*n*n*n,tb);
267       taille1 = n; taille2 = n;taille3 = n;taille4 = n;
268     }
269 // Change la taille du tableau (les nouvelles tailles sont n1 et n2)
270 template <class T>
271 inline void Tableau4<T>::Change_taille (int n1, int n2,int n3, int n4)
272 {
273     #ifdef MISE_AU_POINT
274     if (( n1 < 0) || (n2 < 0) || (n3<0) || (n4<0))
275         { cout << "\nErreur : une des nouvelles tailles demandée est négative , n1= " << n1 << " n2 = " <<
276         n2
277         << " n3 = " << n3 << " n4 = " << n4 << '\n';
278         cout << "Tableau4<T>::Change_taille (int n1,int n2,int n3, int n4) \n";
279         Sortie(1);
280         };
281     #endif
282     if ( ( n1 != taille1) || (n2 != taille2) || (n3 != taille3) || (n4 != taille4))
283         // dans le cas où il y a un changement de taille on répercute
284         { // dimensionnement du nouveau tableau
285           T tb; // def d'une valeur par défaut
286           // déf du tableau avec initialisation à tb
287           this->Tableau4<T>::Change_taille (n1*n2*n3*n4,tb);
288           taille1 = n1; taille2 = n2;taille3 = n3;taille4 = n4;
289         }
290     }
291 // Permet de desallouer l'ensemble des elements du tableau
292 template <class T>
293 inline void Tableau4<T>::Libere ()
294 {
295     this->Tableau4<T>::Libere ();
296     taille1 = 0; taille2 = 0; taille3 = 0;taille4 = 0;
297 }
298 // Surcharge de l'operateur ==
299 template <class T>
300 inline int Tableau4<T>::operator==(const Tableau4<T>& tab) const
301 {
302     if ((tab.taille1 != taille1) || (tab.taille2 != taille2)
303         || (tab.taille3 != taille3) || (tab.taille4 != taille4))
304         return 0;
305     else
306         return this->Tableau4<T>::operator==( tab);
307 }
308 #endif

```

## 7.441 Tableau\_3D.h

```

1 // FICHER : Tableau_3D.h
2 // CLASSE : Tableau_3D
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *   DATE:      23/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *
41 * La classe Tableau_3D permet de declarer des tableaux a trois dimensions dont
42 * les composantes sont de type double.
43 * Les composantes d'une instance de cette classe sont stockees en ligne et la memoire
44 * est allouee dynamiquement.
45 *
46 *   *****
47 *
48 * VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !           !           !
53 *
54 *   *****
55 * MODIFICATIONS:
56 *   ! date !   auteur !           but
57 *   -----
58 *
59 *****/
60
61
62 // La classe Tableau_3D permet de declarer des tableaux a trois dimensions dont
63 // les composantes sont de type double.
64 // Les composantes d'une instance de cette classe sont stockees en ligne et la memoire
65 // est allouee dynamiquement.
66
67
68 #ifndef TABLEAU_3D_H
69 #define TABLEAU_3D_H
70
71
72 #include <iostream>
73 #include <stdlib.h>
74 #include "Sortie.h"
75
76
77 /// @addtogroup Les_Tableaux_generiques
78 /// @{
79 ///
80
81
82 class Tableau_3D
83 {
84
85

```



```

86     public :
87
88
89         // CONSTRUCTEURS :
90
91         // Constructeur par défaut
92         Tableau_3D ();
93
94         // Constructeur utile quand les trois dimensions ainsi qu'une
95         // eventuelle valeur d'initialisation sont connues
96         Tableau_3D (int d1,int d2,int d3,double val=0.0);
97
98         // Constructeur de copie
99         Tableau_3D (const Tableau_3D& tab);
100
101
102         // DESTRUCTEUR :
103
104         ~Tableau_3D ();
105
106
107         // METHODES :
108
109         inline void Affiche () const
110         // Affiche les donnees du tableau a trois dimensions
111         {
112
113             cout << "Tableau 3D\n";
114             cout << "Dimension 1 : " << Taille1() << " .\n";
115             cout << "Dimension 2 : " << Taille2() << " .\n";
116             cout << "Dimension 3 : " << Taille3() << " .\n";
117             cout << "Composante(s) :\n{\n";
118             for (int i=0;i<dim1;i++)
119             {
120                 cout << "\t[ ";
121                 for (int j=0;j<dim2;j++)
122                 {
123                     cout << "[ ";
124                     for (int k=0;k<dim3;k++)
125                     {
126                         cout << *(t+dim3*dim2*i+dim3*j+k) << " ";
127                     };
128                     cout << "]" ";
129                 };
130                 cout << "]" \n";
131             };
132             cout << "}\n\n";
133
134         };
135
136         inline int Taille1 () const
137         // Retourne la taille de la premiere dimension
138         {
139
140             return dim1;
141
142         };
143
144         inline int Taille2 () const
145         // Retourne la taille de la deuxieme dimension
146         {
147
148             return dim2;
149
150         };
151
152         inline int Taille3 () const
153         // Retourne la taille de la troisieme dimension
154         {
155
156             return dim3;
157
158         };
159
160         inline double& operator() (int i,int j,int k)
161         // Retourne la ieme jieme kieme composante
162         {
163
164             if ( (i<1) || (i>dim1) || (j<1) || (j>dim2) || (k<1) || (k>dim3) )
165             {
166                 cout << "\nErreur : composante inexistante !\n";
167                 cout << "TABLEAU_3D::OPERATOR[](int ,int ,int ) \n";
168                 Sortie(1);
169             };
170             return *(t+dim3*dim2*(i-1)+dim3*(j-1)+(k-1));
171
172         };

```

```

173
174 // Initialisation des composantes du tableau
175 void Initialise (double val=0.0);
176
177 // Desallocation des composantes du tableau
178 void Libere ();
179
180 // Surcharge de l'operateur = : egalite entre deux tableaux a trois dimensions
181 Tableau_3D& operator= (const Tableau_3D& tab);
182
183
184 protected :
185
186
187 int dim1; // dimension dans la premiere direction
188 int dim2; // dimension dans la seconde direction
189 int dim3; // dimension dans la troisieme direction
190 double* t; // tableau des composantes
191
192
193 };
194 /// @} // end of group
195
196
197 #endif

```

## 7.442 Tableau\_4D.h

```

1 // FICHER : Tableau_4D.h
2 // CLASSE : Tableau_4D
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *      DATE:      23/01/97
35 *
36 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:     Herezh++
39 *
40 *
41 * La classe Tableau_4D permet de declarer des tableaux a quatre dimensions dont
42 * les composantes sont de type double.
43 * Les composantes d'une instance de cette classe sont stockees en ligne et la memoire
44 * est allouee dynamiquement.
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *
50 *      ! date !   auteur !           but
51 *      -----
52 *      !           !           !
53 *
54 *      *****
55 *
56 *      MODIFICATIONS:
57 *      ! date !   auteur !           but
58 *      -----
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
67 *
68 *
69 *
70 *
71 *
72 *
73 *
74 *
75 *
76 *
77 *
78 *
79 *
80 *
81 *
82 *
83 *
84 *
85 *
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *
118 *
119 *
120 *
121 *
122 *
123 *
124 *
125 *
126 *
127 *
128 *
129 *
130 *
131 *
132 *
133 *
134 *
135 *
136 *
137 *
138 *
139 *
140 *
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 *
151 *
152 *
153 *
154 *
155 *
156 *
157 *
158 *
159 *
160 *
161 *
162 *
163 *
164 *
165 *
166 *
167 *
168 *
169 *
170 *
171 *
172 *
173 *
174 *
175 *
176 *
177 *
178 *
179 *
180 *
181 *
182 *
183 *
184 *
185 *
186 *
187 *
188 *
189 *
190 *
191 *
192 *
193 *
194 *
195 *
196 *
197 *
198 *
199 *
200 *
201 *
202 *
203 *
204 *
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 *
221 *
222 *
223 *
224 *
225 *
226 *
227 *
228 *
229 *
230 *
231 *
232 *
233 *
234 *
235 *
236 *
237 *
238 *
239 *
240 *
241 *
242 *
243 *
244 *
245 *
246 *
247 *
248 *
249 *
250 *
251 *
252 *
253 *
254 *
255 *
256 *
257 *
258 *
259 *
260 *
261 *
262 *
263 *
264 *
265 *
266 *
267 *
268 *
269 *
270 *
271 *
272 *
273 *
274 *
275 *
276 *
277 *
278 *
279 *
280 *
281 *
282 *
283 *
284 *
285 *
286 *
287 *
288 *
289 *
290 *
291 *
292 *
293 *
294 *
295 *
296 *
297 *
298 *
299 *
300 *
301 *
302 *
303 *
304 *
305 *
306 *
307 *
308 *
309 *
310 *
311 *
312 *
313 *
314 *
315 *
316 *
317 *
318 *
319 *
320 *
321 *
322 *
323 *
324 *
325 *
326 *
327 *
328 *
329 *
330 *
331 *
332 *
333 *
334 *
335 *
336 *
337 *
338 *
339 *
340 *
341 *
342 *
343 *
344 *
345 *
346 *
347 *
348 *
349 *
350 *
351 *
352 *
353 *
354 *
355 *
356 *
357 *
358 *
359 *
360 *
361 *
362 *
363 *
364 *
365 *
366 *
367 *
368 *
369 *
370 *
371 *
372 *
373 *
374 *
375 *
376 *
377 *
378 *
379 *
380 *
381 *
382 *
383 *
384 *
385 *
386 *
387 *
388 *
389 *
390 *
391 *
392 *
393 *
394 *
395 *
396 *
397 *
398 *
399 *
400 *
401 *
402 *
403 *
404 *
405 *
406 *
407 *
408 *
409 *
410 *
411 *
412 *
413 *
414 *
415 *
416 *
417 *
418 *
419 *
420 *
421 *
422 *
423 *
424 *
425 *
426 *
427 *
428 *
429 *
430 *
431 *
432 *
433 *
434 *
435 *
436 *
437 *
438 *
439 *
440 *
441 *
442 *
443 *
444 *
445 *
446 *
447 *
448 *
449 *
450 *
451 *
452 *
453 *
454 *
455 *
456 *
457 *
458 *
459 *
460 *
461 *
462 *
463 *
464 *
465 *
466 *
467 *
468 *
469 *
470 *
471 *
472 *
473 *
474 *
475 *
476 *
477 *
478 *
479 *
480 *
481 *
482 *
483 *
484 *
485 *
486 *
487 *
488 *
489 *
490 *
491 *
492 *
493 *
494 *
495 *
496 *
497 *
498 *
499 *
500 *
501 *
502 *
503 *
504 *
505 *
506 *
507 *
508 *
509 *
510 *
511 *
512 *
513 *
514 *
515 *
516 *
517 *
518 *
519 *
520 *
521 *
522 *
523 *
524 *
525 *
526 *
527 *
528 *
529 *
530 *
531 *
532 *
533 *
534 *
535 *
536 *
537 *
538 *
539 *
540 *
541 *
542 *
543 *
544 *
545 *
546 *
547 *
548 *
549 *
550 *
551 *
552 *
553 *
554 *
555 *
556 *
557 *
558 *
559 *
560 *
561 *
562 *
563 *
564 *
565 *
566 *
567 *
568 *
569 *
570 *
571 *
572 *
573 *
574 *
575 *
576 *
577 *
578 *
579 *
580 *
581 *
582 *
583 *
584 *
585 *
586 *
587 *
588 *
589 *
590 *
591 *
592 *
593 *
594 *
595 *
596 *
597 *
598 *
599 *
600 *
601 *
602 *
603 *
604 *
605 *
606 *
607 *
608 *
609 *
610 *
611 *
612 *
613 *
614 *
615 *
616 *
617 *
618 *
619 *
620 *
621 *
622 *
623 *
624 *
625 *
626 *
627 *
628 *
629 *
630 *
631 *
632 *
633 *
634 *
635 *
636 *
637 *
638 *
639 *
640 *
641 *
642 *
643 *
644 *
645 *
646 *
647 *
648 *
649 *
650 *
651 *
652 *
653 *
654 *
655 *
656 *
657 *
658 *
659 *
660 *
661 *
662 *
663 *
664 *
665 *
666 *
667 *
668 *
669 *
670 *
671 *
672 *
673 *
674 *
675 *
676 *
677 *
678 *
679 *
680 *
681 *
682 *
683 *
684 *
685 *
686 *
687 *
688 *
689 *
690 *
691 *
692 *
693 *
694 *
695 *
696 *
697 *
698 *
699 *
700 *
701 *
702 *
703 *
704 *
705 *
706 *
707 *
708 *
709 *
710 *
711 *
712 *
713 *
714 *
715 *
716 *
717 *
718 *
719 *
720 *
721 *
722 *
723 *
724 *
725 *
726 *
727 *
728 *
729 *
730 *
731 *
732 *
733 *
734 *
735 *
736 *
737 *
738 *
739 *
740 *
741 *
742 *
743 *
744 *
745 *
746 *
747 *
748 *
749 *
750 *
751 *
752 *
753 *
754 *
755 *
756 *
757 *
758 *
759 *
760 *
761 *
762 *
763 *
764 *
765 *
766 *
767 *
768 *
769 *
770 *
771 *
772 *
773 *
774 *
775 *
776 *
777 *
778 *
779 *
780 *
781 *
782 *
783 *
784 *
785 *
786 *
787 *
788 *
789 *
790 *
791 *
792 *
793 *
794 *
795 *
796 *
797 *
798 *
799 *
800 *
801 *
802 *
803 *
804 *
805 *
806 *
807 *
808 *
809 *
810 *
811 *
812 *
813 *
814 *
815 *
816 *
817 *
818 *
819 *
820 *
821 *
822 *
823 *
824 *
825 *
826 *
827 *
828 *
829 *
830 *
831 *
832 *
833 *
834 *
835 *
836 *
837 *
838 *
839 *
840 *
841 *
842 *
843 *
844 *
845 *
846 *
847 *
848 *
849 *
850 *
851 *
852 *
853 *
854 *
855 *
856 *
857 *
858 *
859 *
860 *
861 *
862 *
863 *
864 *
865 *
866 *
867 *
868 *
869 *
870 *
871 *
872 *
873 *
874 *
875 *
876 *
877 *
878 *
879 *
880 *
881 *
882 *
883 *
884 *
885 *
886 *
887 *
888 *
889 *
890 *
891 *
892 *
893 *
894 *
895 *
896 *
897 *
898 *
899 *
900 *
901 *
902 *
903 *
904 *
905 *
906 *
907 *
908 *
909 *
910 *
911 *
912 *
913 *
914 *
915 *
916 *
917 *
918 *
919 *
920 *
921 *
922 *
923 *
924 *
925 *
926 *
927 *
928 *
929 *
930 *
931 *
932 *
933 *
934 *
935 *
936 *
937 *
938 *
939 *
940 *
941 *
942 *
943 *
944 *
945 *
946 *
947 *
948 *
949 *
950 *
951 *
952 *
953 *
954 *
955 *
956 *
957 *
958 *
959 *
960 *
961 *
962 *
963 *
964 *
965 *
966 *
967 *
968 *
969 *
970 *
971 *
972 *
973 *
974 *
975 *
976 *
977 *
978 *
979 *
980 *
981 *
982 *
983 *
984 *
985 *
986 *
987 *
988 *
989 *
990 *
991 *
992 *
993 *
994 *
995 *
996 *
997 *
998 *
999 *
1000 *

```

```

58 *                                     $ *
59 *****/
60
61
62 // La classe Tableau_4D permet de declarer des tableaux a quatre dimensions dont
63 // les composantes sont de type double.
64 // Les composantes d'une instance de cette classe sont stockees en ligne et la memoire
65 // est allouee dynamiquement.
66
67
68 #ifndef TABLEAU_4D_H
69 #define TABLEAU_4D_H
70
71
72 #include <iostream>
73 #include <stdlib.h>
74 #include "Sortie.h"
75
76
77 /// @addtogroup Les_Tableaux_generiques
78 /// @{
79 ///
80
81
82 class Tableau_4D
83 {
84
85
86     public :
87
88
89         // CONSTRUCTEURS :
90
91         // Constructeur par default
92         Tableau_4D ();
93
94         // Constructeur utile quand les quatre dimensions ainsi qu'une
95         // eventuelle valeur d'initialisation sont connues
96         Tableau_4D (int d1,int d2,int d3,int d4,double val=0.0);
97
98         // Constructeur de copie
99         Tableau_4D (const Tableau_4D& tab);
100
101
102         // DESTRUCTEUR :
103
104         ~Tableau_4D ();
105
106
107         // METHODES :
108
109         inline void Affiche ()
110         // Affiche les donnees du tableau a trois dimensions
111         {
112
113             cout << "Tableau 4D\n";
114             cout << "Dimension 1 : " << Taille1() << " .\n";
115             cout << "Dimension 2 : " << Taille2() << " .\n";
116             cout << "Dimension 3 : " << Taille3() << " .\n";
117             cout << "Dimension 4 : " << Taille4() << " .\n";
118             cout << "Composante(s) :\n{\n";
119             for (int i=0;i<dim1;i++)
120             {
121                 cout << "( ";
122                 for (int j=0;j<dim2;j++)
123                 {
124                     cout << "\t[ ";
125                     for (int k=0;k<dim3;k++)
126                     {
127                         cout << "[ ";
128                         for (int l=0;l<dim4;l++)
129                         {
130                             cout << *(t+dim4*dim3*dim2*i+dim4*dim3*j+dim4*k+l) << " ";
131                         };
132                         cout << " ]";
133                     };
134                     cout << " ] \n";
135                 };
136                 cout << ") \n";
137             };
138             cout << "}\n\n";
139
140         };
141
142         inline int Taille1 ()
143         // Retourne la taille de la premiere dimension
144         {

```

```

145
146     return dim1;
147
148     };
149
150     inline int Taille2 ()
151     // Retourne la taille de la deuxieme dimension
152     {
153
154         return dim2;
155
156     };
157
158     inline int Taille3 ()
159     // Retourne la taille de la troisieme dimension
160     {
161
162         return dim3;
163
164     };
165
166     inline int Taille4 ()
167     // Retourne la taille de la quatrieme dimension
168     {
169
170         return dim4;
171
172     };
173
174     inline double& operator() (int i,int j,int k,int l)
175     // Retourne la ieme jieme kieme lieme composante
176     {
177
178         if ( (i<1) || (i>dim1) || (j<1) || (j>dim2)
179             || (k<1) || (k>dim3) || (l<1) || (l>dim4) )
180         {
181             cout << "\nErreur : composante inexistante !\n";
182             cout << "TABLEAU_4D::OPERATOR()(int ,int ,int ,int ) \n";
183             Sortie(1);
184         };
185         return *(t+dim4*dim3*dim2*(i-1)+dim4*dim3*(j-1)+dim4*(k-1)+(l-1));
186
187     };
188
189     // Initialisation des composantes du tableau
190     void Initialise (double val=0.0);
191
192     // Desallocation des composantes du tableau
193     void Libere ();
194
195     // Surcharge de l'operateur = : egalite entre deux tableaux a quatre dimensions
196     Tableau_4D& operator= (const Tableau_4D& tab);
197
198
199     protected :
200
201
202     int dim1; // dimension dans la premiere direction
203     int dim2; // dimension dans la seconde direction
204     int dim3; // dimension dans la troisieme direction
205     int dim4; // dimension dans la quatrieme direction
206     double* t; // tableau des composantes
207
208
209 };
210 /// @} // end of group
211
212
213 #endif

```

## 7.443 Tableau\_5D.h

```

1 // FICHER : Tableau_5D.h
2 // CLASSE : Tableau_5D
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)

```

```

14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *   DATE:      23/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 * *****/
41 * La classe Tableau_5D permet de declarer des tableaux a cinq dimensions dont
42 * les composantes sont de type double.
43 * Les composantes d'une instance de cette classe sont stockees en ligne et la memoire
44 * est allouee dynamiquement.
45 *
46 *   *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !       but
51 *   -----
52 *   !       !       !
53 *
54 *   *****
55 *   MODIFICATIONS:
56 *   ! date !   auteur !       but
57 *   -----
58 *
59 * *****/
60
61
62 // La classe Tableau_5D permet de declarer des tableaux a cinq dimensions dont
63 // les composantes sont de type double.
64 // Les composantes d'une instance de cette classe sont stockees en ligne et la memoire
65 // est allouee dynamiquement.
66
67
68 #ifndef TABLEAU_5D_H
69 #define TABLEAU_5D_H
70
71
72 #include <iostream>
73 #include <stdlib.h>
74 #include "Sortie.h"
75
76
77 /// @addtogroup Les_Tableaux_generiques
78 /// @{
79 ///
80
81
82 class Tableau_5D
83 {
84
85
86     public :
87
88
89         // CONSTRUCTEURS :
90
91         // Constructeur par default
92         Tableau_5D ();
93
94         // Constructeur utile quand les cinq dimensions ainsi qu'une
95         // eventuelle valeur d'initialisation sont connues
96         Tableau_5D (int d1,int d2,int d3,int d4,int d5,double val=0.0);
97
98         // Constructeur de copie
99         Tableau_5D (const Tableau_5D& tab);
100

```

```

101
102 // DESTRUCTEUR :
103
104 ~Tableau_5D ();
105
106
107 // METHODES :
108
109 inline void Affiche ()
110 // Affiche les donnees du tableau a cinq dimensions
111 {
112
113     cout << "Tableau 5D\n";
114     cout << "Dimension 1 : " << Taille1() << " .\n";
115     cout << "Dimension 2 : " << Taille2() << " .\n";
116     cout << "Dimension 3 : " << Taille3() << " .\n";
117     cout << "Dimension 4 : " << Taille4() << " .\n";
118     cout << "Dimension 5 : " << Taille5() << " .\n";
119     cout << "Composante(s) :\n{\n";
120     for (int i=0;i<dim1;i++)
121     {
122         cout << "( ";
123         for (int j=0;j<dim2;j++)
124         {
125             cout << "\t[ ";
126             for (int k=0;k<dim3;k++)
127             {
128                 cout << "[ ";
129                 for (int l=0;l<dim4;l++)
130                 {
131                     cout << "[ ";
132                     for (int m=0;m<dim5;m++)
133                     {
134                         cout << *(t+dim5*dim4*dim3*dim2*i+dim5*dim4*dim3*j
135                             +dim5*dim4*k+dim5*1+m) << " ";
136                     };
137                     cout << " ]";
138                 };
139                 cout << " ]";
140             };
141             cout << "] \n";
142         };
143         cout << ") \n";
144     };
145     cout << ")\n\n";
146
147 };
148
149 inline int Taille1 ()
150 // Retourne la taille de la premiere dimension
151 {
152
153     return dim1;
154
155 };
156
157 inline int Taille2 ()
158 // Retourne la taille de la deuxieme dimension
159 {
160
161     return dim2;
162
163 };
164
165 inline int Taille3 ()
166 // Retourne la taille de la troisieme dimension
167 {
168
169     return dim3;
170
171 };
172
173 inline int Taille4 ()
174 // Retourne la taille de la quatrieme dimension
175 {
176
177     return dim4;
178
179 };
180
181 inline int Taille5 ()
182 // Retourne la taille de la cinquieme dimension
183 {
184
185     return dim5;
186
187 };

```

```

188
189     inline double& operator() (int i,int j,int k,int l,int m)
190     // Retourne la ieme jieme kieme lieme mieme composante
191     {
192
193         if ( (i<l) || (i>dim1) || (j<l) || (j>dim2)
194             || (k<l) || (k>dim3) || (l<l) || (l>dim4) || (m<l) || (m>dim5) )
195         {
196             cout << "\nErreur : composante inexistante !\n";
197             cout << "TABLEAU_3D::OPERATOR[](int ,int ,int ) \n";
198             Sortie(l);
199         };
200         return *(t+dim5*dim4*dim3*dim2*(i-1)+dim5*dim4*dim3*(j-1)+dim5*dim4*(k-1)
201                +dim5*(l-1)+(m-1));
202
203     };
204
205     // Initialisation des composantes du tableau
206     void Initialise (double val=0.0);
207
208     // Desallocation des composantes du tableau
209     void Libere ();
210
211     // Surcharge de l'operateur = : egalite entre deux tableaux a cinq dimensions
212     Tableau_5D& operator= (const Tableau_5D& tab);
213
214
215     protected :
216
217
218         int dim1; // dimension dans la premiere direction
219         int dim2; // dimension dans la seconde direction
220         int dim3; // dimension dans la troisieme direction
221         int dim4; // dimension dans la quatrieme direction
222         int dim5; // dimension dans la cinquieme direction
223         double* t; // tableau des composantes
224
225
226 };
227 /// @} // end of group
228
229
230 #endif

```

## 7.444 Tableau\_double.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *
32 *   DATE:           23/01/97
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 *
39 * définition d'un tableau de réel de dimension quelconque. Dans le cas ou la dimension est comprise

```

```

40 * entre 1 et 20, on utilise des listes de réels pour allouer de la mémoire et pour la désallouer, ce qui
41 * permet d'être beaucoup plus rapide que l'opérateur classique new.
42 * la construction de la classe s'appuie sur une spécialisation d'une classe template.
43 *
44 *      *
45 * VERIFICATION:
46 *
47 * ! date ! auteur ! but !
48 * -----
49 * ! ! ! !
50 *
51 *      $
52 *      *
53 * MODIFICATIONS:
54 *
55 * ! date ! auteur ! but !
56 * -----
57 *
58 *      $
59 *
60 *****/
61
62
63 // définition d'un tableau de réel de dimension quelconque. Dans le cas ou la dimension est comprise
64 // entre 1 et 20, on utilise des listes de réels pour allouer de la mémoire et pour la désallouer, ce qui
65 // permet d'être beaucoup plus rapide que l'opérateur classique new.
66
67 // la construction de la classe s'appuie sur une spécialisation d'une classe template.
68
69 #ifndef TABLEAU_DOUBLE_H
70 #define TABLEAU_DOUBLE_H
71
72 #include "Tableau_T.h"
73 #include "PtTabRel.h"
74
75 /// @addtogroup Les_Tableaux_generiques
76 /// @
77 ///
78
79
80 class Tableau_double : public Tableau<double>
81 {
82 public :
83     // VARIABLES PUBLIQUES :
84
85     // CONSTRUCTEURS :
86     // CONSTRUCTEURS :
87
88     // Constructeur par défaut
89     Tableau_double () :
90         Tableau()
91     {};
92
93     // Constructeur fonction de la taille du tableau
94     Tableau_double (int nb);
95
96     // Constructeur fonction de la taille du tableau et d'une
97     // valeur d'initialisation pour les composantes
98     Tableau_double (int nb,double val);
99
100     // Constructeur fonction d'un tableau de double
101     Tableau_double (int nb,double* tab);
102
103     // Constructeur de copie
104     Tableau_double (const Tableau_double & tab);
105
106     // DESTRUCTEUR :
107
108     ~Tableau_double ();
109
110
111     // METHODES :
112
113
114     // Surcharge de l'opérateur d'affectation =
115     Tableau_double& operator= (const Tableau_double& tab);
116
117     // Change la taille du tableau (la nouvelle taille est n)
118     void Change_taille (int n);
119
120     // Enleve la ieme composante du tableau
121     void Enleve (int i);
122
123     // Enleve toutes les composantes du tableau dont la valeur est val
124     void Enleve_val (double val);
125
126     // Permet de desallouer l'ensemble des elements du tableau

```



```

127     void Libere ();
128
129     protected :
130         // allocator dans la liste de data
131         // a priori on utilise un pointeur sur une liste de 3 réels, mais dans
132         // le constructeur et dans le destructeur, on effectue un cast vers le
133         // bon allocator qui normalement à la même dimension
134         ReelsPointe* iapointe; // pointeur générique
135
136
137
138 };
139 ///< @} // end of group
140
141 #endif

```

## 7.445 Tableau\_T.h

```

1 // FICHER : Tableau_T.h
2 // CLASSE : Tableau_T
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32
33 /*****
34 *   DATE:      23/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *****/
41 *   BUT:      La classe Tableau_T permet de declarer des tableaux de
42 *             longueur determinee et dont les composantes sont du type T fixe
43 *             a la declaration d'un objet de cette classe. L'operateur = doit
44 *             etre surcharge pour les objets de la classe T.
45 *             L'allocation memoire est dynamique et permet de declarer des
46 *             tableaux de tailles indifferentes.
47 *
48 *             *****
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !
55 *
56 *             *****
57 *
58 *   MODIFICATIONS:
59 *   ! date !   auteur !           but
60 *   -----
61 *
62 *****/
63 #ifndef TABLEAU_T_H
64 #define TABLEAU_T_H
65
66
67 // #include "Debug.h"

```

```

68 #include <iostream>
69 #include <stdlib.h>
70
71 using namespace std; //introduces namespace std
72
73 #include "Sortie.h"
74 #include <iomanip>
75 #include <string>
76 #include "ParaGlob.h"
77
78 /** @defgroup Les_Tableaux_generiques
79 *
80 *   BUT:   groupe des tableaux génériques: typiquement des templates, mais pas seulement
81 *
82 *
83 * \author   Gérard Rio
84 * \version  1.0
85 * \date    23/01/97
86 * \brief   groupe des tableaux génériques: typiquement des templates, mais pas seulement
87 *
88 */
89
90 /// @addtogroup Les_Tableaux_generiques
91 /// @{
92 ///
93
94
95 template <class T>
96 class Tableau
97 {
98
99     // surcharge de l'operator d'écriture
100     friend ostream & operator << (ostream & sort, const Tableau<T> & tab)
101     { // tout d'abord un indicateur donnant le type
102       sort << "Tableau :taille= " << tab.taille << " ";
103       for (int i=0;i<tab.taille;i++)
104         { sort << setprecision(ParaGlob::NbdigdoCA()) << tab.t[i] ; sort << " "; }
105       // sort << "\n";
106       return sort;
107     }
108
109     // affichage du premier élément pour les messages d'erreur
110     friend void Affiche_premier_elem(ostream &sort , const Tableau<T> & tab )
111     { if (tab.taille > 0)
112       {sort << "\n le premier element du tableau pour info: ";
113         sort << tab.t[0] ;
114         sort << endl;};
115     }
116
117
118     public :
119
120
121         // CONSTRUCTEURS :
122
123         // Constructeur par défaut
124         Tableau ();
125
126         // Constructeur fonction de la taille du tableau
127         Tableau (int nb);
128
129         // Constructeur fonction de la taille du tableau et d'une
130         // valeur d'initialisation pour les composantes
131         Tableau (int nb,T val);
132
133         // Constructeur fonction de la taille du tableau et d'une
134         // référence: pas facile à faire car ensuite on manipule des pointeurs de références !!
135         // bref on ne sait pas ce que l'on fait , dans ce cas il vaut mieux utiliser
136         // un tableau de pointeur de la grandeur en question
137
138         // Constructeur fonction d'un tableau de composantes de type T
139         Tableau (int nb,T* tab);
140
141         // Constructeur de copie
142         Tableau (const Tableau<T> & tab);
143
144
145         // DESTRUCTEUR :
146
147         ~Tableau ();
148
149
150         // METHODES :
151
152         inline int Taille () const
153         // Retourne la taille du tableau
154         { return taille; };

```

```

155
156 // Retourne la ieme composante du tableau : acces en lecture et ecriture
157 T& operator() (int i) const;
158
159 // cas particulier ou l'on ne veut pas modifier les valeurs
160 const T val_const(int i) const;
161 const T& ref_const(int i) const;
162
163 // Surcharge de l'operateur !=
164 // Renvoie 1 si les deux tableaux ne sont pas egaux
165 // Renvoie 0 sinon
166 int operator!=(const Tableau<T>& tab) const;
167
168 // Surcharge de l'operateur ==
169 int operator==(const Tableau<T>& tab) const ;
170
171 // Surcharge de l'operateur d'affectation =
172 Tableau<T>& operator=(const Tableau<T>& tab);
173
174 // Change la taille du tableau (la nouvelle taille est n)
175 void Change_taille (int n);
176 // Change la taille du tableau (la nouvelle taille est n)
177 // et initialisation de toutes les valeurs, anciennes et nouvelles à tb
178 void Change_taille (int n,const T& tb);
179
180 // Enleve la ieme composante du tableau
181 void Enleve (int i);
182
183 // Enleve toutes les composantes du tableau dont la valeur est val
184 void Enleve_val (T val);
185
186 // Permet de desallouer l'ensemble des elements du tableau
187 void Libere ();
188
189 // initialisation d'un tableau à partir d'une liste
190 // premier du tableau == premier de la liste
191 void Init_from_list(const list<T> & liste );
192
193 // opération inverse: remplissage d'une liste à partir du tableau
194 // premier de la liste == premier du tableau
195 void Init_list(list<T> & liste );
196
197 // opération de lecture sur un flot d'entrée
198 // les données sont le type puis la dimension puis les datas
199 istream & Entree(istream &);
200
201 // les données sont le type puis la dimension puis les datas
202 // ici il n'y a pas redimensionnement du tableau si la taille est suffisante
203 istream & Entree_sans_redim(istream &);
204
205 // lectures sur le flot si la taille du tableau est identique ou supérieure à celle lue
206 // retour d'un pointeur nulle
207 // sinon, si la taille lue est supérieure à la taille du tableau actuellement existant
208 // il y a création d'un nouveau tableau de même type avec la taille adéquate
209 // et retour d'un pointeur sur le nouveau tableau
210 Tableau<T> * New_en_lecture_si_taille_superieur_a_lire(istream &);
211
212 // opération d'écriture non formatée
213 // les données sont le type la dimension puis les datas, chacun
214 // séparé par un espace
215 ostream & Sortir(ostream &) const ;
216 // idem mais sans retour chariot à la fin
217 ostream & Sortir_sansRet(ostream &) const;
218
219 // test si la valeur passée en argument appartient au tableau
220 // ramène 0 si ne contient pas
221 // ramène l'indice dans le tableau de la grandeur si appartenance
222 int Contient(const T& e) const;
223
224 // initialisation avec la valeur passée en argument
225 bool Inita(const T& e) ;
226
227
228 protected :
229
230     int taille; // taille du tableau
231     T* t; // pointeur sur les composantes du tableau
232
233
234 };
235 /// @} // end of group
236
237 // cas de la surcharge de la lecture : il y a deux cas à considérer
238 // 1) cas d'élément constant : dans ce cas la lecture est une erreur
239 // 2) cas d'élément non constant : lecture permise
240 // les deux fonctions sont en dehors de la classe car sinon dans la classe il y a pris en compte
241 // du fait d'avoir des éléments constants ou non ce qui n'a pas l'air évident à traité sinon par

```

```

242 // deux classes différentes
243
244
245 template <class T>
246 // surcharge de l'operator de lecture d'élément non constant
247 istream & operator > (istream & entree, Tableau<T> & tab)
248 { // vérification du type
249     string type;
250     entree > type;
251     if (type != "Tableau")
252         {Sortie (1);
253         return entree;
254         };
255     int dim; entree > type > dim;
256     int tabtaille = tab.Taille();
257     if (dim != tabtaille)
258         tab.Change_taille(dim);
259     for (int i=1;i<=dim;i++)
260         entree > tab(i);
261     return entree;
262 }
263
264 //===== def des differents elements =====
265
266
267 template <class T>
268 inline Tableau<T>::Tableau ()
269 // Constructeur par défaut
270 {   taille=0;
271     t=NULL;
272 }
273
274
275 template <class T>
276 inline Tableau<T>::Tableau (int nb)
277 // Constructeur devant etre utilise quand la dimension nb du tableau est connu
278 {
279     #ifndef MISE_AU_POINT
280     if ( nb<0 )
281         // cas ou la taille selectionnee est negative
282         {   cout << "\nErreur : taille invalide !\n";
283             cout << "TABLEAU_T::TABLEAU_T(int ) \n";
284             Sortie(1);
285         };
286     #endif
287     if ( nb==0 )
288         // cas ou la taille selectionnee est nulle : initialisation identique a
289         // l'appel du constructeur par défaut
290         {   taille=0;
291             t=NULL;
292         }
293     else
294         // autres cas
295         {   taille=nb;
296             t=new T [taille]; // allocation de la place memoire
297             T* ptr=t;
298             for (int i=0;i<taille;i++)
299                 // initialisation des composantes a l'aide de l'attribut statique défaut
300                 // (*ptr++)=defaut;
301                 (*ptr++)= T();
302         };
303 }
304
305 template <class T>
306 inline Tableau<T>::Tableau (int nb, T val)
307 // Constructeur permettant de creer un tableau dont toutes les composantes
308 // sont egales a val
309 // N.B: La valeur de l'attribut défaut est fixee a val
310 { // initialisation de l'element défaut
311     #ifndef MISE_AU_POINT
312     if ( nb<0 )
313         // cas ou la taille selectionnee est negative
314         {   cout << "\nErreur : taille invalide !\n";
315             cout << "TABLEAU_T::TABLEAU_T(int ,T ) \n";
316             Sortie(1);
317         };
318     #endif
319     if ( nb==0 )
320         // cas ou la taille selectionnee est nulle : initialisation identique a
321         // l'appel du constructeur par défaut
322         {   taille=0;
323             t=NULL;
324         }
325     else
326         // autres cas
327         {   taille=nb;
328             t=new T [taille]; // allocation de la place memoire

```

```

329     T* ptr=t;
330     for (int i=0;i<taille;i++)
331         // initialisation des composantes a val
332         (*ptr++)=val;
333     };
334 }
335
336 template <class T>
337 inline Tableau<T>::Tableau (int nb,T* tab)
338 // Constructeur permettant de creer un tableau de taille nb dont les composantes
339 // sont initialisees a l'aide des valeurs de tab
340 // N.B.: tab doit avoir au moins nb composantes (aucun test n'est realise pour le
341 // verifier !!!)
342 {
343     #ifdef MISE_AU_POINT
344     if ( nb<0 )
345         // cas ou la taille selectionnee est negative
346         { cout << "\nErreur : taille invalide !\n";
347           cout << "TABLEAU_T::TABLEAU_T(int ,T* ) \n";
348           Sortie(1);
349         };
350     #endif
351     if ( nb==0 )
352         // cas ou la taille selectionnee est nulle : initialisation identique a
353         // l'appel du constructeur par default
354         {
355             taille=0;
356             t=NULL;
357         }
358     else
359         // autres cas
360         {
361             taille=nb;
362             t=new T [taille]; // allocation de la place memoire
363             T* ptr1=t;
364             T* ptr2=tab;
365             for (int i=0;i<taille;i++)
366                 // copie des composantes du tableau tab dans le tableau declare
367                 (*ptr1++)=(*ptr2++);
368         };
369 }
370
371 template <class T>
372 inline Tableau<T>::Tableau (const Tableau<T>& tab)
373 // Constructeur de copie
374 { if ( tab.taille==0 )
375     // cas ou le tableau copie est vide : initialisation identique a l'appel
376     // du constructeur par default
377     { t=NULL;
378       taille=0;
379     }
380     else
381         // autres cas
382         { t=new T [tab.taille]; // allocation de la place memoire
383           T* ptr1=t;
384           T* ptr2=tab.t;
385           for (int i=0;i<tab.taille;i++)
386               // copie des composantes du tableau tab dans le tableau declare
387               (*ptr1++)=(*ptr2++);
388           taille=tab.taille;
389         };
390 }
391
392 template <class T>
393 inline Tableau<T>::~Tableau ()
394 // Destructeur (N.B. : Apres l'appel de ce destructeur le tableau est identique
395 // a ce qu'il aurait ete a la suite d'un appel du constructeur par default)
396 { Libere(); }
397
398 template <class T> void
399 inline Tableau<T>::Change_taille (int n)
400 // Permet de modifier la taille du tableau
401 // N.B. : Si la nouvelle taille est superieure a l'ancienne alors le tableau est
402 // complete par default
403 {
404     #ifdef MISE_AU_POINT
405     if ( n<0 )
406         // cas ou la taille selectionnee est negative
407         { cout << "\nErreur : taille invalide !\n";
408           cout << "TABLEAU_T::CHANGE_TAILLE(int ) \n";
409           cout << " NB: taille actuelle = " << taille << " ";
410           Sortie(1);
411         }
412     #endif
413     if ( n == taille ) // taille identique , on ne fait rien
414         return;
415     if ( n==0 )

```

```

416 // cas ou la taille selectionnee est nulle : initialisation identique a
417 // l'appel du constructeur par default
418 {
419     taille=0;
420     if ( t!= NULL) delete [] t;
421     t=NULL;
422     return ;
423 };
424 T* tempo;
425 tempo=new T [n];
426 T* ptr1=t;
427 T* ptr2=tempo;
428 if ( n<=taille )
429 // cas ou la taille selectionnee est inferieure a la taille d'origine
430 {
431     for (int i=0;i<n;i++)
432         // copie des composantes du tableau d'origine dans le tableau pointe par tempo
433         (*ptr2++)=(*ptr1++);
434 }
435 else if ( n>taille )
436 // cas ou la taille selectionnee est superieure a la taille d'origine
437 {
438     for (int i=0;i<taille;i++)
439         // copie des composantes du tableau d'origine dans le tableau pointe par tempo
440         (*ptr2++)=(*ptr1++);
441     for (int i=taille;i<n;i++)
442         // initialisation des composantes supplementaires a l'aide de l'element par default
443         /// (*ptr2++)=default;
444         (*ptr2++)=T();
445 };
446 delete [] t; // desallocation du tableau pointe par t
447 t=tempo; // affectation du pointeur t au pointeur tempo
448 taille=n;
449 }
450 }
451
452 template <class T> void
453 inline Tableau<T>::Change_taille (int n,const T& tb)
454 // Change la taille du tableau (la nouvelle taille est n)
455 // et initialisation de toutes les valeurs, anciennes et nouvelles à tb
456 {
457     #ifndef MISE_AU_POINT
458     if ( n<0 )
459         // cas ou la taille selectionnee est negative
460         { cout << "\nErreur : taille invalide !\n";
461           cout << "TABLEAU_T::CHANGE_TAILLE(int ) \n";
462           cout << " NB: taille actuelle = " << taille << " ";
463           Sortie(1);
464         }
465     #endif
466     if ( n == taille ) // taille identique , on ne fait que l'initialisation
467     { T* ptr2= t;
468       for (int i=0;i<n;i++)
469           // copie des composantes à tb
470           (*ptr2++)=tb;
471       return;
472     }
473
474     if ( n==0 )
475     // cas ou la taille selectionnee est nulle : initialisation identique a
476     // l'appel du constructeur par default
477     {
478         taille=0;
479         if ( t!= NULL) delete [] t;
480         t=NULL;
481         return ;
482     };
483     T* tempo;
484     tempo=new T [n];
485     T* ptr2=tempo;
486     if ( n<=taille )
487     // cas ou la taille selectionnee est inferieure a la taille d'origine
488     {
489         for (int i=0;i<n;i++)
490             // copie des composantes du tableau d'origine dans le tableau pointe par tempo
491             (*ptr2++)=tb;
492     }
493     else if ( n>taille )
494     // cas ou la taille selectionnee est superieure a la taille d'origine
495     {
496         for (int i=0;i<n;i++)
497             // copie des composantes à tb
498             (*ptr2++)=tb;
499     };
500     delete [] t; // desallocation du tableau pointe par t
501     t=tempo; // affectation du pointeur t au pointeur tempo
502     taille=n;

```

```

503 }
504
505 template <class T>
506 #ifndef MISE_AU_POINT
507     inline
508 #endif
509 T& Tableau<T>::operator() (int i) const
510     // Retourne la ieme composante du tableau : acces en lecture et ecriture
511     {
512     #ifndef MISE_AU_POINT
513         if ( (i<1) || (i>taille) )
514             { cout << "\nErreur : composante inexistante !, nb demande = " << i << '\n';
515             cout << "T& TABLEAU_T::OPERATOR() (int ) \n";
516             cout << "\n pour info: taille actuelle du tableau = " << taille << " ";
517             Sortie(1);
518             };
519         #endif
520     return t[i-1];
521     }
522
523     // cas particulier ou l'on ne veut pas modifier les valeurs
524 template <class T>
525 #ifndef MISE_AU_POINT
526     inline
527 #endif
528 const T Tableau<T>::val_const(int i) const
529 //const T& Tableau<T>::operator() (int i) const
530     // Retourne la ieme composante du tableau : acces en lecture seulement
531     {
532     #ifndef MISE_AU_POINT
533         if ( (i<1) || (i>taille) )
534             { cout << "\nErreur : composante inexistante !, nb demande = " << i << '\n';
535             cout << "T TABLEAU_T::OPERATOR() (int ) const \n";
536             cout << "\n pour info: taille actuelle du tableau = " << taille << " ";
537             Sortie(1);
538             };
539         #endif
540     T retour = t[i-1];
541     return retour;
542     }
543
544 // Retourne la référence constante à la ieme composante du tableau : acces en lecture seulement
545 template <class T>
546 #ifndef MISE_AU_POINT
547     inline
548 #endif
549 const T& Tableau<T>::ref_const(int i) const
550 //const T& Tableau<T>::operator() (int i) const
551     // Retourne la ieme composante du tableau : acces en lecture seulement
552     {
553     #ifndef MISE_AU_POINT
554         if ( (i<1) || (i>taille) )
555             { cout << "\nErreur : composante inexistante !, nb demande = " << i << '\n';
556             cout << "T TABLEAU_T::OPERATOR() (int ) const \n";
557             cout << "\n pour info: taille actuelle du tableau = " << taille << " ";
558             Sortie(1);
559             };
560         #endif
561     return t[i-1];
562     }
563
564 template <class T>
565 #ifndef MISE_AU_POINT
566     inline
567 #endif
568 int Tableau<T>::operator!=(const Tableau<T>& tab) const
569     // Surcharge de l'operateur !=
570     // Renvoie 1 si les deux tableaux ne sont pas egaux
571     // Renvoie 0 sinon
572     { if ( (*this)==tab ) // test de l'egalite des deux tableaux a l'aide
573         // de l'operateur surcharge ==
574         return 0;
575     else
576         return 1;
577     }
578
579 template <class T> void
580 inline Tableau<T>::Enleve (int i)
581 // Enleve la ieme composante du tableau en decremantant la taille
582 {
583     #ifndef MISE_AU_POINT
584         if ( (i<1) || (i>taille) )
585             {
586             cout << "\nErreur : composante inexistante " << i << " !\n";
587             cout << "TABLEAU_T::ENLEVE(int ) \n";
588             cout << "\n pour info: taille actuelle du tableau = " << taille << " ";
589             Sortie(1);

```

```

590     });
591 #endif
592 T* tempo;
593 tempo=new T [taille-1];
594 T* ptr1=t;
595 T* ptr2=tempo;
596 for (int j=0;j<i-1;j++)
597 // copie des elements du tableau pointe par t dans le tableau pointe par tempo
598 // du 1er au (i-1)eme element
599     (*ptr2++)=(*ptr1++);
600 ptr1++; // saut du ieme element du tableau pointe par t
601 for (int j=i;j<taille;j++)
602 // copie des elements du tableau pointe par t dans le tableau pointe par tempo
603 // du (i+1)eme au dernier element
604     (*ptr2++)=(*ptr1++);
605 delete [] t; // desallocation du tableau pointe par t
606 t=tempo; // affectation du pointeur t au pointeur tempo
607 taille=taille-1;
608 }
609
610 template <class T> void
611 inline Tableau<T>::Enleve_val (T val)
612 // Enleve les composantes du tableau egales a val en decremantant la taille
613 {
614
615     T* tempo;
616     tempo=new T [taille];
617     T* ptr1=t;
618     T* ptr2=tempo;
619     int nouvel_taille=taille;
620     for (int j=0;j<taille;j++)
621 // boucle sur les composantes du tableau
622     {
623         if ( (*ptr1)!=val )
624             // l'objet pointe n'est pas egal a val
625             (*ptr2++)=(*ptr1++);
626         else
627             // l'objet pointe est egal a val, saut de celui-ci
628             {
629                 ptr1++;
630                 nouvel_taille=nouvel_taille-1;
631             };
632     };
633
634 #ifdef MISE_AU_POINT
635     if ( nouvel_taille==taille )
636     {
637         cout << "\nErreur : absence de la composante cherchee !\n";
638         cout << "TABLEAU_T::ENLEVE_VAL(T ) \n";
639         Sortie(1);
640     };
641 #endif
642 delete [] t; // desallocation du tableau pointe par t
643 t=tempo; // affectation du pointeur t au pointeur tempo
644 taille=nouvel_taille;
645
646 }
647
648 template <class T> void
649 inline Tableau<T>::Libere ()
650 // Apres l'appel de cette methode, le tableau est identique a ce qu'il aurait
651 // ete a la suite d'un appel du constructeur par default
652 {
653     if ( taille>0 )
654         delete [] t; // desallocation du tableau pointe par t
655     else
656     {
657         #ifdef MISE_AU_POINT
658         if ( t!=NULL )
659             { cout << "\nErreur de liberation de la place memoire\n";
660               cout << "TABLEAU_T::LIBERE() \n";
661               // Affiche_premier_elem( cout);
662               Sortie(1);
663             }
664         #endif
665     };
666     t=NULL;
667     taille=0;
668 }
669
670 template <class T> void
671 inline Tableau<T>::Init_from_list(const list<T> & liste )
672 // initialisation d'un tableau à partir d'une liste
673 { int tail_liste = liste.size(); // récup de la tail de la liste
674   typename list<T>::const_iterator ili,ilifin=liste.end();
675   Change_taille(tail_liste);
676   int i=0;

```



```

677     for (ili=liste.begin();ili!=ilifin;ili++,i++)
678         t[i] = (*ili);
679 }
680
681 template <class T> void
682 inline Tableau<T>::Init_list(list<T> & liste )
683 // initialisation d'une liste à partir du tableau
684 // premier de la liste == premier du tableau
685 { liste.clear(); // on initialise la liste
686   // on balaie le tableau et on remplit la liste
687   T* ptr1=t;
688   for (int i=0;i<taille;i++)
689       liste.push_back(*ptr1++);
690 }
691
692
693 template <class T>
694 inline Tableau<T> & Tableau<T>::operator= (const Tableau<T> & tab)
695 // Surcharge de l'opérateur = : réalise l'égalité entre deux tableaux de pointeurs
696 {
697     // on essaie d'optimiser un peu
698     // -- si les tableaux sont identiques en taille, on ne réaffecte pas
699     // on ne fait que la recopie
700     if (taille == tab.taille)
701     { // on n'examine que si la taille est non nulle
702         if (taille != 0)
703         { T* ptr1=t;
704           T* ptr2=tab.t;
705           for (int i=0;i<taille;i++)
706               // copie des éléments du tableau se trouvant à droite dans le tableau se trouvant
707               // à gauche du signe d'affectation
708               (*ptr1++)=(*ptr2++);
709         };
710         // et retour
711         return (*this);
712     };
713     // sinon on réajuste les tailles
714     if ( t!=NULL ) Libere(); // cas où le tableau se trouvant à gauche du signe =
715                             // n'est pas vide : désallocation de ce tableau
716     if ( tab.taille==0 )
717         // cas où le tableau se trouvant à droite du signe = est vide : initialisation
718         // identique à l'appel du constructeur par défaut
719     {
720         taille=0;
721         t=NULL;
722     }
723     else
724         // autres cas
725     {
726         taille=tab.taille;
727         t=new T [taille];
728         T* ptr1=t;
729         T* ptr2=tab.t;
730         for (int i=0;i<taille;i++)
731             // copie des éléments du tableau se trouvant à droite dans le tableau se trouvant
732             // à gauche du signe d'affectation
733             (*ptr1++)=(*ptr2++);
734     };
735     return (*this);
736 }
737 }
738
739 template <class T>
740 inline int Tableau<T>::operator ==(const Tableau<T> & tab) const
741 // Surcharge de l'opérateur ==
742 // Renvoie 1 si les deux tableaux sont égaux
743 // Renvoie 0 sinon
744 {
745
746     if ( tab.taille!=taille )
747         return 0;
748     else
749     {
750         T* ptr1=t;
751         T* ptr2=tab.t;
752         for (int i=0;i<taille;i++)
753         {
754             if ( (*ptr2++)!=(*ptr1++) )
755                 return 0;
756         };
757         return 1;
758     };
759 }
760 }
761
762 // opération de lecture sur un flot d'entrée
763 // les données sont le type puis la dimension puis les datas

```

```

764 template <class T>
765 inline istream & Tableau<T>::Entree(istream & entree)
766 { // vérification du type
767     string type;
768     entree » type;
769     if (type != "Tableau")
770         {Sortie (1);
771         return entree;
772         }
773     int dime; entree » type » dime;
774     if (dime != taille)
775         Change_taille(dime);
776     for (int i=0;i<taille;i++)
777         entree » t[i];
778     return entree;
779 }
780
781 // opération de lecture sur un flot d'entrée sans redimensionnement
782 // les données sont le type puis la dimension puis les datas
783 // ici il n'y a pas redimensionnement du tableau si la taille est suffisente
784 template <class T>
785 inline istream & Tableau<T>::Entree_sans_redim(istream & entree)
786 { // vérification du type
787     string type;
788     entree » type;
789     if (type != "Tableau")
790         {Sortie (1);
791         return entree;
792         }
793     int dime; entree » type » dime;
794     if (dime > taille) // redimensionnement que si la taille est insuffisente
795         Change_taille(dime);
796     for (int i=0;i<taille;i++)
797         entree » t[i];
798     return entree;
799 }
800
801 // lectures sur le flot si la taille du tableau est identique ou supérieure à celle lue
802 // retour d'un pointeur nulle
803 // sinon, si la taille lue est supérieure à la taille du tableau actuellement existant
804 // il y a création d'un nouveau tableau de même type avec la taille adéquate
805 // et retour d'un pointeur sur le nouveau tableau
806 template <class T>
807 inline Tableau<T> * Tableau<T>::New_en_lecture_si_taille_superieur_a_lire(istream& entree)
808 { Tableau<T> * pt_tableau = NULL; // init au cas courant
809     // vérification du type
810     string type;
811     entree » type;
812     if (type != "Tableau")
813         {Sortie (1);
814         return pt_tableau;
815         }
816     int dime; entree » type » dime;
817     if (dime > taille)
818         { // on crée un nouveau tableau
819         pt_tableau = new Tableau<T>(dime);
820         for (int i=0;i<dime;i++)
821             entree » pt_tableau->t[i];
822         }
823     else
824         { // cas d'une dimension suffisante
825         for (int i=0;i<dime;i++)
826             entree » t[i];
827         };
828     return pt_tableau;
829 }
830
831 // opération d'écriture non formatée
832 // les données sont le type la dimension puis les datas, chacun
833 // séparé par un espace
834 template <class T>
835 inline ostream & Tableau<T>::Sortir( ostream & sort) const
836 { // tout d'abord un indicateur donnant le type
837     sort « "Tableau :taille= " « taille « " ";
838     for (int i=0;i<taille;i++)
839         sort « t[i] « " ";
840     sort « "\n";
841     return sort;
842 }
843
844 // opération d'écriture non formatée, idem précédent mais sans retour chariot à la fin
845 // les données sont le type la dimension puis les datas, chacun
846 // séparé par un espace
847 template <class T>
848 inline ostream & Tableau<T>::Sortir_sansRet( ostream & sort) const
849 { // tout d'abord un indicateur donnant le type
850     sort « "Tableau :taille= " « taille « " ";

```

```

851     for (int i=0;i<taille;i++)
852         sort << t[i] << " ";
853     return sort;
854 }
855
856 // test si la valeur passée en argument appartient au tableau
857 // ramène 0 si n'appartient pas sinon ramène l'indice dans le tableau
858 // t(i) = t[i+1]
859 template <class T>
860 inline int Tableau<T>::Contient(const T& e) const
861 { for (int i=0;i<taille;i++)
862     if (t[i] == e) {return (i+1);}
863     return 0;
864 }
865
866
867 // initialisation avec la valeur passée en argument
868 template <class T>
869 inline bool Tableau<T>::Inita(const T& e)
870 { for (int i=0;i<taille;i++)
871     t[i] = e;
872     return true; // par défaut
873 };
874
875 //// affichage du premier élément pour les messages d'erreur
876 //template <class T>
877 //inline void Tableau<T>::Affiche_premier_elem( ostream & sort) const
878 // { // on indique quelque infos sup pour la compréhension
879 //     sort << "Tableau :taille= " << taille << " ";
880 //     if (taille >0)
881 //         {sort << "\n le premier element du tableau pour info: ";
882 //         sort << t[0] ;
883 //         sort << endl;};
884 // }
885
886
887 // #include "ParaGlob.h"
888
889 #endif

```

## 7.446 TabOper\_T.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *
32 *   DATE:           23/01/97
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 * *****/
39 *   BUT:           une spécialisation du tableau en incluant une
40 *   surcharge d'opérateurs courants. Le template n'est pas utilisable
41 *   par tous les objets: ces derniers doivent posséder également
42 *   les surcharges correspondantes.
43 *

```

```

44 *      *****
45 *      VERIFICATION:
46 *
47 *      ! date !   auteur !           but
48 *      -----
49 *      !           !           !           !
50 *
51 *
52 *      *****
53 *      MODIFICATIONS:
54 *
55 *      ! date !   auteur !           but
56 *      -----
57 *
58 *
59 *
60 *      *****/
61
62
63 #ifndef TABOPER_T_H
64 #define TABOPER_T_H
65
66
67 #include "Tableau_T.h"
68
69
70 /// @addtogroup Les_Tableaux_generiques
71 /// @{
72 ///
73
74
75 template <class T>
76 class TabOper : public Tableau<T>
77 {
78
79     public :
80
81
82         // CONSTRUCTEURS :
83
84         // Constructeur par défaut
85         TabOper () : Tableau<T>() {};
86
87         // Constructeur fonction de la taille du tableau
88         TabOper (int nb) : Tableau<T>(nb) {};
89
90         // Constructeur fonction de la taille du tableau et d'une
91         // valeur d'initialisation pour les composantes
92         TabOper (int nb,T val) : Tableau<T>(nb,val) {};
93
94         // Constructeur fonction d'un tableau de composantes de type T
95         TabOper (int nb,T* tab) : Tableau<T>(nb,tab) {};
96
97         // Constructeur de copie
98         TabOper (const TabOper<T> & tab) : Tableau<T>(tab) {};
99
100
101
102         // DESTRUCTEUR :
103
104         ~TabOper (){};
105
106         // METHODES :
107         // Surcharge de l'operateur + : addition entre deux tableaux
108         TabOper<T> operator+ (const TabOper<T>& vec);
109
110         // Surcharge de l'operateur - : soustraction entre deux tableaux
111         TabOper<T> operator- (const TabOper<T>& vec);
112
113         // Surcharge de l'operateur - : oppose d'un tableau
114         //TabOper operator- ();
115         TabOper<T> operator- ();
116
117         // Surcharge de l'operateur +=
118         void operator+= (const TabOper<T>& vec);
119
120         // Surcharge de l'operateur -=
121         void operator-= (const TabOper<T>& vec);
122
123         // Surcharge de l'operateur *= : multiplication d'un scalaire par un tableau
124         void operator*= (double val);
125
126         // Surcharge de l'operateur * : multiplication d'un vecteur par un scalaire
127         //TabOper<T> operator* (double val);
128         inline TabOper<T> operator* (double val);
129
130         // Surcharge de l'operateur / : division des composantes d'un vecteur par un scalaire

```

```

131     TabOper<T> operator/ (double val);
132
133     // Surcharge de l'operateur /= : division des composantes d'un vecteur par un scalaire
134     void operator/= (double val);
135
136     // mise a zero d'un vecteur
137     void Zero();
138
139 };
140 /// @} // end of group
141
142 template <class T>
143 TabOper<T> operator* (double val, TabOper<T>& vec)
144 // Surcharge de l'operateur * : multiplication entre un scalaire et un tableau
145 { return (vec*val); };
146
147
148 template <class T> TabOper<T>
149 TabOper<T>::operator+ (const TabOper<T>& vec)
150 // Surcharge de l'operateur + : addition entre deux tableaux
151 // N.B.: Les tableaux doivent etre de meme taille
152 {
153     #ifndef MISE_AU_POINT
154     if ( this->taille!=vec.taille )
155     { cout << "\nErreur : tailles non egales !\n";
156       cout << "TabOper::OPERATOR+ (TabOper ) \n";
157       Sortie(1);
158     };
159     #endif
160     TabOper result(this->taille);
161     T* ptr1=this->t;
162     T* ptr2=vec.t;
163     T* ptr3=result.t;
164     for (int i=0;i<this->taille;i++)
165         // somme des composantes des deux tableaux vec et v, puis stockage
166         // dans le tableau result
167         (*ptr3++)=(*ptr1++)+(*ptr2++);
168     return result;
169 }
170
171 template <class T> TabOper<T>
172 TabOper<T>::operator- (const TabOper<T>& vec)
173 // Surcharge de l'operateur - : soustraction entre deux tableaux
174 // N.B.: Les tableaux doivent etre de meme taille
175 {
176     #ifndef MISE_AU_POINT
177     if ( this->taille!=vec.taille )
178     { cout << "\nErreur : tailles non egales !\n";
179       cout << "TabOper::OPERATOR- (TabOper ) \n";
180       Sortie(1);
181     };
182     #endif
183     TabOper result(this->taille);
184     T* ptr1=this->t;
185     T* ptr2=vec.t;
186     T* ptr3=result.t;
187     for (int i=0;i<this->taille;i++)
188         // soustraction des composantes des tableaux vec et v, puis stockage
189         // dans le tableau result
190         (*ptr3++)=(*ptr1++)-(*ptr2++);
191     return result;
192 }
193
194 template <class T> TabOper<T>
195 TabOper<T>::operator- ()
196 // Surcharge de l'operateur - : oppose d'un tableau
197 { TabOper result(this->taille);
198   T* ptr1=this->t;
199   T* ptr2=result.t;
200   for (int i=0;i<this->taille;i++)
201       (*ptr2++)=-(*ptr1++);
202   return result;
203 }
204
205 template <class T> void
206 TabOper<T>::operator+= (const TabOper<T>& vec)
207 // Surcharge de l'operateur += : additionne au tableau courant le tableau vec
208 // N.B.: Les tableaux doivent etre de meme taille
209 {
210     #ifndef MISE_AU_POINT
211     if ( this->taille!=vec.taille )
212     { cout << "\nErreur : tailles non egales !\n";
213       cout << "TabOper::OPERATOR+= (TabOper ) \n";
214       Sortie(1);
215     };
216     #endif
217     T* ptr1=this->t;

```

```

218     T* ptr2=vec.t;
219     for (int i=0;i<this->taille;i++)
220         (*ptr1++)+=(*ptr2++);
221 }
222
223 template <class T> void
224 TabOper<T>::operator-= (const TabOper<T>& vec)
225 // Surcharge de l'operateur -= : soustrait au tableau courant le tableau vec
226 // N.B.: Les tableaux doivent etre de meme taille
227 {
228     #ifdef MISE_AU_POINT
229     if ( this->taille!=vec.taille )
230     { cout << "\nErreur : tailles non egales !\n";
231       cout << "TabOper::OPERATOR-= (TabOper ) \n";
232       Sortie(1);
233     };
234     #endif
235     T* ptr1=this->t;
236     T* ptr2=vec.t;
237     for (int i=0;i<this->taille;i++)
238         (*ptr1++)-=(*ptr2++);
239 }
240
241 template <class T> void
242 TabOper<T>::operator*= (double val)
243 // Surcharge de l'operateur *= : multiplie les composantes du tableau courant par
244 // le scalaire val
245 // N.B.: Les tableaux doivent etre de meme taille
246 { T* ptr=this->t;
247   for (int i=0;i<this->taille;i++)
248       (*ptr++)*=val;
249 }
250
251
252
253 template <class T> TabOper <T>
254 TabOper<T>::operator* (double val)
255 // Surcharge de l'operateur * : multiplication d'un tableau par un scalaire
256 {
257     #ifdef MISE_AU_POINT
258     if ( this->taille==0 )
259     { cout << "\nErreur : taille nulle !\n";
260       cout << "TabOper::OPERATOR* ( T ) \n";
261       Sortie(1);
262     };
263     #endif
264     TabOper result(this->taille);
265     T* ptr1=this->t;
266     T* ptr2=result.t;
267     for (int i=1;i<=this->taille;i++)
268         // stockage dans le tableau result du produit des composantes du tableau courant
269         // avec val
270         (*ptr2++)=val*(*ptr1++);
271     return result;
272 }
273
274 template <class T> TabOper<T>
275 TabOper<T>::operator/ (double val)
276 // Surcharge de l'operateur / : division des composantes d'un tableau par un scalaire
277 {
278     #ifdef MISE_AU_POINT
279     if ( (this->taille==0) || (val==0) )
280     { cout << "\nErreur : taille nulle ou division par zero !\n";
281       cout << "TabOper::OPERATOR/ ( T ) \n";
282       Sortie(1);
283     };
284     #endif
285     TabOper result(this->taille);
286     T* ptr1=this->t;
287     T* ptr2=result.t;
288     for (int i=1;i<=this->taille;i++)
289         // stockage dans le tableau result de la division des composantes du tableau courant
290         // avec val
291         (*ptr2++)=(*ptr1++)/val;
292     return result;
293 }
294
295 template <class T> void
296 TabOper<T>::operator/= (double val)
297 // Surcharge de l'operateur / : division des composantes d'un tableau par un scalaire
298 {
299     #ifdef MISE_AU_POINT
300     if ( (this->taille==0) || (val==0) )
301     { cout << "\nErreur : taille nulle ou division par zero !\n";
302       cout << "TabOper::OPERATOR/ ( T ) \n";
303       Sortie(1);
304     };

```

```

305     #endif
306     T* ptr=this->t;
307     for (int i=0;i<this->taille;i++)
308         (*ptr++) /= val;
309 }
310
311 template <class T>
312 // mise a zero d'un tableau
313 void TabOper<T>::Zero()
314 {     T* ptr=this->t;
315     for (int i=0;i<this->taille;i++)
316         // affectation des composantes du tableau a 0
317         (*ptr++)=0.0;
318 }
319
320
321 #endif

```

## 7.447 utilDebug.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31 *
32 *   DATE:           23/01/97
33 *
34 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *   PROJET:        Herezh++
37 *
38 *
39 *   BUT:           outils pour la visualisation avec le debugger
40 *
41 *   *****
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *
48 *
49 *   *****
50 *   MODIFICATIONS:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *
55 *
56 *
57 *****/
58
59
60 #ifndef TABLE_H
61 #define TABLE_H
62
63
64 class Table33
65 { public :

```

```

66     double A[3][3];
67 };
68
69 #endif

```

## 7.448 Coordonnee.h

```

1 // FICHER : Coordonnee.h
2 // CLASSE : Coordonnee
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          23/01/97
34 *
35 *      AUTEUR:        G RIO
36 *
37 *      PROJET:        Herezh++
38 *
39 *
40 * BUT: Les classes Coordonnee servent a la localisation dans l'espace
41 * des objets tels que les noeuds ou les points. Une instance de
42 * cette classe est caracterisee par le nombre de coordonnees et
43 * par la valeur de celles-ci.
44 * Les valeurs des coordonnees sont de type double et sont stockees
45 * dans un tableau dont la taille depend de la dimension du
46 * probleme.
47 *
48 *      *****
49 *****/
50
51
52
53 #ifndef COORDONNEE_H
54 #define COORDONNEE_H
55
56
57 // #include "Debug.h"
58 #include <iostream>
59 #include <fstream>
60 #include "UtilLecture.h"
61 #include <stdlib.h>
62 #include "Sortie.h"
63 #include "Enum_IO_XML.h"
64 #include "PtTabRel.h"
65
66 class Vecteur; // declare a la fin du fichier, car Vecteur utilise aussi Coordonnee
67 class CoordonneeH; class CoordonneeB;
68
69
70 /** @defgroup Les_classes_cooronnee
71 *
72 * BUT: Les classes de type Coordonnee servent à la localisation dans l'espace
73 * des objets tels que les points ou les noeuds etc.\n Une instance de
74 * cette classe est caracterisee par le nombre de coordonnées et
75 * par la valeur de celles-ci.
76 * \n Les valeurs des coordonnées sont de type double et sont stockées
77 * dans un tableau dont la taille depend de la dimension du
78 * problème.

```



```

79 * \author    Gérard Rio
80 * \version  1.0
81 * \date     23/01/97
82 * \brief    Définition des classes de type Coordonnee, en coordonnées sans variance (ex: absolues)
83 * ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes
84 *
85 */
86
87
88
89 /// @addtogroup Les_classes_coordonnee
90 /// @{
91 ///
92
93 //=====
94 //  cas des coordonnées simples sans variance
95 //=====
96
97 //! Coordonnees simples sans variances
98
99
100 class Coordonnee
101 {
102
103     /// surcharge de l'operator de lecture avec le type
104     friend istream & operator » (istream &, Coordonnee &);
105
106     /// surcharge de l'operator d'écriture
107     friend ostream & operator « (ostream &, const Coordonnee &);
108
109     /// Surcharge de l'opérateur * : multiplication entre un scalaire et des coordonnees
110     inline friend Coordonnee operator* (double val,const Coordonnee& c)
111     { return (c*val); };
112
113     friend class BaseH; friend class BaseB;
114
115     public :
116         // CONSTRUCTEURS :
117
118         /// Constructeur par défaut
119         Coordonnee () ;
120     /*! \brief
121         // Constructeur fonction de la dimension du probleme
122         // les coordonnees sont initialise a zero
123     */
124         Coordonnee (int dimension);
125         /// Constructeur pour une localisation unidimensionnelle
126         Coordonnee (double x);
127         /// Constructeur pour une localisation bidimensionnelle
128         Coordonnee (double x,double y);
129         // Constructeur pour une localisation tridimensionnelle
130         Coordonnee (double x,double y,double z);
131     /*! \brief
132         // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
133         // et d'une dimension ( l'existence de la place memoire est a la charge
134         // de l'utilisateur et ne sera pas détruite par le destructeur.)
135     */
136         Coordonnee (int dimension,double* t);
137         // Constructeur fonction d'un vecteur qui doit avoir une dim = 1 ou 2 ou 3
138     // on supprime car il y a plein de conversion n'ont explicite, le mieux est d'utiliser
139     // l'opérateur Vect(), qui permet "explicitement" de créer un coordonnée et ensuite d'utiliser
140     // le constructeur en fonction de coordonnée
141
142     ///on pourrait faire une fonction à la place
143
144     // Coordonnee ( const Vecteur& vec);
145     //// Construit(const Vecteur& vec);
146
147     /// Constructeur de copie
148     Coordonnee (const Coordonnee& c);
149
150     /// DESTRUCTEUR :
151     virtual ~Coordonnee ();
152
153     // METHODES :
154
155     /*! \brief
156         // fonction équivalente au constructeur: changement pour une place externe via un pointeur
157         // ( l'existence de la place memoire est a la charge
158         // de l'utilisateur et ne sera pas détruite par le destructeur.)
159     */
160     void Change_place(int dimension,double* t);
161
162     // changement des coordonnees
163     // rapide si c'est la même dimension !!
164     /// changement rapide des coordonnées: dimension 1
165     void Change_Coordonnee(int dimension, double x) ; // dimension 1

```

```

166 // changement rapide des coordonnées: dimension 2
167 void Change_Coordonnee(int dimension, double x,double y) ; // dimension 2
168 // changement rapide des coordonnées: dimension 3
169 void Change_Coordonnee(int dimension, double x,double y,double z) ; // dimension 3
170
171 // Renvoie le nombre de coordonnees
172 virtual int Dimension () const ;
173 // Desallocation de la place memoire allouee
174 virtual void Libere ();
175 // Renvoie la ieme coordonnee
176 virtual double& operator() (int i);
177 // Renvoie une copie de la ieme coordonnee
178 virtual double operator() (int i) const;
179 // Surcharge de l'operateur = : realise l'affectation entre deux points
180 Coordonnee& operator= (const Coordonnee& c);
181 // Surcharge de l'operateur = avec un vecteur
182 // trop dangereux, car il y a plein de conversion implicite, donc ce qu'il vaut mieux
183 // faire, c'est une conversion "explicite" à l'aide de Vect()
184 // Coordonnee& operator= ( const Vecteur& c);
185 // Coordonnee& Egale_vecteur ( const Vecteur& c);
186 // Surcharge de l'operateur - : renvoie l'oppose d'un point
187 Coordonnee operator- () const;
188 // Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points
189 Coordonnee operator- (const Coordonnee& c) const;
190 // Surcharge de l'operateur + : realise l'addition des coordonnees de deux points
191 Coordonnee operator+ (const Coordonnee& c) const;
192 // Surcharge de l'operateur +=
193 void operator+= (const Coordonnee& c);
194 // Surcharge de l'operateur -=
195 void operator-= (const Coordonnee& c);
196 // Surcharge de l'operateur *=
197 void operator*= (double val);
198 // Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
199 Coordonnee operator* (double val) const;
200 // Surcharge de l'operateur * : produit scalaire entre coordonnees
201 double operator* (const Coordonnee& c) const ;
202 // Surcharge de l'operateur / : division de coordonnees par un scalaire
203 Coordonnee operator/ (double val) const ;
204 // Surcharge de l'operateur /= : division de coordonnees par un scalaire
205 void operator/= (double val) ;
206 /*! \brief
207 // Surcharge de l'operateur == : test d'egalite
208 // Renvoie 1 si les deux positions sont identiques
209 // Renvoie 0 sinon
210 */
211 int operator== (const Coordonnee& c) const;
212 /*! \brief
213 // Surcharge de l'operateur !=
214 // Renvoie 1 si les deux positions ne sont pas identiques
215 // Renvoie 0 sinon
216 */
217 int operator!= (const Coordonnee& c) const;
218 // Affiche les coordonnees du point à l'écran entre accolades
219 virtual void Affiche () const;
220 // Affiche les coordonnees du point dans sort entre accolades
221 virtual void Affiche (ostream& sort) const ;
222 /*! \brief
223 // Affiche les coordonnees du point dans sort sur nb digit plus un blanc
224 // et rien d'autre (pas d'accolade)
225 */
226 virtual void Affiche (ostream& sort,int nb) const ;
227 // Affiche les coordonnees du point dans sort et rien d'autre (pas d'accolade)
228 virtual void Affiche_1 (ostream& sort) const ;
229 // lecture brut des coordonnées sans la dimension
230 virtual void Lecture (UtilLecture& entreePrinc);
231 // lecture brut des coordonnées sans la dimension dans le flux par défaut
232 virtual void Lecture ();
233 /*! \brief
234 // changement de la dimension
235 // dans le cas d'une nouvelle dimension inferieur on supprime les dernieres coord
236 // dans le cas d'une dimension superieur, on ajoute des coord initialisees a zero
237 */
238 virtual void Change_dim(int dim);
239 // création d'un Vecteur équivalent
240 virtual Vecteur Vect() const ;
241 // mise a zero des coordonnées
242 virtual void Zero();
243 // Calcul de la norme euclidienne des composantes du point
244 virtual double Norme () const ;
245 // norme le vecteur coordonnée
246 Coordonnee& Normer ();
247 // Calcul du maximum en valeur absolu des composantes du vecteur
248 double Max_val_abs () const ;
249 // ramène l'indice du maxi en valeur absolu
250 int Indice_max_val_abs() const;
251 /*! \brief
252 // Calcul du maximum en valeur absolu des composantes du vecteur

```

```

253         // ramene egalement l'indice de tableau du maximum
254     */
255     double Max_val_abs (int& i) const ;
256     /*! \brief
257     // Calcul du maximum en valeur absolu des composantes du vecteur
258     // mais ramène la grandeur signée (avec son signe)
259     */
260     double Max_val_abs_signe() const ;
261     /*! \brief
262     // Calcul du maximum en valeur absolu des composantes du vecteur
263     // mais ramène la grandeur signée (avec son signe)
264     // ramene egalement l'indice de tableau du maximum
265     */
266     double Max_val_abs_signe(int& i) const ;
267     /*! \brief
268     // modifie éventuellement les coordonnées de this pour quelles soient supérieures ou égales
269     // aux coordonnées en paramètre
270     */
271     void Modif_en_max(const Coordonnee& v);
272     /*! \brief
273     // modifie éventuellement les coordonnées de this pour quelles soient inférieures ou égales
274     // aux coordonnées en paramètre
275     */
276     void Modif_en_min(const Coordonnee& v);
277     /// ajoute une même valeur à tous les coordonnées
278     void Ajout_meme_valeur(double val);
279     /// calcul la norme euclidienne au carré
280     double Carre()const {return (*this) * (*this);};
281     /// sortie du schemaXML: en fonction de enu
282     static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
283     // changement de base
284     // beta(i,j) represente les coordonnees de la nouvelle base gp dans l'ancienne g
285     // gp(i) = beta(i,j) * g(j)
286     void ChBase( const Mat_pleine& beta);
287
288     protected :
289     /// dimension
290     short int dim;
291     /// indique s'il y a allocation ou pas
292     bool memoire;
293     /// stockage
294     double* coord;
295
296     /*! \brief
297     // Constructeur inline qui ne fait rien
298     // utilisé par les classes dérivées
299     */
300     Coordonnee (bool ) : dim(-1), coord (NULL),memoire(true) {};
301     /// définit des coordonnées sans variance à la même place que des coordonnées avec variances
302     void Meme_place(CoordonneeB& vB);
303     /// définit des coordonnées sans variance à la même place que des coordonnées avec variances
304     void Meme_place(CoordonneeH& vH);
305
306 };
307 /// @} // end of group
308
309
310
311
312 class CoordonneeB; // défini par la suite ( nécessaire pour le produit scalaire)
313 //class BaseH; // pour déclarer une classe friend
314
315 /// @addtogroup Les_classes_coordonnee
316 /// @{}
317 ///
318
319 //=====
320 //! cas des coordonnées contravariantes
321 //=====
322
323 class CoordonneeH
324 {
325     /// surcharge de l'operator de lecture avec le type
326     friend istream & operator » (istream &, CoordonneeH &);
327     /// surcharge de l'operator d'écriture
328     friend ostream & operator « (ostream &, const CoordonneeH &);
329     /// Surcharge de l'operateur * : multiplication entre un scalaire et des coordonnees
330     inline friend CoordonneeH operator* (double val,const CoordonneeH& c)
331     { return (c*val); };
332
333     // déclaration de la classe BaseH comme friend, pour pouvoir faire une relation entre
334     // un CoordonneeH et un Coordonnee
335     // friend class BaseH;
336     friend class Coordonnee; friend class CoordonneeB;
337
338     public :
339     // CONSTRUCTEURS :

```

```

340
341     /// Constructeur par default
342     CoordonneeH () ;
343  /*! \brief
344     // Constructeur fonction de la dimension du probleme
345     // les coordonnees sont initialise a zero
346  */
347     CoordonneeH (int dimension);
348     /// Constructeur pour une localisation unidimensionnelle
349     CoordonneeH (double x);
350     /// Constructeur pour une localisation bidimensionnelle
351     CoordonneeH (double x,double y);
352     /// Constructeur pour une localisation tridimensionnelle
353     CoordonneeH (double x,double y,double z);
354  /*! \brief
355     // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
356     // et d'une dimension ( l'existence de la place memoire est a la charge
357     // de l'utilisateur et ne sera pas detruite par le destructeur.
358  */
359     CoordonneeH (int dimension,double* t);
360     /// Constructeur fonction d'un vecteur qui doit avoir une dim = 1 ou 2 ou 3
361 // on supprime car il y a plein de conversion n'ont explicite, le mieux est d'utiliser
362 // l'opérateur Vect(), qui permet "explicitement" de créer un coordonnée et ensuite d'utiliser
363 // le constructeur en fonction de coordonnée
364 //     CoordonneeH ( const Vecteur& vec);
365     /// Constructeur de copie
366     CoordonneeH (const CoordonneeH& c);
367
368     /// DESTRUCTEUR :
369     virtual ~CoordonneeH ();
370
371     // METHODES :
372
373  /*! \brief
374     // construction "explicite" à partir d'une instance de CoordonneeB
375     // intéressant si this est initialement construit par default (donc vide)
376     // cela permet de créer un CoordonneeH à partir d'un B, mais de manière explicite,
377     // donc activé quand on le veut (et non pas par le compilé au gré de conversion pas toujours
378     // clair!!)
379  */
380     void ConstructionAPartirDe_B(const CoordonneeB& aB);
381
382     /// Renvoie le nombre de coordonnees
383     virtual int Dimension () const ;
384     /// Desallocation de la place memoire allouee
385     virtual void Libere ();
386     /// Renvoie la ieme coordonnee
387     virtual double& operator() (int i);
388     /// Renvoie une copie de la ieme coordonnee
389     virtual double operator() (int i) const;
390     // Surcharge de l'operateur = : realise l'affectation entre deux points
391     CoordonneeH& operator= (const CoordonneeH& c);
392     /// change les valeurs en fonction d'un point sans variance
393     void Change_val(const Coordonnee& c);
394     // Surcharge de l'operateur = avec un vecteur
395 // trop dangereux, car il y a plein de conversion implicite, donc ce qu'il vaut mieux
396 // faire, c'est une conversion "explicite" à l'aide de Vect()
397 //     CoordonneeH& operator= ( const Vecteur& c);
398 //     CoordonneeH& Egale_vecteur ( const Vecteur& c);
399     /// Surcharge de l'operateur - : renvoie l'oppose d'un point
400     CoordonneeH operator- () const;
401     /// Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points
402     CoordonneeH operator- (const CoordonneeH& c) const;
403     /// Surcharge de l'operateur + : realise l'addition des coordonnees de deux points
404     CoordonneeH operator+ (const CoordonneeH& c) const;
405     /// Surcharge de l'operateur +=
406     void operator+= (const CoordonneeH& c);
407     /// Surcharge de l'operateur -=
408     void operator-= (const CoordonneeH& c);
409     /// Surcharge de l'operateur *=
410     void operator*= (double val);
411     /// Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
412     CoordonneeH operator* (double val) const;
413     /// Surcharge de l'operateur * : produit scalaire entre coordonnees
414     double operator* (const CoordonneeB& c) const ;
415     /// produit scalaire entre coordonnees contravariantes et contravariantes
416     double ScalHH(const CoordonneeH& c) const ;
417     /// Surcharge de l'operateur / : division de coordonnees par un scalaire
418     CoordonneeH operator/ (double val) const ;
419     /// Surcharge de l'operateur /= : division de coordonnees par un scalaire
420     void operator/= (double val) ;
421  /*! \brief
422     // Surcharge de l'operateur == : test d'egalite
423     // Renvoie 1 si les deux positions sont identiques
424     // Renvoie 0 sinon
425  */
426     int operator== (const CoordonneeH& c) const;

```

```

426  /*! \brief
427      // Surcharge de l'operateur !=
428      // Renvoie 1 si les deux positions ne sont pas identiques
429      // Renvoie 0 sinon
430  */
431      int operator!= (const CoordonneeH& c) const;
432      /// Affiche les coordonnees du point à l'écran
433      virtual void Affiche () const;
434      /// Affiche les coordonnees du point dans sort
435      virtual void Affiche (ostream& sort) const ;
436      /// Affiche les coordonnees du point dans sort sur nb digit plus un blanc et rien d'autre
437      virtual void Affiche (ostream& sort,int nb) const ;
438      /// lecture brut des coordonnées sans la dimension
439      virtual void Lecture (UtilLecture& entreePrinc);
440  /*! \brief
441      // changement de la dimension
442      // dans le cas d'une nouvelle dimension inferieur on supprime les dernieres coord
443      // dans le cas d'une dimension superieur, on ajoute des coord initialisees a zero
444  */
445      virtual void Change_dim(int dim);
446      /// création d'un Vecteur équivalent
447      virtual Vecteur Vect() const ;
448      /// création de coordonnées équivalentes sans variance
449      virtual Coordonnee Coord() const;
450  /*! \brief
451      // création explicite en coordonnées sans variance
452      // mais le vecteur est à la même place pour un coût de construction minimum,
453      // il est accessible en lecture uniquement
454  */
455      virtual const Coordonnee Coord_const()const ;
456      /// création explicite de H en B
457      virtual CoordonneeB Haut_bas()const;
458      /// mise a zero des coordonnées
459      virtual void Zero();
460      /// Calcul de la norme euclidienne des composantes du point
461      virtual double Norme () const ;
462      /// norme le vecteur coordonnée
463      CoordonneeH& Normer ();
464      /// Calcul du maximum en valeur absolu des composantes du vecteur
465      double Max_val_abs () const ;
466  /*! \brief
467      // Calcul du maximum en valeur absolu des composantes du vecteur
468      // ramene egalement l'indice de tableau du maximum
469  */
470      double Max_val_abs (int& i) const ;
471  /*! \brief
472      // Calcul du maximum en valeur absolu des composantes du vecteur
473      // mais ramène la grandeur signée (avec son signe)
474  */
475      double Max_val_abs_signe() const ;
476  /*! \brief
477      // Calcul du maximum en valeur absolu des composantes du vecteur
478      // mais ramène la grandeur signée (avec son signe)
479      // ramene egalement l'indice de tableau du maximum
480  */
481      double Max_val_abs_signe(int& i) const ;
482  /*! \brief
483      // modifie éventuellement les coordonnées de this pour quelles soient supérieures ou égales
484      // aux coordonnées en paramètre
485  */
486      void Modif_en_max(const CoordonneeH& v);
487  /*! \brief
488      // modifie éventuellement les coordonnées de this pour quelles soient inférieures ou égales
489      // aux coordonnées en paramètre
490  */
491      void Modif_en_min(const CoordonneeH& v);
492      /// ajoute une même valeur à tous les coordonnées
493      void Ajout_meme_valeur(double val);
494      /// sortie du schemaXML: en fonction de enu
495      static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
496      // changement de base
497      // beta(i,j) represente les coordonnees de la nouvelle base gpB dans l'ancienne gB
498      // gpB(i) = beta(i,j) * gB(j)
499      // void ChBase( const Mat_pleine& beta);
500
501
502      protected :
503
504          short int dim;
505          bool memoire; // indique s'il y a allocation ou pas
506          double* coord;
507          /// Constructeur inline qui ne fait rien
508          CoordonneeH (bool ):dim (-1) , coord (NULL),memoire(true) {};
509
510 };
511 /// @} // end of group
512

```

```

513 /// @addtogroup Les_classes_coordonnee
514 /// @{
515 ///
516 //=====
517 ///!   cas des coordonnées covariantes
518 //=====
519
520 class CoordonneeB
521 {
522     /// surcharge de l'operateur de lecture avec le type
523     friend istream & operator » (istream &, CoordonneeB &);
524     /// surcharge de l'operateur d'écriture
525     friend ostream & operator « (ostream &, const CoordonneeB &);
526     /// Surcharge de l'operateur * : multiplication entre un scalaire et des coordonnees
527     inline friend CoordonneeB operator* (double val,CoordonneeB& c)
528     { return (c*val); };
529
530     friend class Coordonnee;friend class CoordonneeH;
531     public :
532         // CONSTRUCTEURS :
533
534         /// Constructeur par default
535         CoordonneeB () ;
536     /*! \brief
537         // Constructeur fonction de la dimension du probleme
538         // les coordonnees sont initialise a zero
539     */
540         CoordonneeB (int dimension);
541         /// Constructeur pour une localisation unidimensionnelle
542         CoordonneeB (double x);
543         /// Constructeur pour une localisation bidimensionnelle
544         CoordonneeB (double x,double y);
545         /// Constructeur pour une localisation tridimensionnelle
546         CoordonneeB (double x,double y,double z);
547     /*! \brief
548         // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
549         // et d'une dimension ( l'existence de la place memoire est a la charge
550         // de l'utilisateur et ne sera pas détruite par le destructeur.
551     */
552         CoordonneeB (int dimension,double* t);
553         // Constructeur fonction d'un vecteur qui doit avoir une dim = 1 ou 2 ou 3
554     // on supprime car il y a plein de conversion n'ont explicite, le mieux est d'utiliser
555     // l'opérateur Vect(), qui permet "explicitement" de créer un coordonnée et ensuite d'utiliser
556     // le constructeur en fonction de coordonnée
557     // CoordonneeB ( const Vecteur& vec);
558     /// Constructeur de copie
559     CoordonneeB (const CoordonneeB& c);
560
561     /// DESTRUCTEUR :
562     virtual ~CoordonneeB ();
563
564     // METHODES :
565
566
567     /*! \brief
568         // construction "explicite" à partir d'une instance de CoordonneeB
569         // intéressant si this est initialement construit par default (donc vide)
570         // cela permet de créer un CoordonneeH à partir d'un B, mais de manière explicite,
571         // donc activé quand on le veut (et non pas par le compilé au gré de conversion pas toujours
572         // clair!!)
573     */
574     void ConstructionAPartirDe_H(const CoordonneeH& aH);
575
576     /// Renvoie le nombre de coordonnees
577     virtual int Dimension () const ;
578     /// Desallocation de la place memoire allouee
579     virtual void Libere ();
580     /// Renvoie la ieme coordonnee
581     virtual double& operator() (int i);
582     /// Renvoie une copie de la ieme coordonnee
583     virtual double operator() (int i) const;
584     /// Surcharge de l'operateur = : realise l'affectation entre deux points
585     CoordonneeB& operator= (const CoordonneeB& c);
586     /// change les valeurs en fonction d'un point sans variance
587     void Change_val(const Coordonnee& c);
588
589     // Surcharge de l'operateur = avec un vecteur
590     // trop dangereux, car il y a plein de conversion implicite, donc ce qu'il vaut mieux
591     // faire, c'est une conversion "explicite" à l'aide de Vect()
592     CoordonneeB& operator= ( const Vecteur& c);
593     CoordonneeB& Egale_vecteur ( const Vecteur& c);
594     /// Surcharge de l'operateur - : renvoie l'oppose d'un point
595     CoordonneeB operator- () const;
596     /// Surcharge de l'operateur - : realise la soustraction des coordonnees de deux points
597     CoordonneeB operator- (const CoordonneeB& c) const;
598     /// Surcharge de l'operateur + : realise l'addition des coordonnees de deux points
599     CoordonneeB operator+ (const CoordonneeB& c) const;

```

```

599     /// Surcharge de l'operateur +=
600     void operator+= (const CoordonneeB& c);
601     /// Surcharge de l'operateur -=
602     void operator-= (const CoordonneeB& c);
603     /// Surcharge de l'operateur *=
604     void operator*= (double val);
605     /// Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
606     CoordonneeB operator* (double val) const;
607     /// Surcharge de l'operateur * : produit scalaire entre coordonnees
608     double operator* (const CoordonneeH& c) const ;
609     /// produit scalaire entre coordonnees covariantes et covariantes
610     double ScalBB(const CoordonneeB& c) const ;
611     /// Surcharge de l'operateur / : division de coordonnees par un scalaire
612     CoordonneeB operator/ (double val) const ;
613     /// Surcharge de l'operateur /= : division de coordonnees par un scalaire
614     void operator/= (double val) ;
615     /*! \brief
616         /// Surcharge de l'operateur == : test d'egalite
617         /// Renvoie 1 si les deux positions sont identiques
618         /// Renvoie 0 sinon
619     */
620     int operator== (const CoordonneeB& c) const;
621     /*! \brief
622         /// Surcharge de l'operateur !=
623         /// Renvoie 1 si les deux positions ne sont pas identiques
624         /// Renvoie 0 sinon
625     */
626     int operator!= (const CoordonneeB& c) const;
627     /// Affiche les coordonnees du point à l'écran
628     virtual void Affiche () const;
629     /// Affiche les coordonnees du point dans sort
630     virtual void Affiche (ostream& sort) const ;
631     /// Affiche les coordonnees du point dans sort sur nb digit plus un blanc
632     /// et rien d'autre
633     virtual void Affiche (ostream& sort,int nb) const ;
634     /// lecture brut des coordonnées sans la dimension
635     virtual void Lecture (UtilLecture& entreePrinc);
636     /*! \brief
637         /// changement de la dimension
638         /// dans le cas d'une nouvelle dimension inferieur on supprime les dernieres coord
639         /// dans le cas d'une dimension superieur, on ajoute des coord initialisees a zero
640     */
641     virtual void Change_dim(int dim);
642     /// conversion en Vecteur
643     virtual Vecteur Vect() const ;
644     /// conversion explicite en coordonnées sans variance
645     virtual Coordonnee Coord()const ;
646     /*! \brief
647         /// création explicite en coordonnées sans variance
648         /// mais le vecteur est à la même place pour un coût de construction minimum,
649         /// il est accessible en lecture uniquement
650     */
651     virtual const Coordonnee Coord_const()const ;
652     /// conversion explicite de B en H
653     virtual CoordonneeH Bas_haut()const;
654     /// mise a zero des coordonnées
655     virtual void Zero();
656     /// Calcul de la norme euclidienne des composantes du point
657     virtual double Norme () const ;
658     /// norme le vecteur coordonnée
659     CoordonneeB& Normer ();
660     /// Calcul du maximum en valeur absolu des composantes du vecteur
661     double Max_val_abs () const ;
662     /*! \brief
663         /// Calcul du maximum en valeur absolu des composantes du vecteur
664         /// ramene egalement l'indice de tableau du maximum
665     */
666     double Max_val_abs (int& i) const ;
667     /// Calcul du maximum en valeur absolu des composantes du vecteur
668     /// mais ramène la grandeur signée (avec son signe)
669     double Max_val_abs_signe() const ;
670     /*! \brief
671         /// Calcul du maximum en valeur absolu des composantes du vecteur
672         /// mais ramène la grandeur signée (avec son signe)
673         /// ramene egalement l'indice de tableau du maximum
674     */
675     double Max_val_abs_signe(int& i) const ;
676     /*! \brief
677         /// modifie éventuellement les coordonnées de this pour quelles soient supérieures ou égales
678         /// aux coordonnées en paramètre
679     */
680     void Modif_en_max(const CoordonneeB& v);
681     /*! \brief
682         /// modifie éventuellement les coordonnées de this pour quelles soient inférieures ou égales
683         /// aux coordonnées en paramètre
684     */
685     void Modif_en_min(const CoordonneeB& v);

```

```

686         // ajoute une même valeur à tous les coordonnées
687         void Ajout_meme_valeur(double val);
688         // sortie du schemaXML: en fonction de enu
689         static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
690         // changement de base
691         // beta(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
692         // gpH(i) = beta(i,j) * gH(j), i indice de ligne, j indice de colonne
693         // void ChBase( const Mat_pleine& beta);
694
695
696     protected :
697
698         short int dim;
699         bool memoire; // indique s'il y a allocation ou pas
700         double* coord;
701         // Constructeur inline qui ne fait rien
702         CoordonneeB (bool ):dim (-1) , coord (NULL),memoire(true) {};
703
704 };
705 /// @} // end of group
706
707 #ifndef MISE_AU_POINT
708     #include "Coordonnee.cc"
709     #include "CoordonneeH.cc"
710     #include "CoordonneeB.cc"
711     #define COORDONNEE_H_deja_inclus
712 #endif
713
714
715 // #include "Vecteur.h"
716 #endif
717

```

## 7.449 Coordonnee1.h

```

1 // FICHER : Coordonnee1.h
2 // CLASSE : Coordonnee1
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          23/01/97
34 *
35 *      AUTEUR:        G RIO
36 *
37 *      PROJET:        Herezh++
38 *
39 *****/
40 * BUT: Les classes Coordonnee1 servent a la localisation dans l'espace *
41 *      1D des objets tels que les noeuds ou les points. Ces classes *
42 *      dérivent des Classes génériques Coordonnee.
43 *
44 *      *****
45 *****/
46
47
48
49 #ifndef COORDONNEE1_H
50 #define COORDONNEE1_H

```



```

51
52
53 //#include "Debug.h"
54 #include <iostream>
55 #include <fstream>
56 #include <stdlib.h>
57 #include "Sortie.h"
58 #include "Coordonnee.h"
59
60 /** @defgroup Les_classes_cooronneel
61 *
62 * BUT: Les classes Coordonnee1 servent a la localisation dans l'espace
63 * 1D des objets tels que les noeuds ou les points. Ces classes
64 * dérivent des Classes génériques Coordonnee.
65 * \author Gérard Rio
66 * \version 1.0
67 * \date 23/01/97
68 * \brief Définition des classes de type Coordonnee1, en coordonnées sans variance (ex: absolues)
69 * ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont
70 * une spécialisation 1D des classes générales Coordonnee
71 */
72
73
74
75 /// @addtogroup Les_classes_cooronneel
76 /// @{
77 ///
78
79 ///=====
80 /// cas des coordonnées simples sans variance
81 ///=====
82
83 class Coordonnee1 : public Coordonnee
84 {
85
86     public :
87     /// Surcharge de l'opérateur * : multiplication entre un scalaire et des coordonnées
88     inline friend Coordonnee1 operator* (double val,const Coordonnee1& c);
89
90     /// CONSTRUCTEURS :
91
92     /*! \brief
93     /// Constructeur par défaut
94     /// // il y a initialisation des coordonnées à zéro par défaut
95     */
96     Coordonnee1 ();
97     /*! \brief
98     /// Constructeur suivant un booleen
99     /// // quelque soit la valeur du booleen il n'y a pas initialisation des coordonnées
100    /// // ceci pour aller plus vite par rapport au constructeur par défaut
101    */
102    Coordonnee1 (bool test );
103    /// Constructeur pour une localisation monodimensionnelle
104    Coordonnee1 (double x);
105    /*! \brief
106    /// constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
107    /// // ( l'existence de la place mémoire est a la charge
108    /// // de l'utilisateur !!).
109    */
110    Coordonnee1 (double* t);
111    /// Constructeur fonction d'un vecteur qui doit avoir une 1
112    Coordonnee1 ( const Vecteur& vec);
113    /// Constructeur de copie
114    Coordonnee1 (const Coordonnee1& c);
115    /// Constructeur de copie pour une instance indifférenciée
116    Coordonnee1 (const Coordonnee& c);
117
118    /// DESTRUCTEUR :
119    virtual ~Coordonnee1 ();
120
121    /// METHODES :
122
123    /// Renvoie le nombre de coordonnées
124    int Dimension () const ;
125
126    /*! \brief
127    /// Desallocation de la place memoire allouee
128    /// fonction définie dans la classe mère générique mais qui n'a pas de
129    /// sens ici
130    */
131    void Libere ();
132
133    /// Renvoie le nombre de coordonnées
134    /// int Dimension () const ;
135
136    /*! \brief

```

```

137 // changement de la dimension
138 // fonction définie dans la classe mère générique mais qui n'a pas de
139 // sens ici, affiche un message d'erreur
140 */
141     void Change_dim(int dim);
142
143     /// Surcharge de l'opérateur = : réalise l'affectation entre deux points
144     Coordonnee& operator= (const Coordonnee& c);
145
146     /// Surcharge de l'opérateur - : renvoie l'opposé d'un point
147     Coordonnee operator- () const ;
148
149     /*! \brief
150 // Surcharge de l'opérateur - : réalise la soustraction des
151 // coordonnées de deux points
152 */
153     Coordonnee operator- (const Coordonnee& c) const ;
154
155     /*! \brief
156 // Surcharge de l'opérateur + : réalise l'addition des
157 // coordonnées de deux points
158 */
159     Coordonnee operator+ (const Coordonnee& c) const ;
160
161     /// Surcharge de l'opérateur +=
162     void operator+= (const Coordonnee& c);
163
164     /// Surcharge de l'opérateur -=
165     void operator-= (const Coordonnee& c);
166
167     /// Surcharge de l'opérateur *=
168     void operator*= (double val);
169
170     /// Surcharge de l'opérateur * : multiplication de coordonnées par un scalaire
171     Coordonnee operator* (double val) const ;
172
173     /// Surcharge de l'opérateur * : produit scalaire entre coordonnées
174     double operator* (const Coordonnee& c) const ;
175
176     /// Surcharge de l'opérateur / : division de coordonnées par un scalaire
177     Coordonnee operator/ (double val) const ;
178
179     /// Surcharge de l'opérateur /= : division de coordonnées par un scalaire
180     void operator/= (double val) ;
181
182     /*! \brief
183 // Surcharge de l'opérateur == : test d'égalité
184 // Renvoie 1 si les deux positions sont identiques
185 // Renvoie 0 sinon
186 */
187     int operator== (const Coordonnee& c) const;
188
189     /// conversion en Vecteur
190     Vecteur Vect() const ;
191
192     /// mise à zéro des coordonnées
193     void Zero() ;
194
195     /// Calcul de la norme euclidienne des composantes du point
196     double Norme () const ;
197
198     /// norme le vecteur coordonnée
199     Coordonnee& Normer () ;
200
201     /// somme de tous les composantes
202     double Somme() const ;
203     /// sortie du schemaXML: en fonction de enu
204     static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
205
206     protected :
207
208         double coordl[1];
209
210 };
211 /// @} // end of group
212
213 /// @addtogroup Les_classes_coordonnee1
214 /// @{
215 ///
216
217 class Coordonnee1B; // défini par la suite (nécessaire pour le produit scalaire)
218
219 //=====
220 //! cas des coordonnées contravariantes
221 //=====
222
223 class Coordonnee1H : public CoordonneeH

```

```

224 {
225
226     public :
227         friend class Coordonnee1B;
228
229     /// Surcharge de l'operateur * : multiplication entre un scalaire et des coordonnees
230     inline friend Coordonnee1H operator* (double val,const Coordonnee1H& c);
231
232     // CONSTRUCTEURS :
233
234     /*! \brief
235     // Constructeur par défaut
236     // il y a initialisation des coordonnees à zéro par défaut
237     */
238     Coordonnee1H ();
239     /*! \brief
240     // Constructeur suivant un booleen
241     // quelque soit la valeur du booleen il n'y a pas initialisation des coordonnees
242     // ceci pour aller plus vite par rapport au constructeur par défaut
243     */
244     Coordonnee1H (bool test );
245     /// Constructeur pour une localisation monodimensionnelle
246     Coordonnee1H (double x);
247     /*! \brief
248     // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
249     // ( l'existence de la place memoire est a la charge
250     // de l'utilisateur !!).
251     */
252     Coordonnee1H (double* t);
253     /// Constructeur fonction d'un vecteur qui doit avoir une 1
254     Coordonnee1H ( const Vecteur& vec);
255     /// Constructeur de copie
256     Coordonnee1H (const Coordonnee1H& c);
257     /// Constructeur de copie pour une instance indifferenciée
258     Coordonnee1H (const CoordonneeH& c);
259
260     /// DESTRUCTEUR :
261     virtual ~Coordonnee1H ();
262
263     // METHODES :
264
265     /// Renvoie le nombre de coordonnees
266     int Dimension () const ;
267
268     /*! \brief
269     // Desallocation de la place memoire allouee
270     // fonction définie dans la classe mère générique mais qui n'a pas de
271     // sens ici
272     */
273     void Libere ();
274
275     /// Renvoie le nombre de coordonnees
276     //int Dimension () const ;
277
278     /*! \brief
279     // changement de la dimension
280     // fonction définie dans la classe mère générique mais qui n'a pas de
281     // sens ici, affiche un message d'erreur
282     */
283     void Change_dim(int dim);
284
285     /// Surcharge de l'operateur = : realise l'affectation entre deux points
286     Coordonnee1H& operator= (const Coordonnee1H& c);
287
288     /// Surcharge de l'operateur - : renvoie l'oppose d'un point
289     Coordonnee1H operator- () const ;
290
291     /*! \brief
292     // Surcharge de l'operateur - : realise la soustraction des
293     // coordonnees de deux points
294     */
295     Coordonnee1H operator- (const Coordonnee1H& c) const ;
296
297     /*! \brief
298     // Surcharge de l'operateur + : realise l'addition des
299     // coordonnees de deux points
300     */
301     Coordonnee1H operator+ (const Coordonnee1H& c) const ;
302
303     /// Surcharge de l'operateur +=
304     void operator+= (const Coordonnee1H& c);
305
306     /// Surcharge de l'operateur -=
307     void operator-= (const Coordonnee1H& c);
308
309     /// Surcharge de l'operateur *=
310     void operator*= (double val);

```

```

311
312     /// Surcharge de l'opérateur * : multiplication de coordonnées par un scalaire
313     CoordonneeIH operator* (double val) const ;
314
315     /// Surcharge de l'opérateur * : produit scalaire entre coordonnées
316     double operator* (const CoordonneeIB& c) const ;
317
318     /// produit scalaire entre coordonnées contravariantes et contravariantes
319     double ScalHH(const CoordonneeIH& c) const ;
320
321     /// Surcharge de l'opérateur / : division de coordonnées par un scalaire
322     CoordonneeIH operator/ (double val) const ;
323
324     /// Surcharge de l'opérateur /= : division de coordonnées par un scalaire
325     void operator/= (double val) ;
326
327     /*! \brief
328     // Surcharge de l'opérateur == : test d'égalité
329     // Renvoie 1 si les deux positions sont identiques
330     // Renvoie 0 sinon
331     */
332     int operator==(const CoordonneeIH& c) const;
333
334     /// conversion en Vecteur
335     Vecteur Vect() const ;
336
337     /// mise à zéro des coordonnées
338     void Zero() ;
339
340     /// Calcul de la norme euclidienne des composantes du point
341     double Norme () const ;
342
343     /// norme le vecteur coordonnée
344     CoordonneeIH& Normer () ;
345
346     /// somme de tous les composantes
347     double Somme() const ;
348     /// sortie du schemaXML: en fonction de enu
349     static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
350
351     protected :
352
353         double coordl[1];
354
355 };
356 /// @} // end of group
357
358
359 /// @addtogroup Les_classes_coordonneeI
360 /// @
361 ///
362 ///=====
363 ///!   cas des coordonnées covariantes
364 ///=====
365
366 class CoordonneeIB : public CoordonneeB
367 {
368
369     public :
370         friend class CoordonneeIH;
371
372     /// Surcharge de l'opérateur * : multiplication entre un scalaire et des coordonnées
373     inline friend CoordonneeIB operator* (double val,const CoordonneeIB& c);
374
375     // CONSTRUCTEURS :
376
377     /*! \brief
378     // Constructeur par défaut
379     // il y a initialisation des coordonnées à zéro par défaut
380     */
381     CoordonneeIB ();
382     /*! \brief
383     // Constructeur suivant un booléen
384     // quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées
385     // ceci pour aller plus vite par rapport au constructeur par défaut
386     */
387     CoordonneeIB (bool test );
388     /// Constructeur pour une localisation monodimensionnelle
389     CoordonneeIB (double x);
390     /*! \brief
391     // constructeur fonction d'une adresse mémoire où sont stockées les coordonnées
392     // ( l'existence de la place mémoire est à la charge
393     // de l'utilisateur !!).
394     */
395     CoordonneeIB (double* t);
396     /// Constructeur fonction d'un vecteur qui doit avoir une 1
397     CoordonneeIB (const Vecteur& vec);

```

```

398     /// Constructeur de copie
399     Coordonnee1B (const Coordonnee1B& c);
400     /// Constructeur de copie pour une instance indifférenciée
401     Coordonnee1B (const CoordonneeB& c);
402
403     /// DESTRUCTEUR :
404     virtual ~Coordonnee1B () ;
405
406     // METHODES :
407
408     /// Renvoie le nombre de coordonnees
409     int Dimension () const ;
410
411     /*! \brief
412     // Desallocation de la place memoire allouee
413     // fonction définie dans la classe mère générique mais qui n'a pas de
414     // sens ici
415     */
416     void Libere ();
417
418     /// Renvoie le nombre de coordonnees
419     //int Dimension () const ;
420
421     /*! \brief
422     // changement de la dimension
423     // fonction définie dans la classe mère générique mais qui n'a pas de
424     // sens ici, affiche un message d'erreur
425     */
426     void Change_dim(int dim);
427
428     /// Surcharge de l'operateur = : realise l'affectation entre deux points
429     Coordonnee1B& operator= (const Coordonnee1B& c);
430
431     /// Surcharge de l'operateur - : renvoie l'oppose d'un point
432     Coordonnee1B operator- () const ;
433
434     /*! \brief
435     // Surcharge de l'operateur - : realise la soustraction des
436     // coordonnees de deux points
437     */
438     Coordonnee1B operator- (const Coordonnee1B& c) const ;
439
440     /*! \brief
441     // Surcharge de l'operateur + : realise l'addition des
442     // coordonnees de deux points
443     */
444     Coordonnee1B operator+ (const Coordonnee1B& c) const ;
445
446     /// Surcharge de l'operateur +=
447     void operator+= (const Coordonnee1B& c);
448
449     /// Surcharge de l'operateur -=
450     void operator-= (const Coordonnee1B& c);
451
452     /// Surcharge de l'operateur *=
453     void operator*= (double val);
454
455     /// Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
456     Coordonnee1B operator* (double val) const ;
457
458     /// Surcharge de l'operateur * : produit scalaire entre coordonnees
459     double operator* (const Coordonnee1H& c) const ;
460
461     /// produit scalaire entre coordonnees covariantes et covariantes
462     double ScalBB(const Coordonnee1B& c) const ;
463
464     /// Surcharge de l'operateur / : division de coordonnees par un scalaire
465     Coordonnee1B operator/ (double val) const ;
466
467     /// Surcharge de l'operateur /= : division de coordonnees par un scalaire
468     void operator/= (double val) ;
469
470     /*! \brief
471     // Surcharge de l'operateur == : test d'egalite
472     // Renvoie 1 si les deux positions sont identiques
473     // Renvoie 0 sinon
474     */
475     int operator== (const Coordonnee1B& c) const;
476
477     /// conversion en Vecteur
478     Vecteur Vect() const ;
479
480     /// mise a zero des coordonnées
481     void Zero() ;
482
483     /// Calcul de la norme euclidienne des composantes du point
484     double Norme () const ;

```

```

485
486     /// norme le vecteur coordonnée
487     Coordonnee1B& Normer ();
488
489     /// somme de tous les composantes
490     double Somme() const ;
491     /// sortie du schemaXML: en fonction de enu
492     static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
493
494     protected :
495
496         double coord1[1];
497
498 };
499 /// @} // end of group
500
501
502 #ifndef MISE_AU_POINT
503 #include "Coordonnee1.cc"
504 #include "Coordonnee1H.cc"
505 #include "Coordonnee1B.cc"
506 #define COORDONNEE1_H_deja_inclus
507 #endif
508 // #include "Vecteur.h"
509
510 #endif

```

## 7.450 Coordonnee2.h

```

1 // FICHER : Coordonnee2.h
2 // CLASSE : Coordonnee2
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:    G RIO
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 * BUT: Les classes Coordonnee2 servent a la localisation dans l'espace *
41 *      2D des objets tels que les noeuds ou les points. Ces classes *
42 *      dérivent des Classes génériques Coordonnee.
43 *
44 *      *****
45 *****/
46
47
48 #ifndef COORDONNEE2_H
49 #define COORDONNEE2_H
50
51
52 // #include "Debug.h"
53 #include <iostream>
54 #include <fstream>
55 #include <stdlib.h>
56 #include "Sortie.h"

```

```

57 #include "Coordonnee.h"
58
59 /** @defgroup Les_classes_cooronnee2
60 *
61 * BUT: Les classes Coordonnee1 servent a la localisation dans l'espace
62 *      2D des objets tels que les noeuds ou les points. Ces classes
63 *      dérivent des Classes génériques Coordonnee.
64 * \author   Gérard Rio
65 * \version  1.0
66 * \date     23/01/97
67 * \brief    Définition des classes de type Coordonnee2, en coordonnées sans variance (ex: absolues)
68 *      ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont
69 *      une spécialisation 2D des classes générales Coordonnee
70 */
71
72
73
74 /// @addtogroup Les_classes_cooronnee2
75 /// @{
76 ///
77
78 //=====
79 //!   cas des coordonnées simples sans variance
80 //=====
81
82
83 class Coordonnee2 : public Coordonnee
84 {
85
86     public :
87     // Surcharge de l'operateur * : multiplication entre un scalaire et des coordonnées
88     inline friend Coordonnee2 operator* (double val,const Coordonnee2& c)
89         { return Coordonnee2(val*c.coord[0],val*c.coord[1]);};
90         // CONSTRUCTEURS :
91
92     /*! \brief
93     // Constructeur par défaut
94         // il y a initialisation des coordonnées à zéro par défaut
95     */
96     Coordonnee2 ();
97     /*! \brief
98     // Constructeur suivant un booleen
99         // quelque soit la valeur du booleen il n'y a pas initialisation des coordonnées
100        // ceci pour aller plus vite par rapport au constructeur par défaut
101    */
102    Coordonnee2 (bool test );
103        /// Constructeur pour une localisation bidimensionnelle
104    Coordonnee2 (double x,double y);
105    /*! \brief
106    // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
107        // ( l'existence de la place memoire est a la charge
108        // de l'utilisateur !!). */
109    Coordonnee2 (double* t);
110        /// Constructeur fonction d'un vecteur qui doit avoir une 2
111    Coordonnee2 ( const Vecteur& vec);
112        /// Constructeur de copie
113    Coordonnee2 (const Coordonnee2& c);
114        /// Constructeur de copie pour une instance indifférenciée
115    Coordonnee2 (const Coordonnee& c);
116
117        /// DESTRUCTEUR :
118    virtual ~Coordonnee2 () ;
119
120        // METHODES :
121
122        /// Renvoie le nombre de coordonnees
123    int Dimension () const ;
124
125        // Desallocation de la place memoire allouee
126        // fonction définie dans la classe mère générique mais qui n'a pas de
127        // sens ici
128    void Libere ();
129
130        // Renvoie le nombre de coordonnees
131        //int Dimension () const ;
132
133    /*! \brief
134    // changement de la dimension
135    // fonction définie dans la classe mère générique mais qui n'a pas de
136    // sens ici, affiche un message d'erreur */
137    void Change_dim(int dim);
138
139        /// Surcharge de l'operateur = : realise l'affectation entre deux points
140    Coordonnee2& operator= (const Coordonnee2& c);
141
142        /// Surcharge de l'operateur - : renvoie l'oppose d'un point

```

```

143     Coordonnee2 operator- () const ;
144
145     /*! \brief
146     // Surcharge de l'operateur - : realise la soustraction des
147     // coordonnees de deux points */
148     Coordonnee2 operator- (const Coordonnee2& c) const ;
149
150     /*! \brief
151     // Surcharge de l'operateur + : realise l'addition des
152     // coordonnees de deux points */
153     Coordonnee2 operator+ (const Coordonnee2& c) const ;
154
155     /// Surcharge de l'operateur +=
156     void operator+= (const Coordonnee2& c);
157
158     /// Surcharge de l'operateur -=
159     void operator-= (const Coordonnee2& c);
160
161     /// Surcharge de l'operateur *=
162     void operator*= (double val);
163
164     /// Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
165     Coordonnee2 operator* (double val) const ;
166
167     /// Surcharge de l'operateur * : produit scalaire entre coordonnees
168     double operator* (const Coordonnee2& c) const ;
169
170     /// Surcharge de l'operateur / : division de coordonnees par un scalaire
171     Coordonnee2 operator/ (double val) const ;
172
173     /// Surcharge de l'operateur /= : division de coordonnees par un scalaire
174     void operator/= (double val) ;
175
176     /*! \brief
177     // Surcharge de l'operateur == : test d'egalite
178     // Renvoie 1 si les deux positions sont identiques
179     // Renvoie 0 sinon */
180     int operator== (const Coordonnee2& c) const;
181
182     /// conversion en Vecteur
183     Vecteur Vect() const ;
184
185     /// mise a zero des coordonnees
186     void Zero() ;
187
188     /// Calcul de la norme euclidienne des composantes du point
189     double Norme () const ;
190
191     /// norme le vecteur coordonnee
192     Coordonnee2& Normer ();
193
194     /// somme de tous les composantes
195     double Somme() const ;
196
197     /// sortie du schemaXML: en fonction de enu
198     static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
199
200     protected :
201
202         double coord2[2];
203
204 };
205 /// @} // end of group
206
207 /// @addtogroup Les_classes_coordonnee2
208 /// @{
209 ///
210
211
212 class Coordonnee2B; // defini par la suite ( necessaire pour le produit scalaire)
213
214 //=====
215 /*! cas des coordonnees contravariantes
216 //=====
217
218 class Coordonnee2H : public CoordonneeH
219 {
220
221     public :
222     friend class Coordonnee2B;
223     // Surcharge de l'operateur * : multiplication entre un scalaire et des coordonnees
224     inline friend Coordonnee2H operator* (double val,const Coordonnee2H& c)
225     { return Coordonnee2H(val*c.coord2[0],val*c.coord2[1]);};
226
227     // CONSTRUCTEURS :
228
229     /*! \brief

```



```

230 // Constructeur par default
231 // il y a initialisation des coordonnees à zéro par défaut */
232 Coordonnee2H ();
233 /*! \brief
234 // Constructeur suivant un booleen
235 // quelque soit la valeur du booleen il n'y a pas initialisation des coordonnees
236 // ceci pour aller plus vite par rapport au constructeur par défaut */
237 Coordonnee2H (bool test );
238 /// Constructeur pour une localisation bidimensionnelle
239 Coordonnee2H (double x,double y);
240 /*! \brief
241 // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
242 // ( l'existence de la place memoire est a la charge
243 // de l'utilisateur !!). */
244 Coordonnee2H (double* t);
245 /// Constructeur fonction d'un vecteur qui doit avoir une 2
246 Coordonnee2H ( const Vecteur& vec);
247 /// Constructeur de copie
248 Coordonnee2H (const Coordonnee2H& c);
249 /// Constructeur de copie pour une instance indifférenciée
250 Coordonnee2H (const CoordonneeH& c);
251
252 /// DESTRUCTEUR :
253 virtual ~Coordonnee2H ();
254
255 // METHODES :
256
257 /// Renvoie le nombre de coordonnees
258 int Dimension () const ;
259
260 /*! \brief
261 // Desallocation de la place memoire allouee
262 // fonction définie dans la classe mère générique mais qui n'a pas de
263 // sens ici */
264 void Libere ();
265
266 /// Renvoie le nombre de coordonnees
267 //int Dimension () const ;
268
269 /*! \brief
270 // changement de la dimension
271 // fonction définie dans la classe mère générique mais qui n'a pas de
272 // sens ici, affiche un message d'erreur */
273 void Change_dim(int dim);
274
275 /// Surcharge de l'operateur = : realise l'affectation entre deux points
276 Coordonnee2H& operator= (const Coordonnee2H& c);
277
278 /// Surcharge de l'operateur - : renvoie l'oppose d'un point
279 Coordonnee2H operator- () const ;
280
281 /*! \brief
282 // Surcharge de l'operateur - : realise la soustraction des
283 // coordonnees de deux points */
284 Coordonnee2H operator- (const Coordonnee2H& c) const ;
285
286 /*! \brief
287 // Surcharge de l'operateur + : realise l'addition des
288 // coordonnees de deux points */
289 Coordonnee2H operator+ (const Coordonnee2H& c) const ;
290
291 /// Surcharge de l'operateur +=
292 void operator+= (const Coordonnee2H& c);
293
294 /// Surcharge de l'operateur -=
295 void operator-= (const Coordonnee2H& c);
296
297 /// Surcharge de l'operateur *=
298 void operator*= (double val);
299
300 /// Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
301 Coordonnee2H operator* (double val) const ;
302
303 /// Surcharge de l'operateur * : produit scalaire entre coordonnees
304 double operator* (const Coordonnee2B& c) const ;
305
306 /// produit scalaire entre coordonnees contravariantes et contravariantes
307 double ScalHH(const Coordonnee2H& c) const ;
308
309 /// Surcharge de l'operateur / : division de coordonnees par un scalaire
310 Coordonnee2H operator/ (double val) const ;
311
312 /// Surcharge de l'operateur /= : division de coordonnees par un scalaire
313 void operator/= (double val) ;
314
315 /*! \brief
316 // Surcharge de l'operateur == : test d'egalite

```

```

317         // Renvoie 1 si les deux positions sont identiques
318         // Renvoie 0 sinon */
319         int operator==(const Coordonnee2H& c) const;
320
321     /// conversion en Vecteur
322     Vecteur Vect() const ;
323
324     /// mise a zero des coordonnées
325     void Zero() ;
326
327     /// Calcul de la norme euclidienne des composantes du point
328     double Norme () const ;
329
330     /// norme le vecteur coordonnée
331     Coordonnee2H& Normer () ;
332
333     /// somme de tous les composantes
334     double Somme() const ;
335     /// sortie du schemaXML: en fonction de enu
336     static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
337
338     protected :
339
340         double coord2[2];
341
342 };
343 /// @} // end of group
344
345
346 /// @addtogroup Les_classes_coordonnee2
347 /// @{
348 ///
349
350 //=====
351 //!   cas des coordonnées covariantes
352 //=====
353
354 class Coordonnee2B : public CoordonneeB
355 {
356
357     public :
358         friend class Coordonnee2H;
359         /// Surcharge de l'opérateur * : multiplication entre un scalaire et des coordonnees
360         inline friend Coordonnee2B operator* (double val,const Coordonnee2B& c)
361             { return Coordonnee2B(val*c.coord2[0],val*c.coord2[1]);};
362         // CONSTRUCTEURS :
363
364         /*! \brief
365         // Constructeur par défaut
366         // il y a initialisation des coordonnées à zéro par défaut */
367         Coordonnee2B () ;
368         /*! \brief
369         // Constructeur suivant un booleen
370         // quelque soit la valeur du booleen il n'y a pas initialisation des coordonnées
371         // ceci pour aller plus vite par rapport au constructeur par défaut */
372         Coordonnee2B (bool test ) ;
373         /// Constructeur pour une localisation bidimensionnelle
374         Coordonnee2B (double x,double y);
375         /*! \brief
376         // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
377         // ( l'existence de la place mémoire est a la charge
378         // de l'utilisateur !!). */
379         Coordonnee2B (double* t);
380         /// Constructeur fonction d'un vecteur qui doit avoir une 2
381         Coordonnee2B ( const Vecteur& vec);
382         /// Constructeur de copie
383         Coordonnee2B (const Coordonnee2B& c);
384         /// Constructeur de copie pour une instance indifférenciée
385         Coordonnee2B (const CoordonneeB& c);
386
387         /// DESTRUCTEUR :
388         virtual ~Coordonnee2B () ;
389
390         // METHODES :
391
392         /// Renvoie le nombre de coordonnees
393         int Dimension () const ;
394
395         /*! \brief
396         // Desallocation de la place memoire allouee
397         // fonction définie dans la classe mère générique mais qui n'a pas de
398         // sens ici */
399         void Libere ();
400
401         /// Renvoie le nombre de coordonnees
402         //int Dimension () const ;
403

```

```

404  /*! \brief
405  // changement de la dimension
406  // fonction définie dans la classe mère générique mais qui n'a pas de
407  // sens ici, affiche un message d'erreur */
408  void Change_dim(int dim);
409
410  /// Surcharge de l'opérateur = : réalise l'affectation entre deux points
411  Coordonnee2B& operator= (const Coordonnee2B& c);
412
413  /// Surcharge de l'opérateur - : renvoie l'opposé d'un point
414  Coordonnee2B operator- () const ;
415
416  /*! \brief
417  // Surcharge de l'opérateur - : réalise la soustraction des
418  // coordonnées de deux points */
419  Coordonnee2B operator- (const Coordonnee2B& c) const ;
420
421  /*! \brief
422  // Surcharge de l'opérateur + : réalise l'addition des
423  // coordonnées de deux points */
424  Coordonnee2B operator+ (const Coordonnee2B& c) const ;
425
426  /// Surcharge de l'opérateur +=
427  void operator+= (const Coordonnee2B& c);
428
429  /// Surcharge de l'opérateur -=
430  void operator-= (const Coordonnee2B& c);
431
432  /// Surcharge de l'opérateur *=
433  void operator*= (double val);
434
435  /// Surcharge de l'opérateur * : multiplication de coordonnées par un scalaire
436  Coordonnee2B operator* (double val) const ;
437
438  /// Surcharge de l'opérateur * : produit scalaire entre coordonnées
439  double operator* (const Coordonnee2H& c) const ;
440
441  /// produit scalaire entre coordonnées covariantes et covariantes
442  double ScalBB(const Coordonnee2B& c) const ;
443
444  /// Surcharge de l'opérateur / : division de coordonnées par un scalaire
445  Coordonnee2B operator/ (double val) const ;
446
447  /// Surcharge de l'opérateur /= : division de coordonnées par un scalaire
448  void operator/= (double val) ;
449
450  /*! \brief
451  // Surcharge de l'opérateur == : test d'égalité
452  // Renvoie 1 si les deux positions sont identiques
453  // Renvoie 0 sinon */
454  int operator== (const Coordonnee2B& c) const;
455
456  /// conversion en Vecteur
457  Vecteur Vect() const ;
458
459  /// mise à zéro des coordonnées
460  void Zero() ;
461
462  /// Calcul de la norme euclidienne des composantes du point
463  double Norme () const ;
464
465  /// norme le vecteur coordonné
466  Coordonnee2B& Normer () ;
467
468  /// somme de tous les composantes
469  double Somme() const ;
470  /// sortie du schemaXML: en fonction de enu
471  static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
472
473  /// calcul du déterminant de deux vecteurs coordonnées
474  static double Determinant2B( const Coordonnee2B & v1, const Coordonnee2B & v2);
475
476  protected :
477
478  double coord2[2];
479
480 };
481 // @} // end of group
482
483 #ifndef MISE_AU_POINT
484 #include "Coordonnee2.cc"
485 #include "Coordonnee2H.cc"
486 #include "Coordonnee2B.cc"
487 #define COORDONNEE2_H_deja_inclus
488 #endif
489
490

```

```
491 #endif
```

## 7.451 Coordonnee3.h

```
1 // FICHER : Coordonnee3.h
2 // CLASSE : Coordonnee3
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:          23/01/97
34 *
35 *      AUTEUR:        G RIO
36 *
37 *      PROJET:        Herezh++
38 *
39 *
40 * BUT: Les classes Coordonnee3 servent a la localisation dans l'espace
41 *      3D des objets tels que les noeuds ou les points. Ces classes
42 *      dérivent des Classes génériques Coordonnee.
43 *
44 *      *****
45 *****/
46
47
48
49 #ifndef COORDONNEE3_H
50 #define COORDONNEE3_H
51
52
53 // #include "Debug.h"
54 #include <iostream>
55 #include <fstream>
56 #include <stdlib.h>
57 #include "Sortie.h"
58 #include "Coordonnee.h"
59
60 /** @defgroup Les_classes_coordonnee3
61 *
62 * BUT: Les classes Coordonnee1 servent a la localisation dans l'espace
63 *      3D des objets tels que les noeuds ou les points. Ces classes
64 *      dérivent des Classes génériques Coordonnee.
65 * \author Gérard Rio
66 * \version 1.0
67 * \date 23/01/97
68 * \brief Définition des classes de type Coordonnee3, en coordonnées sans variance (ex: absolues)
69 * ou en coordonnées locales c'est-à-dire en coordonnées covariantes ou contravariantes. Ces classes sont
70 *      une spécialisation 3D des classes générales Coordonnee
71 */
72
73
74
75 /// @addtogroup Les_classes_coordonnee3
76 /// @{
77 ///
78 //=====
79 //! cas des coordonnées simples sans variance
80 //=====
```

```
81
82
83 class Coordonnee3 : public Coordonnee
84 {
85
86     public :
87     /// Surcharge de l'operateur * : multiplication entre un scalaire et des coordonnees
88     inline friend Coordonnee3 operator* (double val,const Coordonnee3& c)
89     { return Coordonnee3(val*c.coord3[0],val*c.coord3[1],val*c.coord3[2]);};
90     // CONSTRUCTEURS :
91
92     /*! \brief
93     // Constructeur par default
94     // il y a initialisation des coordonnees à zero par default */
95     Coordonnee3 ();
96     /*! \brief
97     // Constructeur suivant un boolean
98     // quelque soit la valeur du boolean il n'y a pas initialisation des coordonnees
99     // ceci pour aller plus vite par rapport au constructeur par default */
100    Coordonnee3 (bool test );
101    /// Constructeur pour une localisation tridimensionnelle
102    Coordonnee3 (double x,double y,double z);
103    /*! \brief
104    // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
105    // ( l'existence de la place memoire est a la charge
106    // de l'utilisateur !!). */
107    Coordonnee3 (double* t);
108    /// Constructeur fonction d'un vecteur qui doit avoir une 3
109    Coordonnee3 ( const Vecteur& vec);
110    /// Constructeur de copie
111    Coordonnee3 (const Coordonnee3& c);
112    /// Constructeur de copie pour une instance indifférenciée
113    Coordonnee3 (const Coordonnee& c);
114
115    /// DESTRUCTEUR :
116    virtual ~Coordonnee3 ();
117
118    // METHODES :
119
120    /// Renvoie le nombre de coordonnees
121    int Dimension () const ;
122
123    /*! \brief
124    // Desallocation de la place memoire allouee
125    // fonction définie dans la classe mère générique mais qui n'a pas de
126    // sens ici */
127    void Libere ();
128
129    // Renvoie le nombre de coordonnees
130    //int Dimension () const ;
131
132    /*! \brief
133    // changement de la dimension
134    // fonction définie dans la classe mère générique mais qui n'a pas de
135    // sens ici, affiche un message d'erreur */
136    void Change_dim(int dim);
137
138    /// Surcharge de l'operateur = : realise l'affectation entre deux points
139    Coordonnee3& operator= (const Coordonnee3& c);
140
141    /// Surcharge de l'operateur - : renvoie l'oppose d'un point
142    Coordonnee3 operator- () const ;
143
144    /*! \brief
145    // Surcharge de l'operateur - : realise la soustraction des
146    // coordonnees de deux points */
147    Coordonnee3 operator- (const Coordonnee3& c) const ;
148
149    /*! \brief
150    // Surcharge de l'operateur + : realise l'addition des
151    // coordonnees de deux points */
152    Coordonnee3 operator+ (const Coordonnee3& c) const ;
153
154    /// Surcharge de l'operateur +=
155    void operator+= (const Coordonnee3& c);
156
157    /// Surcharge de l'operateur -=
158    void operator-= (const Coordonnee3& c);
159
160    /// Surcharge de l'operateur *=
161    void operator*= (double val);
162
163    /// Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
164    Coordonnee3 operator* (double val) const ;
165
166    /// Surcharge de l'operateur * : produit scalaire entre coordonnees
167    double operator* (const Coordonnee3& c) const ;
```

```

168
169 // Surcharge de l'operateur / : division de coordonnees par un scalaire
170     Coordonnee3 operator/ (double val) const ;
171
172 // Surcharge de l'operateur /= : division de coordonnees par un scalaire
173     void operator/= (double val) ;
174
175 /*! \brief
176 // Surcharge de l'operateur == : test d'egalite
177     // Renvoie 1 si les deux positions sont identiques
178     // Renvoie 0 sinon */
179     int operator==(const Coordonnee3& c) const;
180
181 // conversion en Vecteur
182     Vecteur Vect() const ;
183
184 // mise a zero des coordonnees
185     void Zero() ;
186
187 // Calcul de la norme euclidienne des composantes du point
188     double Norme () const ;
189
190 // norme le vecteur coordonnee
191     Coordonnee3& Normer ();
192
193 // somme de tous les composantes
194     double Somme() const ;
195
196 // sortie du schemaXML: en fonction de enu
197     static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
198
199     protected :
200         double coord3[3];
201
202 };
203 // @} // end of group
204
205 // @addtogroup Les_classes_coordonnee3
206 // @{
207 //
208 //
209
210
211 class Coordonnee3B; // defini par la suite ( necessaire pour le produit scalaire)
212
213 //=====
214 //!   cas des coordonnees contravariantes
215 //=====
216
217 class Coordonnee3H : public CoordonneeH
218 {
219     public :
220         friend class Coordonnee3B;
221         // Surcharge de l'operateur * : multiplication entre un scalaire et des coordonnees
222         inline friend Coordonnee3H operator* (double val,const Coordonnee3H& c)
223         { return Coordonnee3H(val*c.coord3[0],val*c.coord3[1],val*c.coord3[2]);};
224
225         // CONSTRUCTEURS :
226
227         /*! \brief
228         // Constructeur par default
229         // il y a initialisation des coordonnees a zero par default */
230         Coordonnee3H ();
231
232         /*! \brief
233         // Constructeur suivant un booleen
234         // quelque soit la valeur du booleen il n'y a pas initialisation des coordonnees
235         // ceci pour aller plus vite par rapport au constructeur par default */
236         Coordonnee3H (bool test );
237
238         // Constructeur pour une localisation tridimensionnelle
239         Coordonnee3H (double x,double y,double z);
240
241         /*! \brief
242         // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
243         // ( l'existence de la place memoire est a la charge
244         // de l'utilisateur !!). */
245         Coordonnee3H (double* t);
246
247         // Constructeur fonction d'un vecteur qui doit avoir une 3
248         Coordonnee3H ( const Vecteur& vec);
249
250         // Constructeur de copie
251         Coordonnee3H (const Coordonnee3H& c);
252
253         // Constructeur de copie pour une instance indifferenciee
254         Coordonnee3H (const CoordonneeH& c);
255
256         // DESTRUCTEUR :
257         virtual ~Coordonnee3H ();
258
259         // METHODES :

```

```

255
256 // Renvoie le nombre de coordonnees
257     int Dimension () const ;
258
259 /*! \brief
260 // Desallocation de la place memoire allouee
261 // fonction définie dans la classe mère générique mais qui n'a pas de
262 // sens ici */
263 void Libere ();
264
265 // Renvoie le nombre de coordonnees
266 //int Dimension () const ;
267
268 /*! \brief
269 // changement de la dimension
270 // fonction définie dans la classe mère générique mais qui n'a pas de
271 // sens ici, affiche un message d'erreur */
272     void Change_dim(int dim);
273
274 // Surcharge de l'operateur = : realise l'affectation entre deux points
275     Coordonnee3H& operator= (const Coordonnee3H& c);
276
277 // Surcharge de l'operateur - : renvoie l'oppose d'un point
278     Coordonnee3H operator- () const ;
279
280 /*! \brief
281 // Surcharge de l'operateur - : realise la soustraction des
282 // coordonnees de deux points */
283     Coordonnee3H operator- (const Coordonnee3H& c) const ;
284
285 /*! \brief
286 // Surcharge de l'operateur + : realise l'addition des
287 // coordonnees de deux points */
288     Coordonnee3H operator+ (const Coordonnee3H& c) const ;
289
290 // Surcharge de l'operateur +=
291     void operator+= (const Coordonnee3H& c);
292
293 // Surcharge de l'operateur -=
294     void operator-= (const Coordonnee3H& c);
295
296 // Surcharge de l'operateur *=
297     void operator*= (double val);
298
299 // Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
300     Coordonnee3H operator* (double val) const ;
301
302 // Surcharge de l'operateur * : produit scalaire entre coordonnees
303     double operator* (const Coordonnee3B& c) const ;
304
305 // produit scalaire entre coordonnees contravariantes et contravariantes
306     double ScalHH(const Coordonnee3H& c) const ;
307
308 // Surcharge de l'operateur / : division de coordonnees par un scalaire
309     Coordonnee3H operator/ (double val) const ;
310
311 // Surcharge de l'operateur /= : division de coordonnees par un scalaire
312     void operator/= (double val) ;
313
314 /*! \brief
315 // Surcharge de l'operateur == : test d'egalite
316 // Renvoie 1 si les deux positions sont identiques
317 // Renvoie 0 sinon */
318     int operator== (const Coordonnee3H& c) const;
319
320 // conversion en Vecteur
321     Vecteur Vect() const ;
322
323 // mise a zero des coordonnées
324     void Zero() ;
325
326 // Calcul de la norme euclidienne des composantes du point
327     double Norme () const ;
328
329 // norme le vecteur coordonnée
330     Coordonnee3H& Normer () ;
331
332 // somme de tous les composantes
333     double Somme() const ;
334
335 // sortie du schemaXML: en fonction de enu
336     static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
337
338     protected :
339
340         double coord3[3];
341

```

```

342 };
343
344 /// @} // end of group
345
346 /// @addtogroup Les_classes_coordonnee3
347 /// @{
348 ///
349 //=====
350 //!   cas des coordonnées covariantes
351 //=====
352
353 class Coordonnee3B : public CoordonneeB
354 {
355     public :
356     friend class Coordonnee3H;
357     /// Surcharge de l'opérateur * : multiplication entre un scalaire et des coordonnées
358     inline friend Coordonnee3B operator* (double val,const Coordonnee3B& c)
359     { return Coordonnee3B(val*c.coord3[0],val*c.coord3[1],val*c.coord3[2]);};
360
361     // CONSTRUCTEURS :
362
363     /*! \brief
364     // Constructeur par défaut
365     // il y a initialisation des coordonnées à zéro par défaut */
366     Coordonnee3B ();
367     /*! \brief
368     // Constructeur suivant un booléen
369     // quelque soit la valeur du booléen il n'y a pas initialisation des coordonnées
370     // ceci pour aller plus vite par rapport au constructeur par défaut */
371     Coordonnee3B (bool test );
372     /// Constructeur pour une localisation tridimensionnelle
373     Coordonnee3B (double x,double y,double z);
374     /*! \brief
375     // constructeur fonction d'une adresse memoire ou sont stockee les coordonnees
376     // ( l'existence de la place memoire est a la charge
377     // de l'utilisateur !!). */
378     Coordonnee3B (double* t);
379     /// Constructeur fonction d'un vecteur qui doit avoir une 3
380     Coordonnee3B ( const Vecteur& vec);
381     /// Constructeur de copie
382     Coordonnee3B (const Coordonnee3B& c);
383     /// Constructeur de copie pour une instance indifférenciée
384     Coordonnee3B (const CoordonneeB& c);
385
386     /// DESTRUCTEUR :
387     virtual ~Coordonnee3B ();
388
389     // METHODES :
390
391     /// Renvoie le nombre de coordonnees
392     int Dimension () const ;
393
394     /*! \brief
395     // Desallocation de la place memoire allouee
396     // fonction définie dans la classe mère générique mais qui n'a pas de
397     // sens ici */
398     void Libere ();
399
400     // Renvoie le nombre de coordonnees
401     //int Dimension () const ;
402
403     /*! \brief
404     // changement de la dimension
405     // fonction définie dans la classe mère générique mais qui n'a pas de
406     // sens ici, affiche un message d'erreur */
407     void Change_dim(int dim);
408
409     /// Surcharge de l'opérateur = : realise l'affectation entre deux points
410     Coordonnee3B& operator= (const Coordonnee3B& c);
411
412     /// Surcharge de l'opérateur - : renvoie l'opposé d'un point
413     Coordonnee3B operator- () const ;
414
415     /*! \brief
416     // Surcharge de l'opérateur - : realise la soustraction des
417     // coordonnees de deux points */
418     Coordonnee3B operator- (const Coordonnee3B& c) const ;
419
420     /*! \brief
421     // Surcharge de l'opérateur + : realise l'addition des
422     // coordonnees de deux points */
423     Coordonnee3B operator+ (const Coordonnee3B& c) const ;
424
425     /// Surcharge de l'opérateur +=
426     void operator+= (const Coordonnee3B& c);
427
428

```



```

429  // Surcharge de l'operateur -=
430      void operator-= (const Coordonnee3B& c);
431
432  // Surcharge de l'operateur *=
433      void operator*= (double val);
434
435  // Surcharge de l'operateur * : multiplication de coordonnees par un scalaire
436      Coordonnee3B operator* (double val) const ;
437
438  // Surcharge de l'operateur * : produit scalaire entre coordonnees
439      double operator* (const Coordonnee3H& c) const ;
440
441  // produit scalaire entre coordonnees covariantes et covariantes
442      double ScalBB(const Coordonnee3B& c) const ;
443
444  // Surcharge de l'operateur / : division de coordonnees par un scalaire
445      Coordonnee3B operator/ (double val) const ;
446
447  // Surcharge de l'operateur /= : division de coordonnees par un scalaire
448      void operator/= (double val) ;
449
450  /*! \brief
451  // Surcharge de l'operateur == : test d'egalite
452      // Renvoie 1 si les deux positions sont identiques
453      // Renvoie 0 sinon */
454      int operator== (const Coordonnee3B& c) const;
455
456  // conversion en Vecteur
457      Vecteur Vect() const ;
458
459  // mise a zero des coordonnees
460      void Zero() ;
461
462  // Calcul de la norme euclidienne des composantes du point
463      double Norme () const ;
464
465  // norme le vecteur coordonnée
466      Coordonnee3B& Normer ();
467
468  // somme de tous les composantes
469      double Somme() const ;
470
471  // sortie du schemaXML: en fonction de enu
472      static void SchemaXML_Coordonnee(ofstream& sort,const Enum_IO_XML enu) ;
473
474  // calcul du produit mixte de trois vecteurs coordonnées
475      static double Determinant3B( const Coordonnee3B & v1, const Coordonnee3B & v2, const Coordonnee3B &
v3);
476
477  // calcul du produit vectoriel de 2 vecteurs coordonnées
478      static Coordonnee3H Vectoriel( const Coordonnee3B & v1, const Coordonnee3B & v2);
479
480      protected :
481
482          double coord3[3];
483
484 };
485 /// @} // end of group
486
487 #ifndef MISE_AU_POINT
488     #include "Coordonnee3.cc"
489     #include "Coordonnee3H.cc"
490     #include "Coordonnee3B.cc"
491     #define COORDONNEE3_H_deja_inclus
492 #endif
493
494 #endif

```

## 7.452 Base.h

```

1 // fichier: Base.h
2
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr

```

```

16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      23/01/97
34 *
35 *   AUTEUR:    G RIO
36 *
37 *   PROJET:    Herezh++
38 *
39 *****/
40 * BUT: Les classes Base servent à définir des bases en 1D 2D 3D
41 *   qui permettent d'exprimer des coordonnées, en absolue ou en locale.
42 *   Ces bases correspondent à des bases naturelles ou des bases duales.
43 *
44 *   *****
45 *   MODIFICATIONS:
46 *   ! date !   auteur !           but
47 *   -----
48 *
49 *****/
50
51 #ifndef BASE_H
52 #define BASE_H
53
54 #include "Tableau_T.h"
55 #include "Coordonnee.h"
56 #include "Mat_pleine.h"
57
58
59 /** @defgroup Les_classes_Base
60 *
61 * BUT: Les classes Base servent à définir des bases en 1D 2D 3D
62 *   qui permettent d'exprimer des coordonnées, en absolue ou en locale.
63 *   Ces bases correspondent à des bases naturelles ou des bases duales.
64 * \author Gérard Rio
65 * \version 1.0
66 * \date 23/01/97
67 * \brief Définition des bases naturelles ou absolues,
68 *   ou des bases duales.
69 *
70 */
71
72
73
74 /// @addtogroup Les_classes_Base
75 /// @{
76 ///
77
78 class BaseH;
79 class Util;
80
81 //=====
82 //! Les base covariantes et absolus
83 //=====
84 class BaseB
85 { // surcharge de l'operator de lecture avec le type
86   friend istream & operator » (istream &, BaseB &);
87   // surcharge de l'operator d'écriture
88   friend ostream & operator « (ostream &, const BaseB &);
89
90 public :
91   // VARIABLES PUBLIQUES :
92
93   // CONSTRUCTEURS :
94
95   /*! \brief
96   // constructeur par défaut, défini la Base absolu en dimension 3
97   // les vecteurs sont unitaires, ex en dim 2 ->
98   // la base : 1,0 et 0,1 */
99   BaseB () ;
100
101   // défini la Base absolu en dimension dim
102   BaseB (int dim);

```

```

103
104     /*! \brief
105     // defini une Base relative vl(dim)
106     // c-a-d differente de la base triviale absolu */
107     BaseB (int dim, const Tableau<CoordonneeB > & vl);
108
109     /*! \brief
110     // defini une Base locale absolue (triviale) de n vecteurs
111     // de dimension dim */
112     BaseB (int dim,int n);
113
114     /// idem dessus mais avec toutes les composantes = s
115     BaseB (int dim,int n,double s);
116
117     /*! \brief
118     // defini une Base locale relative vl(dim)
119     // de n vecteurs de dimension dim */
120     BaseB (int dim,int n,const Tableau<CoordonneeB > & vl);
121
122     /// constructeur de copie
123     BaseB (const BaseB &);
124
125     /// DESTRUCTEUR :
126     ~BaseB ();
127     // METHODES PUBLIQUES :
128
129     /// surcharge de l'affectation
130     BaseB & operator = (const BaseB & aB);
131
132     // retourne la dimension des vecteurs
133     inline const int Dimension () const
134     {return dimension; };
135
136     /// retourne le nombre de vecteurs de la base
137     inline const int NbVecteur () const
138     {return nb_vecteur; };
139     /// retourne le i ieme vecteurs en lecture only
140     const CoordonneeB & operator () (int i) const;
141     /// retourne le i ieme vecteurs en I/O
142     CoordonneeB& CoordoB(int i) ;
143     /// acces directe contrôlé à des vecteurs sans variance: uniquement en lecture
144     const Coordonnee & Coordo(int i) const ;
145
146     /// affichage à l'écran des infos
147     void Affiche() const;
148
149     /*! \brief
150     // affectation trans_variance: utile pour une recopie de mêmes valeurs
151     // mais ici l'appel est explicite donc a priori on sait ce que l'on fait
152     // il n'y a pas de redimensionnement, donc la dimension et le nombre des vecteurs
153     // doivent être identiques */
154     void Affectation_trans_variance(const BaseH& aH);
155
156     /*! \brief
157     // changement de base
158     // on suppose que this(i) correspond aux coordonnées dans un premier repère I_a
159     // IpH correspond aux coordonnées dans I_a d'un nouveau repère
160     // en sortie: apB(i) correspond aux coordonnées dans ipB de this(i) */
161     void Change_repere(const BaseH& IpH, BaseB& apB);
162
163     /*! \brief
164     // la méthode qui suit a pour objectif de calculer les vecteurs de la base naturelle finale
165     // associée à un paramétrage cartésien initial */
166
167     /// donc soit connue une base naturelle \hat{g}_i associée à un paramétrage theta^i
168     /// \hat{g}_i est représenté par les vecteurs de this, et représente la situation déformée
169     /// soit la base duale de g_i : gammaH^i = g^i c-a-d la base duale en situation non déformée
170     /// maintenant: si on considère les coordonnées initiales X^i on cherche
171     /// les vecteurs des bases naturelles associées à X^i avant g'_i et après déformation \hat{g}'_i
172     /// \n on a (cf. annexe dans le document théorique d'Herezh)
173     /// \n g'_i = I_i par définition et
174     /// \n \hat{g}'_i = (\vec{I}_i . \vec{g}^j) ~\hat{g}_j c-a-d
175     /// \n \hat{g}'_i = gammaH^j(i) * (*this)(j)
176     /// \n c'est le résultat de la méthode qui suit
177
178     /// \n changement de base: retourne apB(a) = (*this)(j) . gamma^j_a
179     /// il faut que le nombre de vecteur de apB soit identique à la dimension de gammaH
180     /// et que le nombre de vecteur de gammaH soit identique au nombre de vecteur de this
181     /// et que la dimension de apB soit identique à la dimension de this
182     void ChangeBase_theta_vers_Xi(BaseB& apB, const BaseH& gammaH);
183
184     /*! \brief
185     // la méthode calcule à partir de this qui correspond à g_alpha dans I_a:
186     // apB : g_alpha dans I'^alpha
187     // apH : g^alpha dans I'^alpha
188     // IpB : I'_beta dans I_a */
189

```

```

190     /// soient une base globale orthonormée : I_a et une base locale orthonormée I'_alpha
191     /// pour l'instant: I_a est en 3 dimensions et alpha varie de 1 à 2
192     /// soient une base naturelle g_alpha et duale associée g^beta
193     /// telles que : I'_alpha appartient à l'espace des g_alpha ou g^beta
194     /// \n A) on a:
195     /// g^alpha = gamma(alpha,beta) * I'^beta c-a-d : gamma^alpha_beta
196     /// \n NB: 1) comme I'est orthonormée: I'^beta = I'_beta
197     /// \n      2) chaque ligne de gamma représente un g^alpha
198     /// \n B) dans ce contexte on a:
199     /// inverse de gamma -> une matrice beta et chaque ligne de beta représente
200     /// les coordonnées de g_alpha dans I'^alpha
201     /// c-a-d g_alpha = alpha (delta,alpha) * I'_delta
202     /// \n C) on peut également calculer les coordonnées de I'_beta dans le repère globale
203     /// \n I'_beta = gamma(alpha,beta) * g_alpha
204     ///
205     void ChangeBase_curviligne( const Mat_pleine& gamma, BaseB& apB , BaseH& apH, BaseB& IpB ) const;
206
207     /*! \brief
208     // calcul des composantes de coordonnées locales dans la base absolue
209     // en argument : A -> une reference sur les coordonnées résultat qui peut avoir une dimension
210     // différente des coordonnées locale, retour d'une reference sur A */
211     Coordonnee & BaseAbsolue(Coordonnee & A,const CoordonneeH & B) const;
212
213     /*! \brief
214     // une partie des vecteurs de B est affectée à this,
215     // si this contient plus de vecteur que B, les autres vecteurs sont mis à 0 ou non
216     // suivant la valeur de plusZero: = false: les autres vecteurs sont inchangées,
217     // plusZero = true: les autres vecteurs sont mis à 0 */
218     void Affectation_partielle(int nb_vecteur_a_affecter, const BaseB & B,bool plusZero);
219
220 private :
221     // VARIABLES PROTEGEES :
222     int dimension; // dimension des vecteurs de la Base
223     int nb_vecteur; // nombre de vecteur de la base
224     Tableau<CoordonneeB > v ; // pointeur des vecteurs covariants de la Base
225     Tableau<Coordonnee > v_sans ; // pointeur des vecteurs sans variance de la Base
226
227     /// on met à la même place les v_sans que les v
228     void Meme_place_cooronnee();
229 };
230 /// @} // end of group
231
232 /// @addtogroup Les_classes_Base
233 /// @{
234 ///
235
236 //=====
237 ///! Les base duales
238 //=====
239
240 class BaseH
241 { /// surcharge de l'operator de lecture avec le type
242     friend istream & operator » (istream &, BaseH &);
243     /// surcharge de l'operator d'écriture
244     friend ostream & operator « (ostream &, const BaseH &);
245
246 public :
247     // VARIABLES PUBLIQUES :
248
249     // CONSTRUCTEURS :
250     /*! \brief
251     // par défaut, defini une Base absolu en dimension 3
252     // les vecteurs sont unitaires, ex en dim 2 ->
253     // la base : 1,0 et 0,1 */
254     BaseH () ;
255     /// defini une Base absolu en dimension dim
256     BaseH (int dim);
257     /*! \brief
258     // defini une Base relative vl(dim)
259     // c-a-d differente de la base triviale absolue */
260     BaseH (int dim, const Tableau<CoordonneeH > & vl);
261     /*! \brief
262     // defini une Base locale absolue (triviale) de n vecteurs
263     // de dimension dim */
264     BaseH (int dim,int n);
265     /// idem dessus mais avec toutes les composantes = s
266     BaseH (int dim,int n,double s);
267     /// defini une Base locale relative vl(dim) de n vecteurs de dimension dim
268     BaseH (int dim,int n,const Tableau<CoordonneeH >& vl);
269     /// constructeur de copie
270     BaseH (const BaseH & ) ;
271     /// DESTRUCTEUR :
272     ~BaseH ();
273     // METHODES PUBLIQUES :
274
275     /// surcharge de l'affectation
276     BaseH & operator = (const BaseH & aB);

```

```

277 // retourne la dimension des vecteurs
278 inline const int Dimension () const
279 {return dimension; };
280 // retourne le nombre de vecteurs de la base
281 inline const int NbVecteur () const
282 {return nb_vecteur; };
283 // retourne le i ieme vecteurs en lecture only
284 const CoordonneeH & operator () (int i) const;
285 // retourne le i ieme vecteurs en I/O
286 CoordonneeH& CoordoH(int i) ;
287 // acces directe contrôlé à des vecteurs sans variance: uniquement en lecture
288 const Coordonnee & Coordo(int i) const ;
289
290 // affichage à l'écran des infos
291 void Affiche() const;
292
293 /*! \brief
294 // affectation trans_variance: utile pour une recopie de mêmes valeurs
295 // mais ici l'appel est explicite donc a priori on sait ce que l'on fait
296 // il n'y a pas de redimensionnement, donc la dimension et le nombre des vecteurs
297 // doivent être identiques */
298 void Affectation_trans_variance(const BaseB& aB);
299
300 /*! \brief
301 // changement de base
302 // on suppose que this(i) correspond aux coordonnées dans un premier repère I_a
303 // IpB correspond aux coordonnées dans I_a d'un nouveau repère
304 // en sortie: apH(i) correspond aux coordonnées dans ipH de this(i) */
305 void Change_repere(const BaseB& IpB, BaseH& apH);
306
307 /*! \brief
308 // la méthode qui suit a pour objectif de calcul les vecteurs de la base naturelle finale
309 // associée à un paramétrage cartésien initiale */
310
311 // donc soit connue une base duale \hat{g}^i associée à un paramétrage theta^i
312 // \hat{g}^i est représenté par les vecteurs de this, et représente la situation déformée
313 // soit la base naturel de g_i : betaB_i = g_i c-à-d la base naturelle en situation non déformée
314 // maintenant: si on considère les coordonnées initiales X^i on cherche
315 // les vecteurs des bases duales associées à X^i avant g'^i et après déformation \hat{g}'^i
316 // on a (cf. annexe dans le document théorique d'Herezh)
317 // \n g'^i = g'_i = I_i = I^i par définition et
318 // \n \hat{g}'^i = (\vec{I}_i . \vec{g}_j) ~\hat{g}^j c-a-d
319 // \n \hat{g}'^i = betaB^j(i) * (*this)(j)
320 // \n c'est le résultat de la méthode qui suit
321
322 // \n changement de base: retourne apH(a) = (*this)(j) . beta_j^a
323 // il faut que le nombre de vecteur de apH soit identique à la dimension de betaB
324 // et que le nombre de vecteur de betaB soit identique au nombre de vecteur de this
325 // et que la dimension de apB soit identique à la dimension de this
326 void ChangeBase_theta_vers_Xi(BaseH& apH, const BaseB& betaB);
327
328 /*! \brief
329 // la méthode calcule à partir de this qui correspond à g^alpha dans I_a:
330 // apB : g_alpha dans I'^alpha
331 // apH : g^alpha dans I'^alpha
332 // IpH : I'^beta dans I_a */
333
334 // soient une base globale orthonormée : I_a et une base locale orthonormée I'_alpha
335 // pour l'instant: I_a est en 3 dimensions et alpha varie de 1 à 2
336 // soient une base naturelle g_alpha et duale associée g^beta
337 // telles que : I'_alpha appartient à l'espace des g_alpha ou g^beta
338 // \n A) on a:
339 // g^alpha = gamma(alpha,beta) * I'^beta c-a-d : gamma^alpha_beta
340 // \n NB: 1) comme I'est orthonormée: I'^beta = I'_beta
341 // 2) chaque ligne de gamma représente un g^alpha
342 // \n B) dans ce contexte on a:
343 // inverse de gamma -> une matrice beta et chaque ligne de beta représente
344 // les coordonnées de g_alpha dans I'^alpha
345 // c-a-d g_alpha = alpha (delta,alpha) * I'_delta
346 // \n C) on peut également calculer les coordonnées de I'_beta dans le repère globale
347 // \n I'_beta = gamma(alpha,beta) * g_alpha
348 //
349 void ChangeBase_curviligne( const Mat_pleine& gamma, BaseB& apB , BaseH& apH, BaseH& IpH ) const;
350
351 /*! \brief
352 // calcul des composantes de coordonnées locales dans la base absolue
353 // en argument : A -> une reference sur les coordonnées résultat qui peut avoir une dimension
354 // différente des coordonnées locale, retour d'une reference sur A */
355 Coordonnee & BaseAbsolue(Coordonnee & A,const CoordonneeB & B) const;
356
357 /*! \brief
358 // une partie des vecteurs de B est affectée à this,
359 // si this contient plus de vecteur que B, les autres vecteurs sont mis à 0 ou non
360 // suivant la valeur de plusZero:= false: les autres vecteurs sont inchangées,
361 // plusZero = true: les autres vecteurs sont mis à 0 */
362 void Affectation_partielle(int nb_vecteur_a_affecter, const BaseH & B,bool plusZero);
363

```

```

364 private :
365     // VARIABLES PROTEGEES :
366     int dimension; // dimension des vecteurs de la Base
367     int nb_vecteur; // nombre de vecteur de la base
368     Tableau<CoordonneeH > v ; // pointeur des vecteurs contravariant de la Base
369     Tableau<Coordonnee > v_sans ; // pointeur des vecteurs sans variance de la Base
370
371     // on met à la même place les v_sans que les v
372     void Meme_place_cooronnee();
373 };
374 /// @} // end of group
375
376 /// @addtogroup Les_classes_Base
377 /// @{
378
379 //=====
380 ///! un groupe de 3 bases covariantes et absolus à 0, t et tdt
381 //=====
382 /// il s'agit ici essentiellement d'un conteneur pour optimiser le stockage
383 //=====
384
385 class BaseB_0_t_tdt
386 { // surcharge de l'operator de lecture avec le type
387     friend istream & operator » (istream &, BaseB_0_t_tdt &);
388     // surcharge de l'operator d'écriture
389     friend ostream & operator « (ostream &, const BaseB_0_t_tdt &);
390
391 public :
392     // VARIABLES PUBLIQUES :
393
394     // CONSTRUCTEURS :
395
396     /// par défaut, défini les Bases en absolu en dimension 3
397     /// les vecteurs sont unitaires, ex en dim 2 ->
398     /// la base : 1,0 et 0,1
399     BaseB_0_t_tdt () :
400         baseB_0(),baseB_t(), baseB() {};
401     /// défini les Bases en absolu en dimension dim
402     BaseB_0_t_tdt (int dim) :
403         baseB_0(dim),baseB_t(dim), baseB(dim) {};
404     /// défini 3 Bases relatives v1(dim)
405     /// c-a-d différente de la base triviale absolu
406     BaseB_0_t_tdt (int dim, const Tableau<CoordonneeB > & v1) :
407         baseB_0(dim,v1),baseB_t(dim,v1), baseB(dim,v1) {};
408     /// défini les Bases locale absolue (triviale) de n vecteurs
409     /// de dimension dim
410     BaseB_0_t_tdt (int dim,int n) :
411         baseB_0(dim,n),baseB_t(dim,n), baseB(dim,n) {};
412     /// idem dessus mais avec toutes les composantes = s
413     BaseB_0_t_tdt (int dim,int n,double s) :
414         baseB_0(dim,n,s),baseB_t(dim,n,s), baseB(dim,n,s) {};
415     /// défini les Bases locale relative v1(dim)
416     /// de n vecteurs de dimension dim
417     BaseB_0_t_tdt (int dim,int n,const Tableau<CoordonneeB > & v1) :
418         baseB_0(dim,n,v1),baseB_t(dim,n,v1), baseB(dim,n,v1) {};
419     /// constructeur de copie
420     BaseB_0_t_tdt (const BaseB_0_t_tdt & b):
421         baseB_0(b.baseB_0),baseB_t(b.baseB_t), baseB(b.baseB) {};
422
423     /// DESTRUCTEUR :
424     ~BaseB_0_t_tdt () {};
425     // METHODES PUBLIQUES :
426     /// surcharge de l'affectation
427     BaseB_0_t_tdt & operator = (const BaseB_0_t_tdt & b)
428     {baseB_0 = b.baseB_0; baseB_t = b.baseB_t; baseB = b.baseB;
429     };
430
431     /// récupération de la base,
432     /// a) en constant, la base actuelle
433     const BaseB& Const_BaseB_Noed() const {return baseB;};
434     /// b) en constant, la base à t
435     const BaseB& Const_BaseB_Noed_t() const {return baseB_t;};
436     /// c) en constant, la base initiale
437     const BaseB& Const_BaseB_Noed_0() const {return baseB_0;};
438
439     /// d) en lecture écriture, la base actuelle
440     BaseB& BaseB_Noed() {return baseB;};
441     /// e) en lecture écriture, la base à t
442     BaseB& BaseB_Noed_t() {return baseB_t;};
443     /// f) en lecture écriture, la base à 0
444     BaseB& BaseB_Noed_0() {return baseB_0;};
445
446
447 private :
448     BaseB baseB_0; // à 0
449     BaseB baseB_t; // à t
450     BaseB baseB; // actuelle

```

```

451 };
452 /// @} // end of group
453
454 /// @addtogroup Les_classes_Base
455 /// @{
456 ///
457
458
459 //=====
460 //! un groupe de 3 bases contravariant et absolus à 0, t et tdt
461 //=====
462 /// il s'agit ici essentiellement d'un conteneur pour optimiser le stockage
463 //=====
464
465 class BaseH_0_t_tdt
466 { /// surcharge de l'operator de lecture avec le type
467     friend istream & operator > (istream & ent, BaseH_0_t_tdt &)
468     {
469
470     };
471     /// surcharge de l'operator d'écriture
472     friend ostream & operator < (ostream & sort, const BaseH_0_t_tdt &);
473
474 public :
475     // VARIABLES PUBLIQUES :
476
477     /// CONSTRUCTEURS :
478     /// par défaut, défini les Bases en absolu en dimension 3
479     /// les vecteurs sont unitaires, ex en dim 2 ->
480     /// la base : 1,0 et 0,1
481     BaseH_0_t_tdt () :
482         baseH_0(),baseH_t(), baseH() {};
483     /// défini les Bases en absolu en dimension dim
484     BaseH_0_t_tdt (int dim) :
485         baseH_0(dim),baseH_t(dim), baseH(dim) {};
486     /// défini 3 Bases relatives vl(dim)
487     /// c-a-d différente de la base triviale absolu
488     BaseH_0_t_tdt (int dim, const Tableau<CoordonneeH > & vl) :
489         baseH_0(dim,vl),baseH_t(dim,vl), baseH(dim,vl) {};
490     /// défini les Bases locale absolue (triviale) de n vecteurs
491     /// de dimension dim
492     BaseH_0_t_tdt (int dim,int n) :
493         baseH_0(dim,n),baseH_t(dim,n), baseH(dim,n) {};
494     /// idem dessus mais avec toutes les composantes = s
495     BaseH_0_t_tdt (int dim,int n,double s) :
496         baseH_0(dim,n,s),baseH_t(dim,n,s), baseH(dim,n,s) {};
497     /// défini les Bases locale relative vl(dim)
498     /// de n vecteurs de dimension dim
499     BaseH_0_t_tdt (int dim,int n,const Tableau<CoordonneeH > & vl) :
500         baseH_0(dim,n,vl),baseH_t(dim,n,vl), baseH(dim,n,vl) {};
501     /// constructeur de copie
502     BaseH_0_t_tdt (const BaseH_0_t_tdt & b) :
503         baseH_0(b.baseH_0),baseH_t(b.baseH_t), baseH(b.baseH) {};
504
505     /// DESTRUCTEUR :
506     ~BaseH_0_t_tdt () {};
507     // METHODES PUBLIQUES :
508     /// surcharge de l'affectation
509     BaseH_0_t_tdt & operator = (const BaseH_0_t_tdt & b)
510     {baseH_0 = b.baseH_0; baseH_t = b.baseH_t; baseH = b.baseH; return *this;
511     };
512
513     /// récupération de la base,
514     /// a) en constant, la base actuelle
515     const BaseH& Const_BaseH_Noeud() const {return baseH;};
516     /// b) en constant, la base à t
517     const BaseH& Const_BaseH_Noeud_t() const {return baseH_t;};
518     /// c) en constant, la base initiale
519     const BaseH& Const_BaseH_Noeud_0() const {return baseH_0;};
520
521     /// d) en lecture écriture, la base actuelle
522     BaseH& BaseH_Noeud() {return baseH;};
523     /// e) en lecture écriture, la base à t
524     BaseH& BaseH_Noeud_t() {return baseH_t;};
525     /// f) en lecture écriture, la base à 0
526     BaseH& BaseH_Noeud_0() {return baseH_0;};
527
528
529 private :
530     BaseH baseH_0; // à 0
531     BaseH baseH_t; // à t
532     BaseH baseH; // actuelle
533 };
534 /// @} // end of group
535
536
537 #ifndef MISE_AU_POINT

```

```

538 #include "Base.cc"
539 #define BASE_HetB_deja_inclus
540 #endif
541
542 #endif

```

## 7.453 Base3D3.h

```

1 // fichier: Base3D3.h
2
3
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:    G RIO
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 *      BUT:  definition des bases de 3 vecteurs de dim 3
41 *      qui permettent d'exprimer des coordonnées, en absolue ou en locale.
42 *      Ces bases correspondent à des bases naturelles ou des bases duales.
43 *
44 *      *****
45 *      MODIFICATIONS:
46 *      ! date !  auteur !      but
47 *      -----
48 *
49 *****/
50
51 #ifndef BASE3D3_H
52 #define BASE3D3_H
53
54 #include "Coordonnee3.h"
55 #include "Tableau_T.h"
56
57
58
59 /** @defgroup Les_classes_Base3D3
60 *
61 * BUT: Les classes Base3D3 servent à définir des bases de 3 vecteurs de dim 3
62 * qui permettent d'exprimer des coordonnées, en absolue ou en locale.
63 * Ces bases correspondent à des bases naturelles ou des bases duales.
64 * \author Gérard Rio
65 * \version 1.0
66 * \date 23/01/97
67 * \brief Définition des bases naturelles ou absolues
68 * ou des bases duales, de 3 vecteurs de dim 3.
69 *
70 */
71
72
73
74 /// @addtogroup Les_classes_Base3D3
75 /// @{
76 ///
77 //=====

```



```

78 /// Les base covariantes et absolus
79 //=====
80
81 class Base3D3B
82 { /// surcharge de l'operator de lecture avec le type
83     friend istream & operator » (istream &, Base3D3B &);
84     /// surcharge de l'operator d'écriture
85     friend ostream & operator « (ostream &, const Base3D3B &);
86
87 public :
88     // VARIABLES PUBLIQUES :
89
90     /// CONSTRUCTEURS :
91     /// par défaut, défini une Base de coordonnées nulles
92     Base3D3B () ;
93     /// défini une Base unitaire en dimension 3 :100,010,001
94     Base3D3B (bool ) ;
95     /// défini une Base donnée a partir d'un tableau
96     Base3D3B ( const Tableau<Coordonnee3B > & v1);
97     /// défini une Base donnée a partir de trois vecteur
98     Base3D3B ( const Coordonnee3B& v1,const Coordonnee3B& v2,const Coordonnee3B& v3);
99     /// constructeur de copie
100    Base3D3B (const Base3D3B &) ;
101    /// DESTRUCTEUR :
102    ~Base3D3B ();
103    /// METHODES PUBLIQUES :
104    /// surcharge de l'affectation
105    Base3D3B & operator = (const Base3D3B & aB);
106    /// retourne la dimension des Coordonnee3s
107    inline const int Dimension ()
108        {return 3; };
109    /// retourne le nombre de Coordonnee3s de la base
110    inline const int NbCoordonnee ()
111        {return 3; };
112    /// retourne le i ieme vecteur en I/O
113    Coordonnee3B & operator () (int i);
114    /// retourne le i ieme vecteur en lecture only
115    Coordonnee3B operator () (int i) const ;
116    /// retourne la j ieme composante du i ieme vecteur en I/O
117    double & operator () (int i,int j);
118    /// retourne la j ieme composante du i ieme vecteur en lecture only
119    double operator () (int i,int j) const ;
120
121
122 private :
123     // VARIABLES PROTEGEES :
124     // Coordonnee3 covariantes de la Base
125     // pour pouvoir initialiser directement on drfini trois valeurs plutqt
126     // qu'un tableau
127     Coordonnee3B v1,v2,v3;
128     Coordonnee3B * v[3];
129 };
130
131 /// @} // end of group
132
133 /// @addtogroup Les_classes_Base3D3
134 /// @{
135 ///
136 //=====
137 /// Les base duales
138 //=====
139
140 class Base3D3H
141 { /// surcharge de l'operator de lecture avec le type
142     friend istream & operator » (istream &, Base3D3H &);
143     /// surcharge de l'operator d'écriture
144     friend ostream & operator « (ostream &, const Base3D3H &);
145
146 public :
147     // VARIABLES PUBLIQUES :
148
149     /// CONSTRUCTEURS :
150     /// par défaut, défini une Base de coordonnées nulles
151     Base3D3H () ;
152     /// défini une Base unitaire en dimension 3 :100,010,001
153     Base3D3H (bool ) ;
154     /// défini une Base donnée a partir d'un tableau
155     Base3D3H ( const Tableau<Coordonnee3H > & v1);
156     /// défini une Base donnée a partir de trois vecteur
157     Base3D3H ( const Coordonnee3H& v1,const Coordonnee3H& v2,const Coordonnee3H& v3);
158     /// constructeur de copie
159     Base3D3H (const Base3D3H &) ;
160     /// DESTRUCTEUR :
161     ~Base3D3H ();
162     /// METHODES PUBLIQUES :
163     /// surcharge de l'affectation
164     Base3D3H & operator = (const Base3D3H & aB);

```

```

165 // retourne la dimension des Coordonnee3s
166 inline const int Dimension ()
167 {return 3; };
168 // retourne le nombre de Coordonnee3s de la base
169 inline const int NbCoordonnee ()
170 {return 3; };
171 // retourne le i ieme vecteur en I/O
172 Coordonnee3H & operator () (int i);
173 // retourne le i ieme vecteur en lecture only
174 Coordonnee3H operator () (int i) const ;
175 // retourne la j ieme composante du i ieme vecteur en I/O
176 double & operator () (int i,int j);
177 // retourne la j ieme composante du i ieme vecteur en lecture only
178 double operator () (int i,int j) const ;
179
180
181 private :
182 // VARIABLES PROTEGEES :
183 // Coordonnee3 covariantes de la Base
184 // pour pouvoir initialiser directement on defini trois valeurs plutot
185 // qu'un tableau
186 Coordonnee3H v1,v2,v3;
187 Coordonnee3H * v[3];
188 };
189 /// @} // end of group
190
191 #ifndef MISE_AU_POINT
192 #include "Base3D3.cc"
193 #define BASE3D3_H_deja_inclus
194 #endif
195
196
197 #endif

```

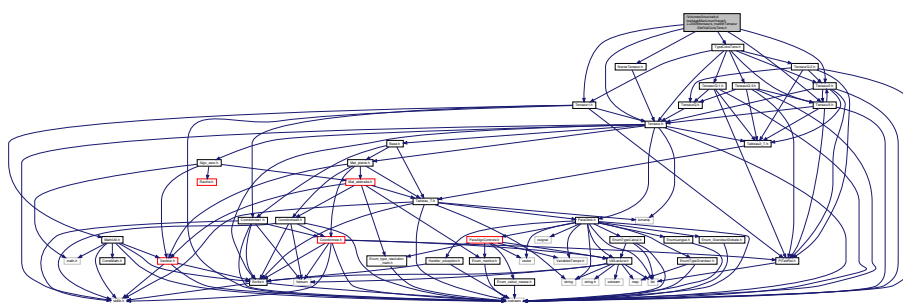
## 7.454 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/tenseurs\_mai99/Tenseur/DefValConsTens.h

```

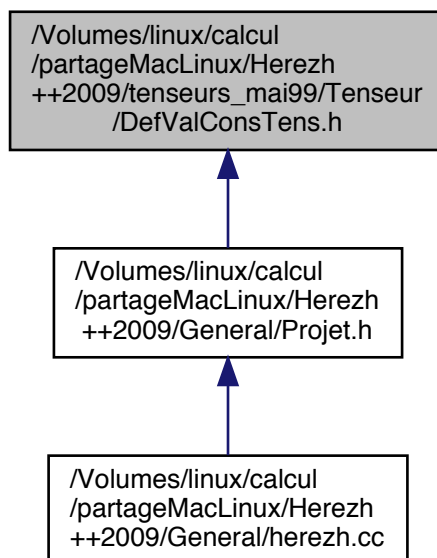
#include "NevezTenseur.h"
#include "Tenseur.h"
#include "Tenseur1.h"
#include "Tenseur2.h"
#include "Tenseur3.h"
#include "TypeConsTens.h"

```

Graphe des dépendances par inclusion de DefValConsTens.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

— void `ConstantesTenseur` (int dim)

### 7.454.1 Description détaillée

définition d'une fonction qui crée des constantes de tenseur utilisables globalement

### 7.454.2 Documentation des fonctions

#### 7.454.2.1 ConstantesTenseur()

```
void ConstantesTenseur (
    int dim )
```

la fonction qui définit les tenseurs constants d'une dimension "dim" utilisables globalement

## 7.455 DefValConsTens.h

[Aller à la documentation de ce fichier.](#)

```

1 /** \file DefValConsTens.h
2  * définition d'une fonction qui crée des constantes de tenseur utilisables globalement
3  */
4
5
6
7  // This file is part of the Herezh++ application.
8  //
9  // The finite element software Herezh++ is dedicated to the field
10 // of mechanics for large transformations of solid structures.
11 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
12 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
13 //
```

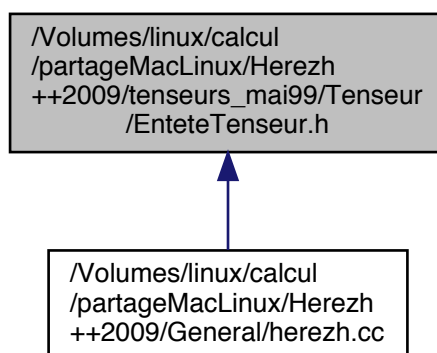
```

14 // Herezh++ is distributed under GPL 3 license ou ultérieure.
15 //
16 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
17 // AUTHOR : Gérard Rio
18 // E-MAIL : gerardrio56@free.fr
19 //
20 // This program is free software: you can redistribute it and/or modify
21 // it under the terms of the GNU General Public License as published by
22 // the Free Software Foundation, either version 3 of the License,
23 // or (at your option) any later version.
24 //
25 // This program is distributed in the hope that it will be useful,
26 // but WITHOUT ANY WARRANTY; without even the implied warranty
27 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
28 // See the GNU General Public License for more details.
29 //
30 // You should have received a copy of the GNU General Public License
31 // along with this program. If not, see <https://www.gnu.org/licenses/>.
32 //
33 // For more information, please consult: <https://herezh.irdl.fr/>.
34
35
36 #ifndef CONSTATESTENSEUR_H
37 #define CONSTATESTENSEUR_H
38 // pour la definition de nouveaux tenseurs
39 #include "NevezTenseur.h"
40 #include "Tenseur.h"
41 #include "Tenseur1.h"
42 #include "Tenseur2.h"
43 #include "Tenseur3.h"
44 // def des constantes tenseur
45 # include "TypeConsTens.h"
46
47 /**
48 * la fonction qui définit les tenseurs constants d'une dimension "dim" utilisables globalement
49 */
50 void ConstantesTenseur(int dim);
51
52 #endif

```

## 7.456 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/tenseurs\_mai99/Tenseur/EnteteTenseur.h

Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### 7.456.1 Description détaillée

définition de différentes grandeurs, en particulier tensorielles, utilisables globalement

## 7.457 EnteteTenseur.h

[Aller à la documentation de ce fichier.](#)

```

1 /** \file EnteteTenseur.h
2 * définition de différentes grandeurs, en particulier tensorielles, utilisables globalement
3 */
4
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 //=====
36 // fichier d'entete des tenseurs. A ne placer qu'une fois dans le fichier
37 // du programme principal
38 //=====
39
40 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du second ordre
41 PtTenseurHH * LesMaillonsHH::maille = NULL ;
42 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du second ordre
43 PtTenseurBB * LesMaillonsBB::maille = NULL ;
44 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du second ordre
45 PtTenseurHB * LesMaillonsHB::maille = NULL ;
46 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du second ordre
47 PtTenseurBH * LesMaillonsBH::maille = NULL ;
48 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre
49 PtTenseurHHHH * LesMaillonsHHHH::maille = NULL ;
50 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre
51 PtTenseurBBBB * LesMaillonsBBBB::maille = NULL ;
52 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre
53 PtTenseurHHBB * LesMaillonsHHBB::maille = NULL ;
54 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre
55 PtTenseurBBHH * LesMaillonsBBHH::maille = NULL ;
56 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre
57 PtTenseurBHHB * LesMaillonsBHHB::maille = NULL ;
58 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre
59 PtTenseurHBHB * LesMaillonsHBHB::maille = NULL ;
60 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre
61 PtTenseurBHHH * LesMaillonsBHHH::maille = NULL ;
62 /// initialisation de la variable static pour la gestion des tenseurs intermediaires du quatrieme ordre
63 PtTenseurHBBH * LesMaillonsHBBH::maille = NULL ;
64
65 /// initialisation d'une instance d'outil permettant la recherche de zero
66 Algo_zero TenseurHB::alg_zero;
67 /// initialisation d'une instance d'outil permettant la recherche de zero
68 Algo_zero TenseurBH::alg_zero;
69
70 /// initialisation des tableaux d'index pour tenseur du second ordre
71 const Tenseur3HH::ChangementIndex Tenseur3HH::cdex;
72 /// initialisation des tableaux d'index pour tenseur du second ordre
73 const Tenseur_ns3HH::ChangementIndex Tenseur_ns3HH::cdex;
74 /// initialisation des tableaux d'index pour tenseur du second ordre
75 const Tenseur3BB::ChangementIndex Tenseur3BB::cdex;
76 /// initialisation des tableaux d'index pour tenseur du second ordre
77 const Tenseur_ns3BB::ChangementIndex Tenseur_ns3BB::cdex;
78 /// initialisation des tableaux d'index pour tenseur du second ordre
79 const Tenseur3HB::ChangementIndex Tenseur3HB::cdex;
80 /// initialisation des tableaux d'index pour tenseur du second ordre
81 const Tenseur3BH::ChangementIndex Tenseur3BH::cdex;
82
83 /// initialisation des tableaux d'index pour tenseur du second ordre

```

```

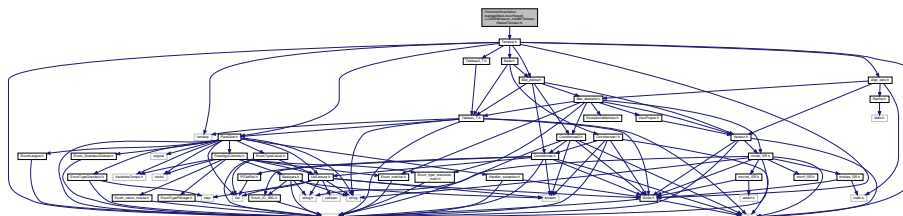
84 const Tenseur2HH::ChangementIndex Tenseur2HH::cdex;
85 /// initialisation des tableaux d'index pour tenseur du second ordre
86 const Tenseur_ns2HH::ChangementIndex Tenseur_ns2HH::cdex;
87 /// initialisation des tableaux d'index pour tenseur du second ordre
88 const Tenseur2BB::ChangementIndex Tenseur2BB::cdex;
89 /// initialisation des tableaux d'index pour tenseur du second ordre
90 const Tenseur_ns2BB::ChangementIndex Tenseur_ns2BB::cdex;
91 /// initialisation des tableaux d'index pour tenseur du second ordre
92 const Tenseur2HB::ChangementIndex Tenseur2HB::cdex;
93 /// initialisation des tableaux d'index pour tenseur du second ordre
94 const Tenseur2BH::ChangementIndex Tenseur2BH::cdex;
95
96 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3
97 const Tenseur3HHHH::ChangementIndex Tenseur3HHHH::cdex3HHHH;
98 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3
99 const Tenseur3BBBB::ChangementIndex Tenseur3BBBB::cdex3BBBB;
100 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3
101 const Tenseur3HHBB::ChangementIndex Tenseur3HHBB::cdex3HHBB;
102 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3
103 const Tenseur3BBHH::ChangementIndex Tenseur3BBHH::cdex3BBHH;
104
105 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3
106 const TenseurQ3_troisSym_HHHH::ChangementIndex TenseurQ3_troisSym_HHHH::cdex;
107 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 3
108 const TenseurQ3_troisSym_BBBB::ChangementIndex TenseurQ3_troisSym_BBBB::cdex;
109
110 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 1
111 const Tenseur1HHHH::ChangementIndex Tenseur1HHHH::cdex1HHHH;
112 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 1
113 const Tenseur1BBBB::ChangementIndex Tenseur1BBBB::cdex1BBBB;
114 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 1
115 const Tenseur1HHBB::ChangementIndex Tenseur1HHBB::cdex1HHBB;
116 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 1
117 const Tenseur1BBHH::ChangementIndex Tenseur1BBHH::cdex1BBHH;
118
119 //const TenseurQ1_troisSym_HHHH::ChangementIndex TenseurQ1_troisSym_HHHH::cdex;
120 //const TenseurQ1_troisSym_BBBB::ChangementIndex TenseurQ1_troisSym_BBBB::cdex;
121
122 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 2
123 const Tenseur2HHHH::ChangementIndex Tenseur2HHHH::cdex2HHHH;
124 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 2
125 const Tenseur2BBBB::ChangementIndex Tenseur2BBBB::cdex2BBBB;
126 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 2
127 const Tenseur2HHBB::ChangementIndex Tenseur2HHBB::cdex2HHBB;
128 /// initialisation des tableaux d'index pour tenseur du quatrième ordre dim 2
129 const Tenseur2BBHH::ChangementIndex Tenseur2BBHH::cdex2BBHH;
130
131 //const TenseurQ2_troisSym_HHHH::ChangementIndex TenseurQ2_troisSym_HHHH::cdex;
132 //const TenseurQ2_troisSym_BBBB::ChangementIndex TenseurQ2_troisSym_BBBB::cdex;
133
134

```

## 7.458 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/tenseurs\_mai99/Tenseur/NevezTenseur.h

```
#include "Tenseur.h"
```

Graphe des dépendances par inclusion de NevezTenseur.h:



### Fonctions

— **TenseurHH \* NevezTenseurHH** (int dim, double val=0.)

la méthode retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension "dim" par défaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes le signe de la dimension indique: dim > 0 : tenseurs symétriques dim < 0 : tenseurs non symétriques

- **TenseurBB \* NevezTenseurBB** (int dim, double val=0.)  
la méthode retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension "dim" par défaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes le signe de la dimension indique: dim > 0 : tenseurs symétriques dim < 0 : tenseurs non symétriques
- **TenseurHB \* NevezTenseurHB** (int dim, double val=0.)  
la méthode retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension "dim" par défaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes le signe de la dimension indique: dim > 0 : tenseurs symétriques dim < 0 : tenseurs non symétriques
- **TenseurBH \* NevezTenseurBH** (int dim, double val=0.)  
la méthode retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension "dim" par défaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes le signe de la dimension indique: dim > 0 : tenseurs symétriques dim < 0 : tenseurs non symétriques
- **TenseurHH \* NevezTenseurHH** (const TenseurHH &a)  
en entree on fourni une référence sur un element de la classe generique en sortie la méthode retourne le pointeur affecte sur un tenseur de la bonne taille en fonction du tenseur du meme type les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre
- **TenseurBB \* NevezTenseurBB** (const TenseurBB &a)  
en entree on fourni une référence sur un element de la classe generique en sortie la méthode retourne le pointeur affecte sur un tenseur de la bonne taille en fonction du tenseur du meme type les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre
- **TenseurHB \* NevezTenseurHB** (const TenseurHB &a)  
en entree on fourni une référence sur un element de la classe generique en sortie la méthode retourne le pointeur affecte sur un tenseur de la bonne taille en fonction du tenseur du meme type les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre
- **TenseurBH \* NevezTenseurBH** (const TenseurBH &a)  
en entree on fourni une référence sur un element de la classe generique en sortie la méthode retourne le pointeur affecte sur un tenseur de la bonne taille en fonction du tenseur du meme type les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre
- **TenseurBB \* Produit\_tensorielBB** (const CoordonneeB &aB, const CoordonneeB &bB)  
définition de tenseur d'ordre 2 à partir d'un produit tensoriel de vecteur en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension du pb
- **TenseurHH \* Produit\_tensorielHH** (const CoordonneeH &aH, const CoordonneeH &bH)  
définition de tenseur d'ordre 2 à partir d'un produit tensoriel de vecteur en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension du pb
- **TenseurHB \* Produit\_tensorielHB** (const CoordonneeH &aH, const CoordonneeB &bB)  
définition de tenseur d'ordre 2 à partir d'un produit tensoriel de vecteur en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension du pb
- **TenseurBH \* Produit\_tensorielBH** (const CoordonneeB &aB, const CoordonneeH &bH)  
définition de tenseur d'ordre 2 à partir d'un produit tensoriel de vecteur en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension du pb

### 7.458.1 Description détaillée

déclaration des méthodes externes aux classes de tenseur d'ordre 2, qui permettent de définir un nouveau tenseur

## 7.459 NevezTenseur.h

[Aller à la documentation de ce fichier.](#)

```

1 /** \file NevezTenseur.h
2 * déclaration des méthodes externes aux classes de tenseur d'ordre 2, qui permettent de définir un nouveau
   tenseur
3 */
4
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio

```

```

17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35 /*****
36 *      DATE:          23/01/97
37 *
38 *      AUTEUR:        G RIO
39 *
40 *      PROJET:        Herezh++
41 *
42 *
43 *
44 *      BUT:   definir un nouveau tenseur en fonction de la dimension
45 *             ceci pour des tenseurs d'ordre 2.
46 *
47 *      *****
48 *****/
49
50 #ifndef NEVEZTENSEUR_H
51 #define NEVEZTENSEUR_H
52 #include "Tenseur.h"
53
54 /// la méthode retourne le pointeur affecte sur un tenseur de
55 /// la bonne taille en fonction de la dimension "dim"
56 /// par default les composantes du nouveau tenseur sont mise a zero
57 /// mais on peut indiquer une valeur qui sera affectee a tous les composantes
58 /// le signe de la dimension indique:
59 /// dim > 0 : tenseurs symétriques
60 /// dim < 0 : tenseurs non symétriques
61 TenseurHH * NevezTenseurHH(int dim,double val = 0.);
62 /// la méthode retourne le pointeur affecte sur un tenseur de
63 /// la bonne taille en fonction de la dimension "dim"
64 /// par default les composantes du nouveau tenseur sont mise a zero
65 /// mais on peut indiquer une valeur qui sera affectee a tous les composantes
66 /// le signe de la dimension indique:
67 /// dim > 0 : tenseurs symétriques
68 /// dim < 0 : tenseurs non symétriques
69 TenseurBB * NevezTenseurBB(int dim,double val = 0.);
70 /// la méthode retourne le pointeur affecte sur un tenseur de
71 /// la bonne taille en fonction de la dimension "dim"
72 /// par default les composantes du nouveau tenseur sont mise a zero
73 /// mais on peut indiquer une valeur qui sera affectee a tous les composantes
74 /// le signe de la dimension indique:
75 /// dim > 0 : tenseurs symétriques
76 /// dim < 0 : tenseurs non symétriques
77 TenseurHB * NevezTenseurHB(int dim,double val = 0.);
78 /// la méthode retourne le pointeur affecte sur un tenseur de
79 /// la bonne taille en fonction de la dimension "dim"
80 /// par default les composantes du nouveau tenseur sont mise a zero
81 /// mais on peut indiquer une valeur qui sera affectee a tous les composantes
82 /// le signe de la dimension indique:
83 /// dim > 0 : tenseurs symétriques
84 /// dim < 0 : tenseurs non symétriques
85 TenseurBH * NevezTenseurBH(int dim,double val = 0.);
86
87 /// en entree on fourni une référence sur un element de la classe generique
88 /// en sortie la méthode retourne le pointeur affecte sur un tenseur de
89 /// la bonne taille en fonction du tenseur du meme type
90 /// les composantes du nouveau tenseur sont identiques a celle du tenseur
91 /// passé en parametre
92 TenseurHH * NevezTenseurHH(const TenseurHH& a);
93 /// en entree on fourni une référence sur un element de la classe generique
94 /// en sortie la méthode retourne le pointeur affecte sur un tenseur de
95 /// la bonne taille en fonction du tenseur du meme type
96 /// les composantes du nouveau tenseur sont identiques a celle du tenseur
97 /// passé en parametre
98 TenseurBB * NevezTenseurBB(const TenseurBB& a);
99 /// en entree on fourni une référence sur un element de la classe generique
100 /// en sortie la méthode retourne le pointeur affecte sur un tenseur de
101 /// la bonne taille en fonction du tenseur du meme type
102 /// les composantes du nouveau tenseur sont identiques a celle du tenseur
103 /// passé en parametre

```



```

104 TenseurHB * NevezTenseurHB(const TenseurHB& a);
105 /// en entree on fourni une référence sur un element de la classe generique
106 /// en sortie la méthode retourne le pointeur affecte sur un tenseur de
107 /// la bonne taille en fonction du tenseur du meme type
108 /// les composantes du nouveau tenseur sont identiques a celle du tenseur
109 /// passé en parametre
110 TenseurBH * NevezTenseurBH(const TenseurBH& a);
111
112 /// définition de tenseur d'ordre 2 à partir d'un produit tensoriel de vecteur
113 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
114 /// la bonne taille en fonction de la dimension du pb
115 TenseurBB * Produit_tensorielBB(const CoordonneeB & aB, const CoordonneeB & bB);
116 /// définition de tenseur d'ordre 2 à partir d'un produit tensoriel de vecteur
117 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
118 /// la bonne taille en fonction de la dimension du pb
119 TenseurHH * Produit_tensorielHH(const CoordonneeH & aH, const CoordonneeH & bH);
120 /// définition de tenseur d'ordre 2 à partir d'un produit tensoriel de vecteur
121 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
122 /// la bonne taille en fonction de la dimension du pb
123 TenseurHB * Produit_tensorielHB(const CoordonneeH & aH, const CoordonneeB & bB);
124 /// définition de tenseur d'ordre 2 à partir d'un produit tensoriel de vecteur
125 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
126 /// la bonne taille en fonction de la dimension du pb
127 TenseurBH * Produit_tensorielBH(const CoordonneeB & aB, const CoordonneeH & bH);
128
129
130
131
132 #endif

```

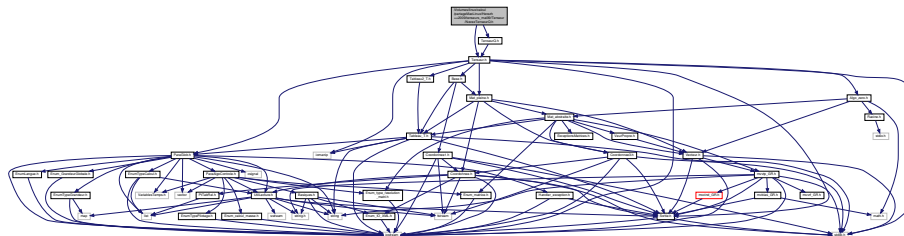
## 7.460 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/tenseurs\_mai99/Tenseur/NevezTenseurQ.h

```

#include "Tenseur.h"
#include "TenseurQ.h"

```

Graphe des dépendances par inclusion de NevezTenseurQ.h:



### Fonctions

- `TenseurHHHH * NevezTenseurHHHH` (int dim, double val=0.)  
définition et création d'un nouveau tenseur d'ordre 4
- `TenseurBBBB * NevezTenseurBBBB` (int dim, double val=0.)  
définition et création d'un nouveau tenseur d'ordre 4
- `TenseurHHBB * NevezTenseurHHBB` (int dim, double val=0.)  
définition et création d'un nouveau tenseur d'ordre 4
- `TenseurBBHH * NevezTenseurBBHH` (int dim, double val=0.)  
définition et création d'un nouveau tenseur d'ordre 4
- `TenseurBHBH * NevezTenseurBHBH` (int dim, double val=0.)  
définition et création d'un nouveau tenseur d'ordre 4
- `TenseurHBHB * NevezTenseurHBHB` (int dim, double val=0.)  
définition et création d'un nouveau tenseur d'ordre 4
- `TenseurBHBB * NevezTenseurBHBB` (int dim, double val=0.)  
définition et création d'un nouveau tenseur d'ordre 4
- `TenseurHBBH * NevezTenseurHBBH` (int dim, double val=0.)  
définition et création d'un nouveau tenseur d'ordre 4
- `TenseurHHHH * NevezTenseurHHHH` (const `TenseurHHHH` &a)  
définition et création d'un nouveau tenseur d'ordre 4

- `TenseurBBBB * NevezTenseurBBBB` (const `TenseurBBBB` &a)  
*définition et création d'un nouveau tenseur d'ordre 4*
- `TenseurHHBB * NevezTenseurHHBB` (const `TenseurHHBB` &a)  
*définition et création d'un nouveau tenseur d'ordre 4*
- `TenseurBBHH * NevezTenseurBBHH` (const `TenseurBBHH` &a)  
*définition et création d'un nouveau tenseur d'ordre 4*
- `TenseurBHBH * NevezTenseurBHBH` (const `TenseurBHBH` &a)  
*définition et création d'un nouveau tenseur d'ordre 4*
- `TenseurHBHB * NevezTenseurHBHB` (const `TenseurHBHB` &a)  
*définition et création d'un nouveau tenseur d'ordre 4*
- `TenseurBHHB * NevezTenseurBHHB` (const `TenseurBHHB` &a)  
*définition et création d'un nouveau tenseur d'ordre 4*
- `TenseurHBBH * NevezTenseurHBBH` (const `TenseurHBBH` &a)  
*définition et création d'un nouveau tenseur d'ordre 4*

### 7.460.1 Description détaillée

déclaration des méthodes externes aux classes de tenseur d'ordre 4, qui permettent de définir un nouveau tenseur

### 7.460.2 Documentation des fonctions

#### 7.460.2.1 NevezTenseurBBBB() [1/2]

```
TenseurBBBB * NevezTenseurBBBB (
    const TenseurBBBB & a )
```

définition et création d'un nouveau tenseur d'ordre 4

en entree on fourni une référence sur un element de la classe generique

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction d'un tenseur du meme type.

Les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre

#### 7.460.2.2 NevezTenseurBBBB() [2/2]

```
TenseurBBBB * NevezTenseurBBBB (
    int dim,
    double val = 0. )
```

définition et création d'un nouveau tenseur d'ordre 4

en sortie le programme retourne le pointeur affecte sur un tenseur de

la bonne taille en fonction de la dimension du pb

par default les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes

- dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)

- dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport aux deux premiers indices, et symétriques par rapport aux deux derniers indices ( 1 ou 9 ou 36 coeff)

- dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:

$$(i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)$$

d'où (1 ou 6 ou 21 coeff)

#### 7.460.2.3 NevezTenseurBBHH() [1/2]

```
TenseurBBHH * NevezTenseurBBHH (
    const TenseurBBHH & a )
```

définition et création d'un nouveau tenseur d'ordre 4

en entree on fourni une référence sur un element de la classe generique

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction d'un tenseur du meme type.

Les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre

**7.460.2.4 NevezTenseurBBHH()** [2/2]

```
TenseurBBHH * NevezTenseurBBHH (
    int dim,
    double val = 0. )
```

définition et création d'un nouveau tenseur d'ordre 4

en sortie le programme retourne le pointeur affecte sur un tenseur de

la bonne taille en fonction de la dimension du pb

par défaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes

- dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)

- dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport aux deux premiers indices, et symétriques par rapport aux deux derniers indices ( 1 ou 9 ou 36 coeff)

- dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:

$(i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)$

d'où (1 ou 6 ou 21 coeff)

**7.460.2.5 NevezTenseurBHBH()** [1/2]

```
TenseurBHBH * NevezTenseurBHBH (
    const TenseurBHBH & a )
```

définition et création d'un nouveau tenseur d'ordre 4

en entree on fourni une référence sur un element de la classe generique

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction d'un tenseur du meme type.

Les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre

**7.460.2.6 NevezTenseurBHHB()** [2/2]

```
TenseurBHHB * NevezTenseurBHHB (
    int dim,
    double val = 0. )
```

définition et création d'un nouveau tenseur d'ordre 4

en sortie le programme retourne le pointeur affecte sur un tenseur de

la bonne taille en fonction de la dimension du pb

par défaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes

- dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)

- dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport aux deux premiers indices, et symétriques par rapport aux deux derniers indices ( 1 ou 9 ou 36 coeff)

- dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:

$(i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)$

d'où (1 ou 6 ou 21 coeff)

**7.460.2.7 NevezTenseurBHBB()** [1/2]

```
TenseurBHBB * NevezTenseurBHBB (
    const TenseurBHBB & a )
```

définition et création d'un nouveau tenseur d'ordre 4

en entree on fourni une référence sur un element de la classe generique

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction d'un tenseur du meme type.

Les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre

**7.460.2.8 NevezTenseurBHBB()** [2/2]

```
TenseurBHBB * NevezTenseurBHBB (
```

```
int dim,
double val = 0. )
```

définition et création d'un nouveau tenseur d'ordre 4

en sortie le programme retourne le pointeur affecte sur un tenseur de

la bonne taille en fonction de la dimension du pb

par défaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes

- dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)

- dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport aux deux premiers indices, et symétriques par rapport aux deux derniers indices ( 1 ou 9 ou 36 coeff)

- dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:

$(i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)$

d'où (1 ou 6 ou 21 coeff)

#### 7.460.2.9 NevezTenseurHBBH() [1/2]

```
TenseurHBBH * NevezTenseurHBBH (
const TenseurHBBH & a )
```

définition et création d'un nouveau tenseur d'ordre 4

en entree on fourni une référence sur un element de la classe generique

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction d'un tenseur du meme type.

Les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre

#### 7.460.2.10 NevezTenseurHBBH() [2/2]

```
TenseurHBBH * NevezTenseurHBBH (
int dim,
double val = 0. )
```

définition et création d'un nouveau tenseur d'ordre 4

en sortie le programme retourne le pointeur affecte sur un tenseur de

la bonne taille en fonction de la dimension du pb

par défaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes

- dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)

- dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport aux deux premiers indices, et symétriques par rapport aux deux derniers indices ( 1 ou 9 ou 36 coeff)

- dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:

$(i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)$

d'où (1 ou 6 ou 21 coeff)

#### 7.460.2.11 NevezTenseurHBHB() [1/2]

```
TenseurHBHB * NevezTenseurHBHB (
const TenseurHBHB & a )
```

définition et création d'un nouveau tenseur d'ordre 4

en entree on fourni une référence sur un element de la classe generique

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction d'un tenseur du meme type.

Les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre

#### 7.460.2.12 NevezTenseurHBHB() [2/2]

```
TenseurHBHB * NevezTenseurHBHB (
int dim,
double val = 0. )
```

définition et création d'un nouveau tenseur d'ordre 4

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction de la dimension du pb  
par defaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes

- dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
- dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport aux deux premiers indices, et symétriques par rapport aux deux derniers indices ( 1 ou 9 ou 36 coeff)
- dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:  
 $(i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)$   
d'où (1 ou 6 ou 21 coeff)

#### 7.460.2.13 NevezTenseurHHBB() [1/2]

```
TenseurHHBB * NevezTenseurHHBB (
    const TenseurHHBB & a )
```

définition et création d'un nouveau tenseur d'ordre 4

en entree on fourni une référence sur un element de la classe generique

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction d'un tenseur du meme type.

Les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre

#### 7.460.2.14 NevezTenseurHHBB() [2/2]

```
TenseurHHBB * NevezTenseurHHBB (
    int dim,
    double val = 0. )
```

définition et création d'un nouveau tenseur d'ordre 4

en sortie le programme retourne le pointeur affecte sur un tenseur de

la bonne taille en fonction de la dimension du pb

par defaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee a tous les composantes

- dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
- dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport aux deux premiers indices, et symétriques par rapport aux deux derniers indices ( 1 ou 9 ou 36 coeff)
- dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:  
 $(i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)$   
d'où (1 ou 6 ou 21 coeff)

#### 7.460.2.15 NevezTenseurHHHH() [1/2]

```
TenseurHHHH * NevezTenseurHHHH (
    const TenseurHHHH & a )
```

définition et création d'un nouveau tenseur d'ordre 4

en entree on fourni une référence sur un element de la classe generique

en sortie le programme retourne le pointeur affecte sur un tenseur de la bonne taille en fonction d'un tenseur du meme type.

Les composantes du nouveau tenseur sont identiques a celle du tenseur passé en parametre

#### 7.460.2.16 NevezTenseurHHHH() [2/2]

```
TenseurHHHH * NevezTenseurHHHH (
    int dim,
    double val = 0. )
```

définition et création d'un nouveau tenseur d'ordre 4

en sortie le programme retourne le pointeur affecte sur un tenseur de

la bonne taille en fonction de la dimension du pb

par defaut les composantes du nouveau tenseur sont mise a zero mais on peut indiquer une valeur qui sera affectee

a tous les composantes

- dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
- dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport aux deux premiers indices, et symétriques par rapport aux deux derniers indices ( 1 ou 9 ou 36 coeff)
- dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:  
 $(i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)$   
 d'où (1 ou 6 ou 21 coeff)

## 7.461 NevezTenseurQ.h

[Aller à la documentation de ce fichier.](#)

```

1 /** \file NevezTenseurQ.h
2 * déclaration des méthodes externes aux classes de tenseur d'ordre 4, qui permettent de définir un nouveau
   tenseur
3 */
4
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35
36 /*****
37 *      DATE:          23/01/97
38 *
39 *      AUTEUR:        G RIO
40 *
41 *      PROJET:        Herezh++
42 *
43 *****/
44 *      BUT:  définir un nouveau tenseur d'ordre 4 en fonction de la
45 *           dimension
46 *
47 *      *****
48 *****/
49
50
51 #ifndef NEVEZTENSEURQ_H
52 #define NEVEZTENSEURQ_H
53 #include "Tenseur.h"
54 #include "TenseurQ.h"
55
56 /// définition et création d'un nouveau tenseur d'ordre 4
57 ///
58 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
59 /// \n la bonne taille en fonction de la dimension du pb
60 /// \n par défaut les composantes du nouveau tenseur sont mise a zero
61 /// \n mais on peut indiquer une valeur qui sera affectee a tous les composantes
62 /// \n
63 /// \n - dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
64 /// \n - dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport
65 /// \n aux deux premiers indices, et symétriques par rapport aux deux derniers indices
66 /// \n ( 1 ou 9 ou 36 coeff)
67 /// \n - dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:
68 /// \n (i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)

```

```

69 /// \n    d'où (1 ou 6 ou 21 coeff)
70 TenseurHHHH * NevezTenseurHHHH(int dim,double val = 0.);
71
72 /// définition et création d'un nouveau tenseur d'ordre 4
73 ///
74 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
75 /// \n la bonne taille en fonction de la dimension du pb
76 /// \n par default les composantes du nouveau tenseur sont mise a zero
77 ///    mais on peut indiquer une valeur qui sera affectee a tous les composantes
78 /// \n
79 /// \n - dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
80 /// \n - dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport
81 ///    aux deux premiers indices, et symétriques par rapport aux deux derniers indices
82 ///    ( 1 ou 9 ou 36 coeff)
83 /// \n - dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:
84 /// \n    (i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)
85 /// \n    d'où (1 ou 6 ou 21 coeff)
86 TenseurBBBB * NevezTenseurBBBB(int dim,double val = 0.);
87
88 /// définition et création d'un nouveau tenseur d'ordre 4
89 ///
90 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
91 /// \n la bonne taille en fonction de la dimension du pb
92 /// \n par default les composantes du nouveau tenseur sont mise a zero
93 ///    mais on peut indiquer une valeur qui sera affectee a tous les composantes
94 /// \n
95 /// \n - dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
96 /// \n - dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport
97 ///    aux deux premiers indices, et symétriques par rapport aux deux derniers indices
98 ///    ( 1 ou 9 ou 36 coeff)
99 /// \n - dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:
100 /// \n    (i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)
101 /// \n    d'où (1 ou 6 ou 21 coeff)
102 TenseurHHBB * NevezTenseurHHBB(int dim,double val = 0.);
103
104 /// définition et création d'un nouveau tenseur d'ordre 4
105 ///
106 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
107 /// \n la bonne taille en fonction de la dimension du pb
108 /// \n par default les composantes du nouveau tenseur sont mise a zero
109 ///    mais on peut indiquer une valeur qui sera affectee a tous les composantes
110 /// \n
111 /// \n - dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
112 /// \n - dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport
113 ///    aux deux premiers indices, et symétriques par rapport aux deux derniers indices
114 ///    ( 1 ou 9 ou 36 coeff)
115 /// \n - dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:
116 /// \n    (i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)
117 /// \n    d'où (1 ou 6 ou 21 coeff)
118 TenseurBBHH * NevezTenseurBBHH(int dim,double val = 0.);
119
120 /// définition et création d'un nouveau tenseur d'ordre 4
121 ///
122 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
123 /// \n la bonne taille en fonction de la dimension du pb
124 /// \n par default les composantes du nouveau tenseur sont mise a zero
125 ///    mais on peut indiquer une valeur qui sera affectee a tous les composantes
126 /// \n
127 /// \n - dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
128 /// \n - dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport
129 ///    aux deux premiers indices, et symétriques par rapport aux deux derniers indices
130 ///    ( 1 ou 9 ou 36 coeff)
131 /// \n - dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:
132 /// \n    (i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)
133 /// \n    d'où (1 ou 6 ou 21 coeff)
134 TenseurBHHB * NevezTenseurBHHB(int dim,double val = 0.);
135
136 /// définition et création d'un nouveau tenseur d'ordre 4
137 ///
138 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
139 /// \n la bonne taille en fonction de la dimension du pb
140 /// \n par default les composantes du nouveau tenseur sont mise a zero
141 ///    mais on peut indiquer une valeur qui sera affectee a tous les composantes
142 /// \n
143 /// \n - dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
144 /// \n - dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport
145 ///    aux deux premiers indices, et symétriques par rapport aux deux derniers indices
146 ///    ( 1 ou 9 ou 36 coeff)
147 /// \n - dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:
148 /// \n    (i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)
149 /// \n    d'où (1 ou 6 ou 21 coeff)
150 TenseurHBHB * NevezTenseurHBHB(int dim,double val = 0.);
151
152 /// définition et création d'un nouveau tenseur d'ordre 4
153 ///
154 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
155 /// \n la bonne taille en fonction de la dimension du pb

```

```

156 /// \n par default les composantes du nouveau tenseur sont mise a zero
157 /// mais on peut indiquer une valeur qui sera affectee a tous les composantes
158 /// \n
159 /// \n - dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
160 /// \n - dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport
161 /// aux deux premiers indices, et symétriques par rapport aux deux derniers indices
162 /// ( 1 ou 9 ou 36 coeff)
163 /// \n - dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:
164 /// \n (i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)
165 /// \n d'où (1 ou 6 ou 21 coeff)
166 TenseurBBBB * NevezTenseurBBBB(int dim,double val = 0.);
167
168 /// définition et création d'un nouveau tenseur d'ordre 4
169 ///
170 /// en sortie le programme retourne le pointeur affecte sur un tenseur de
171 /// \n la bonne taille en fonction de la dimension du pb
172 /// \n par default les composantes du nouveau tenseur sont mise a zero
173 /// mais on peut indiquer une valeur qui sera affectee a tous les composantes
174 /// \n
175 /// \n - dans le cas ou dim = 1 ou 2 ou 3 -> def de tenseurs généraux (1 ou 16 ou 81 coeff)
176 /// \n - dans le cas ou dim = 11 ou 22 ou 33 -> def de tenseurs symétriques par rapport
177 /// aux deux premiers indices, et symétriques par rapport aux deux derniers indices
178 /// ( 1 ou 9 ou 36 coeff)
179 /// \n - dans le cas ou dim = 106 ou 206 ou 306 -> def de tenseurs avec les 3 symétries:
180 /// \n (i,j,k,l) = (j,i,k,l) = (i,j,l,k) = (k,l,i,j)
181 /// \n d'où (1 ou 6 ou 21 coeff)
182 TenseurBBBH * NevezTenseurBBBH(int dim,double val = 0.);
183
184 /// définition et création d'un nouveau tenseur d'ordre 4
185 ///
186 /// en entree on fourni une référence sur un element de la classe generique
187 /// \n en sortie le programme retourne le pointeur affecte sur un tenseur de
188 /// la bonne taille en fonction d'un tenseur du meme type.
189 /// \n Les composantes du nouveau tenseur sont identiques a celle du tenseur
190 /// passé en parametre
191 TenseurHHHH * NevezTenseurHHHH(const TenseurHHHH& a);
192
193 /// définition et création d'un nouveau tenseur d'ordre 4
194 ///
195 /// en entree on fourni une référence sur un element de la classe generique
196 /// \n en sortie le programme retourne le pointeur affecte sur un tenseur de
197 /// la bonne taille en fonction d'un tenseur du meme type.
198 /// \n Les composantes du nouveau tenseur sont identiques a celle du tenseur
199 /// passé en parametre
200 TenseurBBBB * NevezTenseurBBBB(const TenseurBBBB& a);
201
202 /// définition et création d'un nouveau tenseur d'ordre 4
203 ///
204 /// en entree on fourni une référence sur un element de la classe generique
205 /// \n en sortie le programme retourne le pointeur affecte sur un tenseur de
206 /// la bonne taille en fonction d'un tenseur du meme type.
207 /// \n Les composantes du nouveau tenseur sont identiques a celle du tenseur
208 /// passé en parametre
209 TenseurHHBB * NevezTenseurHHBB(const TenseurHHBB& a);
210
211 /// définition et création d'un nouveau tenseur d'ordre 4
212 ///
213 /// en entree on fourni une référence sur un element de la classe generique
214 /// \n en sortie le programme retourne le pointeur affecte sur un tenseur de
215 /// la bonne taille en fonction d'un tenseur du meme type.
216 /// \n Les composantes du nouveau tenseur sont identiques a celle du tenseur
217 /// passé en parametre
218 TenseurBBHH * NevezTenseurBBHH(const TenseurBBHH& a);
219
220 /// définition et création d'un nouveau tenseur d'ordre 4
221 ///
222 /// en entree on fourni une référence sur un element de la classe generique
223 /// \n en sortie le programme retourne le pointeur affecte sur un tenseur de
224 /// la bonne taille en fonction d'un tenseur du meme type.
225 /// \n Les composantes du nouveau tenseur sont identiques a celle du tenseur
226 /// passé en parametre
227 TenseurBHHB * NevezTenseurBHHB(const TenseurBHHB& a);
228
229 /// définition et création d'un nouveau tenseur d'ordre 4
230 ///
231 /// en entree on fourni une référence sur un element de la classe generique
232 /// \n en sortie le programme retourne le pointeur affecte sur un tenseur de
233 /// la bonne taille en fonction d'un tenseur du meme type.
234 /// \n Les composantes du nouveau tenseur sont identiques a celle du tenseur
235 /// passé en parametre
236 TenseurHBHB * NevezTenseurHBHB(const TenseurHBHB& a);
237
238 /// définition et création d'un nouveau tenseur d'ordre 4
239 ///
240 /// en entree on fourni une référence sur un element de la classe generique
241 /// \n en sortie le programme retourne le pointeur affecte sur un tenseur de
242 /// la bonne taille en fonction d'un tenseur du meme type.

```



```

243 /// \n Les composantes du nouveau tenseur sont identiques a celle du tenseur
244 /// passé en parametre
245 TenseurBHHB * NevezTenseurBHHB(const TenseurBHHB& a);
246
247 /// définition et création d'un nouveau tenseur d'ordre 4
248 ///
249 /// en entree on fourni une référence sur un element de la classe generique
250 /// \n en sortie le programme retourne le pointeur affecte sur un tenseur de
251 /// la bonne taille en fonction d'un tenseur du meme type.
252 /// \n Les composantes du nouveau tenseur sont identiques a celle du tenseur
253 /// passé en parametre
254 TenseurHBBH * NevezTenseurHBBH(const TenseurHBBH& a);
255
256
257
258 #endif

```

## 7.462 Tenseur.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *      DATE:          23/01/97
34 *
35 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:        Herezh++
38 *
39 *
40 *      BUT:           Definir les tenseurs de differentes composantes.
41 *      Les classes sont virtuelles pures.
42 *      Elles se declinent en fonction de la dimension du probleme.
43 *      L'objectif principal est de surcharger les differentes operations
44 *      classiques.
45 *
46 *      *****
47 *****/
48
49 #ifndef Tenseur_H
50 #define Tenseur_H
51
52 // #include "Debug.h"
53 #include <iostream>
54 #include <stdlib.h>
55 #include "Sortie.h"
56 #include "ParaGlob.h"
57 #include "Mat_pleine.h"
58 #include "Base.h"
59 #include <iomanip>
60 #include "Algo_zero.h"
61 #include "Tableau2_T.h"
62
63 /** @defgroup Les_classes_tenseurs_virtuelles_ordre2 Les_classes_tenseurs_virtuelles_ordre2
64 *
65 *      BUT:           Definir les tenseurs d'ordre 2 de differentes composantes.
66 *      Les classes sont virtuelles pures.

```

```

67 *      Elles se declinent en fonction de la dimension du probleme.
68 *      L'objectif principal est de surcharger les differentes operations
69 *      classiques.
70 *
71 *      concernant le produit contracte un fois, en particulier pour les tenseurs mixtes
72 *      il y a contraction du 2ieme indice du premier tenseur avec le premier indice du second
73 *      tenseur :  $A_{ij} * B_{jk} = C_{ik} \leftrightarrow A * B = C$ 
74 *      le tenseur inverse par rapport au produit contracte est defini de la maniere suivante
75 *       $Inverse(A) * A = Id$ , ainsi l'inverse d'un tenseur BH est un BH idem pour les HB
76 *      mais l'inverse d'un BB est un HH, et l'inverse d'un HH est un BB
77 *
78 *      NB: lorsque les tenseurs mixtes sont issues de tenseurs HH ou BB symetrique, l'ordre
79 *      de contraction des indices n'a pas d'importance sur le resultat !!
80 *
81 *      le produit contracte de deux tenseurs symetriques quelconques ne donne pas un tenseur
82 *      symetrique !!, donc par exemple la contraction d'un tenseur HB avec HH n'est pas forcement
83 *      symetrique. Le resultat est symetrique SEULEMENT lorsque ces operations sont effectues
84 *      avec le tenseur metrique.
85 *
86 * \author      Gérard Rio
87 * \version    1.0
88 * \date      23/01/97
89 * \brief      Définition des classes virtuelles pures de type Tenseur d'ordre 2, en coordonnées avec
              différentes variances.
90 *
91 */
92
93
94
95
96      class TenseurBB; // def anticipée
97      class TenseurHB; // pour l'utilisation dans les produits contractes
98      class TenseurBH; //
99
100 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre2
101 /// @{
102 ///
103
104 //-----
105 //!      TenseurHH: cas des composantes deux fois contravariantes
106 //-----
107 /// \author      Gérard Rio
108 /// \version    1.0
109 /// \date      23/01/97
110
111 class TenseurHH
112 { friend TenseurHH & operator * (double r, const TenseurHH & t)
113     { return (t*r); };
114   /// produit contracte avec un vecteur
115   friend CoordonneeH operator * (const CoordonneeB & v, const TenseurHH & t)
116     { return (t*v); };
117
118   public :
119     /// DESTRUCTEUR :
120     virtual ~TenseurHH() {};
121
122     // METHODES PUBLIQUES :
123     //1) non virtuelles
124     /// retourne la dimension du tenseur
125     int Dimension() const { return dimension; };
126     //2) virtuelles
127     /// initialise toutes les composantes à val
128     virtual void Inita(double val) = 0;
129     // //fonctions définissant le produit tensoriel normal de deux vecteurs
130     // // *this=aH(i).bH(j) gBi \otimes gBj
131     // // le résultat est a priori non symétrique
132     // static TenseurHH & Prod_tensoriel(const CoordonneeH & aH, const CoordonneeH & bH) ;
133     // //fonctions définissant le produit tensoriel d'un vecteur avec lui-même
134     // // *this=aH(i).aH(j) gBi \otimes gBj
135     // // le résultat est symétrique
136     // static TenseurHH & Prod_tensoriel(const CoordonneeH & aH) ;
137     /// operations +
138     virtual TenseurHH & operator + (const TenseurHH &) const = 0;
139     /// operations +=
140     virtual void operator += (const TenseurHH &) = 0;
141     /// operations -
142     virtual TenseurHH & operator - () const = 0; // oppose du tenseur
143     /// operations - tens
144     virtual TenseurHH & operator - (const TenseurHH &) const = 0;
145     /// operations -=
146     virtual void operator -= (const TenseurHH &) = 0;
147     /// operations =
148     virtual TenseurHH & operator = (const TenseurHH &) = 0;
149     /// operations * double
150     virtual TenseurHH & operator * (const double &) const = 0 ;
151     /// operations *= double
152     virtual void operator *= (const double &) = 0;

```

```

153     /// operations /
154     virtual TenseurHH & operator / ( const double & ) const = 0;
155     /// operations +/=
156     virtual void operator /= ( const double & ) = 0;
157
158     /// produit contracte avec un vecteur
159     virtual CoordonneeH operator * ( const CoordonneeB & ) const = 0;
160
161     /// produit contracte contracté une fois A(i,j)*B(j,k)=A..B
162     /// -> donc c'est l'indice du milieu qui est contracté
163     /// avec BH
164     virtual TenseurHH & operator * ( const TenseurBH & ) const = 0;
165     /// idem avec BB
166     virtual TenseurHB & operator * ( const TenseurBB & ) const = 0;
167     /// produit contracté deux fois A(i,j)*B(j,i)=A..B
168     /// -> on contracte d'abord l'indice du milieu puis l'indice externe
169     virtual double operator && ( const TenseurBB & ) const = 0;
170
171     /// test
172     virtual int operator == ( const TenseurHH & ) const = 0;
173     /// test
174     virtual int operator != ( const TenseurHH & ) const = 0;
175     /// determinant de la matrice des coordonnees
176     virtual double Det() const = 0;
177
178     /// calcul du tenseur inverse par rapport au produit contracte
179     virtual TenseurBB & Inverse() const = 0;
180
181     /// changement de base (cf. théorie)
182     ///
183
184     /// changement de base (cf. théorie) : la matrice beta est telle que:
185     /// \n par défaut inverse = false : (c'est l'utilisation historique)
186     /// \n beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne gB
187     /// \n gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
188     /// \n gpB(i) = beta(i,j) * gB(j) <==> gp_i = beta_i^j * g_j
189     /// \n et on a la matrice gamma telle que:
190     /// \n gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
191     /// \n gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
192     /// \n c-a-d= gp^i = gamma^i_j * g^j
193     /// \n rappel des différentes relations entre beta et gamma
194     /// \n [beta]^{-1} = [gamma]^T ; [beta]^{-1T} = [gamma]
195     /// \n [beta] = [gamma]^{-1T} ; [beta]^T = [gamma]^{-1}
196     /// \n changement de base pour de deux fois contravariants:
197     /// \n [Ap^kl] = [gamma] * [A^ij] * [gamma]^T
198     /// \n donc le changement de base s'effectue directement à l'aide de gamma
199
200     /// \n si inverse = true:
201     /// \n beta(i,j) représente les coordonnées de l'ancienne base gB dans la nouvelle gpB
202     /// \n gB(i) = beta(i,j) * gpB(j), i indice de ligne, j indice de colonne
203     /// \n la formule de changement de base est alors:
204     /// \n [Ap^kl] = [beta]^T * [A^ij] * [beta]
205     ///
206     /// \n nécessite d'inverser la matrice gamma pour calculer beta
207     ///
208     void ChBase( const Mat_pleine& gamma, bool inverse = false) ;
209
210     /// il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
211     ///
212     /// \n il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
213     /// \n connaissant sa variation dans la base actuelle
214     /// \n this : le tenseur
215     /// \n var_tensBB : en entrée: la variation du tenseur dans la base initiale qu'on appelle g_i
216     /// \n var_tensHH : en sortie: la variation du tenseur dans la base finale qu'on appelle gp_i
217     /// \n gamma : en entrée gpH(i) = gamma(i,j) * gH(j)
218     /// \n var_gamma : en entrée : la variation de gamma
219     /// \n [Ap^kl] = [gamma] * [A_ij] * [gamma]^T
220     void Var_tenseur_dans_nouvelle_base
221         (const Mat_pleine& gamma, TenseurHH& var_tensHH, const Mat_pleine& var_gamma);
222
223     /// Affectation_trans_dimension
224     ///
225     /// \n affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
226     /// \n plusZero = true: les données manquantes sont mises à 0
227     /// \n si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
228     /// \n des données possibles
229     virtual void Affectation_trans_dimension(const TenseurHH & B, bool plusZero) = 0;
230
231     /// calcul des composantes du tenseur dans la base absolue: cas HH
232     ///
233     /// \n calcul des composantes du tenseur dans la base absolue
234     /// \n la variance du résultat peut-être quelconque d'où quatre possibilités
235     /// \n en fonction de l'argument A. Dans tous les cas les composantes sont identiques
236     /// \n car la sortie est en absolue
237     /// \n en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
238     /// \n différente du tenseur courant suivant que la dimension absolue et la dimension locale
239     /// \n sont égales ou différentes, retour d'une reference sur A

```

```

240
241 TenseurHH & BaseAbsolue(TenseurHH & A,const BaseB & gi) const;
242 /// idem: cas BB
243 TenseurBB & BaseAbsolue(TenseurBB & A,const BaseB & gi) const;
244 /// idem: cas BH
245 TenseurBH & BaseAbsolue(TenseurBH & A,const BaseB & gi) const;
246 /// idem: cas HB
247 TenseurHB & BaseAbsolue(TenseurHB& A,const BaseB & gi) const;
248
249 /// calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
250 ///
251 /// \n calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
252 /// \n en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
253 /// \n différente du tenseur courant suivant que la dimension absolue et la dimension locale
254 /// \n sont égales ou différentes , retour d'une reference sur A
255 TenseurHH & BaseLocale(TenseurHH & A,const BaseB & gi) const;
256
257 /// ATTENTION creation d'un tenseur transpose qui est supprime par Libere
258 virtual TenseurHH & Transpose() const = 0;
259
260 /// calcul du maximum en valeur absolu des composantes du tenseur
261 virtual double MaxiComposante() const = 0;
262
263 /// ---- manipulation d'indice ---- -> création de nouveaux tenseurs
264 virtual TenseurBB& Baisse2Indices() const = 0;
265 /// ---- manipulation d'indice ---- -> création de nouveaux tenseurs
266 virtual TenseurBH& BaissePremierIndice() const = 0;
267 /// ---- manipulation d'indice ---- -> création de nouveaux tenseurs
268 virtual TenseurHB& BaisseDernierIndice() const = 0;
269
270 // // conversion de type
271 // operator TenseurHH & (void)
272 // {cout << " appel de la conversion de type\n";
273 // return *this;};
274
275 /// Retourne la composante i,j du tenseur
276 /// acces en lecture et en ecriture
277 virtual double& Coor(int i, int j) = 0;
278
279 /// retourne la matrice contenant les composantes du tenseur
280 Mat_pleine& Matrice_composante(Mat_pleine& a) const;
281 /// opération inverse: affectation en fonction d'une matrice
282 /// donc sans vérif de variance !
283 void Affectation(const Mat_pleine& a);
284
285 /// Retourne la composante i,j du tenseur
286 /// acces en lecture seule
287 virtual double operator () (int i, int j) const = 0;
288
289 /// lecture et écriture de données
290 virtual istream & Lecture(istream & entree) = 0;
291 /// lecture et écriture de données
292 virtual ostream & Ecriture(ostream & sort) const = 0;
293
294 // protected :
295
296 // variables
297 int dimension; // dimension du tenseur
298 double * t; // pointeur des données
299
300 protected :
301
302 /// sortie d'un message standard
303 /// dim = dimension du tenseur argument
304 void Message(int dim, string mes) const ;
305
306 };
307 };
308 /// @} // end of group
309
310
311 /** @defgroup Les_classes_Maillons_tenseurs
312 *
313 * BUT: On gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsXX"
314 * qui stocke les pointeurs sur les tenseurs intermédiaire
315 *
316 * \author Gérard Rio
317 * \version 1.0
318 * \date 23/01/97
319 * \brief Définition des classes qui stockent les pointeurs sur les tenseurs intermédiaire.
320 *
321 */
322
323 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsHH"
324 // qui stocke les pointeurs sur les tenseurs intermédiaire
325
326 class PtTenseurHH; // def de la liste chainee des tenseurs intermediaires

```

```

327
328 /// @addtogroup Les_classes_Maillons_tenseurs
329 /// @{
330
331 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
332 class LesMaillonsHH
333 { public :
334     ///liberation de la place occupee par des tenseurs crees pour les
335     /// operations intermediaires
336     static void Libere();
337     /// enregistrement de l'ajout d'un tenseur
338     static void NouveauMaillon(const TenseurHH *);
339
340     /// dernier maillon
341     static PtTenseurHH * maille ;
342     #ifndef MISE_AU_POINT
343     /// nombre de maillon courant sauvegarde
344     static int nbmailHH;
345     #endif
346 };
347 /// @} // end of group
348
349 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre2
350 /// @{///
351
352 //
353 //-----
354 //! TenseurBB: cas des composantes deux fois covariantes
355 //-----
356     /// \author Gérard Rio
357     /// \version 1.0
358     /// \date 23/01/97
359
360 class TenseurBB
361 { friend TenseurBB & operator * (double r, const TenseurBB & t)
362     { return ( t*r); };
363     /// produit contracte avec un vecteur
364     friend CoordonneeB operator * ( const CoordonneeH & v , const TenseurBB & t)
365     { return (t*v); };
366     public :
367     /// DESTRUCTEUR :
368     virtual ~TenseurBB() {};
369
370     // METHODES PUBLIQUES :
371 //1) non virtuelles
372     /// retourne la dimension du tenseur
373     int Dimension() const{ return dimension;}; // retourne la dimension du tenseur
374 //2) virtuelles
375     /// initialise toutes les composantes à val
376     virtual void InitA(double val) = 0;
377     /// operations +
378     virtual TenseurBB & operator + ( const TenseurBB & ) const = 0;
379     /// operations +=
380     virtual void operator += (const TenseurBB & ) = 0;
381     /// operations -
382     virtual TenseurBB & operator - ( ) const = 0; // oppose du tenseur
383     /// operations - tens
384     virtual TenseurBB & operator - ( const TenseurBB & ) const = 0;
385     /// operations -=
386     virtual void operator -= ( const TenseurBB & ) = 0;
387     /// operations =
388     virtual TenseurBB & operator = ( const TenseurBB & ) = 0;
389     /// operations * double
390     virtual TenseurBB & operator * ( const double & ) const = 0;
391     /// operations *= double
392     virtual void operator *= ( const double & ) = 0;
393     /// operations /
394     virtual TenseurBB & operator / (const double & ) const = 0;
395     /// operations /=
396     virtual void operator /= ( const double & ) = 0;
397
398     /// produit contracte avec un vecteur
399     virtual CoordonneeB operator * ( const CoordonneeH & ) const =0;
400
401     /// produit contracte contracté une fois A(i,j)*B(j,k)=A.B
402     /// -> donc c'est l'indice du milieu qui est contracté
403     virtual TenseurBB & operator * ( const TenseurBH & ) const = 0;
404     /// idem en BH
405     virtual TenseurBH & operator * ( const TenseurHH & ) const = 0;
406     /// produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
407     /// -> on contracte d'abord l'indice du milieu puis l'indice externe
408     virtual double operator && ( const TenseurHH & ) const = 0;
409
410     /// test
411     virtual int operator == ( const TenseurBB & ) const = 0;
412     /// test
413     virtual int operator != ( const TenseurBB & ) const = 0;

```

```

414 // determinant de la matrice des coordonnees
415 virtual double Det() const = 0; // determinant de la matrice des coordonnees
416
417 // changement de base (cf. théorie)
418 //
419
420 // changement de base (cf. théorie) : la matrice beta est telle que:
421 // \n gpB(i) = beta(i,j) * gB(j) <=> gp_i = beta_i^j * g_j
422 // \n et la matrice gamma telle que:
423 // \n gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
424 // \n gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
425 // \n c-a-d= gp^i = gamma^i_j * g^j
426 // \n rappel des différentes relations entre beta et gamma
427 // \n [beta]^{-1} = [gamma]^T ; [beta]^{-1T} = [gamma]
428 // \n [beta] = [gamma]^{-1T} ; [beta]^T = [gamma]^{-1}
429 // \n changement de base pour de deux fois covariants:
430 // \n [Ap_kl] = [beta] * [A_ij] * [beta]^T
431 void ChBase( const Mat_pleine& beta);
432
433 // il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
434 //
435 // \n il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
436 // \n connaissant sa variation dans la base actuelle
437 // \n this : le tenseur
438 // \n var_tensBB : en entrée: la variation du tenseur dans la base initiale qu'on appelle g^i
439 // \n var_tensBB : en sortie: la variation du tenseur dans la base finale qu'on appelle gp^i
440 // \n beta : en entrée gpB(i) = beta(i,j) * gB(j)
441 // \n var_beta : en entrée : la variation de beta
442 void Var_tenseur_dans_nouvelle_base
443 (const Mat_pleine& beta, TenseurBB& var_tensBB, const Mat_pleine& var_beta);
444
445 // Affectation_trans_dimension
446 //
447 // \n affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
448 // \n plusZero = true: les données manquantes sont mises à 0
449 // \n si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
450 // \n des données possibles
451 virtual void Affectation_trans_dimension(const TenseurBB & B, bool plusZero) = 0;
452
453 // calcul des composantes du tenseur dans la base absolue: cas BB
454 //
455 // \n calcul des composantes du tenseur dans la base absolue
456 // \n la variance du résultat peut-être quelconque d'où quatre possibilités
457 // \n en fonction de l'argument A. Dans tous les cas les composantes sont identiques
458 // \n car la sortie est en absolue
459 // \n en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
460 // \n différente du tenseur courant suivant que la dimension absolue et la dimension locale
461 // \n sont égales ou différentes, retour d'une reference sur A
462 TenseurBB & BaseAbsolue(TenseurBB & A, const BaseH & gi) const ;
463 // idem pour HH
464 TenseurHH & BaseAbsolue(TenseurHH & A, const BaseH & gi) const ;
465 // idem pour BH
466 TenseurBH & BaseAbsolue(TenseurBH & A, const BaseH & gi) const ;
467 // idem pour HB
468 TenseurHB & BaseAbsolue(TenseurHB & A, const BaseH & gi) const ;
469
470 // calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
471 //
472 // \n calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
473 // \n en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
474 // \n différente du tenseur courant suivant que la dimension absolue et la dimension locale
475 // \n sont égales ou différentes , retour d'une reference sur A
476 TenseurBB & Baselocale(TenseurBB & A, const BaseB & gi) const;
477
478 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
479 virtual TenseurBB & Transpose() const = 0;
480
481 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
482 virtual TenseurHH& Monte2Indices() const = 0;
483 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
484 virtual TenseurHB& MontePremierIndice() const = 0;
485 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
486 virtual TenseurBH& MonteDernierIndice() const = 0;
487
488 // calcul du maximum en valeur absolu des composantes du tenseur
489 virtual double MaxiComposante() const = 0;
490
491 // calcul du tenseur inverse par rapport au produit contracte
492 virtual TenseurHH & Inverse() const = 0;
493
494 // // conversion de type
495 // operator TenseurBB & (void)
496 // {cout << " appel de la conversion de type\n";
497 // return *this;};
498
499 // Retourne la composante i,j du tenseur
500 // acces en lecture et en ecriture

```

```

501     virtual double& Coor(int i, int j) = 0;
502
503     /// retourne la matrice contenant les composantes du tenseur
504     Mat_pleine& Matrice_composante(Mat_pleine& a) const;
505     /// opération inverse: affectation en fonction d'une matrice
506     /// donc sans vérif de variance !
507     void Affectation(const Mat_pleine& a);
508
509     /// Retourne la composante i,j du tenseur
510     /// acces en lecture seulement
511     virtual double operator () (int i, int j) const = 0;
512
513     /// lecture et écriture de données
514     virtual istream & Lecture(istream & entree) = 0;
515     /// lecture et écriture de données
516     virtual ostream & Ecriture(ostream & sort) const = 0;
517
518     // protected :
519
520     // variables
521     int dimension; // dimension du tenseur
522     double * t; // pointeur des données
523
524     protected :
525
526     /// sortie d'un message standard
527     /// dim = dimension du tenseur argument
528     void Message(int dim, string mes) const ;
529 };
530 /// @} // end of group
531
532 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsBB"
533 // qui stocke les pointeurs sur les tenseurs intermédiaire
534
535 class PtTenseurBB; // pour la liste chaineé des tenseurs intermediaires
536
537 /// @addtogroup Les_classes_Maillons_tenseurs
538 /// @{
539
540 /// def d'un maillon de liste chaineé pour memoriser les differents tenseurs intermediaires
541 class LesMaillonsBB
542 { public :
543     ///liberation de la place occupee par des tenseurs creés pour les
544     /// operations intermediaires
545     static void Libere();
546     /// enregistrement de l'ajout d'un tenseur
547     static void NouveauMaillon(const TenseurBB *); // enregistrement de l'ajout d'un tenseur
548
549     /// dernier maillon
550     static PtTenseurBB * maille ;// dernier maillon
551     #ifdef MISE_AU_POINT
552     /// nombre de maillon courant sauvegarde
553     static int nbmailBB; // nombre de maillon courant sauvegarde
554     #endif
555 };
556 /// @} // end of group
557
558 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre2
559 /// @{///
560
561 //-----
562 //!      TenseurBH: cas des composantes mixtes BH
563 //-----
564 /// \author   Gérard Rio
565 /// \version  1.0
566 /// \date    23/01/97
567 class TenseurBH
568 { friend TenseurBH & operator * (double r, const TenseurBH & t)
569     { return ( t*r); };
570     /// produit contracte avec un vecteur
571     friend CoordonneeH operator * ( const CoordonneeH & v , const TenseurBH & t);
572
573     public :
574     /// DESTRUCTEUR :
575     virtual ~TenseurBH() {};
576
577     // METHODES PUBLIQUES :
578     //1) non virtuelles
579     /// retourne la dimension du tenseur
580     int Dimension() const { return dimension;}; // retourne la dimension du tenseur
581     //2) virtuelles
582     /// initialise toutes les composantes à val
583     virtual void Init(double val) = 0;
584     /// operations +
585     virtual TenseurBH & operator + ( const TenseurBH &) const = 0;
586     /// operations +=
587     virtual void operator += ( const TenseurBH &) = 0;

```

```

588 // operations -
589 virtual TenseurBH & operator - ( const TenseurBH & ) const = 0;
590 // operations opposé
591 virtual TenseurBH & operator - () const = 0; // oppose du tenseur
592 // operations -=
593 virtual void operator -= ( const TenseurBH & ) = 0;
594 // operations =
595 virtual TenseurBH & operator = ( const TenseurBH & ) = 0;
596 // operations *
597 virtual TenseurBH & operator * ( const double & ) const = 0;
598 // operations *=
599 virtual void operator *= ( const double & ) = 0;
600 // operations /
601 virtual TenseurBH & operator / ( const double & ) const = 0;
602 // operations /=
603 virtual void operator /= ( const double & ) = 0;
604
605 // produit contracte avec un vecteur
606 virtual CoordonneeB operator * ( const CoordonneeB & ) const = 0;
607
608 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
609 // -> donc c'est l'indice du milieu qui est contracté
610 virtual TenseurBB & operator * ( const TenseurBB & ) const = 0;
611 virtual TenseurBH & operator * ( const TenseurBH & ) const = 0; // produit une fois contracte
612 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
613 // -> on contracte d'abord l'indice du milieu puis l'indice externe
614 virtual double operator && ( const TenseurBH & ) const = 0; // produit deux fois contracte
615
616 // test
617 virtual int operator == ( const TenseurBH & ) const = 0;
618 virtual int operator != ( const TenseurBH & ) const = 0;
619
620 // calcul du tenseur inverse par rapport au produit contracte
621 virtual TenseurBH & Inverse() const = 0;
622
623 // trace du tenseur ou premier invariant
624 virtual double Trace() const = 0;
625 // second invariant = trace (A*A)
626 virtual double II() const = 0;
627 // troisieme invariant = trace ((A*A)*A)
628 virtual double III() const = 0;
629 // determinant de la matrice des coordonnees
630 virtual double Det() const = 0;
631
632 // calcul des valeurs propres
633 //
634 // \n valeurs propre dans le vecteur de retour, classée par ordres "décroissants"
635 // \n cas indique le cas de valeur propre:
636 // \n quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
637 // \n dans ce cas les valeurs propres de retour sont nulles par défaut
638 // \n dim = 1, cas=1 pour une valeur propre;
639 // \n dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques
640 // \n dim = 3 , cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)
641 // \n , cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),
642 // \n , cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du
retour)
643 // \n , cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du
retour)
644 virtual Coordonnee ValPropre(int& cas) const = 0 ;
645
646 // calcul des valeurs propres
647 //
648 // \n idem met en retour la matrice mat contiend par colonne les vecteurs propre
649 // \n elle doit avoir la dimension du tenseur
650 // \n les vecteurs propre sont exprime dans le repere dual (contrairement au HB) pour les tenseurs
dim 3
651 // \n pour dim=2:le premier vecteur propre est exprime dans le repere dual
652 // \n le second vecteur propre est exprimé dans le repère naturel
653 // \n pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
654 virtual Coordonnee ValPropre(int& cas, Mat_pleine& mat) const = 0 ;
655
656 // \n calcul des vecteurs propres, les valeurs propres
657 // \n étant déjà connues
658 //
659 // \n ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres
660 // \n étant déjà connues
661 // \n en retour VP les vecteurs propre : doivent avoir la dimension du tenseur
662 // \n les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
663 // \n pour dim=2:le premier vecteur propre est exprime dans le repere naturel
664 // \n le second vecteur propre est exprimé dans le repère dual
665 // \n pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
666 // \n en sortie cas = -1 s'il y a eu un problème, dans ce cas, V_P est quelconque
667 // \n sinon si tout est ok, cas est identique en sortie avec l'entrée
668 virtual void VecteursPropres(const Coordonnee& Val_P,int& cas, Tableau <Coordonnee>& V_P) const = 0;
669
670 // changement de base (cf. théorie)
671 //

```



```

672
673     /// \n changement de base (cf. théorie) : la matrice beta est telle que:
674     /// \n gpB(i) = beta(i,j) * gB(j) <=> gp_i = beta_i^j * g_j
675     /// \n et la matrice gamma telle que:
676     /// \n gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
677     /// \n gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
678     /// \n c-a-d= gp^i = gamma^i_j * g^j
679     /// \n rappel des différentes relations entre beta et gamma
680     /// \n [beta]^{-1} = [gamma]^T ; [beta]^{-1T} = [gamma]
681     /// \n [beta] = [gamma]^{-1T} ; [beta]^T = [gamma]^{-1}
682     /// \n formule de changement de base
683     /// \n [Ap_k^l] = [beta] * [A_i^j] * [gamma]^T
684     /// \n beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne
    gB
685     /// \n gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
686     void ChBase( const Mat_pleine& beta,const Mat_pleine& gamma);
687
688     /// il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
689     ///
690     /// \n il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
691     /// \n connaissant sa variation dans la base actuelle
692     /// \n this : le tenseur
693     /// \n var_tensBB : en entrée: la variation du tenseur dans les bases initiales qu'on appelle g_i
    et g^i
694     /// \n var_tensBH : en sortie: la variation du tenseur dans les bases finales qu'on appelle gp_i et
    gp^j
695     /// \n beta : en entrée gpB(i) = beta(i,j) * gB(j)
696     /// \n var_beta : en entrée : la variation de beta
697     /// \n gamma : en entrée gpH(i) = gamma(i,j) * gH(j)
698     /// \n var_gamma : en entrée : la variation de gamma
699     /// \n [Ap_k^l] = [beta] * [A_i^j] * [gamma]^T
700     void Var_tenseur_dans_nouvelle_base
701         (const Mat_pleine& beta,TenseurBH& var_tensBH, const Mat_pleine& var_beta
702         ,const Mat_pleine& gamma,const Mat_pleine& var_gamma);
703
704
705     /// Affectation_trans_dimension
706     ///
707     /// \n affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
708     /// \n plusZero = true: les données manquantes sont mises à 0
709     /// \n si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
710     /// \n des données possibles
711     virtual void Affectation_trans_dimension(const TenseurBH & B,bool plusZero) = 0;
712
713     /// calcul des composantes du tenseur dans la base absolue: cas HH
714     ///
715     /// \n calcul des composantes du tenseur dans la base absolue
716     /// \n la variance du résultat peut-être quelconque d'où quatre possibilités
717     /// \n mais on n'en conserve que les non symétriques qui sont le cas général
718     /// \n se qui évite de symétriser accidentellement un tenseur non symétrique
719     /// \n en fonction de l'argument A.Dans tous les cas les composantes sont identiques
720     /// \n car la sortie est en absolue.
721     /// \n en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
722     /// \n différente du tenseur courant, retour d'une reference sur A
723     TenseurBH & BaseAbsolue(TenseurBH & A,const BaseB & giB,const BaseH & giH) const;
724     /// idem cas HB
725     TenseurHB & BaseAbsolue(TenseurHB & A,const BaseB & giB,const BaseH & giH) const;
726
727     /// calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
728     ///
729     /// \n calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
730     /// \n en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
731     /// \n différente du tenseur courant suivant que la dimension absolue et la dimension locale
732     /// \n sont égales ou différentes , retour d'une reference sur A
733     TenseurBH & BaseLocale(TenseurBH & A,const BaseH & giH,const BaseB & gib) const;
734
735     /// ATTENTION creation d'un tenseur transpose qui est supprime par Libere
736     virtual TenseurHB & Transpose() const = 0;
737     /// permute Bas Haut, mais reste dans le même tenseur
738     virtual void PermuteHautBas() =0;
739
740     /// calcul du maximum en valeur absolu des composantes du tenseur
741     virtual double MaxiComposante() const = 0;
742
743     // // conversion de type
744     // operator TenseurBH & (void)
745     // { return *this;};
746
747     /// Retourne la composante i,j du tenseur
748     /// acces en lecture et en ecriture
749     virtual double& Coord(int i, int j) = 0;
750
751     /// retourne la matrice contenant les composantes du tenseur
752     Mat_pleine& Matrice_composante(Mat_pleine& a) const;
753     /// opération inverse: affectation en fonction d'une matrice
754     /// donc sans vérif de variance !
755     void Affectation(const Mat_pleine& a);

```

```

756
757     /// Retourne la composante i,j du tenseur
758     /// acces en lecture seulement
759     virtual double operator () (int i, int j) const = 0;
760
761     /// lecture et écriture de données
762     virtual istream & Lecture(istream & entree) = 0;
763     /// lecture et écriture de données
764     virtual ostream & Ecriture(ostream & sort) const = 0;
765
766 // protected :
767
768 // variables
769     int dimension; // dimension du tenseur
770     double * t; // pointeur des données
771
772 protected :
773
774     static Algo_zero alg_zero; // algo pour la recherche de zero
775     /// sortie d'un message standard
776     /// dim = dimension du tenseur argument
777     void Message(int dim, string mes) const ;
778
779     /// méthode interne pour le calcul de vecteurs propres
780     Coordonnee ValPropre_int(int& cas, Mat_pleine& mat) const;
781 };
782 /// @} // end of group
783
784 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsBH"
785 // qui stocke les pointeurs sur les tenseurs intermédiaire
786
787 class PtTenseurBH; // pour la liste chainee des tenseurs intermediaires
788
789 /// @addtogroup Les_classes_Maillons_tenseurs
790 /// @{
791
792 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
793 class LesMaillonsBH
794 { public :
795     ///liberation de la place occupee par des tenseurs crees pour les
796     /// operations intermediaires
797     static void Libere();
798     /// enregistrement de l'ajout d'un tenseur
799     static void NouveauMaillon(const TenseurBH *); // enregistrement de l'ajout d'un tenseur
800
801     /// dernier maillon
802     static PtTenseurBH * maille ;// dernier maillon
803     #ifdef MISE_AU_POINT
804     /// nombre de maillon courant sauvegarde
805     static int nbmailBH; // nombre de maillon courant sauvegarde
806     #endif
807 };
808 /// @} // end of group
809
810 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre2
811 /// @{///
812
813 //-----
814 ///! TenseurHB: cas des composantes mixtes HB
815 //-----
816 /// \author Gérard Rio
817 /// \version 1.0
818 /// \date 23/01/97
819 class TenseurHB
820 { friend TenseurHB & operator * (double r,const TenseurHB & t)
821     { return ( t*r); };
822     /// produit contracte avec un vecteur
823     friend CoordonneeB operator * (const CoordonneeB & v ,const TenseurHB & t)
824     { return (t.Transpose() * v) ; };
825
826 public :
827     /// DESTRUCTEUR :
828     virtual ~TenseurHB() { };
829
830     // METHODES PUBLIQUES :
831     //1) non virtuelles
832     /// retourne la dimension du tenseur
833     int Dimension() const { return dimension;}; // retourne la dimension du tenseur
834     //2) virtuelles
835     /// initialise toutes les composantes à val
836     virtual void Inita(double val) = 0;
837     /// operations +
838     virtual TenseurHB & operator + ( const TenseurHB &) const = 0;
839     /// operations +=
840     virtual void operator += ( const TenseurHB &) = 0;
841     /// operations opposé
842     virtual TenseurHB & operator - () const = 0; // oppose du tenseur

```

```

843     /// operations -
844     virtual TenseurHB & operator - ( const TenseurHB & ) const = 0;
845     /// operations -=
846     virtual void operator -= ( const TenseurHB & ) = 0;
847     /// operations =
848     virtual TenseurHB & operator = ( const TenseurHB & ) = 0;
849     /// operations *
850     virtual TenseurHB & operator * ( const double & ) const = 0;
851     /// operations *=
852     virtual void operator *= ( const double & ) = 0;
853     /// operations /
854     virtual TenseurHB & operator / ( const double & ) const = 0;
855     /// operations /=
856     virtual void operator /= ( const double & ) = 0;
857
858     /// produit contracte avec un vecteur
859     virtual CoordonneeH operator * ( const CoordonneeH & ) const =0;
860
861     /// produit contracte contracté une fois A(i,j)*B(j,k)=A.B
862     /// -> donc c'est l'indice du milieu qui est contracté
863     virtual TenseurHH & operator * ( const TenseurHH & ) const = 0;
864     virtual TenseurHB & operator * ( const TenseurHB & ) const = 0; // produit une fois contracte
865     /// produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
866     /// -> on contracte d'abord l'indice du milieu puis l'indice externe
867     virtual double operator && ( const TenseurHB & ) const = 0; // produit deux fois contracte
868
869     /// test
870     virtual int operator == ( const TenseurHB & ) const = 0;
871     virtual int operator != ( const TenseurHB & ) const = 0;
872
873     /// calcul du tenseur inverse par rapport au produit contracte
874     virtual TenseurHB & Inverse() const = 0;
875
876     /// trace du tenseur ou premier invariant
877     virtual double Trace() const = 0;
878     /// second invariant = trace (A*A)
879     virtual double II() const = 0;
880     /// troisieme invariant = trace ((A*A)*A)
881     virtual double III() const = 0;
882     /// determinant de la matrice des coordonnees
883     virtual double Det() const = 0;
884
885     /// calcul des valeurs propres
886     ///
887
888     /// \n valeurs propre dans le vecteur de retour, classée par ordres "décroissants"
889     /// \n cas indique le cas de valeur propre:
890     /// \n quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
891     /// \n dans ce cas les valeurs propres de retour sont nulles par défaut
892     /// \n dim = 1, cas=1 pour une valeur propre;
893     /// \n dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques
894     /// \n dim = 3 , cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)
895     /// \n , cas = 0 si les 3 sont identiques (= la première composantes du vecteur de retour),
896     /// \n , cas = 2 si val(1)=val(2) ( val(1), et val(3) dans les 2 premières composantes du
897     /// \n , cas = 3 si val(2)=val(3) ( val(1), et val(2) dans les 2 premières composantes du
898     virtual Coordonnee ValPropre(int& cas) const =0 ;
899
900     /// calcul des valeurs propres
901     ///
902     /// \n idem met en retour la matrice mat contiend par colonne les vecteurs propre
903     /// \n elle doit avoir la dimension du tenseur
904     /// \n les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
905     /// \n pour dim=2:le premier vecteur propre est exprime dans le repere naturel
906     /// \n le second vecteur propre est exprimé dans le repère dual
907     /// \n pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
908     virtual Coordonnee ValPropre(int& cas, Mat_pleine& mat) const = 0;
909
910
911     /// \n calcul des vecteurs propres, les valeurs propres
912     /// \n étant déjà connues
913     ///
914     /// \n ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres
915     /// \n étant déjà connues
916     /// \n en retour VP les vecteurs propre : doivent avoir la dimension du tenseur
917     /// \n les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
918     /// \n pour dim=2:le premier vecteur propre est exprime dans le repere naturel
919     /// \n le second vecteur propre est exprimé dans le repère dual
920     /// \n pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
921     /// \n en sortie cas = -1 s'il y a eu un problème, dans ce cas, V_P est quelconque
922     /// \n sinon si tout est ok, cas est identique en sortie avec l'entrée
923     virtual void VecteursPropres(const Coordonnee& Val_P,int& cas, Tableau <Coordonnee>& V_P) const = 0;
924
925     // // conversion de type
926     // operator TenseurHB & (void)
927     // {return *this;};

```

```

928
929
930
931     /// changement de base (cf. théorie)
932     ///
933
934     /// \n changement de base (cf. théorie) : la matrice beta est telle que:
935     /// \n gpB(i) = beta(i,j) * gB(j) <==> gp_i = beta_i^j * g_j
936     /// \n et la matrice gamma telle que:
937     /// \n gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
938     /// \n gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
939     /// \n c-a-d= gp^i = gamma^i_j * g^j
940     /// \n rappel des différentes relations entre beta et gamma
941     /// \n [beta]^{-1} = [gamma]^T      ; [beta]^{-1T} = [gamma]
942     /// \n [beta] = [gamma]^{-1T}      ; [beta]^T = [gamma]^{-1}
943     /// \n formule de changement de base
944     /// \n [Ap^k_l] = [gamma] * [A^i_j] * [beta]^T
945     /// \n      beta(i,j) represente les coordonnees de la nouvelle base naturelle gpB dans l'ancienne
gpB
946     /// \n      gpB(i) = beta(i,j) * gB(j), i indice de ligne, j indice de colonne
947     void ChBase( const Mat_pleine& beta, const Mat_pleine& gamma);
948
949
950     /// il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
951     ///
952     /// \n il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
953     /// \n connaissant sa variation dans la base actuelle
954     /// \n this      : le tenseur
955     /// \n var_tensBB : en entrée: la variation du tenseur dans les bases initiales qu'on appelle g_i
et g^i
956     /// \n var_tensHB : en sortie: la variation du tenseur dans les bases finales qu'on appelle gp_i et
gp^j
957     /// \n beta      : en entrée gpB(i) = beta(i,j) * gB(j)
958     /// \n var_beta   : en entrée : la variation de beta
959     /// \n gamma      : en entrée gpH(i) = gamma(i,j) * gH(j)
960     /// \n var_gamma  : en entrée : la variation de gamma
961     /// \n [Ap^k_l] = [gamma] * [A^i_j] * [beta]^T
962     void Var_tenseur_dans_nouvelle_base
963         (const Mat_pleine& beta, TenseurBH& var_tensHB, const Mat_pleine& var_beta
964         , const Mat_pleine& gamma, const Mat_pleine& var_gamma);
965
966
967     /// Affectation_trans_dimension
968     ///
969     /// \n affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
970     /// \n plusZero = true: les données manquantes sont mises à 0
971     /// \n si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
972     /// \n des données possibles
973     virtual void Affectation_trans_dimension(const TenseurHB & B, bool plusZero) = 0;
974
975     /// calcul des composantes du tenseur dans la base absolue: cas HH
976     ///
977     /// \n calcul des composantes du tenseur dans la base absolue
978     /// \n la variance du résultat peut-être quelconque d'où quatre possibilités
979     /// \n mais on n'en conserve que les non symétriques qui sont le cas général
980     /// \n se qui évite de symétriser accidentellement un tenseur non symétrique
981     /// \n en fonction de l'argument A. Dans tous les cas les composantes sont identiques
982     /// \n car la sortie est en absolue.
983     /// \n en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension
984     /// \n différente du tenseur courant, retour d'une référence sur A
985     TenseurHB & BaseAbsolue(TenseurHB & A, const BaseH & giH, const BaseB & giB) const;
986     TenseurBH & BaseAbsolue(TenseurBH & A, const BaseH & giH, const BaseB & giB) const;
987     /// \n calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
988     /// \n en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension
989     /// \n différente du tenseur courant suivant que la dimension absolue et la dimension locale
990     /// \n sont égales ou différentes , retour d'une référence sur A
991     TenseurHB & BaseLocale(TenseurHB & A, const BaseB & gib, const BaseH & giH) const;
992
993     /// ATTENTION creation d'un tenseur transpose qui est supprime par Libere
994     virtual TenseurBH & Transpose() const = 0;
995     /// permute Bas Haut, mais reste dans le même tenseur
996     virtual void PermuteHautBas() = 0;
997
998     /// calcul du maximum en valeur absolu des composantes du tenseur
999     virtual double MaxiComposante() const = 0;
1000
1001     /// Retourne la composante i,j du tenseur
1002     /// acces en lecture et en ecriture
1003     virtual double& Coord(int i, int j) = 0;
1004
1005     /// retourne la matrice contenant les composantes du tenseur
1006     Mat_pleine& Matrice_composante(Mat_pleine& a) const;
1007     /// opération inverse: affectation en fonction d'une matrice
1008     /// donc sans vérif de variance !
1009     void Affectation(const Mat_pleine& a);
1010
1011     /// Retourne la composante i,j du tenseur

```

```

1012     /// acces en lecture seulement
1013     virtual double operator () (int i, int j) const = 0;
1014
1015     /// lecture et écriture de données
1016     virtual istream & Lecture(istream & entree) = 0;
1017     /// lecture et écriture de données
1018     virtual ostream & Ecriture(ostream & sort) const = 0;
1019
1020 // protected :
1021
1022     // variables
1023     int dimension; // dimension du tenseur
1024     double * t; // pointeur des données
1025
1026 protected :
1027
1028     static Algo_zero alg_zero; // algo pour la recherche de zero
1029     /// sortie d'un message standard
1030     /// dim = dimension du tenseur argument
1031     void Message(int dim, string mes) const ;
1032
1033     /// méthode interne pour le calcul de vecteurs propres
1034     Coordonnee ValPropre_int(int& cas, Mat_pleine& mat) const;
1035
1036 };
1037 /// @} // end of group
1038
1039 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsHB"
1040 // qui stocke les pointeurs sur les tenseurs intermédiaire
1041
1042 class PtTenseurHB; // pour la liste chainee des tenseurs intermediaires
1043
1044 /// @addtogroup Les_classes_Maillons_tenseurs
1045 /// @{
1046
1047 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
1048 class LesMaillonsHB
1049 { public :
1050     ///liberation de la place occupee par des tenseurs crees pour les
1051     /// operations intermediaires
1052     static void Libere();
1053     /// enregistrement de l'ajout d'un tenseur
1054     static void NouveauMaillon(const TenseurHB *); // enregistrement de l'ajout d'un tenseur
1055
1056     /// dernier maillon
1057     static PtTenseurHB * maille ;// dernier maillon
1058     #ifdef MISE_AU_POINT
1059         /// nombre de maillon courant sauvegarde
1060         static int nbmailHB; // nombre de maillon courant sauvegarde
1061     #endif
1062 };
1063 /// @} // end of group
1064
1065
1066
1067 //=====
1068 // probleme de gestion de la definition de tenseurs supplementaires lors
1069 // d'écriture de grandes expressions. Il est necessaire de liberer l'espace
1070 // apres les calculs
1071 // la fonction libereTenseur libere tout l'espace de tous les types de tenseurs
1072 //=====
1073
1074 void LibereTenseur() ;
1075
1076
1077 #ifndef MISE_AU_POINT
1078     #include "Tenseur.cc"
1079     #define Tenseur_H_deja_inclus
1080 #endif
1081
1082 #endif

```

## 7.463 Tenseur1.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //

```

```

12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30 //
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:    Herezh++
38 *
39 *****/
40 *      BUT:      Definition des classes derivees de dimension 1.
41 *
42 *      *****
43 *****/
44
45 #ifndef Tenseur1_H
46 #define Tenseur1_H
47
48 // #include "Debug.h"
49 #include <iostream>
50 #include "Tenseur.h"
51 #include "PtTabRel.h"
52 #include "MathUtil.h"
53 #include "Coordonnee1.h"
54
55
56 /** @defgroup Les_classes_tenseurs_dim1_ordre2
57 *
58 *      BUT:      Définir les tenseurs d'ordre 2 de différentes composantes,
59 *      spécifiquement à la dimension 1.
60 *      L'objectif principal est de surcharger les différentes opérations
61 *      classiques.
62 *
63 *      concernant le produit contracté un fois, en particulier pour les tenseurs mixtes
64 *      il y a contraction du 2ième indice du premier tenseur avec le premier indice du second
65 *      tenseur :  $A_{ij} * B_{jk} = C_{ik} \Leftrightarrow A * B = C$ 
66 *      le tenseur inverse par rapport au produit contracté est défini de la manière suivante
67 *      Inverse(A) * A = Id, ainsi l'inverse d'un tenseur BH est un BH idem pour les HB
68 *      mais l'inverse d'un BB est un HH, et l'inverse d'un HH est un BB
69 *
70 *      NB: lorsque les tenseurs mixtes sont issues de tenseurs HH ou BB symétrique, l'ordre
71 *      de contraction des indices n'a pas d'importance sur le résultat !!
72 *
73 *      le produit contracté de deux tenseurs symétriques quelconques ne donne pas un tenseur
74 *      symétrique !!, donc par exemple la contraction d'un tenseur HB avec HH n'est pas forcément
75 *      symétrique. Le résultat est symétrique SEULEMENT lorsque ces opérations sont effectuées
76 *      avec le tenseur métrique.
77 *
78 * \author      Gérard Rio
79 * \version    1.0
80 * \date       23/01/97
81 * \brief      Définition des classes de dimension 1 de type Tenseur d'ordre 2, en coordonnées avec
82 *             différentes variances.
83 */
84
85
86
87 class TenseurHB; // pour l'utilisation dans les produits contractés
88
89 /// @addtogroup Les_classes_tenseurs_dim1_ordre2
90 /// @{
91
92 //-----
93 /// Définition des tenseur dérivées de dimension 1.
94 /// cas des composantes deux fois contravariantes
95 //-----
96 /// \author      Gérard Rio
97 /// \version    1.0

```

```

98 /// \date      23/01/97
99 class Tenseur1HH : public TenseurHH
100 {
101 // surcharge de l'operator de lecture
102 friend istream & operator » (istream &, Tenseur1HH &);
103 // surcharge de l'operator d'écriture
104 friend ostream & operator « (ostream &, const Tenseur1HH &);
105 public :
106 // Constructeur
107 Tenseur1HH() ; // par défaut
108 // initialisation de la composante (1,1)
109 Tenseur1HH(double val) ;
110 // DESTRUCTEUR :
111 ~Tenseur1HH() ;
112 // constructeur a partir d'une instance non differenciee
113 Tenseur1HH (const TenseurHH &);
114 // constructeur de copie
115 Tenseur1HH ( const Tenseur1HH & B);
116
117 // METHODES PUBLIQUES :
118 // initialise toutes les composantes à val
119 void InitA(double val);
120 // operations
121 TenseurHH & operator + ( const TenseurHH &) const ;
122 void operator += ( const TenseurHH &);
123 TenseurHH & operator - ( ) const ;
124 TenseurHH & operator - ( const TenseurHH &) const ;
125 void operator -= ( const TenseurHH &);
126 TenseurHH & operator = ( const TenseurHH &);
127 TenseurHH & operator = ( const Tenseur1HH & B)
128 { return this->operator=(*(const TenseurHH*) & B));};
129 TenseurHH & operator * ( const double &) const ;
130 void operator *= ( const double &);
131 TenseurHH & operator / ( const double &) const ;
132 void operator /= ( const double &);
133
134 // produit contracte avec un vecteur
135 CoordonneeH operator * ( const CoordonneeB & ) const ;
136
137 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
138 // -> donc c'est l'indice du milieu qui est contracté
139 TenseurHH & operator * ( const TenseurBH &) const ;
140 TenseurHB & operator * ( const TenseurBB &) const ;
141 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
142 // -> on contracte d'abord l'indice du milieu puis l'indice externe
143 double operator && ( const TenseurBB &) const ;
144
145 // test
146 int operator == ( const TenseurHH &) const ;
147 int operator != ( const TenseurHH &) const ;
148
149 double Det() const ; // determinant de la matrice des coordonnees
150
151 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
152 TenseurHH & Transpose() const ;
153
154 // calcul du maximum en valeur absolu des composantes du tenseur
155 double MaxiComposante() const {return Dabs(*t);};
156
157 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
158 virtual TenseurBB& Baisse2Indices() const;
159 virtual TenseurBH& BaissePremierIndice() const;
160 virtual TenseurHB& BaisseDernierIndice() const;
161
162 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
163 // plusZero = true: les données manquantes sont mises à 0
164 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
165 // des données possibles
166 void Affectation_trans_dimension(const TenseurHH & B,bool plusZero);
167 // calcul du tenseur inverse par rapport au produit contracte
168 TenseurBB & Inverse() const ;
169
170 // retourne la composante i,j en lecture/écriture
171 #ifdef MISE_AU_POINT
172 inline double& Coor(int i,int j)
173 #else
174 inline double& Coor(int ,int )
175 #endif
176 {
177 #ifdef MISE_AU_POINT
178 if ( (i!=1) || (j!=1) )
179 { cout << "\nErreur : composante " << i << ", " << j << " inexistante !\n";
180 cout << "Tenseur1HH::OPERATOR() (int,int ) \n";
181 Sortie(1);
182 }
183 #endif
184 return *t;

```

```

185     };
186     // retourne la composante i,j en lecture seule
187 #ifdef MISE_AU_POINT
188     inline double operator () (int i,int j) const
189 #else
190     inline double operator () (int ,int ) const
191 #endif
192     {
193 #ifdef MISE_AU_POINT
194         if ( (i!=1) || (j!=1) )
195             { cout << "\nErreur : composante " << i << ", " << j << " inexistante !\n";
196               cout << "Tenseur1HH::OPERATOR() (int,int ) \n";
197               Sortie(1);
198             };
199 #endif
200         return *t;
201     };
202
203     //fonctions static définissant le produit tensoriel de deux vecteurs
204     static TenseurHH & Prod_tensoriel(const CoordonneeH & aH, const CoordonneeH & bH);
205
206     // lecture et écriture de données
207     istream & Lecture(istream & entree);
208     ostream & Ecriture(ostream & sort) const ;
209
210     protected :
211         // allocator dans la liste de data
212         listdoubleIter ipointe;
213
214 };
215 /// @} // end of group
216
217 class TenseurBH; // pour l'utilisation dans TenseurHH
218
219 /// @addtogroup Les_classes_tenseurs_dim1_ordre2
220 /// @{
221
222 //-----
223 /// Definition des tenseur derivees de dimension1.
224 /// cas des composantes deux fois covariantes
225 //-----
226
227 /// \author Gérard Rio
228 /// \version 1.0
229 /// \date 23/01/97
230 class Tenseur1BB : public TenseurBB
231 {
232     // surcharge de l'operator de lecture
233     friend istream & operator » (istream &, Tenseur1BB &);
234     // surcharge de l'operator d'écriture
235     friend ostream & operator « (ostream &, const Tenseur1BB &);
236     public :
237         // constructeur
238         Tenseur1BB(); // par défaut
239         // initialisation de la compoante (1,1)
240         Tenseur1BB(double val);
241         // DESTRUCTEUR :
242         ~Tenseur1BB();
243         // constructeur a partir d'une instance non differenciee
244         Tenseur1BB ( const TenseurBB &);
245         // constructeur de copie
246         Tenseur1BB ( const Tenseur1BB & B);
247
248         // METHODES PUBLIQUES :
249         // initialise toutes les composantes à val
250         void InitA(double val);
251         // operations
252         TenseurBB & operator + ( const TenseurBB &) const ;
253         void operator += ( const TenseurBB &);
254         TenseurBB & operator - () const ;
255         TenseurBB & operator - ( const TenseurBB &) const ;
256         void operator -= ( const TenseurBB &);
257         TenseurBB & operator = ( const TenseurBB &);
258         TenseurBB & operator = ( const Tenseur1BB & B)
259         { return this->operator=(*(const TenseurBB*) & B); };
260         TenseurBB & operator * ( const double &) const ;
261         void operator *= ( const double &);
262         TenseurBB & operator / ( const double &) const ;
263         void operator /= ( const double &);
264
265         // produit contracte avec un vecteur
266         CoordonneeB operator * ( const CoordonneeH & ) const ;
267
268         // produit contracte contracté une fois A(i,j)+B(j,k)=A.B
269         // -> donc c'est l'indice du milieu qui est contracté
270         TenseurBB & operator * ( const TenseurHB &) const ;
271         TenseurBH & operator * ( const TenseurHH &) const ;

```



```

272 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
273 // -> on contracte d'abord l'indice du milieu puis l'indice externe
274 double operator && ( const TenseurHH &) const ;
275
276 // test
277 int operator == ( const TenseurBB &) const ;
278 int operator != ( const TenseurBB &) const ;
279
280 double Det() const ; // determinant de la matrice des coordonnees
281
282 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
283 TenseurBB & Transpose() const ;
284
285 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
286 virtual TenseurHH& Monte2Indices() const ;
287 virtual TenseurHB& MontePremierIndice() const ;
288 virtual TenseurBH& MonteDernierIndice() const ;
289
290 // calcul du maximum en valeur absolu des composantes du tenseur
291 double MaxiComposante() const {return Dabs(*t);};
292
293 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
294 // plusZero = true: les données manquantes sont mises à 0
295 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
296 // des données possibles
297 void Affectation_trans_dimension(const TenseurBB & B,bool plusZero);
298
299 // calcul du tenseur inverse par rapport au produit contracte
300 TenseurHH & Inverse() const ;
301
302 // retourne la composante i,j en lecture/écriture
303 #ifndef MISE_AU_POINT
304 inline double& Coor(int i,int j)
305 #else
306 inline double& Coor(int ,int )
307 #endif
308 {
309 #ifdef MISE_AU_POINT
310 if ( (i!=1) || (j!=1) )
311 { cout << "\nErreur : composante " << i <<"," << j << " inexistante !\n";
312 cout << " Tenseur1BB::OPERATOR() (int,int ) \n";
313 Sortie(1);
314 };
315 #endif
316 return *t;
317 };
318
319 // retourne la composante i,j en écriture seule
320 #ifndef MISE_AU_POINT
321 inline double operator () (int i,int j) const
322 #else
323 inline double operator () (int ,int ) const
324 #endif
325 {
326 #ifdef MISE_AU_POINT
327 if ( (i!=1) || (j!=1) )
328 { cout << "\nErreur : composante " << i <<"," << j << " inexistante !\n";
329 cout << " Tenseur1BB::OPERATOR() (int,int ) \n";
330 Sortie(1);
331 };
332 #endif
333 return *t;
334 };
335
336 //fonctions static définissant le produit tensoriel de deux vecteurs
337 static TenseurBB & Prod_tensoriel(const CoordonneeB & aB, const CoordonneeB & bB);
338
339 // lecture et écriture de données
340 istream & Lecture(istream & entree);
341 ostream & Ecriture(ostream & sort) const ;
342
343 protected :
344 // allocator dans la liste de data
345 listdoubleIter ipointe;
346
347 };
348 /// @} // end of group
349
350 class Tenseur3BH; // pour Affectation_3D_a_1D
351
352 /// @addtogroup Les_classes_tenseurs_dim1_ordre2
353 /// @{
354
355 //-----
356 /// Definition des tenseur derivees de dimension1.
357 /// cas des composantes mixtes BH
358 //-----

```

```

359 /// \author   Gérard Rio
360 /// \version  1.0
361 /// \date     23/01/97
362 class Tenseur1BH : public TenseurBH
363 {
364 // surcharge de l'operator de lecture
365 friend istream & operator » (istream &, Tenseur1BH &);
366 // surcharge de l'operator d'écriture
367 friend ostream & operator « (ostream &, const Tenseur1BH &);
368 friend class Tenseur3BH; // pour le passage 3D 2D: méthode Affectation_3D_a_2D(..
369 public :
370 // constructeur
371 Tenseur1BH() ; // par défaut
372 // initialisation de la compoante (1,1)
373 Tenseur1BH(double val) ;
374 // DESTRUCTEUR :
375 ~Tenseur1BH();
376 // constructeur a partir d'une instance non differenciee
377 Tenseur1BH ( const TenseurBH &);
378 // constructeur de copie
379 Tenseur1BH ( const Tenseur1BH & B);
380
381 // METHODES PUBLIQUES :
382 // initialise toutes les composantes à val
383 void InitA(double val);
384 // operations
385 TenseurBH & operator + ( const TenseurBH &) const ;
386 void operator += ( const TenseurBH &);
387 TenseurBH & operator - () const ;
388 TenseurBH & operator - ( const TenseurBH &) const ;
389 void operator -= ( const TenseurBH &);
390 TenseurBH & operator = ( const TenseurBH &);
391 TenseurBH & operator = ( const Tenseur1BH & B)
392 { return this->operator=(*(const TenseurBH*) & B) ); };
393 TenseurBH & operator * ( const double &) const ;
394 void operator *= ( const double &);
395 TenseurBH & operator / ( const double &) const ;
396 void operator /= ( const double &);
397
398 // produit contracte avec un vecteur
399 CoordonneeB operator * ( const CoordonneeB & ) const ;
400 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
401 // -> donc c'est l'indice du milieu qui est contracté
402 TenseurBH & operator * ( const TenseurBH &) const ; // produit une fois contracte
403 TenseurBH & operator * ( const TenseurBH &) const ; // produit une fois contracte
404 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
405 // -> on contracte d'abord l'indice du milieu puis l'indice externe
406 double operator && ( const TenseurBH &) const ; // produit deux fois contracte
407
408 double Trace() const ; // trace du tenseur ou premier invariant
409 double II() const ; // second invariant = trace (A*A)
410 double III() const ; // troisieme invariant = trace ((A*A)*A)
411 double Det() const ; // determinant de la matrice des coordonnees
412 // valeurs propre dans le vecteur de retour, classée par ordres "décroissants"
413 // cas indique le cas de valeur propre:
414 // quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
415 // dans ce cas les valeurs propres de retour sont nulles par défaut
416 // dim = 1, cas=1 pour une valeur propre ;
417 virtual Coordonnee ValPropre(int& cas) const ;
418 // idem met en retour la matrice mat contiend par colonne les vecteurs propre
419 // elle doit avoir la dimension du tenseur
420 // les vecteurs propre sont exprime dans le repere naturel
421 virtual Coordonnee ValPropre(int& cas, Mat_pleine& mat) const ;
422
423 // ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres
424 // étant déjà connues
425 // en retour VP les vecteurs propre : doivent avoir la dimension du tenseur
426 // les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
427 // pour dim=2:le premier vecteur propre est exprime dans le repere naturel
428 // le second vecteur propre est exprimé dans le repère dual
429 // pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
430 // en sortie cas = -1 s'il y a eu un problème, dans ce cas, V_P est quelconque
431 // sinon si tout est ok, cas est identique en sortie avec l'entrée
432 virtual void VecteursPropres(const Coordonnee& Val_P,int& cas, Tableau <Coordonnee>& V_P) const;
433
434 // test
435 int operator == ( const TenseurBH &) const ;
436 int operator != ( const TenseurBH &) const ;
437
438 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
439 // plusZero = true: les données manquantes sont mises à 0
440 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
441 // des données possibles
442 void Affectation_trans_dimension(const TenseurBH & B,bool plusZero);
443
444 // calcul du tenseur inverse par rapport au produit contracte
445 TenseurBH & Inverse() const ;

```

```

446
447 // tenseur transpose
448 TenseurHB & Transpose() const ;
449 // symétrie formelle  $A^i_j = A_j^i$  ==> création du tenseur en HB identique
450 TenseurHB & Identique() const;
451 // permute Bas Haut, mais reste dans le même tenseur
452 // ici ne fait rien
453 void PermuteHautBas() {};
454
455 // calcul du maximum en valeur absolu des composantes du tenseur
456 double MaxiComposante() const {return Dabs(*t);};
457
458 // retourne la composante i,j en lecture/écriture
459 #ifndef MISE_AU_POINT
460 inline double& Coor(int i,int j)
461 #else
462 inline double& Coor(int ,int )
463 #endif
464 {
465     #ifndef MISE_AU_POINT
466     if ( (i!=1) || (j!=1) )
467     { cout << "\nErreur : composante " << i << ", " << j << " inexistante !\n";
468       cout << "Tenseur1BH::OPERATOR() (int,int ) \n";
469       Sortie(1);
470     };
471     #endif
472     return *t;
473 };
474
475 // retourne la composante i,j en lecture seule
476 #ifndef MISE_AU_POINT
477 inline double operator () (int i,int j) const
478 #else
479 inline double operator () (int ,int ) const
480 #endif
481 {
482     #ifndef MISE_AU_POINT
483     if ( (i!=1) || (j!=1) )
484     { cout << "\nErreur : composante " << i << ", " << j << " inexistante !\n";
485       cout << "Tenseur1BH::OPERATOR() (int,int ) \n";
486       Sortie(1);
487     };
488     #endif
489     return *t;
490 };
491
492 //fonctions static définissant le produit tensoriel de deux vecteurs
493 static TenseurBH & Prod_tensoriel(const CoordonneeB & aB, const CoordonneeH & bH);
494
495 // lecture et écriture de données
496 istream & Lecture(istream & entree);
497 ostream & Ecriture(ostream & sort) const ;
498
499 protected :
500 // allocator dans la liste de data
501 listdoublelIter ipointe;
502
503
504
505 };
506 /// @} // end of group
507
508 /// @addtogroup Les_classes_tenseurs_dim1_ordre2
509 /// @{}
510
511 //-----
512 /// Definition des tenseur derivees de dimension1.
513 /// cas des composantes mixtes HB
514 //-----
515 /// \author Gérard Rio
516 /// \version 1.0
517 /// \date 23/01/97
518 class Tenseur1HB : public TenseurHB
519 {
520 // surcharge de l'operator de lecture
521 friend istream & operator » (istream &, Tenseur1HB &);
522 // surcharge de l'operator d'écriture
523 friend ostream & operator « (ostream &, const Tenseur1HB &);
524 public :
525 // Constructeur
526 Tenseur1HB() ; // par défaut
527 // initialisation de la compoante (1,1)
528 Tenseur1HB(double val) ;
529 // DESTRUCTEUR :
530 ~Tenseur1HB();
531 // constructeur a partir d'une instance non differenciee
532 Tenseur1HB ( const TenseurHB &);

```

```

533 // constructeur de copie
534 Tenseur1HB ( const Tenseur1HB & B);
535
536 // METHODES PUBLIQUES :
537 // initialise toutes les composantes à val
538 void InitA(double val);
539 // operations
540 TenseurHB & operator + ( const TenseurHB &) const ;
541 void operator += ( const TenseurHB &);
542 TenseurHB & operator - () const ;
543 TenseurHB & operator - ( const TenseurHB &) const ;
544 void operator -= ( const TenseurHB &);
545 TenseurHB & operator = ( const TenseurHB &);
546 TenseurHB & operator = ( const Tenseur1HB & B);
547 { return this->operator=(*(const TenseurHB*) & B));};
548 TenseurHB & operator * ( const double &) const ;
549 void operator *= ( const double &);
550 TenseurHB & operator / ( const double &) const ;
551 void operator /= ( const double &);
552
553 // produit contracte avec un vecteur
554 CoordonneeH operator * ( const CoordonneeH & ) const ;
555
556 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
557 // -> donc c'est l'indice du milieu qui est contracté
558 TenseurHH & operator * ( const TenseurHH &) const ;
559 TenseurHB & operator * ( const TenseurHB &) const ; // produit une fois contracte
560 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
561 // -> on contracte d'abord l'indice du milieu puis l'indice externe
562 double operator && ( const TenseurHB &) const ; // produit deux fois contracte
563
564 double Trace() const ; // trace du tenseur ou premier invariant
565 double II() const ; // second invariant = trace (A*A)
566 double III() const ; // troisieme invariant = trace ((A*A)*A)
567 double Det() const ; // determinant de la matrice des coordonnees
568 // valeurs propre dans le vecteur de retour, classée par ordres "décroissants"
569 // cas indique le cas de valeur propre:
570 // quelque soit la dimension: cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
571 // dans ce cas les valeurs propres de retour sont nulles par défaut
572 // dim = 1, cas=1 pour une valeur propre ;
573 virtual Coordonnee ValPropre(int& cas) const ;
574 // idem met en retour la matrice mat contient par colonne les vecteurs propre
575 // elle doit avoir la dimension du tenseur
576 // les vecteurs propre sont exprime dans le repere naturel
577 virtual Coordonnee ValPropre(int& cas, Mat_pleine& mat) const ;
578
579 // ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres
580 // étant déjà connues
581 // en retour VP les vecteurs propre : doivent avoir la dimension du tenseur
582 // les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
583 // pour dim=2:le premier vecteur propre est exprime dans le repere naturel
584 // le second vecteur propre est exprimé dans le repère dual
585 // pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
586 // en sortie cas = -1 s'il y a eu un problème, dans ce cas, V_P est quelconque
587 // sinon si tout est ok, cas est identique en sortie avec l'entrée
588 virtual void VecteursPropres(const Coordonnee& Val_P,int& cas, Tableau <Coordonnee>& V_P) const;
589
590 // test
591 int operator == ( const TenseurHB &) const ;
592 int operator != ( const TenseurHB &) const ;
593
594 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
595 // plusZero = true: les données manquantes sont mises à 0
596 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
597 // des données possibles
598 void Affectation_trans_dimension(const TenseurHB & B,bool plusZero);
599
600 // calcul du tenseur inverse par rapport au produit contracte
601 TenseurHB & Inverse() const ;
602
603 // tenseur transpose
604 TenseurBH & Transpose() const ;
605 // symétrie formelle A^i_j = A_j^i : ==> création du tenseur en BH identique
606 TenseurBH & Identique() const;
607 // permute Bas Haut, mais reste dans le même tenseur
608 // ici ne fait rien
609 void PermuteHautBas() {};
610
611 // calcul du maximum en valeur absolu des composantes du tenseur
612 double MaxiComposante() const {return Dabs(*t);};
613
614 // retourne la composante i,j en lecture/écriture
615 #ifdef MISE_AU_POINT
616 inline double& Coor(int i,int j)
617 #else
618 inline double& Coor(int ,int )
619 #endif

```

```

620     {
621         #ifdef MISE_AU_POINT
622             if ( (i!=1) || (j!=1) )
623                 { cout << "\nErreur : composante " << i << ", " << j << " inexistante !\n";
624                   cout << "Tenseur1HB::OPERATOR() (int,int ) \n";
625                     Sortie(1);
626                 };
627             #endif
628             return *t;
629         };
630         // retourne la composante i,j en lecture seule
631 #ifdef MISE_AU_POINT
632     inline double operator () (int i,int j) const
633 #else
634     inline double operator () (int ,int ) const
635 #endif
636     {
637         #ifdef MISE_AU_POINT
638             if ( (i!=1) || (j!=1) )
639                 { cout << "\nErreur : composante " << i << ", " << j << " inexistante !\n";
640                   cout << "Tenseur1HB::OPERATOR() (int,int ) \n";
641                     Sortie(1);
642                 };
643             #endif
644             return *t;
645         };
646
647         //fonctions static définissant le produit tensoriel de deux vecteurs
648         static TenseurHB & Prod_tensoriel(const CoordonneeH & aH, const CoordonneeB & bB);
649
650         // lecture et écriture de données
651         istream & Lecture(istream & entree);
652         ostream & Ecriture(ostream & sort) const ;
653
654     protected :
655         // allocator dans la liste de data
656         listdoubleIter ipointe;
657
658 };
659 /// @} // end of group
660
661 #ifndef MISE_AU_POINT
662     #include "Tenseur1-1.cc"
663     #include "Tenseur1-2.cc"
664     #define Tenseur1_H_deja_inclus
665 #endif
666
667
668 #endif

```

## 7.464 Tenseur1\_TroisSym.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *          LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)          *

```

```

34 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
35 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
36 *****
37 * DATE: 8/6/2003 *
38 * $ *
39 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
40 * Tel 0297874571 fax : 02.97.87.45.72 *
41 * $ *
42 * PROJET: Herezh++ *
43 * $ *
44 *****
45 * BUT: Definition d'une classe derivee de tenseur du 4ieme ordre *
46 * de dimension 1 . Il s'agit ici de classes *
47 * très particulières. Les tenseurs possèdent les trois *
48 * symétrie: (ijkl) = (jikl) = (ijlk) = (klij) *
49 * Ainsi en composantes 4 fois covariantes ou 4 fois *
50 * contravariantes, en 1D il y a 1 composantes indépendantes.*
51 * La classe est plutôt à considérer comme un conteneur, *
52 * ainsi seules les méthodes principales sont utilisables. *
53 * $ *
54 * *****
55 * VERIFICATION: *
56 * *
57 * ! date ! auteur ! but ! *
58 * ----- *
59 * ! ! ! ! *
60 * $ *
61 * *****
62 * MODIFICATIONS: *
63 * ! date ! auteur ! but ! *
64 * ----- *
65 * $ *
66 *****/
67 #ifndef TenseurQ1_TROIS_SYM_H
68 #define TenseurQ1_TROIS_SYM_H
69
70 #include <iostream>
71 #include "TenseurQ.h"
72 #include "PtTabRel.h"
73 #include "Tableau2_T.h"
74 #include "Tenseur.h"
75
76 // les tenseurs en mixte: BHHB, HBBH etc. ne sont pas définis ici car normalement il n'ont plus
77 // de symétrie, par contre ils sont définis dans le cas générale : cf. TenseurQlgene
78 //
79 // les tenseurs mixtes BBHH et HHBB ne sont pas encore définis car pas employés pour l'instant (à faire
80 // si nécessaire)
81
82 //-----
83 // cas des composantes 4 fois contravariantes HHHH
84 //-----
85 class TenseurQ1_troisSym_HHHH : public TenseurHHHH
86 { // surcharge de l'opérateur de lecture
87 friend istream & operator > (istream &, TenseurQ1_troisSym_HHHH &);
88 // surcharge de l'opérateur d'écriture
89 friend ostream & operator < (ostream &, const TenseurQ1_troisSym_HHHH &);
90
91 public :
92 // Constructeur
93 TenseurQ1_troisSym_HHHH() ; // par défaut
94 // initialisation de toutes les composantes (ici une seule) a une même valeur val
95 TenseurQ1_troisSym_HHHH(const double& x1111);
96
97 // DESTRUCTEUR :
98 ~TenseurQ1_troisSym_HHHH() ;
99 // constructeur à partir d'une instance non différenciée
100 TenseurQ1_troisSym_HHHH (const TenseurHHHH &);
101 // constructeur de copie
102 TenseurQ1_troisSym_HHHH (const TenseurQ1_troisSym_HHHH &);
103
104 // METHODES PUBLIQUES :
105 //2) virtuelles
106 // initialise toutes les composantes à val
107 void Init(double val);
108 // operations
109 TenseurHHHH & operator + ( const TenseurHHHH &) const ;
110 void operator += ( const TenseurHHHH &);
111 TenseurHHHH & operator - () const ; // oppose du tenseur
112 TenseurHHHH & operator - ( const TenseurHHHH &) const ;
113 void operator -= ( const TenseurHHHH &);
114 TenseurHHHH & operator = ( const TenseurHHHH &);
115 TenseurHHHH & operator * (const double &) const ;
116 void operator *= ( const double &);
117 TenseurHHHH & operator / ( const double &) const ;
118 void operator /= ( const double &);
119

```

```

120 // produit deux fois contracte à droite avec un tenseur du second ordre
121 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
122 TenseurHH& operator && ( const TenseurBB & ) const ;
123 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
124 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
125 TenseurHHHH& operator && ( const TenseurBBHH & ) const
126 { cout << "\n non implante !! , TenseurHHHH& TenseurQ1_troisSym_HHHH::operator && ( const
TenseurBBHH & ) const ";
127 Sortie(1); TenseurHHHH* toto; return *toto; /* toto pour eviter un warning*/ };
128 TenseurHHBB& operator && ( const TenseurBBBB & ) const
129 { cout << "\n non implante !! , TenseurHHBB& TenseurQ1_troisSym_HHHH::operator && ( const
TenseurBBBB & ) const ";
130 Sortie(1); TenseurHHBB* toto; return *toto; /* toto pour eviter un warning*/ };
131
132 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
133 // les 2 premiers indices sont échangés avec les deux derniers indices
134 // ici en fait c'est le même tenseur grace aux symétries constitutives
135 TenseurHHHH & Transposelet2avec3et4() const ;
136
137 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
138 // plusZero = true: les données manquantes sont mises à 0
139 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
140 // des données possibles
141 void Affectation_trans_dimension(const TenseurHHHH & B,bool plusZero);
142
143 // test
144 int operator == ( const TenseurHHHH & ) const ;
145
146 // change la composante i,j,k,l du tenseur
147 // acces en ecriture,
148 void Change (int i, int j, int k, int l,const double& val) ;
149 // en cumul : équivalent de +=
150 void ChangePlus (int i, int j, int k, int l,const double& val);
151
152 // Retourne la composante i,j,k,l du tenseur
153 // acces en lecture seule
154 double operator () (int i, int j, int k, int l) const ;
155
156 // calcul du maximum en valeur absolu des composantes du tenseur
157 double MaxiComposante() const;
158
159 // lecture et écriture de données
160 istream & Lecture(istream & entree);
161 ostream & Ecriture(ostream & sort) const ;
162
163 protected :
164 // allocator dans la liste de data
165 listdoubleIter ipointe;
166
167 // fonction pour le produit contracté à gauche
168 TenseurHH& Prod_gauche( const TenseurBB & F) const { return ((*this) && F); };
169 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
170 TenseurBBHH& Prod_gauche( const TenseurBBBB & ) const
171 { cout << "\n non implante !! , TenseurBBHH& TenseurQ1_troisSym_HHHH::Prod_gauche( const
TenseurBBBB & ) const ";
172 Sortie(1); TenseurBBHH* toto; return *toto; /* toto pour eviter un warning*/ };
173 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
174 TenseurHHHH& Prod_gauche( const TenseurHHBB & ) const
175 { cout << "\n non implante !! , TenseurHHHH& TenseurQ1_troisSym_HHHH::Prod_gauche( const TenseurHHBB
& ) const ";
176 Sortie(1); TenseurHHHH* toto; return *toto; /* toto pour eviter un warning*/ };
177 };
178 //
179 //-----
180 // cas des composantes 4 fois covariantes BBBB
181 //-----
182 class TenseurQ1_troisSym_BBBB : public TenseurBBBB
183 { // surcharge de l'operator de lecture
184 friend istream & operator » (istream &, TenseurQ1_troisSym_BBBB &);
185 // surcharge de l'operator d'ecriture
186 friend ostream & operator « (ostream &, const TenseurQ1_troisSym_BBBB &);
187
188 public :
189 // Constructeur
190 TenseurQ1_troisSym_BBBB() ; // par défaut
191 // initialisation de toutes les composantes a une meme valeur val
192 TenseurQ1_troisSym_BBBB(const double& x1111);
193
194 // DESTRUCTEUR :
195 ~TenseurQ1_troisSym_BBBB() ;
196 // constructeur a partir d'une instance non differenciee
197 TenseurQ1_troisSym_BBBB (const TenseurBBBB &);
198 // constructeur de copie
199 TenseurQ1_troisSym_BBBB (const TenseurQ1_troisSym_BBBB &);

```

```

200
201 // METHODES PUBLIQUES :
202 //2) virtuelles
203 // initialise toutes les composantes à val
204 void Initia(double val) ;
205 // operations
206 TenseurBBBB & operator + ( const TenseurBBBB & ) const ;
207 void operator += ( const TenseurBBBB & );
208 TenseurBBBB & operator - ( ) const ; // oppose du tenseur
209 TenseurBBBB & operator - ( const TenseurBBBB & ) const ;
210 void operator -= ( const TenseurBBBB & );
211 TenseurBBBB & operator = ( const TenseurBBBB & );
212 TenseurBBBB & operator * (const double & ) const ;
213 void operator *= ( const double & );
214 TenseurBBBB & operator / ( const double & ) const ;
215 void operator /= ( const double & );
216
217 // produit deux fois contracte à droite avec un tenseur du second ordre
218 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
219 TenseurBB& operator && ( const TenseurHH & ) const ;
220 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
221 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
222 TenseurBBHH& operator && ( const TenseurHHHH & ) const
223 { cout << "\n non implante !! , TenseurBBHH& TenseurQ1_troisSym_BBBB::operator && ( const
TenseurHHHH & ) const ";
224 Sortie(1); TenseurBBHH* toto; return *toto; /* toto pour eviter un warning*/ };
225 TenseurBBBB& operator && ( const TenseurHHBB & ) const
226 { cout << "\n non implante !! , TenseurBBBB& TenseurQ1_troisSym_BBBB::operator && ( const
TenseurHHBB & ) const ";
227 Sortie(1); TenseurBBBB* toto; return *toto; /* toto pour eviter un warning*/ };
228
229 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
230 // les 2 premiers indices sont échangés avec les deux derniers indices
231 // ici en fait c'est le même tenseur grace aux symétries constitutives
232 TenseurBBBB & Transposelet2avec3et4() const ;
233
234 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
235 // plusZero = true: les données manquantes sont mises à 0
236 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
237 // des données possibles
238 void Affectation_trans_dimension(const TenseurBBBB & B,bool plusZero);
239
240 // test
241 int operator == ( const TenseurBBBB & ) const ;
242
243 // change la composante i,j,k,l du tenseur
244 // acces en ecriture,
245 void Change (int i, int j, int k, int l,const double& val) ;
246 // en cumul : équivalent de +=
247 void ChangePlus (int i, int j, int k, int l,const double& val);
248
249 // Retourne la composante i,j,k,l du tenseur
250 // acces en lecture seule
251 double operator () (int i, int j, int k, int l) const ;
252
253 // calcul du maximum en valeur absolu des composantes du tenseur
254 double MaxiComposante() const;
255
256 // lecture et écriture de données
257 istream & Lecture(istream & entree);
258 ostream & Ecriture(ostream & sort) const ;
259
260 protected :
261 // allocator dans la liste de data
262 listdoubleIter ipointe;
263
264 // fonction pour le produit contracté à gauche
265 TenseurBB& Prod_gauche( const TenseurHH & F) const { return ((*this) && F); };
266 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
267 TenseurHHBB& Prod_gauche( const TenseurHHHH & ) const
268 { cout << "\n non implante !! , TenseurHHBB& TenseurQ1_troisSym_BBBB::Prod_gauche( const
TenseurHHHH & ) const ";
269 Sortie(1); TenseurHHBB* toto; return *toto; /* toto pour eviter un warning*/ };
270 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
271 TenseurBBBB& Prod_gauche( const TenseurBBHH & ) const
272 { cout << "\n non implante !! , TenseurBBBB& TenseurQ1_troisSym_BBBB::Prod_gauche( const TenseurBBHH
& ) const ";
273 Sortie(1); TenseurBBBB* toto; return *toto; /* toto pour eviter un warning*/ };
274 };
275
276
277 #ifndef MISE_AU_POINT
278 #include "Tenseur1_TroisSym.cc"
279 #define TenseurQ1_TroisSym_H_deja_inclus

```



```

280 #endif
281
282
283 #endif

```

## 7.465 Tenseur2.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31 /*****
32 *      DATE:      23/01/97
33 *
34 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:    Herezh++
37 *
38 *****/
39 *      BUT:      Definition des classes derivees de dimension 2.
40 *
41 *      *****
42 *****/
43
44 #ifndef Tenseur2_H
45 #define Tenseur2_H
46
47 #include <iostream>
48 #include "Tenseur.h"
49 #include "PtTabRel.h"
50 #include "Tableau2_T.h"
51
52
53 /** @defgroup Les_classes_tenseurs_dim2_ordre2
54 *
55 *      BUT:      Définir les tenseurs d'ordre 2 de differentes composantes,
56 *      spécifiquement à la dimension 2.
57 *      L'objectif principal est de surcharger les differentes operations
58 *      classiques.
59 *
60 *      concernant le produit contracte un fois, en particulier pour les tenseurs mixtes
61 *      il y a contraction du 2ieme indice du premier tenseur avec le premier indice du second
62 *      tenseur :  $A_{ij} * B_{jk} = C_{ik} \rightarrow A * B = C$ 
63 *      le tenseur inverse par rapport au produit contracte est defini de la maniere suivante
64 *       $Inverse(A) * A = Id$ , ainsi l'inverse d'un tenseur BH est un BH idem pour les HB
65 *      mais l'inverse d'un BB est un HH, et l'inverse d'un HH est un BB
66 *
67 *      NB: lorsque les tenseurs mixtes sont issues de tenseurs HH ou BB symetrique, l'ordre
68 *      de contraction des indices n'a pas d'importance sur le resultat !!
69 *
70 *      le produit contracte de deux tenseurs symetriques quelconques ne donne pas un tenseur
71 *      symetrique !!, donc par exemple la contraction d'un tenseur HB avec HH n'est pas forcement
72 *      symetrique. Le resultat est symetrique SEULEMENT lorsque ces operations sont effectues
73 *      avec le tenseur metrique.
74 *
75 *      \author      Gérard Rio
76 *      \version    1.0
77 *      \date       23/01/97

```

```

78 * \brief      Définition des classes de dimension 2 de type Tenseur d'ordre 2, en coordonnées avec
                différentes variances.
79 *
80 */
81
82
83 class Tenseur3HH;
84
85 /// @addtogroup Les_classes_tenseurs_dim2_ordre2
86 /// @{
87
88 //-----
89 /// Définition des tenseur derivees de dimension 2.
90 ///      cas des composantes deux fois contravariantes symetriques
91 //-----
92 /// \author    Gérard Rio
93 /// \version   1.0
94 /// \date     23/01/97
95 class Tenseur2HH : public TenseurHH
96 {
97 // surcharge de l'operator de lecture
98 friend istream & operator » (istream &, Tenseur2HH &);
99 // surcharge de l'operator d'écriture
100 friend ostream & operator « (ostream &, const Tenseur2HH &);
101 friend class Tenseur3HH; // pour le passage 3D 2D: méthode Affectation_2D_a_3D(..
102 public :
103     // Constructeur
104     Tenseur2HH() ; // par défaut
105     // initialisation de toutes les composantes a une meme valeur val
106     Tenseur2HH(const double val);
107     // initialisation avec 3 valeurs différentes correspondantes a
108     // (1,1) (2,2) (1,2) qui est identique a (2,1)
109     Tenseur2HH
110     (const double val1,const double val2,const double val3) ;
111     // DESTRUCTEUR :
112     ~Tenseur2HH() ;
113     // constructeur a partir d'une instance non differenciee
114     Tenseur2HH (const TenseurHH &);
115     // constructeur de copie
116     Tenseur2HH (const Tenseur2HH &);
117
118     // METHODES PUBLIQUES :
119     // initialise toutes les composantes à val
120     void Inita(double val);
121     // operations
122     TenseurHH & operator + (const TenseurHH &) const ;
123     void operator += (const TenseurHH &);
124     TenseurHH & operator - () const ; // oppose du tenseur
125     TenseurHH & operator - (const TenseurHH &) const ;
126     void operator -= (const TenseurHH &);
127     TenseurHH & operator = (const TenseurHH &);
128     TenseurHH & operator = (const Tenseur2HH & B)
129     { return this->operator=(*(const TenseurHH*) & B) ;};
130     TenseurHH & operator * (const double &) const ;
131     void operator *= (const double &);
132     TenseurHH & operator / (const double &) const ;
133     void operator /= (const double &);
134     // affectation de B dans this, les données en trop sont ignorées
135     void Affectation_3D_a_2D(const Tenseur3HH & B);
136
137     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
138     // plusZero = true: les données manquantes sont mises à 0
139     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
140     // des données possibles
141     void Affectation_trans_dimension(const TenseurHH & B,bool plusZero);
142
143     // produit contracte avec un vecteur
144     CoordonneeH operator * (const CoordonneeB & ) const ;
145
146     // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
147     // -> donc c'est l'indice du milieu qui est contracté
148     TenseurHH & operator * (const TenseurBH &) const ;
149     TenseurHB & operator * (const TenseurBB &) const ;
150     // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
151     // -> on contracte d'abord l'indice du milieu puis l'indice externe
152     double operator && (const TenseurBB &) const ;
153
154     // test
155     int operator == (const TenseurHH &) const ;
156     int operator != (const TenseurHH &) const ;
157
158     double Det() const ; // determinant de la matrice des coordonnees
159
160     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
161     TenseurHH & Transpose() const ;
162
163     // ---- manipulation d'indice ---- -> création de nouveaux tenseurs

```

```

164 virtual TenseurBB& Baisse2Indices() const;
165 virtual TenseurBH& BaissePremierIndice() const;
166 virtual TenseurHB& BaisseDernierIndice() const;
167
168 // calcul du maximum en valeur absolu des composantes du tenseur
169 double MaxiComposante() const;
170
171 // calcul du tenseur inverse par rapport au produit contracte
172 TenseurBB & Inverse() const ;
173
174 // retourne la composante i,j accessible en lecture/ecriture
175 double& Coor(const int i,const int j);
176
177 // retourne la composante i,j accessible uniquement en lecture
178 double operator () (const int i,const int j) const;
179
180 //fonctions static définissant le produit tensoriel de deux vecteurs
181 // si les vecteurs sont égaux le tenseur est symétrique sinon il est non symétrique
182 static TenseurHH & Prod_tensoriel(const CoordonneeH & aH, const CoordonneeH & bH);
183
184 // lecture et écriture de données
185 istream & Lecture(istream & entree);
186 ostream & Ecriture(ostream & sort) const ;
187
188 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
189 static int OdVect(const int i, const int j) {return cdex.odVect(i,j);};
190 // transformation 1 indice -> 2 indices
191 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
192 static int idx_i(const int k) {return cdex.idx_i(k);};
193 static int idx_j(const int k) {return cdex.idx_j(k);};
194
195 protected :
196 // allocator dans la liste de data
197 listdouble3Iter ipointe;
198 // --- gestion d'index ----
199 class ChangementIndex
200 { public:
201     ChangementIndex();
202     // passage pour les index de la forme vecteur à la forme i,j
203     Tableau<int> idx_i,idx_j;
204     // passage pour les index de la forme i,j à la forme vecteur
205     Tableau2<int> odVect;
206 };
207 static const ChangementIndex cdex;
208 };
209 /// @} // end of group
210
211 class Tenseur_ns3HH;
212
213 /// @addtogroup Les_classes_tenseurs_dim2_ordre2
214 /// @{
215
216 //-----
217 /// Definition des tenseur derivees de dimension 2.
218 /// cas des composantes deux fois contravariantes non symetriques
219 /// pour les differencier la dimension = -2
220 //-----
221 /// \author Gérard Rio
222 /// \version 1.0
223 /// \date 23/01/97
224 class Tenseur_ns2HH : public TenseurHH
225 {
226 // surcharge de l'operator de lecture
227 friend istream & operator » (istream &, Tenseur_ns2HH &);
228 // surcharge de l'operator d'ecriture
229 friend ostream & operator « (ostream &, const Tenseur_ns2HH &);
230 friend class Tenseur_ns3HH; // pour le passage 3D 2D: méthode Affectation_3D_a_2D(..
231 friend class Tenseur_ns2BB;
232 public :
233 // Constructeur
234 Tenseur_ns2HH() ; // par défaut
235 // initialisation de toutes les composantes a une meme valeur val
236 Tenseur_ns2HH(const double val);
237 // initialisation avec 4 valeurs différentes correspondantes a
238 // (1,1) (2,2) (2,1) (1,2), car le tableau n'est pas symetrique !!!!
239 // le premier indice = ligne, le second = colonne
240 Tenseur_ns2HH
241 (const double val1, const double val2, const double val3,const double val4) ;
242 // DESTRUCTEUR :
243 ~Tenseur_ns2HH() ;
244 // constructeur a partir d'une instance non differenciee
245 Tenseur_ns2HH (const TenseurHH &);
246 // constructeur de copie
247 Tenseur_ns2HH (const Tenseur_ns2HH &);
248
249 // METHODES PUBLIQUES :
250 // initialise toutes les composantes à val

```

```

251 void Inita(double val);
252 // operations
253 TenseurHH & operator + (const TenseurHH &) const ;
254 void operator += (const TenseurHH &);
255 TenseurHH & operator - () const ; // oppose du tenseur
256 TenseurHH & operator - (const TenseurHH &) const ;
257 void operator -= (const TenseurHH &);
258 TenseurHH & operator = ( const TenseurHH &);
259 TenseurHH & operator = ( const Tenseur_ns2HH & B)
260 { return this->operator=(*(const TenseurHH*) & B)) ;};
261 TenseurHH & operator * ( const double &) const ;
262 void operator *= ( const double &);
263 TenseurHH & operator / ( const double &) const ;
264 void operator /= ( const double &);
265 // affectation de B dans this, les données en trop sont ignorées
266 void Affectation_3D_a_2D(const Tenseur_ns3HH & B);
267
268 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
269 // plusZero = true: les données manquantes sont mises à 0
270 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
271 // des données possibles
272 void Affectation_trans_dimension(const TenseurHH & B,bool plusZero);
273
274 // produit contracte avec un vecteur
275 CoordonneeH operator * ( const CoordonneeB & ) const ;
276
277 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
278 // -> donc c'est l'indice du milieu qui est contracté
279 TenseurHH & operator * ( const TenseurBH &) const ;
280 TenseurHB & operator * ( const TenseurBB &) const ;
281 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
282 // -> on contracte d'abord l'indice du milieu puis l'indice externe
283 double operator && ( const TenseurBB &) const ;
284
285 // test
286 int operator == ( const TenseurHH &) const ;
287 int operator != ( const TenseurHH &) const ;
288 double Det() const ; // determinant de la matrice des coordonnees
289 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
290 TenseurHH & Transpose() const ;
291
292 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
293 virtual TenseurBB& Baisse2Indices() const;
294 virtual TenseurBH& BaissePremierIndice() const;
295 virtual TenseurHB& BaisseDernierIndice() const;
296
297 // calcul du maximum en valeur absolu des composantes du tenseur
298 double MaxiComposante() const;
299
300 // calcul du tenseur inverse par rapport au produit contracte
301 TenseurBB & Inverse() const ;
302
303 // retourne la composante i,j en lecture/écriture
304 double& Coor( const int i, const int j);
305
306 // retourne la composante i,j en lecture seule
307 double operator () ( const int i, const int j) const;
308
309 // lecture et écriture de données
310 istream & Lecture(istream & entree);
311 ostream & Ecriture(ostream & sort) const ;
312
313 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
314 static int OdVect(const int i, const int j) {return cdex.odVect(i,j)};
315 // transformation 1 indice -> 2 indices
316 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
317 static int idx_i(const int k) {return cdex.idx_i(k)};
318 static int idx_j(const int k) {return cdex.idx_j(k)};
319
320 protected :
321 // allocator dans la liste de data
322 listdouble4Iter ipointe;
323 // --- gestion d'index ----
324 class ChangementIndex
325 { public:
326     ChangementIndex();
327     // passage pour les index de la forme vecteur à la forme i,j
328     Tableau <int> idx_i,idx_j;
329     // passage pour les index de la forme i,j à la forme vecteur
330     Tableau2 <int> odVect;
331 };
332 static const ChangementIndex cdex;
333 };
334 /// @} // end of group
335
336 class Tenseur3BB;
337

```

```

338 /// @addtogroup Les_classes_tenseurs_dim2_ordre2
339 /// @{
340
341 //-----
342 /// Definition des tenseur derivees de dimension 2.
343 ///          cas des composantes deux fois covariantes symetriques
344 //-----
345 /// \author   Gérard Rio
346 /// \version  1.0
347 /// \date    23/01/97
348 class Tenseur2BB : public TenseurBB
349 {
350 // surcharge de l'operator de lecture
351 friend istream & operator » (istream &, Tenseur2BB &);
352 // surcharge de l'operator d'écriture
353 friend ostream & operator « (ostream &, const Tenseur2BB &);
354 friend class Tenseur3BB; // pour le passage 3D 2D: méthode Affectation_3D_a_2D(..
355 public :
356     // constructeur
357     Tenseur2BB() ; // par défaut
358     // initialisation de toutes les composantes a une meme valeur val
359     Tenseur2BB( const double val);
360     // initialisation avec 3 valeurs différentes correspondantes a
361     // (1,1) (2,2) (1,2) qui est identique a (2,1)
362     Tenseur2BB
363     ( const double val1, const double val2, const double val3) ;
364     // DESTRUCTEUR :
365     ~Tenseur2BB() ;
366     // constructeur a partir d'une instance non differenciee
367     Tenseur2BB ( const TenseurBB &);
368     // constructeur de copie
369     Tenseur2BB ( const Tenseur2BB &);
370
371     // METHODES PUBLIQUES :
372     // initialise toutes les composantes à val
373     void InitA(double val);
374     // operations
375     TenseurBB & operator + ( const TenseurBB &) const ;
376     void operator += ( const TenseurBB &);
377     TenseurBB & operator - () const ; // oppose du tenseur
378     TenseurBB & operator - (const TenseurBB &) const ;
379     void operator -- ( const TenseurBB &);
380     TenseurBB & operator = ( const TenseurBB &);
381     TenseurBB & operator = ( const Tenseur2BB & B)
382     { return this->operator=(*(const TenseurBB*) & B)) ;};
383     TenseurBB & operator * ( const double &) const ;
384     void operator *= ( const double &) ;
385     TenseurBB & operator / ( const double &) const ;
386     void operator /= ( const double &);
387     // affectation de B dans this, les données en trop sont ignorées
388     void Affectation_3D_a_2D(const Tenseur3BB & B);
389
390     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
391     // plusZero = true: les données manquantes sont mises à 0
392     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
393     // des données possibles
394     void Affectation_trans_dimension(const TenseurBB & B,bool plusZero);
395
396     // produit contracte avec un vecteur
397     CoordonneeB operator * ( const CoordonneeH &) const ;
398
399     // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
400     // -> donc c'est l'indice du milieu qui est contracté
401     TenseurBB & operator * ( const TenseurHB &) const ;
402     TenseurBH & operator * ( const TenseurHH &) const ;
403     // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
404     // -> on contracte d'abord l'indice du milieu puis l'indice externe
405     double operator && ( const TenseurHH &) const ;
406
407     // test
408     int operator == ( const TenseurBB &) const ;
409     int operator != ( const TenseurBB &) const ;
410
411     double Det() const ; // determinant de la matrice des coordonnees
412
413     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
414     TenseurBB & Transpose() const ;
415
416     // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
417     virtual TenseurHH& Monte2Indices() const ;
418     virtual TenseurHB& MontePremierIndice() const ;
419     virtual TenseurBH& MonteDernierIndice() const ;
420
421     // calcul du maximum en valeur absolu des composantes du tenseur
422     double MaxiComposante() const;
423
424     // calcul du tenseur inverse par rapport au produit contracte

```

```

425     TenseurHH & Inverse() const ;
426
427     // retourne la composante i,j en lecture/écriture
428     double& Coor( const int i, const int j);
429
430     // retourne la composante i,j en lecture seule
431     double operator () ( const int i, const int j) const;
432
433     //fonctions static définissant le produit tensoriel de deux vecteurs
434     // si les vecteurs sont égaux le tenseur est symétrique sinon il est non symétrique
435     static TenseurBB & Prod_tensoriel(const CoordonneeB & aB, const CoordonneeB & bB);
436
437     // lecture et écriture de données
438     istream & Lecture(istream & entree);
439     ostream & Ecriture(ostream & sort) const ;
440
441     // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
442     static int OdVect(const int i, const int j) {return cdex.odVect(i,j);};
443     // transformation 1 indice -> 2 indices
444     // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
445     static int idx_i(const int k) {return cdex.idx_i(k);};
446     static int idx_j(const int k) {return cdex.idx_j(k);};
447
448     protected :
449         // allocator dans la liste de data
450         listdouble3Iter ipointe;
451         // --- gestion d'index ----
452         class ChangementIndex
453         { public:
454             ChangementIndex();
455             // passage pour les index de la forme vecteur à la forme i,j
456             Tableau <int> idx_i,idx_j;
457             // passage pour les index de la forme i,j à la forme vecteur
458             Tableau2 <int> odVect;
459         };
460         static const ChangementIndex cdex;
461 };
462 /// @} // end of group
463
464 class Tenseur_ns3BB;
465
466 /// @addtogroup Les_classes_tenseurs_dim2_ordre2
467 /// @{
468
469 //-----
470 /// Definition des tenseur derivees de dimension 2.
471 /// cas des composantes deux fois covariantes non symetriques
472 /// pour les differencier la dimension = -2
473 //-----
474 /// \author Gérard Rio
475 /// \version 1.0
476 /// \date 23/01/97
477 class Tenseur_ns2BB : public TenseurBB
478 {
479     // surcharge de l'operator de lecture
480     friend istream & operator » (istream &, Tenseur_ns2BB &);
481     // surcharge de l'operator d'écriture
482     friend ostream & operator « (ostream &, const Tenseur_ns2BB &);
483     friend class Tenseur_ns3HH; // pour le passage 3D 2D: méthode Affectation_3D_a_2D(..
484     friend class Tenseur_ns2HH;
485     public :
486         // constructeur
487         Tenseur_ns2BB() ; // par défaut
488         // initialisation de toutes les composantes a une meme valeur val
489         Tenseur_ns2BB( const double val);
490         // initialisation avec 4 valeurs différentes correspondantes a
491         // (1,1) (2,2) (2,1) (1,2), car le tableau n'est pas symétrique !!!!
492         // le premier indice = ligne, le second = colonne
493         Tenseur_ns2BB
494         ( const double val1, const double val2, const double val3, const double val4) ;
495     // DESTRUCTEUR :
496     ~Tenseur_ns2BB() ;
497     // constructeur a partir d'une instance non differenciée
498     Tenseur_ns2BB ( const TenseurBB &);
499     // constructeur de copie
500     Tenseur_ns2BB ( const Tenseur_ns2BB &);
501
502     // METHODES PUBLIQUES :
503     // initialise toutes les composantes à val
504     void InitA(double val);
505     // operations
506     TenseurBB & operator + ( const TenseurBB &) const ;
507     void operator += ( const TenseurBB &);
508     TenseurBB & operator - () const ; // oppose du tenseur
509     TenseurBB & operator - ( const TenseurBB &) const ;
510     void operator -= ( const TenseurBB &);
511     TenseurBB & operator = ( const TenseurBB &);

```

```

512 TenseurBB & operator = ( const Tenseur_ns2BB & B)
513 { return this->operator=(*(const TenseurBB*) & B) );};
514 TenseurBB & operator * ( const double & ) const ;
515 void operator *= ( const double & );
516 TenseurBB & operator / (const double & ) const ;
517 void operator /= ( const double & );
518 // affectation de B dans this, les données en trop sont ignorées
519 void Affectation_3D_a_2D(const Tenseur_ns3BB & B);
520
521 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
522 // plusZero = true: les données manquantes sont mises à 0
523 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
524 // des données possibles
525 void Affectation_trans_dimension(const TenseurBB & B,bool plusZero);
526
527 // produit contracte avec un vecteur
528 CoordonneeB operator * ( const CoordonneeH & ) const ;
529
530 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
531 // -> donc c'est l'indice du milieu qui est contracté
532 TenseurBB & operator * ( const TenseurHB & ) const ;
533 TenseurBH & operator * ( const TenseurHH & ) const ;
534 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
535 // -> on contracte d'abord l'indice du milieu puis l'indice externe
536 double operator && ( const TenseurHH & ) const ;
537
538 // test
539 int operator == ( const TenseurBB & ) const ;
540 int operator != ( const TenseurBB & ) const ;
541
542 double Det() const ; // determinant de la matrice des coordonnees
543
544 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
545 TenseurBB & Transpose() const ;
546
547 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
548 virtual TenseurHH& Monte2Indices() const ;
549 virtual TenseurHB& MontePremierIndice() const ;
550 virtual TenseurBH& MonteDernierIndice() const ;
551
552 // calcul du maximum en valeur absolu des composantes du tenseur
553 double MaxiComposante() const;
554
555 // calcul du tenseur inverse par rapport au produit contracte
556 TenseurHH & Inverse() const ;
557
558 // retourne la composante i,j en lecture/écriture
559 double& Coor( const int i, const int j);
560
561 // retourne la composante i,j en lecture seule
562 double operator () ( const int i, const int j) const;
563
564 // lecture et écriture de données
565 istream & Lecture(istream & entree);
566 ostream & Ecriture(ostream & sort) const ;
567
568 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
569 static int OdVect(const int i, const int j) {return cdex.odVect(i,j);};
570 // transformation 1 indice -> 2 indices
571 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
572 static int idx_i(const int k) {return cdex.idx_i(k);};
573 static int idx_j(const int k) {return cdex.idx_j(k);};
574
575 protected :
576 // allocator dans la liste de data
577 listdouble4Iter ipointe;
578 // --- gestion d'index ---
579 class ChangementIndex
580 { public:
581     ChangementIndex();
582     // passage pour les index de la forme vecteur à la forme i,j
583     Tableau <int> idx_i,idx_j;
584     // passage pour les index de la forme i,j à la forme vecteur
585     Tableau2 <int> odVect;
586 };
587 static const ChangementIndex cdex;
588 };
589 /// @} // end of group
590
591 class Tenseur3BH;
592
593 /// @addtogroup Les_classes_tenseurs_dim2_ordre2
594 /// @{
595
596 //-----
597 /// Definition des tenseur derivees de dimension 2.
598 /// cas des composantes mixtes BH

```

```

599 //-----
600 /// \author   Gérard Rio
601 /// \version  1.0
602 /// \date    23/01/97
603 class Tenseur2BH : public TenseurBH
604 {
605 // surcharge de l'operator de lecture
606 friend istream & operator » (istream &, Tenseur2BH &);
607 // surcharge de l'operator d'écriture
608 friend ostream & operator « (ostream &, const Tenseur2BH &);
609 friend class Tenseur3BH; // pour le passage 3D 2D: méthode Affectation_3D_a_2D(..
610 public :
611 // constructeur
612 Tenseur2BH() ; // par défaut
613 // initialisation de toutes les composantes a une meme valeur val
614 Tenseur2BH ( const double val) ;
615 // initialisation avec 4 valeurs différentes correspondantes a
616 // (1,1) (2,2) (2,1) (1,2), car le tableau n'est pas symetrique !!!!
617 // le premier indice = ligne, le second = colonne
618 Tenseur2BH
619 ( const double val1, const double val2, const double val3, const double val4) ;
620
621 // DESTRUCTEUR :
622 ~Tenseur2BH();
623 // constructeur a partir d'une instance non differenciée
624 Tenseur2BH ( const TenseurBH &);
625 // constructeur de copie
626 Tenseur2BH ( const Tenseur2BH &);
627
628 // METHODES PUBLIQUES :
629 // initialise toutes les composantes à val
630 void Initia(double val);
631 // operations
632 TenseurBH & operator + ( const TenseurBH &) const ;
633 void operator += ( const TenseurBH &);
634 TenseurBH & operator - () const ; // oppose du tenseur
635 TenseurBH & operator - ( const TenseurBH &) const ;
636 void operator -= ( const TenseurBH &);
637 TenseurBH & operator = ( const TenseurBH &);
638 TenseurBH & operator = ( const Tenseur2BH & B)
639 { return this->operator=(*(const TenseurBH*) & B) ;};
640 TenseurBH & operator * ( const double &) const ;
641 void operator *= ( const double &);
642 TenseurBH & operator / ( const double &) const ;
643 void operator /= ( const double &);
644 // affectation de B dans this, les données en trop sont ignorées
645 void Affectation_3D_a_2D(const Tenseur3BH & B);
646
647 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
648 // plusZero = true: les données manquantes sont mises à 0
649 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
650 // des données possibles
651 void Affectation_trans_dimension(const TenseurBH & B,bool plusZero);
652
653 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
654 // -> donc c'est l'indice du milieu qui est contracté
655 CoordonneeB operator * (const CoordonneeB &) const ;
656 TenseurBH & operator * ( const TenseurBH &) const ; // produit une fois contracte
657 TenseurBH & operator * ( const TenseurBH &) const ; // produit une fois contracte
658 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
659 // -> on contracte d'abord l'indice du milieu puis l'indice externe
660 double operator && ( const TenseurBH &) const ; // produit deux fois contracte
661
662 // test
663 int operator == ( const TenseurBH &) const ;
664 int operator != ( const TenseurBH &) const ;
665 // calcul du tenseur inverse par rapport au produit contracte
666 TenseurBH & Inverse() const ;
667
668 double Trace() const ; // trace du tenseur ou premier invariant
669 double II() const ; // second invariant = trace (A*A)
670 double III() const ; // troisieme invariant = trace ((A*A)*A)
671 double Det() const ; // determinant de la matrice des coordonnees
672 // valeurs propre dans le vecteur de retour, classée par ordres "décroissants"
673 // cas indique le cas de valeur propre:
674 // cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
675 // dans ce cas les valeurs propres de retour sont nulles par défaut
676 // dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques
677 virtual Coordonnee ValPropre(int& cas) const ;
678 // idem met en retour la matrice mat contiennd par colonne les vecteurs propre
679 // elle doit avoir la dimension du tenseur
680 // le premier vecteur propre est exprime dans le repere dual
681 // le second vecteur propre est exprimé dans le repère naturel
682 virtual Coordonnee ValPropre(int& cas, Mat_pleine& mat) const ;
683
684 // ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres
685 // étant déjà connues

```



```

686 // en retour VP les vecteurs propre : doivent avoir la dimension du tenseur
687 // les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
688 // pour dim=2:le premier vecteur propre est exprime dans le repere naturel
689 // le second vecteur propre est exprimé dans le repère dual
690 // pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
691 // en sortie cas = -1 s'il y a eu un problème, dans ce cas, V_P est quelconque
692 // sinon si tout est ok, cas est identique en sortie avec l'entrée
693 virtual void VecteursPropres(const Coordonnee& Val_P,int& cas, Tableau <Coordonnee>& V_P) const;
694
695 // tenseur transpose
696 TenseurHB & Transpose() const ;
697 // permute Bas Haut, mais reste dans le même tenseur
698 void PermuteHautBas();
699
700 // calcul du maximum en valeur absolu des composantes du tenseur
701 double MaxiComposante() const;
702
703 // retourne la composante i,j en lecture/écriture
704 double& Coord(const int i, const int j);
705
706 // retourne la composante i,j en lecture seule
707 double operator () (const int i, const int j) const;
708
709 //fonctions static définissant le produit tensoriel de deux vecteurs
710 static TenseurBH & Prod_tensoriel(const CoordonneeB & aB, const CoordonneeH & bH);
711
712 // lecture et écriture de données
713 istream & Lecture(istream & entree);
714 ostream & Ecriture(ostream & sort) const ;
715
716 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
717 static int OdVect(const int i, const int j) {return cdex.odVect(i,j);};
718 // transformation 1 indice -> 2 indices
719 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
720 static int idx_i(const int k) {return cdex.idx_i(k);};
721 static int idx_j(const int k) {return cdex.idx_j(k);};
722
723 protected :
724 // allocator dans la liste de data
725 listdouble4Iter ipointe;
726
727 // --- gestion d'index ----
728 class ChangementIndex
729 { public:
730     ChangementIndex();
731     // passage pour les index de la forme vecteur à la forme i,j
732     Tableau <int> idx_i,idx_j;
733     // passage pour les index de la forme i,j à la forme vecteur
734     Tableau2 <int> odVect;
735 };
736 static const ChangementIndex cdex;
737 };
738 /// @} // end of group
739
740 class Tenseur3HB;
741
742 /// @addtogroup Les_classes_tenseurs_dim2_ordre2
743 /// @{
744
745 //-----
746 /// Definition des tenseur derivees de dimension 2.
747 /// cas des composantes mixtes HB
748 //-----
749 /// \author Gérard Rio
750 /// \version 1.0
751 /// \date 23/01/97
752 class Tenseur2HB : public TenseurHB
753 {
754 // surcharge de l'operator de lecture
755 friend istream & operator » (istream &, Tenseur2HB &);
756 // surcharge de l'operator d'écriture
757 friend ostream & operator « (ostream &, const Tenseur2HB &);
758 friend class Tenseur3HB; // pour le passage 3D 2D: méthode Affectation_3D_a_2D(..
759 public :
760 // Constructeur
761 Tenseur2HB() ; // par défaut
762 // initialisation de toutes les composantes a une meme valeur val
763 Tenseur2HB (const double val) ;
764 // initialisation avec 4 valeurs différentes correspondantes a
765 // (1,1) (2,2) (2,1) (1,2), car le tableau n'est pas symetrique !!!!
766 // le premier indice = ligne, le second = colonne
767 Tenseur2HB
768 (const double val1, const double val2, const double val3, const double val4) ;
769 // constructeur a partir d'une instance non differenciee
770 Tenseur2HB (const TenseurHB &);
771 // constructeur de copie
772 Tenseur2HB (const Tenseur2HB &);

```

```

773
774 // DESTRUCTEUR :
775 ~Tenseur2HB();
776
777 // METHODES PUBLIQUES :
778 // initialise toutes les composantes à val
779 void InitA(double val);
780 // operations
781 TenseurHB & operator + ( const TenseurHB & ) const ;
782 void operator += ( const TenseurHB & );
783 TenseurHB & operator - ( ) const ; // oppose du tenseur
784 TenseurHB & operator - ( const TenseurHB & ) const ;
785 void operator -= ( const TenseurHB & );
786 TenseurHB & operator = ( const TenseurHB & );
787 TenseurHB & operator = ( const Tenseur2HB & B)
788 { return this->operator=(*(const TenseurHB*) & B) ;};
789 TenseurHB & operator * ( const double & ) const ;
790 void operator *= ( const double & );
791 TenseurHB & operator / (const double & ) const ;
792 void operator /= ( const double & );
793 // affectation de B dans this, les données en trop sont ignorées
794 void Affectation_3D_a_2D(const Tenseur3HB & B);
795
796 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
797 // plusZero = true: les données manquantes sont mises à 0
798 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
799 // des données possibles
800 void Affectation_trans_dimension(const TenseurHB & B,bool plusZero);
801
802 // produit contracte avec un vecteur
803 CoordonneeH operator * ( const CoordonneeH & ) const ;
804
805 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
806 // -> donc c'est l'indice du milieu qui est contracté
807 TenseurHH & operator * ( const TenseurHH & ) const ;
808 TenseurHB & operator * ( const TenseurHB & ) const ; // produit une fois contracte
809 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
810 // -> on contracte d'abord l'indice du milieu puis l'indice externe
811 double operator && ( const TenseurHB & ) const ; // produit deux fois contracte
812
813 // test
814 int operator == ( const TenseurHB & ) const ;
815 int operator != ( const TenseurHB & ) const ;
816 // calcul du tenseur inverse par rapport au produit contracte
817 TenseurHB & Inverse() const ;
818
819 double Trace() const ; // trace du tenseur ou premier invariant
820 double II() const ; // second invariant = trace (A*A)
821 double III() const ; // troisieme invariant = trace ((A*A)*A)
822 double Det() const ; // determinant de la matrice des coordonnees
823 // valeurs propre dans le vecteur de retour, classée par ordres "décroissants"
824 // cas indique le cas de valeur propre:
825 // cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
826 // dans ce cas les valeurs propres de retour sont nulles par défaut
827 // dim = 2 , cas = 1 si deux valeurs propres distinctes, cas = 0 si deux val propres identiques
828 virtual Coordonnee ValPropre(int& cas) const ;
829 // idem met en retour la matrice mat contient par colonne les vecteurs propre
830 // elle doit avoir la dimension du tenseur
831 // le premier vecteur propre est exprimé dans le repere naturel
832 // le second vecteur propre est exprimé dans le repère dual
833 virtual Coordonnee ValPropre(int& cas, Mat_pleine& mat) const ;
834
835 // ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres
836 // étant déjà connues
837 // en retour VP les vecteurs propre : doivent avoir la dimension du tenseur
838 // les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
839 // pour dim=2:le premier vecteur propre est exprime dans le repere naturel
840 // le second vecteur propre est exprimé dans le repère dual
841 // pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
842 // en sortie cas = -1 s'il y a eu un problème, dans ce cas, V_P est quelconque
843 // sinon si tout est ok, cas est identique en sortie avec l'entrée
844 virtual void VecteursPropres(const Coordonnee& Val_P,int& cas, Tableau <Coordonnee>& V_P) const;
845
846 // tenseur transpose
847 TenseurBH & Transpose() const ;
848 // permute Bas Haut, mais reste dans le même tenseur
849 void PermuteHautBas();
850
851 // calcul du maximum en valeur absolu des composantes du tenseur
852 double MaxiComposante() const;
853
854 // retourne la composante i,j en lecture/écriture
855 double& Coor( const int i, const int j);
856
857 // retourne la composante i,j en lecture seule
858 double operator () ( const int i, const int j) const;
859

```

```

860 //   operator TenseurHB & (void)
861 //       { return *this;};
862
863 //fonctions static définissant le produit tensoriel de deux vecteurs
864 static TenseurHB & Prod_tensoriel(const CoordonneeH & aH, const CoordonneeB & bB);
865
866 // lecture et écriture de données
867 istream & Lecture(istream & entree);
868 ostream & Ecriture(ostream & sort) const ;
869
870 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
871 static int OdVect(const int i, const int j) {return cdex.odVect(i,j);};
872 // transformation 1 indice -> 2 indices
873 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
874 static int idx_i(const int k) {return cdex.idx_i(k);};
875 static int idx_j(const int k) {return cdex.idx_j(k);};
876
877     protected :
878         // allocator dans la liste de data
879         listdouble4Iter ipointe;
880
881         // --- gestion d'index ---
882         class ChangementIndex
883         { public:
884             ChangementIndex();
885             // passage pour les index de la forme vecteur à la forme i,j
886             Tableau <int> idx_i,idx_j;
887             // passage pour les index de la forme i,j à la forme vecteur
888             Tableau2 <int> odVect;
889         };
890         static const ChangementIndex cdex;
891 };
892 /// @} // end of group
893
894 #include "Tenseur3.h"
895
896 #ifndef MISE_AU_POINT
897 #include "Tenseur2-1.cc"
898 #include "Tenseur2-2.cc"
899 #include "Tenseur2_ns.cc"
900 #define Tenseur2_H_deja_inclus
901 #endif
902
903
904 #endif

```

## 7.466 Tenseur2\_TroisSym.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *           LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)           *
34 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex          *
35 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
36 *****/
37 *           DATE:           8/6/2003           *

```

```

38 *
39 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) $ *
40 * Tel 0297874571 fax : 02.97.87.45.72 *
41 * $ *
42 * PROJET: Herezh++ *
43 * $ *
44 *****
45 * BUT: Definition d'une classe derivee de tenseur du 4ieme ordre *
46 * de dimension 2 . Il s'agit ici de classes *
47 * très particulières. Les tenseurs possèdent les trois *
48 * symétrie: (ijkl) = (jikl) = (ijlk) = (klij) *
49 * Ainsi en composantes 4 fois covariantes ou 4 fois *
50 * contravariantes, en 2D il y a 6 composantes indépendantes.*
51 * La classe est plutôt à considérer comme un conteneur, *
52 * ainsi seules les méthodes principales sont utilisables. *
53 * $ *
54 * ***** *
55 * VERIFICATION: *
56 * *
57 * ! date ! auteur ! but ! *
58 * ----- *
59 * ! ! ! ! *
60 * $ *
61 * ***** *
62 * MODIFICATIONS: *
63 * ! date ! auteur ! but ! *
64 * ----- *
65 * $ *
66 *****/
67 #ifndef TENSEURQ2_TROIS_SYM_H
68 #define TENSEURQ2_TROIS_SYM_H
69
70 #include <iostream>
71 #include "TenseurQ.h"
72 #include "PtTabRel.h"
73 #include "Tableau2_T.h"
74 #include "Tenseur.h"
75
76 -----
77 // cas des composantes 4 fois contravariantes HHHH
78 -----
79 class TenseurQ2_troisSym_HHHH : public TenseurHHHH
80 { // surcharge de l'operator de lecture
81 friend istream & operator >> (istream &, TenseurQ2_troisSym_HHHH &);
82 // surcharge de l'operator d'écriture
83 friend ostream & operator << (ostream &, const TenseurQ2_troisSym_HHHH &);
84
85 public :
86 // Constructeur
87 TenseurQ2_troisSym_HHHH() ; // par défaut
88 // initialisation de toutes les composantes a une meme valeur val
89 TenseurQ2_troisSym_HHHH(const double& val);
90 // initialisation à partir des 6 coefficients indépendants
91 // (1111) (2222) (1122) (1212) (1211) (1222)
92 TenseurQ2_troisSym_HHHH(const double& x1111,const double& x2222,const double& x1122
93 ,const double& x1212,const double& x1211,const double& x1222);
94
95 // DESTRUCTEUR :
96 ~TenseurQ2_troisSym_HHHH() ;
97 // constructeur a partir d'une instance non differenciee
98 // il n'y a pas de vérification des symétries seules les grandeurs
99 // (1111) (2222) (1122) (1212) (1211) (1222) sont utilisées
100 TenseurQ2_troisSym_HHHH (const TenseurHHHH &);
101 // constructeur de copie
102 TenseurQ2_troisSym_HHHH (const TenseurQ2_troisSym_HHHH &);
103
104 // METHODES PUBLIQUES :
105 //2) virtuelles
106 // initialise toutes les composantes à val
107 void InitA(double val) ;
108 // operations
109 TenseurHHHH & operator + ( const TenseurHHHH & ) const ;
110 void operator += ( const TenseurHHHH &);
111 TenseurHHHH & operator - ( ) const ; // oppose du tenseur
112 TenseurHHHH & operator - ( const TenseurHHHH & ) const ;
113 void operator -= ( const TenseurHHHH &);
114 TenseurHHHH & operator = ( const TenseurHHHH &);
115 TenseurHHHH & operator * (const double & ) const ;
116 void operator *= ( const double &);
117 TenseurHHHH & operator / ( const double & ) const ;
118 void operator /= ( const double &);
119
120 // produit deux fois contracte à droite avec un tenseur du second ordre
121 // diffèrent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
122 TenseurHH& operator && ( const TenseurBB & ) const ;
123 // produit deux fois contracte à droite avec un tenseur du quatrième ordre

```

```

124 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
125 // symétrie
126 TenseurHHHH& operator && ( const TenseurBBHH & ) const
127 { cout << "\n non implante !! , TenseurHHHH& TenseurQ2_troisSym_HHHH::operator && ( const
128 TenseurBBHH & ) const ";
129 Sortie(1); TenseurHHHH* toto; return *toto; /* toto pour eviter un warning*/ };
130 TenseurHHBB& operator && ( const TenseurBBBB & ) const
131 { cout << "\n non implante !! , TenseurHHBB& TenseurQ2_troisSym_HHHH::operator && ( const
132 TenseurBBBB & ) const ";
133 Sortie(1); TenseurHHBB* toto; return *toto; /* toto pour eviter un warning*/ };
134 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
135 // les 2 premiers indices sont échangés avec les deux derniers indices
136 // ici en fait c'est le même tenseur grace aux symétries constitutives
137 TenseurHHHH & Transposelet2avec3et4() const ;
138 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
139 // plusZero = true: les données manquantes sont mises à 0
140 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
141 // des données possibles
142 void Affectation_trans_dimension(const TenseurHHHH & B,bool plusZero);
143 // test
144 int operator == ( const TenseurHHHH & ) const ;
145 // change la composante i,j,k,l du tenseur
146 // acces en ecriture,
147 void Change (int i, int j, int k, int l,const double& val) ;
148 // en cumul : équivalent de +=
149 void ChangePlus (int i, int j, int k, int l,const double& val);
150 // Retourne la composante i,j,k,l du tenseur
151 // acces en lecture seule
152 double operator () (int i, int j, int k, int l) const ;
153 // calcul du maximum en valeur absolu des composantes du tenseur
154 double MaxiComposante() const;
155 // lecture et écriture de données
156 istream & Lecture(istream & entree);
157 ostream & Ecriture(ostream & sort) const ;
158 protected :
159 // allocator dans la liste de data
160 listdouble6Iter ipointe;
161 // fonction pour le produit contracté à gauche
162 TenseurHH& Prod_gauche( const TenseurBB & F) const { return ((*this) && F); };
163 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
164 // symétrie
165 TenseurBBHH& Prod_gauche( const TenseurBBBB & ) const
166 { cout << "\n non implante !! , TenseurBBHH& TenseurQ2_troisSym_HHHH::Prod_gauche( const
167 TenseurBBBB & ) const ";
168 Sortie(1); TenseurBBHH* toto; return *toto; /* toto pour eviter un warning*/ };
169 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
170 // symétrie
171 TenseurHHHH& Prod_gauche( const TenseurHHBB & ) const
172 { cout << "\n non implante !! , TenseurHHHH& TenseurQ2_troisSym_HHHH::Prod_gauche( const TenseurHHBB
173 & ) const ";
174 Sortie(1); TenseurHHHH* toto; return *toto; /* toto pour eviter un warning*/ };
175 };
176 //-----
177 // cas des composantes 4 fois covariantes BBBB
178 //-----
179 class TenseurQ2_troisSym_BBBB : public TenseurBBBB
180 { // surcharge de l'operator de lecture
181 friend istream & operator » (istream &, TenseurQ2_troisSym_BBBB &);
182 // surcharge de l'operator d'écriture
183 friend ostream & operator « (ostream &, const TenseurQ2_troisSym_BBBB &);
184 public :
185 // Constructeur
186 TenseurQ2_troisSym_BBBB() ; // par défaut
187 // initialisation de toutes les composantes a une meme valeur val
188 TenseurQ2_troisSym_BBBB(const double& val);
189 // initialisation à partir des 6 coefficients indépendants
190 // (1111) (2222) (1122) (1212) (1211) (1222)
191 TenseurQ2_troisSym_BBBB(const double& x1111,const double& x2222,const double& x1122
192 ,const double& x1212,const double& x1211,const double& x1222);
193 // DESTRUCTEUR :
194 ~TenseurQ2_troisSym_BBBB() ;
195 // constructeur a partir d'une instance non differenciee
196 // il n'y a pas de vérification des symétries seules les grandeurs
197 // (1111) (2222) (1122) (1212) (1211) (1222) sont utilisées

```

```

204 TenseurQ2_troisSym_BBBB (const TenseurBBBB &);
205 // constructeur de copie
206 TenseurQ2_troisSym_BBBB (const TenseurQ2_troisSym_BBBB &);
207
208 // METHODES PUBLIQUES :
209 //2) virtuelles
210 // initialise toutes les composantes à val
211 void Initia(double val) ;
212 // operations
213 TenseurBBBB & operator + ( const TenseurBBBB & ) const ;
214 void operator += ( const TenseurBBBB & );
215 TenseurBBBB & operator - ( ) const ; // oppose du tenseur
216 TenseurBBBB & operator - ( const TenseurBBBB & ) const ;
217 void operator -= ( const TenseurBBBB & );
218 TenseurBBBB & operator = ( const TenseurBBBB & );
219 TenseurBBBB & operator * (const double & ) const ;
220 void operator *= ( const double & );
221 TenseurBBBB & operator / ( const double & ) const ;
222 void operator /= ( const double & );
223
224 // produit deux fois contracte à droite avec un tenseur du second ordre
225 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
226 TenseurBB& operator && ( const TenseurHH & ) const ;
227 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
228 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
229 TenseurBBHH& operator && ( const TenseurHHHH & ) const
230 { cout << "\n non implante !! , TenseurBBHH& TenseurQ2_troisSym_BBBB::operator && ( const
TenseurHHHH & ) const ";
231 Sortie(1); TenseurBBHH* toto; return *toto; /* toto pour eviter un warning*/ };
232 TenseurBBBB& operator && ( const TenseurHHBB & ) const
233 { cout << "\n non implante !! , TenseurBBBB& TenseurQ2_troisSym_BBBB::operator && ( const
TenseurHHBB & ) const ";
234 Sortie(1); TenseurBBBB* toto; return *toto; /* toto pour eviter un warning*/ };
235
236 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
237 // les 2 premiers indices sont échangés avec les deux derniers indices
238 // ici en fait c'est le même tenseur grace aux symétries constitutives
239 TenseurBBBB & Transposelet2avec3et4() const ;
240
241 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
242 // plusZero = true: les données manquantes sont mises à 0
243 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
244 // des données possibles
245 void Affectation_trans_dimension(const TenseurBBBB & B,bool plusZero);
246
247
248 // test
249 int operator == ( const TenseurBBBB & ) const ;
250
251 // change la composante i,j,k,l du tenseur
252 // acces en ecriture,
253 void Change (int i, int j, int k, int l,const double& val) ;
254 // en cumul : équivalent de +=
255 void ChangePlus (int i, int j, int k, int l,const double& val);
256
257 // Retourne la composante i,j,k,l du tenseur
258 // acces en lecture seule
259 double operator () (int i, int j, int k, int l) const ;
260
261 // calcul du maximum en valeur absolu des composantes du tenseur
262 double MaxiComposante() const;
263
264 // lecture et écriture de données
265 istream & Lecture(istream & entree);
266 ostream & Ecriture(ostream & sort) const ;
267
268 protected :
269 // allocator dans la liste de data
270 listdouble6Iter ipointe;
271
272 // fonction pour le produit contracté à gauche
273 TenseurBB& Prod_gauche( const TenseurHH & F) const { return ((*this) && F); };
274 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
275 TenseurHHBB& Prod_gauche( const TenseurHHHH & ) const
276 { cout << "\n non implante !! , TenseurHHBB& TenseurQ2_troisSym_BBBB::Prod_gauche( const
TenseurHHHH & ) const ";
277 Sortie(1); TenseurHHBB* toto; return *toto; /* toto pour eviter un warning*/ };
278 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
279 TenseurBBBB& Prod_gauche( const TenseurBBHH & ) const
280 { cout << "\n non implante !! , TenseurBBBB& TenseurQ2_troisSym_BBBB::Prod_gauche( const TenseurBBHH
& ) const ";
281 Sortie(1); TenseurBBBB* toto; return *toto; /* toto pour eviter un warning*/ };
282 };
283

```

```

284
285 #ifndef MISE_AU_POINT
286 #include "Tenseur2_TroisSym.cc"
287 #define TenseurQ2_TroisSym_H_deja_inclus
288 #endif
289
290
291 #endif

```

## 7.467 Tenseur3.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *      DATE:      23/01/97
34 *
35 *      AUTEUR:      G RIO      (mailto:gerardrio56@free.fr)
36 *
37 *      PROJET:      Herezh++
38 *
39 *****/
40 *      BUT:      Definition des classes derivees de dimension 3.
41 *
42 *      *****
43 *****/
44
45 #ifndef Tenseur3_H
46 #define Tenseur3_H
47
48 #include <iostream>
49 #include "Tenseur.h"
50 #include "PtTabRel.h"
51 #include "Tableau2_T.h"
52
53
54
55 /** @defgroup Les_classes_tenseurs_dim3_ordre2
56 *
57 *      BUT:      Définir les tenseurs d'ordre 2 de différentes composantes,
58 *      spécifiquement à la dimension 3.
59 *      L'objectif principal est de surcharger les différentes opérations
60 *      classiques.
61 *
62 *      concernant le produit contracté un fois, en particulier pour les tenseurs mixtes
63 *      il y a contraction du 2ième indice du premier tenseur avec le premier indice du second
64 *      tenseur :  $A_{ij} * B_{jk} = C_{ik} \Leftrightarrow A * B = C$ 
65 *      le tenseur inverse par rapport au produit contracté est défini de la manière suivante
66 *       $Inverse(A) * A = Id$ , ainsi l'inverse d'un tenseur BH est un BH idem pour les HB
67 *      mais l'inverse d'un BB est un HH, et l'inverse d'un HH est un BB
68 *
69 *      NB: lorsque les tenseurs mixtes sont issues de tenseurs HH ou BB symétrique, l'ordre
70 *      de contraction des indices n'a pas d'importance sur le résultat !!
71 *
72 *      le produit contracté de deux tenseurs symétriques quelconques ne donne pas un tenseur
73 *      symétrique !!, donc par exemple la contraction d'un tenseur HB avec HH n'est pas forcément
74 *      symétrique. Le résultat est symétrique SEULEMENT lorsque ces opérations sont effectuées

```

```

75 *   avec le tenseur metrique.
76 *
77 * \author   Gérard Rio
78 * \version  1.0
79 * \date     23/01/97
80 * \brief    Définition des classes de dimension 3 de type Tenseur d'ordre 2, en coordonnées avec
             différentes variances.
81 *
82 */
83
84
85 class Tenseur_ns3HH;class Tenseur2HH;
86 /// @addtogroup Les_classes_tenseurs_dim3_ordre2
87 /// @{
88
89 -----
90 /// Définition des tenseur derivees de dimension 3.
91 ///          cas des composantes deux fois contravariantes
92 -----
93 /// \author   Gérard Rio
94 /// \version  1.0
95 /// \date     23/01/97
96
97 class Tenseur3HH : public TenseurHH
98 {
99 // surcharge de l'operator de lecture
100 friend istream & operator » (istream &, Tenseur3HH &);
101 // surcharge de l'operator d'écriture
102 friend ostream & operator « (ostream &, const Tenseur3HH &);
103 friend class Tenseur2HH; // pour le passage 2D 3D: méthode Affectation_2D_a_3D(..
104 friend class Tenseur3BB;
105
106 public :
107     // Constructeur
108     Tenseur3HH() ; // par défaut
109     // initialisation de toutes les composantes a une meme valeur val
110     Tenseur3HH(const double val);
111     // initialisation avec 6 valeurs différentes correspondantes a
112     // (1,1) (2,2) (3,3) (2,1)=(1,2) (3,2)=(2,3) (3,1)=(1,3)
113     Tenseur3HH
114     (const double val1,const double val2,const double val3,const double val4,
115      const double val5,const double val6) ;
116     // DESTRUCTEUR :
117     ~Tenseur3HH() ;
118     // constructeur a partir d'une instance non differenciee
119     Tenseur3HH (const TenseurHH &);
120     // constructeur de copie
121     Tenseur3HH (const Tenseur3HH &);
122
123     // METHODES PUBLIQUES :
124     // initialise toutes les composantes à val
125     void InitA(double val);
126     // operations
127     TenseurHH & operator + (const TenseurHH &) const ;
128     void operator += ( const TenseurHH &);
129     TenseurHH & operator - () const ; // oppose du tenseur
130     TenseurHH & operator - ( const TenseurHH &) const ;
131     void operator -= ( const TenseurHH &);
132     TenseurHH & operator = ( const TenseurHH &);
133     TenseurHH & operator = ( const Tenseur3HH & B)
134     { return this->operator=((TenseurHH &) B); };
135     TenseurHH & operator = ( const Tenseur_ns3HH & B)
136     { return this->operator=((TenseurHH &) B); };
137     TenseurHH & operator * ( const double &) const ;
138     void operator *= ( const double &);
139     TenseurHH & operator / ( const double &) const ;
140     void operator /= ( const double &);
141     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
142     // plusZero = true: les données manquantes sont mises à 0
143     void Affectation_2D_a_3D(const Tenseur2HH & B,bool plusZero);
144
145     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
146     // plusZero = true: les données manquantes sont mises à 0
147     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
148     // des données possibles
149     void Affectation_trans_dimension(const TenseurHH & B,bool plusZero);
150
151     // produit contracte avec un vecteur
152     CoordonneeH operator * ( const CoordonneeB & ) const ;
153
154     // produit contracte contracté une fois A(i,j)+B(j,k)=A.B
155     // -> donc c'est l'indice du milieu qui est contracté
156     TenseurHH & operator * ( const TenseurBH &) const ;
157     TenseurHB & operator * ( const TenseurBB &) const ;
158     // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
159     // -> on contracte d'abord l'indice du milieu puis l'indice externe
160     double operator && ( const TenseurBB &) const ;

```



```

161
162 // test
163 int operator == ( const TenseurHH &) const ;
164 int operator != ( const TenseurHH &) const ;
165
166 double Det() const ; // determinant de la matrice des coordonnees
167
168 // calcul du tenseur inverse par rapport au produit contracte
169 TenseurBB & Inverse() const ;
170
171 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
172 TenseurHH & Transpose() const ;
173
174 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
175 virtual TenseurBB& Baisse2Indices() const;
176 virtual TenseurBH& BaissePremierIndice() const;
177 virtual TenseurHB& BaisseDernierIndice() const;
178
179 // calcul du maximum en valeur absolu des composantes du tenseur
180 double MaxiComposante() const;
181
182 // retourne la composante i,j en lecture et écriture
183 double& Coor( const int i, const int j);
184
185 // retourne la composante i,j en lecture seulement
186 double operator () ( const int i, const int j) const ;
187
188 //fonctions static définissant le produit tensoriel de deux vecteurs
189 // si les vecteurs sont égaux le tenseur est symétrique sinon il est non symétrique
190 static TenseurHH & Prod_tensoriel(const CoordonneeH & aH, const CoordonneeH & bH);
191
192 // lecture et écriture de données
193 istream & Lecture(istream & entree);
194 ostream & Ecriture(ostream & sort) const ;
195
196 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
197 // dans l'ordre: (1,1) (2,2) (3,3) (2,1)=(1,2) (3,2)=(2,3) (3,1)=(1,3)
198 static int OdVect(const int i, const int j) {return Tenseur3HH::cdex.odVect(i,j)};
199 // transformation 1 indice -> 2 indices
200 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
201 // dans l'ordre: (1,1) (2,2) (3,3) (2,1)=(1,2) (3,2)=(2,3) (3,1)=(1,3)
202 static int idx_i(const int k) {return Tenseur3HH::cdex.idx_i(k)};
203 static int idx_j(const int k) {return Tenseur3HH::cdex.idx_j(k)};
204
205 protected :
206 // allocator dans la liste de data
207 listdouble6Iter ipointe;
208 // --- gestion d'index ----
209 class ChangementIndex
210 { public:
211     ChangementIndex();
212     // passage pour les index de la forme vecteur à la forme i,j
213     Tableau <int> idx_i,idx_j;
214     // passage pour les index de la forme i,j à la forme vecteur
215     Tableau2 <int> odVect;
216 };
217 static const ChangementIndex cdex;
218 };
219 /// @} // end of group
220
221 class Tenseur_ns2HH;
222
223 /// @addtogroup Les_classes_tenseurs_dim3_ordre2
224 /// @{
225
226 //-----
227 /// Definition des tenseur derivees de dimension 3.
228 /// cas des composantes deux fois contravariantes non symetriques
229 /// pour les differencier la dimension = -3
230 //-----
231 /// \author Gérard Rio
232 /// \version 1.0
233 /// \date 23/01/97
234 class Tenseur_ns3HH : public TenseurHH
235 {
236 // surcharge de l'operator de lecture
237 friend istream & operator » (istream &, Tenseur_ns3HH &);
238 // surcharge de l'operator d'écriture
239 friend ostream & operator « (ostream &, const Tenseur_ns3HH &);
240 friend class Tenseur_ns2HH; // pour le passage 2D 3D: méthode Affectation_2D_a_3D(..
241 friend class Tenseur_ns3BB;
242 public :
243 // constructeur
244 Tenseur_ns3HH() ; // par défaut
245 // initialisation de toutes les composantes a une meme valeur val
246 Tenseur_ns3HH (double val) ;
247 // initialisation avec 9 valeurs différentes correspondantes aux trois

```

```

248 // lignes : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
249 // car le tableau n'est pas symetrique !!!!
250 // le premier indice = ligne, le second = colonne
251 Tenseur_ns3HH
252     ( const double val1,  const double val2, const double val3,  // 1ere ligne
253       const double val4,  const double val5, const double val6,  // 2ieme ligne
254       const double val7,  const double val8, const double val9) ; // 3ieme ligne
255
256 // DESTRUCTEUR :
257 ~Tenseur_ns3HH();
258 // constructeur a partir d'une instance non differenciee
259 Tenseur_ns3HH ( const TenseurHH &);
260 // constructeur de copie
261 Tenseur_ns3HH ( const Tenseur_ns3HH &);
262
263 // METHODES PUBLIQUES :
264 // initialise toutes les composantes à val
265 void InitA(double val);
266 // operations
267 TenseurHH & operator + ( const TenseurHH &) const ;
268 void operator += ( const TenseurHH &);
269 TenseurHH & operator - () const ; // oppose du tenseur
270 TenseurHH & operator - ( const TenseurHH &) const ;
271 void operator -= ( const TenseurHH &);
272 TenseurHH & operator = ( const TenseurHH &);
273 TenseurHH & operator = ( const Tenseur3HH & B)
274     { return this->operator=((TenseurHH &) B); };
275 TenseurHH & operator = ( const Tenseur_ns3HH & B)
276     { return this->operator=((TenseurHH &) B); };
277 TenseurHH & operator * ( const double &) const ;
278 void operator *= ( const double &);
279 TenseurHH & operator / ( const double &) const ;
280 void operator /= ( const double &);
281 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
282 // plusZero = true: les données manquantes sont mises à 0
283 void Affectation_2D_a_3D(const Tenseur_ns2HH & B,bool plusZero);
284
285 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
286 // plusZero = true: les données manquantes sont mises à 0
287 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
288 // des données possibles
289 void Affectation_trans_dimension(const TenseurHH & B,bool plusZero);
290
291 // produit contracte avec un vecteur
292 CoordonneeH operator * ( const CoordonneeB & ) const ;
293
294 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
295 // -> donc c'est l'indice du milieu qui est contracté
296 TenseurHH & operator * ( const TenseurBH &) const ;
297 TenseurHB & operator * ( const TenseurBB &) const ;
298 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
299 // -> on contracte d'abord l'indice du milieu puis l'indice externe
300 double operator && ( const TenseurBB &) const ;
301
302 // test
303 int operator == ( const TenseurHH &) const ;
304 int operator != ( const TenseurHH &) const ;
305
306 double Det() const ; // determinant de la matrice des coordonnees
307
308 // calcul du tenseur inverse par rapport au produit contracte
309 TenseurBB & Inverse() const ;
310
311 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
312 TenseurHH & Transpose() const ;
313
314 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
315 virtual TenseurBB& Baisse2Indices() const;
316 virtual TenseurBH& BaissePremierIndice() const;
317 virtual TenseurHB& BaisseDernierIndice() const;
318
319 // calcul du maximum en valeur absolu des composantes du tenseur
320 double MaxiComposante() const;
321
322 // retourne la composante i,j en lecture et écriture
323 double& Coor( const int i, const int j);
324
325 // retourne la composante i,j en lecture seulement
326 double operator () ( const int i, const int j) const ;
327
328 // lecture et écriture de données
329 istream & Lecture(istream & entree);
330 ostream & Ecriture(ostream & sort) const ;
331
332
333 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
334 // dans l'ordre : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)

```

```

335     static int OdVect(const int i, const int j) {return Tenseur_ns3HH::cdex.odVect(i,j);};
336     // transformation 1 indice -> 2 indices
337     // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
338     // dans l'ordre : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
339     static int idx_i(const int k) {return Tenseur_ns3HH::cdex.idx_i(k);};
340     static int idx_j(const int k) {return Tenseur_ns3HH::cdex.idx_j(k);};
341
342     protected :
343         // allocator dans la liste de data
344         // static list <Reels4> listdouble4;
345     listdouble9Iter ipointe;
346     // --- gestion d'index ----
347     class ChangementIndex
348     { public:
349         ChangementIndex();
350         // passage pour les index de la forme vecteur à la forme i,j
351         Tableau <int> idx_i,idx_j;
352         // passage pour les index de la forme i,j à la forme vecteur
353         Tableau2 <int> odVect;
354     };
355     static const ChangementIndex cdex;
356 };
357 /// @} // end of group
358
359 class Tenseur_ns3BB;class Tenseur2BB;
360
361 /// @addtogroup Les_classes_tenseurs_dim3_ordre
362 /// @{
363
364 //-----
365 /// Definition des tenseur derivees de dimension 3.
366 /// cas des composantes deux fois covariantes
367 //-----
368 /// \author Gérard Rio
369 /// \version 1.0
370 /// \date 23/01/97
371 class Tenseur3BB : public TenseurBB
372 {
373     // surcharge de l'operator de lecture
374     friend istream & operator » (istream &, Tenseur3BB &);
375     // surcharge de l'operator d'écriture
376     friend ostream & operator « (ostream &, const Tenseur3BB &);
377     friend class Tenseur2BB; // pour le passage 2D 3D: méthode Affectation_2D_a_3D(..
378     friend class Tenseur3HH;
379     public :
380         // constructeur
381         Tenseur3BB(); // par défaut
382         // initialisation de toutes les composantes a une meme valeur val
383         Tenseur3BB( const double val);
384         // initialisation avec 6 valeurs différentes correspondantes a
385         // (1,1) (2,2) (3,3) (2,1)=(1,2) (3,2)=(2,3) (3,1)=(1,3)
386         Tenseur3BB
387         ( const double val1, const double val2, const double val3,
388         const double val4, const double val5, const double val6) ;
389         // DESTRUCTEUR :
390         ~Tenseur3BB() ;
391         // constructeur a partir d'une instance non differenciee
392         Tenseur3BB ( const TenseurBB &);
393         // constructeur de copie
394         Tenseur3BB ( const Tenseur3BB &);
395
396         // METHODES PUBLIQUES :
397         // initialise toutes les composantes à val
398         void InitA(double val);
399         // operations
400         TenseurBB & operator + ( const TenseurBB &) const ;
401         void operator += ( const TenseurBB &);
402         TenseurBB & operator - () const ; // oppose du tenseur
403         TenseurBB & operator - ( const TenseurBB &) const ;
404         void operator -= ( const TenseurBB &);
405         TenseurBB & operator = ( const TenseurBB &) ;
406         TenseurBB & operator = ( const Tenseur3BB & B)
407         { return this->operator=((TenseurBB &) B); };
408         TenseurBB & operator = ( const Tenseur_ns3BB & B)
409         { return this->operator=((TenseurBB &) B); };
410         TenseurBB & operator * ( const double &) const ;
411         void operator *= ( const double &);
412         TenseurBB & operator / ( const double &) const ;
413         void operator /= ( const double &);
414         // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
415         // plusZero = true: les données manquantes sont mises à 0
416         void Affectation_2D_a_3D(const Tenseur2BB & B,bool plusZero);
417
418         // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
419         // plusZero = true: les données manquantes sont mises à 0
420         // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
421         // des données possibles

```

```

422 void Affectation_trans_dimension(const TenseurBB & B,bool plusZero);
423
424 // produit contracte avec un vecteur
425 CoordonneeB operator * ( const CoordonneeH & ) const ;
426
427 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
428 // -> donc c'est l'indice du milieu qui est contracté
429 TenseurBB & operator * ( const TenseurHB & ) const ;
430 TenseurBH & operator * ( const TenseurHH & ) const ;
431 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
432 // -> on contracte d'abord l'indice du milieu puis l'indice externe
433 double operator && ( const TenseurHH & ) const ;
434
435 // test
436 int operator == ( const TenseurBB & ) const ;
437 int operator != ( const TenseurBB & ) const ;
438
439 double Det() const ; // determinant de la matrice des coordonnees
440
441 // calcul du tenseur inverse par rapport au produit contracte
442 TenseurHH & Inverse() const ;
443
444 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
445 TenseurBB & Transpose() const ;
446
447 // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
448 virtual TenseurHH& Monte2Indices() const ;
449 virtual TenseurHB& MontePremierIndice() const ;
450 virtual TenseurBH& MonteDernierIndice() const ;
451
452 // calcul du maximum en valeur absolu des composantes du tenseur
453 double MaxiComposante() const;
454
455 // retourne la composante i,j en lecture et écriture
456 double& Coor( const int i, const int j);
457
458 // retourne la composante i,j en lecture seulement
459 double operator () ( const int i, const int j) const ;
460
461 //fonctions static définissant le produit tensoriel de deux vecteurs
462 // si les vecteurs sont égaux le tenseur est symétrique sinon il est non symétrique
463 static TenseurBB & Prod_tensoriel(const CoordonneeB & aB, const CoordonneeB & bB);
464
465 // lecture et écriture de données
466 istream & Lecture(istream & entree);
467 ostream & Ecriture(ostream & sort) const ;
468
469
470 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
471 // dans l'ordre: (1,1) (2,2) (3,3) (2,1)=(1,2) (3,2)=(2,3) (3,1)=(1,3)
472 static int OdVect(const int i, const int j) {return Tenseur3BB::cdex.odVect(i,j);};
473 // transformation 1 indice -> 2 indices
474 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
475 // dans l'ordre: (1,1) (2,2) (3,3) (2,1)=(1,2) (3,2)=(2,3) (3,1)=(1,3)
476 static int idx_i(const int k) {return Tenseur3BB::cdex.idx_i(k);};
477 static int idx_j(const int k) {return Tenseur3BB::cdex.idx_j(k);};
478
479 protected :
480 // allocator dans la liste de data
481 listdouble6Iter ipointe;
482 // --- gestion d'index ----
483 class ChangementIndex
484 { public:
485     ChangementIndex();
486     // passage pour les index de la forme vecteur à la forme i,j
487     Tableau <int> idx_i,idx_j;
488     // passage pour les index de la forme i,j à la forme vecteur
489     Tableau2 <int> odVect;
490 };
491 static const ChangementIndex cdex;
492 };
493 /// @} // end of group
494
495 class Tenseur_ns2BB;
496
497 /// @addtogroup Les_classes_tenseurs_dim3_ordre2
498 /// @{
499
500 //-----
501 /// Definition des tenseur derivees de dimension 3.
502 /// cas des composantes deux fois covariantes non symetriques
503 /// pour les differencier la dimension = -3
504 //-----
505 /// \author Gérard Rio
506 /// \version 1.0
507 /// \date 23/01/97
508 class Tenseur_ns3BB : public TenseurBB

```

```

509 {
510 // surcharge de l'operator de lecture
511 friend istream & operator >> (istream &, Tenseur_ns3BB &);
512 // surcharge de l'operator d'écriture
513 friend ostream & operator << (ostream &, const Tenseur_ns3BB &);
514 friend class Tenseur_ns2BB; // pour le passage 2D 3D: méthode Affectation_2D_a_3D(..
515 friend class Tenseur_ns3HH;
516 public :
517     // Constructeur
518     Tenseur_ns3BB() ; // par défaut
519     // initialisation de toutes les composantes a une meme valeur val
520     Tenseur_ns3BB ( const double val) ;
521     // initialisation avec 9 valeurs différentes correspondantes aux trois
522     // lignes : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
523     // car le tableau n'est pas symetrique !!!!
524     // le premier indice = ligne, le second = colonne
525     Tenseur_ns3BB
526     ( const double val1, const double val2, const double val3, // 1ere ligne
527       const double val4, const double val5, const double val6, // 2ieme ligne
528       const double val7, const double val8, const double val9) ; // 3ieme ligne
529     // constructeur a partir d'une instance non differenciee
530     Tenseur_ns3BB ( const TenseurBB &);
531     // constructeur de copie
532     Tenseur_ns3BB ( const Tenseur_ns3BB &);
533
534     // DESTRUCTEUR :
535     ~Tenseur_ns3BB();
536
537     // METHODES PUBLIQUES :
538     // initialise toutes les composantes à val
539     void InitA(double val);
540     // operations
541     TenseurBB & operator + ( const TenseurBB &) const ;
542     void operator += ( const TenseurBB &);
543     TenseurBB & operator - ( ) const ; // oppose du tenseur
544     TenseurBB & operator - ( const TenseurBB &) const ;
545     void operator -= ( const TenseurBB &);
546     TenseurBB & operator = ( const TenseurBB &);
547     TenseurBB & operator = ( const Tenseur3BB & B)
548     { return this->operator=((TenseurBB &) B); };
549     TenseurBB & operator = ( const Tenseur_ns3BB & B)
550     { return this->operator=((TenseurBB &) B); };
551     TenseurBB & operator * ( const double &) const ;
552     void operator *= ( const double &);
553     TenseurBB & operator / ( const double &) const ;
554     void operator /= ( const double &);
555     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
556     // plusZero = true: les données manquantes sont mises à 0
557     void Affectation_2D_a_3D(const Tenseur_ns2BB & B,bool plusZero);
558
559     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
560     // plusZero = true: les données manquantes sont mises à 0
561     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
562     // des données possibles
563     void Affectation_trans_dimension(const TenseurBB & B,bool plusZero);
564
565     // produit contracte avec un vecteur
566     CoordonneeB operator * ( const CoordonneeH & ) const ;
567
568     // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
569     // -> donc c'est l'indice du milieu qui est contracté
570     TenseurBB & operator * ( const TenseurHB &) const ;
571     TenseurBH & operator * ( const TenseurHH &) const ;
572     // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B
573     // -> on contracte d'abord l'indice du milieu puis l'indice externe
574     double operator && ( const TenseurHH &) const ;
575
576     // test
577     int operator == ( const TenseurBB &) const ;
578     int operator != ( const TenseurBB &) const ;
579
580     double Det() const ; // determinant de la matrice des coordonnees
581
582     // calcul du tenseur inverse par rapport au produit contracte
583     TenseurHH & Inverse() const ;
584
585     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
586     TenseurBB & Transpose() const ;
587
588     // ---- manipulation d'indice ---- -> création de nouveaux tenseurs
589     virtual TenseurHH& Monte2Indices() const ;
590     virtual TenseurHB& MontePremierIndice() const ;
591     virtual TenseurBH& MonteDernierIndice() const ;
592
593     // calcul du maximum en valeur absolu des composantes du tenseur
594     double MaxiComposante() const;
595

```

```

596 // retourne la composante i,j en lecture et écriture
597 double& Coord( const int i, const int j);
598
599 // retourne la composante i,j en lecture seulement
600 double operator () ( const int i, const int j) const ;
601
602 // lecture et écriture de données
603 istream & Lecture(istream & entree);
604 ostream & Ecriture(ostream & sort) const ;
605
606
607 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
608 // dans l'ordre : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
609 static int OdVect(const int i, const int j) {return Tenseur_ns3BB::cdex.odVect(i,j);};
610 // transformation 1 indice -> 2 indices
611 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
612 // dans l'ordre : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
613 static int idx_i(const int k) {return Tenseur_ns3BB::cdex.idx_i(k);};
614 static int idx_j(const int k) {return Tenseur_ns3BB::cdex.idx_j(k);};
615
616 protected :
617 // allocator dans la liste de data
618 listdouble9Iter ipointe;
619 // --- gestion d'index ----
620 class ChangementIndex
621 { public:
622   ChangementIndex();
623   // passage pour les index de la forme vecteur à la forme i,j
624   Tableau <int> idx_i,idx_j;
625   // passage pour les index de la forme i,j à la forme vecteur
626   Tableau2 <int> odVect;
627 };
628 static const ChangementIndex cdex;
629 };
630 /// @} // end of group
631
632 class Tenseur2BH;
633
634 /// @addtogroup Les_classes_tenseurs_dim3_ordre2
635 /// @{
636
637 //-----
638 /// Definition des tenseur derivees de dimension 3.
639 /// cas des composantes mixtes BH
640 //-----
641 /// \author Gérard Rio
642 /// \version 1.0
643 /// \date 23/01/97
644 class Tenseur3BH : public TenseurBH
645 {
646 // surcharge de l'operator de lecture
647 friend istream & operator » (istream &, Tenseur3BH &);
648 // surcharge de l'operator d'écriture
649 friend ostream & operator « (ostream &, const Tenseur3BH &);
650 friend class Tenseur2BH; // pour le passage 2D 3D: méthode Affectation_2D_a_3D(..
651 public :
652 // constructeur
653 Tenseur3BH() ; // par défaut
654 // initialisation de toutes les composantes a une meme valeur val
655 Tenseur3BH ( const double val) ;
656 // initialisation avec 9 valeurs différentes correspondantes aux trois
657 // lignes : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
658 // car le tableau n'est pas symetrique !!!!
659 // le premier indice = ligne, le second = colonne
660 Tenseur3BH
661 ( const double val1, const double val2, const double val3, // 1ere ligne
662 const double val4, const double val5, const double val6, // 2ieme ligne
663 const double val7, const double val8, const double val9) ; // 3ieme ligne
664
665 // DESTRUCTEUR :
666 ~Tenseur3BH();
667 // constructeur a partir d'une instance non differenciee
668 Tenseur3BH ( const TenseurBH &);
669 // constructeur de copie
670 Tenseur3BH ( const Tenseur3BH &);
671
672 // METHODES PUBLIQUES :
673 // initialise toutes les composantes à val
674 void Inita(double val);
675 // operations
676 TenseurBH & operator + ( const TenseurBH &) const ;
677 void operator += ( const TenseurBH &);
678 TenseurBH & operator - () const ; // oppose du tenseur
679 TenseurBH & operator - ( const TenseurBH &) const ;
680 void operator -= ( const TenseurBH &);
681 TenseurBH & operator = ( const TenseurBH &);
682 TenseurBH & operator = ( const Tenseur3BH & B)

```

```

683     { return this->operator=((TenseurBH &) B); };
684 TenseurBH & operator * ( const double &) const ;
685 void operator *= ( const double &);
686 TenseurBH & operator / ( const double &) const ;
687 void operator /= ( const double &);
688 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
689 // plusZero = true: les données manquantes sont mises à 0
690 void Affectation_2D_a_3D(const Tenseur2BH & B,bool plusZero);
691
692 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
693 // plusZero = true: les données manquantes sont mises à 0
694 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
695 // des données possibles
696 void Affectation_trans_dimension(const TenseurBH & B,bool plusZero);
697
698 // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
699 // -> donc c'est l'indice du milieu qui est contracté
700 CoordonneeB operator * ( const CoordonneeB & ) const ;
701 TenseurBB & operator * ( const TenseurBB &) const ; // produit une fois contracte
702 TenseurBH & operator * ( const TenseurBH &) const ; // produit une fois contracte
703 // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B c-c-d : A_i^{.k} * B_k^{.i}
704 // -> on contracte d'abord l'indice du milieu puis l'indice externe
705 // cela permet d'utiliser les mêmes variances pour les deux tenseurs
706
707 double operator && ( const TenseurBH &) const ; // produit deux fois contracte
708
709 // test
710 int operator == ( const TenseurBH &) const ;
711 int operator != ( const TenseurBH &) const ;
712
713 // calcul du tenseur inverse par rapport au produit contracte
714 TenseurBH & Inverse() const ;
715
716 double Trace() const ; // trace du tenseur ou premier invariant
717 double II() const ; // second invariant = trace (A*A)
718 double III() const ; // troisieme invariant = trace (A*A)*A)
719 double Det() const ; // determinant de la matrice des coordonnees
720
721 // valeurs propre dans le vecteur de retour, classée par ordres "décroissants"
722 // cas indique le cas de valeur propre:
723 // cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
724 // dans ce cas les valeurs propres de retour sont nulles par défaut
725 // dim = 3 , cas = 1 si 3 val propres différentes, cas = 0 si les 3 sont identiques,
726 // cas = 2 si val(1)=val(2), cas = 3 si val(2)=val(3)
727 virtual Coordonnee ValPropre(int& cas) const ;
728 // idem met en retour la matrice mat contiend par colonne les vecteurs propre
729 // elle doit avoir la dimension du tenseur
730 // les vecteurs propre sont exprime dans le repere dual (contrairement au HB)
731 virtual Coordonnee ValPropre(int& cas, Mat_pleine& mat) const ;
732
733 // ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres
734 // étant déjà connues
735 // en retour VP les vecteurs propre: doivent avoir la dimension du tenseur
736 // les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
737 // pour dim=2:le premier vecteur propre est exprime dans le repere naturel
738 // le second vecteur propre est exprimé dans le repère dual
739 // pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
740 // en sortie cas = -1 s'il y a eu un problème, dans ce cas, V_P est quelconque
741 // sinon si tout est ok, cas est identique en sortie avec l'entrée
742 virtual void VecteursPropres(const Coordonnee& Val_P,int& cas, Tableau <Coordonnee>& V_P) const;
743
744 // tenseur transpose
745 TenseurHB & Transpose() const ;
746 // permute Bas Haut, mais reste dans le même tenseur
747 void PermuteHautBas();
748
749 // calcul du maximum en valeur absolu des composantes du tenseur
750 double MaxiComposante() const;
751
752 // retourne la composante i,j en lecture et écriture
753 double& Coor( const int i, const int j);
754
755 // retourne la composante i,j en lecture seulement
756 double operator () ( const int i, const int j) const ;
757
758 //fonctions static définissant le produit tensoriel de deux vecteurs
759 static TenseurBH & Prod_tensoriel(const CoordonneeB & aB, const CoordonneeH & bH);
760
761 // lecture et écriture de données
762 istream & Lecture(istream & entree);
763 ostream & Ecriture(ostream & sort) const ;
764
765
766 // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
767 // dans l'ordre : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
768 static int OdVect(const int i, const int j) {return Tenseur3BH::cdex.odVect(i,j);};
769 // transformation 1 indice -> 2 indices

```

```

770 // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
771 // dans l'ordre : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
772 static int idx_i(const int k) {return Tenseur3BH::cdex.idx_i(k);};
773 static int idx_j(const int k) {return Tenseur3BH::cdex.idx_j(k);};
774
775 protected :
776 // allocator dans la liste de data
777 listdouble9Iter ipointe;
778 // --- gestion d'index ----
779 class ChangementIndex
780 { public:
781   ChangementIndex();
782   // passage pour les index de la forme vecteur à la forme i,j
783   Tableau<int> idx_i,idx_j;
784   // passage pour les index de la forme i,j à la forme vecteur
785   Tableau2<int> odVect;
786 };
787 static const ChangementIndex cdex;
788 };
789 class Tenseur2HB;
790 /// @} // end of group
791
792 /// @addtogroup Les_classes_tenseurs_dim3_ordre2
793 /// @{
794
795 //-----
796 /// Definition des tenseur derivees de dimension 3.
797 /// cas des composantes mixtes HB
798 //-----
799 /// \author Gérard Rio
800 /// \version 1.0
801 /// \date 23/01/97
802 class Tenseur3HB : public TenseurHB
803 {
804 // surcharge de l'operator de lecture
805 friend istream & operator » (istream &, Tenseur3HB &);
806 // surcharge de l'operator d'écriture
807 friend ostream & operator « (ostream &, const Tenseur3HB &);
808 friend class Tenseur2HB; // pour le passage 2D 3D: méthode Affectation_2D_a_3D(..
809 public :
810 // Constructeur
811 Tenseur3HB() ; // par défaut
812 // initialisation de toutes les composantes a une meme valeur val
813 Tenseur3HB ( const double val) ;
814 // initialisation avec 9 valeurs différentes correspondantes aux trois
815 // lignes : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
816 // car le tableau n'est pas symetrique !!!!
817 // le premier indice = ligne, le second = colonne
818 Tenseur3HB
819 ( const double val1, const double val2, const double val3, // 1ere ligne
820 const double val4, const double val5, const double val6, // 2ieme ligne
821 const double val7, const double val8, const double val9) ; // 3ieme ligne
822 // constructeur a partir d'une instance non differenciee
823 Tenseur3HB ( const TenseurHB &);
824 // constructeur de copie
825 Tenseur3HB ( const Tenseur3HB &);
826
827 // DESTRUCTEUR :
828 ~Tenseur3HB();
829
830 // METHODES PUBLIQUES :
831 // initialise toutes les composantes à val
832 void Inita(double val);
833 // operations
834 TenseurHB & operator + ( const TenseurHB &) const ;
835 void operator += ( const TenseurHB &);
836 TenseurHB & operator - () const ; // oppose du tenseur
837 TenseurHB & operator - ( const TenseurHB &) const ;
838 void operator -= ( const TenseurHB &);
839 TenseurHB & operator = ( const TenseurHB &);
840 TenseurHB & operator = ( const Tenseur3HB & B)
841 { return this->operator=((TenseurHB &) B); };
842 TenseurHB & operator * ( const double &) const ;
843 void operator *= ( const double &);
844 TenseurHB & operator / ( const double &) const ;
845 void operator /= ( const double &);
846 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
847 // plusZero = true: les données manquantes sont mises à 0
848 void Affectation_2D_a_3D(const Tenseur2HB & B,bool plusZero);
849
850 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
851 // plusZero = true: les données manquantes sont mises à 0
852 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
853 // des données possibles
854 void Affectation_trans_dimension(const TenseurHB & B,bool plusZero);
855
856 // produit contracte avec un vecteur

```



```

857   CoordonneeH operator * ( const CoordonneeH & ) const ;
858
859   // produit contracte contracté une fois A(i,j)*B(j,k)=A.B
860   // -> donc c'est l'indice du milieu qui est contracté
861   TenseurHH & operator * ( const TenseurHH & ) const ;
862   TenseurHB & operator * ( const TenseurHB & ) const ; // produit une fois contracte
863   // produit contracte contracté deux fois A(i,j)*B(j,i)=A..B c-c-d : A^i_{.k} * B^k_{.i}
864   // -> on contracte d'abord l'indice du milieu puis l'indice externe
865   // cela permet d'utiliser les mêmes variances pour les deux tenseurs
866   double operator && ( const TenseurHB & ) const ; // produit deux fois contracte
867
868   // test
869   int operator == ( const TenseurHB & ) const ;
870   int operator != ( const TenseurHB & ) const ;
871
872   // calcul du tenseur inverse par rapport au produit contracte
873   TenseurHB & Inverse() const ;
874
875   double Trace() const ; // trace du tenseur ou premier invariant
876   double II() const ; // second invariant = trace (A*A)
877   double III() const ; // troisieme invariant = trace ((A*A)*A)
878   double Det() const ; // determinant de la matrice des coordonnees
879
880   // valeurs propre dans le vecteur de retour, classée par ordres "décroissants"
881   // cas indique le cas de valeur propre:
882   // cas = -1, si l'extraction des valeurs propres n'a pas pu se faire
883   // dans ce cas les valeurs propres de retour sont nulles par défaut
884   // dim = 3 , cas = 1 si 3 val propres différentes, cas = 0 si les 3 sont identiques,
885   // cas = 2 si val(1)=val(2), cas = 3 si val(2)=val(3)
886   virtual Coordonnee ValPropre(int& cas) const ;
887   // idem met en retour la matrice mat contient par colonne les vecteurs propre
888   // elle doit avoir la dimension du tenseur
889   // les vecteurs propre sont exprime dans le repere naturel
890   virtual Coordonnee ValPropre(int& cas, Mat_pleine& mat) const ;
891
892   // ici il s'agit uniquement de calculer les vecteurs propres, les valeurs propres
893   // étant déjà connues
894   // en retour VP les vecteurs propre: doivent avoir la dimension du tenseur
895   // les vecteurs propre sont exprime dans le repere naturel (pour les tenseurs dim 3
896   // pour dim=2:le premier vecteur propre est exprime dans le repere naturel
897   // le second vecteur propre est exprimé dans le repère dual
898   // pour dim=1 le vecteur est dans la base naturelle (mais ça importe pas)
899   // en sortie cas = -1 s'il y a eu un problème, dans ce cas, V_P est quelconque
900   // sinon si tout est ok, cas est identique en sortie avec l'entrée
901   virtual void VecteursPropres(const Coordonnee& Val_P,int& cas, Tableau <Coordonnee>& V_P) const ;
902
903   // tenseur transpose
904   TenseurBH & Transpose() const ;
905   // permute Bas Haut, mais reste dans le même tenseur
906   void PermuteHautBas();
907
908   // calcul du maximum en valeur absolu des composantes du tenseur
909   double MaxiComposante() const;
910
911   // retourne la composante i,j en lecture et écriture
912   double& Coor( const int i, const int j);
913
914   // retourne la composante i,j en lecture seulement
915   double operator () ( const int i, const int j) const ;
916
917   // inline operator TenseurHB & (void)
918   // { return *this;};
919
920   //fonctions static définissant le produit tensoriel de deux vecteurs
921   static TenseurHB & Prod_tensoriel(const CoordonneeH & aH, const CoordonneeB & bB);
922
923   // lecture et écriture de données
924   istream & Lecture(istream & entree);
925   ostream & Ecriture(ostream & sort) const ;
926
927   // transformation 2 indices -> 1 indice (c-a-d vecteur unicolonne)
928   // dans l'ordre : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
929   static int OdVect(const int i, const int j) {return Tenseur3HB::cdex.odVect(i,j);};
930   // transformation 1 indice -> 2 indices
931   // en retour 2 indices : (k)(1) et (k)(2) -> les indices (i,j)
932   // dans l'ordre : (1,1) (1,2) (1,3) ; (2,1) (2,2) (2,3) ; (3,1) (3,2) (3,3)
933   static int idx_i(const int k) {return Tenseur3HB::cdex.idx_i(k);};
934   static int idx_j(const int k) {return Tenseur3HB::cdex.idx_j(k);};
935
936   protected :
937   // allocator dans la liste de data
938   listdouble9Iter ipointe;
939   // --- gestion d'index ----
940   class ChangementIndex
941   { public:
942     ChangementIndex();
943     // passage pour les index de la forme vecteur à la forme i,j

```

```

944     Tableau <int> idx_i,idx_j;
945     // passage pour les index de la forme i,j à la forme vecteur
946     Tableau2 <int> odVect;
947     };
948     static const ChangementIndex cdex;
949 };
950 /// @} // end of group
951
952 #include "Tenseur2.h"
953
954 #ifndef MISE_AU_POINT
955     #include "Tenseur3-1.cc"
956     #include "Tenseur3-2.cc"
957     #include "Tenseur3_ns.cc"
958     #define Tenseur3_H_deja_inclus
959 #endif
960
961
962 #endif

```

## 7.468 Tenseur3\_TroisSym.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *           LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)
34 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex
35 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr
36 *****/
37 *   DATE:           8/6/2003
38 *
39 *   AUTEUR:         G RIO (mailto:gerard.rio@univ-ubs.fr)
40 *                   Tel 0297874571 fax : 02.97.87.45.72
41 *
42 *   PROJET:         Herezh++
43 *
44 *****/
45 *   BUT:   Definition d'une classe derivee de tenseur du 4ieme ordre
46 *          de dimension 3 . Il s'agit ici de classes
47 *          très particulières. Les tenseurs possèdent les trois
48 *          symétrie: (ijkl) = (jikl) = (klij)
49 *          Ainsi en composantes 4 fois covariantes ou 4 fois
50 *          contravariantes, en 3D il y a 21 composantes indépendantes.
51 *          La classe est plutôt à considérer comme un conteneur,
52 *          ainsi seules les méthodes principales sont utilisables.
53 *
54 *   *****
55 *
56 *   VERIFICATION:
57 *
58 *   ! date ! auteur ! but
59 *   ! ! !
60 *
61 *   *****
62 *   MODIFICATIONS:

```

```

63 *      ! date !      auteur !      but      !      *
64 *      -----      *
65 *      *      $      *
66 *****/
67 #ifndef TENSEURQ3_TROIS_SYM_H
68 #define TENSEURQ3_TROIS_SYM_H
69
70 #include <iostream>
71 #include "TenseurQ.h"
72 #include "PtTabRel.h"
73 #include "Tableau2_T.h"
74 #include "Tableau4_T.h"
75 #include "Tenseur.h"
76
77 //-----
78 //      cas des composantes 4 fois contravariantes HHHH
79 //-----
80 class TenseurQ3_troisSym_HHHH : public TenseurHHHH
81 { // surcharge de l'operator de lecture
82     friend istream & operator >> (istream &, TenseurQ3_troisSym_HHHH &);
83     // surcharge de l'operator d'écriture
84     friend ostream & operator << (ostream &, const TenseurQ3_troisSym_HHHH &);
85
86 public :
87     // Constructeur
88     TenseurQ3_troisSym_HHHH() ; // par défaut
89     // initialisation de toutes les composantes a une meme valeur val
90     TenseurQ3_troisSym_HHHH(const double& val);
91     // initialisation à partir des 21 coefficients indépendants
92     // (1111) (2222) (3333) (1122) (1133) (2233)
93     // (1211) (1222) (1233) (1311) (1322) (1333) (2311) (2322) (2333)
94     // (1212) (1313) (2323) (1213) (1223) (1323)
95     TenseurQ3_troisSym_HHHH
96     ( const double& x1111,const double& x2222,const double& x3333,const double& x1122,const double&
97     x1133,const double& x2233
98     ,const double& x1211,const double& x1222,const double& x1233,const double& x1311,const double&
99     x1322,const double& x1333
100     ,const double& x2311,const double& x2322,const double& x2333
101     ,const double& x1212,const double& x1313,const double& x2323,const double& x1213,const double&
102     x1223,const double& x1323);
103
104 // DESTRUCTEUR :
105 ~TenseurQ3_troisSym_HHHH() ;
106 // constructeur a partir d'une instance non differenciee
107 // il n'y a pas de vérification des symétries seules les grandeurs suivantes sont utilisés:
108 // (1111) (2222) (3333) (1122) (1133) (2233)
109 // (1211) (1222) (1233) (1311) (1322) (1333) (2311) (2322) (2333)
110 // (1212) (1313) (2323) (1213) (1223) (1323)
111 TenseurQ3_troisSym_HHHH (const TenseurHHHH &);
112 // constructeur de copie
113 TenseurQ3_troisSym_HHHH (const TenseurQ3_troisSym_HHHH &);
114
115 // METHODES PUBLIQUES :
116 //2) virtuelles
117 // initialise toutes les composantes à val
118 void Init(double val) ;
119 // operations
120 TenseurHHHH & operator + ( const TenseurHHHH & ) const ;
121 void operator += ( const TenseurHHHH & );
122 TenseurHHHH & operator - ( ) const ; // oppose du tenseur
123 TenseurHHHH & operator - ( const TenseurHHHH & ) const ;
124 void operator -= ( const TenseurHHHH & );
125 TenseurHHHH & operator = ( const TenseurHHHH & );
126 TenseurHHHH & operator * (const double & ) const ;
127 void operator *= ( const double & );
128 TenseurHHHH & operator / ( const double & ) const ;
129 void operator /= ( const double & );
130
131 // produit deux fois contracte à droite avec un tenseur du second ordre
132 // diffèrent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
133 TenseurHH& operator && ( const TenseurBB & ) const ;
134 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
135 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
136 // symétrie
137 TenseurHHHH& operator && ( const TenseurBBHH & ) const
138 { cout << "\n non implante !! , TenseurHHHH& TenseurQ3_troisSym_HHHH::operator && ( const
139 TenseurBBHH & ) const ";
140     Sortie(1); TenseurHHHH* toto; return *toto; /* toto pour éviter un warning*/ };
141 TenseurHHBB& operator && ( const TenseurBBBB & ) const
142 { cout << "\n non implante !! , TenseurHHBB& TenseurQ3_troisSym_HHHH::operator && ( const
143 TenseurBBBB & ) const ";
144     Sortie(1); TenseurHHBB* toto; return *toto; /* toto pour éviter un warning*/ };
145
146 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
147 // les 2 premiers indices sont échangés avec les deux derniers indices
148 // ici en fait c'est le même tenseur grace aux symétries constitutives
149 TenseurHHHH & Transposelet2avec3et4() const ;

```

```

144
145 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
146 // plusZero = true: les données manquantes sont mises à 0
147 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
148 // des données possibles
149 void Affectation_trans_dimension(const TenseurHHHH & B,bool plusZero);
150
151
152 // test
153 int operator == ( const TenseurHHHH & const );
154
155 // change la composante i,j,k,l du tenseur
156 // acces en ecriture,
157 void Change (int i, int j, int k, int l,const double& val) ;
158 // en cumul : équivalent de +=
159 void ChangePlus (int i, int j, int k, int l,const double& val);
160
161 // Retourne la composante i,j,k,l du tenseur
162 // acces en lecture seule
163 double operator () (int i, int j, int k, int l) const ;
164
165 // calcul du maximum en valeur absolu des composantes du tenseur
166 double MaxiComposante() const;
167
168 // lecture et écriture de données
169 istream & Lecture(istream & entree);
170 ostream & Ecriture(ostream & sort) const ;
171
172 protected :
173 // allocator dans la liste de data
174 listdouble2lIter ipointe;
175 // --- gestion d'index ----
176 class ChangementIndex
177 { public:
178     ChangementIndex();
179     // passage pour les index de la forme i,j,k,l à la forme vecteur
180     Tableau4 <int> odVect;
181 };
182 static const ChangementIndex cdex;
183
184 // fonction pour le produit contracté à gauche
185 TenseurHH& Prod_gauche( const TenseurBB & F) const { return ((*this) && F); };
186 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
187 // symétrie
188 TenseurBBHH& Prod_gauche( const TenseurBBBB & ) const
189 { cout << "\n non implante !! , TenseurBBHH& TenseurQ3_troisSym_HHHH::Prod_gauche( const
190 TenseurBBBB & ) const ";
191     Sortie(1); TenseurBBHH* toto; return *toto; /* toto pour eviter un warning*/ };
192 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
193 // symétrie
194 TenseurHHHH& Prod_gauche( const TenseurHHBB & ) const
195 { cout << "\n non implante !! , TenseurHHHH& TenseurQ3_troisSym_HHHH::Prod_gauche( const TenseurHHBB
196 & ) const ";
197     Sortie(1); TenseurHHHH* toto; return *toto; /* toto pour eviter un warning*/ };
198 };
199 //
200 //-----
201 //          cas des composantes 4 fois covariantes BBBB
202 //-----
203 class TenseurQ3_troisSym_BBBB : public TenseurBBBB
204 { // surcharge de l'operator de lecture
205     friend istream & operator » (istream &, TenseurQ3_troisSym_BBBB &);
206     // surcharge de l'operator d'ecriture
207     friend ostream & operator « (ostream &, const TenseurQ3_troisSym_BBBB &);
208 };
209 public :
210     // Constructeur
211     TenseurQ3_troisSym_BBBB() ; // par défaut
212     // initialisation de toutes les composantes a une meme valeur val
213     TenseurQ3_troisSym_BBBB(const double& val);
214     // initialisation à partir des 21 coefficients indépendants
215     // (1111) (2222) (3333) (1122) (1133) (2233)
216     // (1211) (1222) (1233) (1311) (1322) (1333) (2311) (2322) (2333)
217     // (1212) (1313) (2323) (1213) (1223) (1323)
218     TenseurQ3_troisSym_BBBB
219     ( const double& x1111,const double& x2222,const double& x3333,const double& x1122,const double&
220 x1133,const double& x2233
221     ,const double& x1211,const double& x1222,const double& x1233,const double& x1311,const double&
222 x1322,const double& x1333
223     ,const double& x2311,const double& x2322,const double& x2333
224     ,const double& x1212,const double& x1313,const double& x2323,const double& x1213,const double&
225 x1223,const double& x1323);
226
227 // DESTRUCTEUR :
228 ~TenseurQ3_troisSym_BBBB() ;
229 // constructeur a partir d'une instance non differenciee
230 // il n'y a pas de vérification des symétries seules les grandeurs

```

```

224 // (1111) (2222) (3333) (1122) (1133) (2233)
225 // (1211) (1222) (1233) (1311) (1322) (1333) (2311) (2322) (2333)
226 // (1212) (1313) (2323) (1213) (1223) (1323)
227 TenseurQ3_troisSym_BBBB (const TenseurBBBB &);
228 // constructeur de copie
229 TenseurQ3_troisSym_BBBB (const TenseurQ3_troisSym_BBBB &);
230
231 // METHODES PUBLIQUES :
232 //(2) virtuelles
233 // initialise toutes les composantes à val
234 void Initia(double val) ;
235 // operations
236 TenseurBBBB & operator + ( const TenseurBBBB & ) const ;
237 void operator += ( const TenseurBBBB & );
238 TenseurBBBB & operator - ( ) const ; // oppose du tenseur
239 TenseurBBBB & operator - ( const TenseurBBBB & ) const ;
240 void operator -= ( const TenseurBBBB & );
241 TenseurBBBB & operator = ( const TenseurBBBB & );
242 TenseurBBBB & operator * (const double & ) const ;
243 void operator *= ( const double & );
244 TenseurBBBB & operator / ( const double & ) const ;
245 void operator /= ( const double & );
246
247 // produit deux fois contracte à droite avec un tenseur du second ordre
248 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
249 TenseurBB& operator && ( const TenseurHH & ) const ;
250 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
251 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
252 TenseurBBHH& operator && ( const TenseurHHHH & ) const
253 { cout << "\n non implante !! , TenseurBBHH& TenseurQ3_troisSym_BBBB::operator && ( const
TenseurHHHH & ) const ";
254 Sortie(1); TenseurBBHH* toto; return *toto; /* toto pour éviter un warning*/ };
255 TenseurBBBB& operator && ( const TenseurHHBB & ) const
256 { cout << "\n non implante !! , TenseurBBBB& TenseurQ3_troisSym_BBBB::operator && ( const
TenseurHHBB & ) const ";
257 Sortie(1); TenseurBBBB* toto; return *toto; /* toto pour éviter un warning*/ };
258
259 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
260 // les 2 premiers indices sont échangés avec les deux derniers indices
261 // ici en fait c'est le même tenseur grace aux symétries constitutives
262 TenseurBBBB & Transposelet2avec3et4() const ;
263
264 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
265 // plusZero = true: les données manquantes sont mises à 0
266 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
267 // des données possibles
268 void Affectation_trans_dimension(const TenseurBBBB & B,bool plusZero);
269
270
271 // test
272 int operator == ( const TenseurBBBB & ) const ;
273
274 // change la composante i,j,k,l du tenseur
275 // acces en ecriture,
276 void Change (int i, int j, int k, int l,const double& val) ;
277 // en cumul : équivalent de +=
278 void ChangePlus (int i, int j, int k, int l,const double& val);
279
280 // Retourne la composante i,j,k,l du tenseur
281 // acces en lecture seule
282 double operator () (int i, int j, int k, int l) const ;
283
284 // calcul du maximum en valeur absolu des composantes du tenseur
285 double MaxiComposante() const;
286
287 // lecture et écriture de données
288 istream & Lecture(istream & entree);
289 ostream & Ecriture(ostream & sort) const ;
290
291 protected :
292 // allocator dans la liste de data
293 listdouble2lIter ipointe;
294 // --- gestion d'index ----
295 class ChangementIndex
296 { public:
297 ChangementIndex();
298 // passage pour les index de la forme i,j,k,l à la forme vecteur
299 Tableau4 <int> odVect;
300 };
301 static const ChangementIndex cdex;
302
303 // fonction pour le produit contracté à gauche
304 TenseurBB& Prod_gauche( const TenseurHH & F) const { return ((*this) && F); };
305 // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
306 TenseurHHBB& Prod_gauche( const TenseurHHHH & ) const

```

```

307     { cout << "\n non implante !! , TenseurHHBB& TenseurQ3_troisSym_BBBB::Prod_gauche( const
TenseurHHHH & ) const ";
308     Sortie(1); TenseurHHBB* toto; return *toto; /* toto pour eviter un warning*/ };
309     // a priori pas intéressant pour cette classe de tenseur car cela donne un tenseur qui n'a plus de
symétrie
310     TenseurBBBB& Prod_gauche( const TenseurBBHH & ) const
311     { cout << "\n non implante !! , TenseurBBBB& TenseurQ3_troisSym_BBBB::Prod_gauche( const TenseurBBHH
& ) const ";
312     Sortie(1); TenseurBBBB* toto; return *toto; /* toto pour eviter un warning*/ };
313 };
314
315
316 #ifndef MISE_AU_POINT
317 #include "Tenseur3_TroisSym.cc"
318 #define TenseurQ3_TroisSym_H_deja_inclus
319 #endif
320
321
322 #endif

```

## 7.469 TenseurO4.h

```

1 /*****
2 * UNIVERSITE DE BRETAGNE SUD (UBS) --- I.U.P/I.U.T. DE LORIENT *
3 *****/
4 * LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M) *
5 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
6 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
7 *****/
8 * DATE: 23/03/2002 *
9 * $ *
10 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
11 * Tel 0297874571 fax : 02.97.87.45.72 *
12 * $ *
13 * PROJET: Herezh++ *
14 * $ *
15 *****/
16 * BUT: Définir les tenseurs d'ordre 4 de différentes composantes.*
17 * La classe est virtuelle pure. *
18 * Elle se déclare en fonction de la dimension du problème *
19 * L'objectif principal est de surcharger les différentes *
20 * opérations classiques. *
21 * $ *
22 * ***** *
23 * VERIFICATION: *
24 * ! date ! auteur ! but ! *
25 * ----- *
26 * ! ! ! ! *
27 * $ *
28 * ***** *
29 * MODIFICATIONS: *
30 * ! date ! auteur ! but ! *
31 * ----- *
32 * $ *
33 * *****/
34 *****/
35 #ifndef TenseurO4_H
36 #define TenseurO4_H
37
38 #include "Tenseur.h"
39
40
41
42
43 //-----
44 // cas des composantes quatre fois contravariantes
45 //-----
46 class TenseurBBBB; // def anticipée
47 class TenseurHHBB; // pour l'utilisation dans les produits contractés
48 class TenseurBBHH; //
49
50
51 class TenseurHHHH
52 { friend TenseurHHHH & operator * (double r, const TenseurHHHH & t)
53 { return (t*r); };
54 // produit contracté à gauche avec un tenseur d'ordre 2
55 // différent à droite
56 friend TenseurHH& operator && ( const TenseurBB & F , const TenseurHHHH & T);
57
58 // surcharge de l'opérateur de lecture
59 friend istream & operator >> (istream &, TenseurHHHH &);
60 // surcharge de l'opérateur d'écriture
61 friend ostream & operator << (ostream &, const TenseurHHHH &);
62

```

```

63 public :
64     // Constructeur
65     TenseurHHHH() ; // par défaut
66     // construction d'un tenseur par produit tensoriel de tenseur
67     // si le pointeur Z1==NULL => on ne tiend pas compte de Z1 et Z2
68     // sinon on fait le produit tensoriel de Z1 et Z2
69     // si le pointeur W1== NULL => on ne tiend pas compte de W1 et W2
70     // sinon on ajoute le produit tensoriel barre de W1 et W2
71     // le résultat est la somme des Zi et Wi éventuelles
72     // en composantes: Z1(i,j).Z2(k,l) + W1(i,k).W2(j,l)
73     TenseurHHHH(const TenseurHH * Z1,const TenseurHH * Z2,const TenseurHH * W1,const TenseurHH * W2)
74     ;
75     // constructeur de copie
76     TenseurHHHH (const TenseurHHHH &);
77     // DESTRUCTEUR :
78     ~TenseurHHHH() ;
79     // libération du la place mémoire
80     // cas = 0 : on libère tout
81     // cas = 1 : on libère les Zi
82     // cas = 2 : on libère les Wi
83     void Libere(int cas);
84
85     // METHODES PUBLIQUES :
86
87     int Dimension() const; // retourne la dimension du tenseur
88     // operations
89     TenseurHHHH operator + ( const TenseurHHHH &) const ;
90     void operator += ( const TenseurHHHH &);
91     TenseurHHHH operator - ( const TenseurHHHH &) const ; // oppose du tenseur
92     void operator -= ( const TenseurHHHH &);
93     TenseurHHHH& operator = ( const TenseurHHHH &);
94     TenseurHHHH operator * (const double &) const ;
95     void operator *= ( const double &);
96     TenseurHHHH operator / ( const double &) const ;
97     void operator /= ( const double &);
98
99     // produit contracte à droite avec un tenseur du second ordre
100    // différent à gauche !!
101    TenseurHH& operator && ( const TenseurBB &) const ;
102    //fonctions static définissant le produit tensoriel normal de deux tenseurs
103    // aHH(i,j).bHH(k,l) gB1 gBj gBk gBl
104    static TenseurHHHH Prod_tensoriel(const TenseurHH & aHH, const TenseurHH & bHH);
105    //fonctions static définissant le produit tensoriel barre de deux tenseurs
106    // aHH(i,k).bHH(j,l) gB1 gBj gBk gBl
107    static TenseurHHHH Prod_tensoriel_barre(const TenseurHH & aHH, const TenseurHH & bHH);
108
109    // test
110    bool operator == ( const TenseurHHHH &) const ;
111    bool operator != ( const TenseurHHHH &) const;
112
113    // conversion de type
114    operator TenseurHHHH & (void)
115    {cout << " appel de la conversion de type\n";
116     return *this;};
117
118    // Retourne la composante i,j,k,l du tenseur
119    // acces en ecriture,
120    // possible uniquement si c'est un seul type de produit tensoriel
121    // si c'est un mixte alors il y a erreur
122    void change (int i, int j, int k, int l, double& val);
123
124    // Retourne la composante i,j,k,l du tenseur
125    // acces en lecture seule
126    double operator () (int i, int j, int k, int l) const ;
127
128    // acces en lecture aux tenseurs de base
129    const TenseurHH & Z_1() const {return *Z1;};
130    const TenseurHH & Z_2() const {return *Z2;};
131    const TenseurHH & W_1() const {return *W1;};
132    const TenseurHH & W_2() const {return *W2;};
133    const TenseurHH * Z_1P() const {return Z1;};
134    const TenseurHH * Z_2P() const {return Z2;};
135    const TenseurHH * W_1P() const {return W1;};
136    const TenseurHH * W_2P() const {return W2;};
137
138    // variables
139
140 protected :
141
142     TenseurHH * Z1,* Z2,* W1,* W2; // les quatres tenseurs de construction
143     // &Z1 tensoriel &Z2 + &W1 {tensoriel}_barre &W2
144     // Z1 != NULL : le tenseur contient &Z1 tensoriel &Z2, sinon Z1=Z2= NULL
145     // W1 != NULL : le tenseur contient &W1 {tensoriel}_barre &W2, sinon W1=W2=NULL
146
147     // sortie d'un message standard
148     // dim = dimension du tenseur argument

```

```

149     void Message(int dim, string mes) const ;
150
151 };
152
153 //-----
154 //             cas des composantes quatre fois covariantes
155 //-----
156
157 class TenseurBBBB
158 { friend TenseurBBBB & operator * (double r, const TenseurBBBB & t)
159   { return ( t*r); } ;
160 // produit contracte à gauche avec un tenseur d'ordre 2
161 // différent à droite
162   friend TenseurBB& operator && ( const TenseurHH & F , const TenseurBBBB & T);
163
164 // surcharge de l'operator de lecture
165   friend istream & operator » (istream &, TenseurBBBB &);
166 // surcharge de l'operator d'écriture
167   friend ostream & operator « (ostream &, const TenseurBBBB &);
168
169 public :
170     // Constructeur
171     TenseurBBBB() ; // par défaut
172     // construction d'un tenseur par produit tensoriel de tenseur
173     // si le pointeur Zi==NULL => on ne tiend pas compte de Zi1 et Zi2
174     // sinon on fait le produit tensoriel de Zi1 et Zi2
175     // si le pointeur Wi1== NULL => on ne tiend pas compte de Wi1 et Wi2
176     // sinon on ajoute le produit tensoriel barre de Wi1 et Wi2
177     // le résultat est la somme des Zi et Wi éventuelles
178     // en composantes: Zi(i,j).Z2(k,l) + W1(i,k).W2(j,l)
179     TenseurBBBB(const TenseurBB * Zi1,const TenseurBB * Zi2,const TenseurBB * Wi1,const TenseurBB * Wi2)
180 ;
181 // constructeur de copie
182 TenseurBBBB (const TenseurBBBB &);
183 // DESTRUCTEUR :
184 ~TenseurBBBB() ;
185 // libération du la place mémoire
186 // cas = 0 : on libère tout
187 // cas = 1 : on libère les Zi
188 // cas = 2 : on libère les Wi
189 void Libere(int cas);
190
191 // METHODES PUBLIQUES :
192
193 int Dimension() const; // retourne la dimension du tenseur
194 // operations
195 TenseurBBBB operator + ( const TenseurBBBB &) const ;
196 void operator += ( const TenseurBBBB &);
197 TenseurBBBB operator - () const ; // oppose du tenseur
198 TenseurBBBB operator - ( const TenseurBBBB &) const ;
199 void operator -= ( const TenseurBBBB &);
200 TenseurBBBB& operator = ( const TenseurBBBB &);
201 TenseurBBBB operator * (const double &) const ;
202 void operator *= ( const double &);
203 TenseurBBBB operator / ( const double &) const ;
204 void operator /= ( const double &);
205
206 // produit contracte à droite avec un tenseur du second ordre
207 // différent à gauche !!
208 TenseurBB& operator && ( const TenseurHH &) const ;
209 //fonctions static définissant le produit tensoriel normal de deux tenseurs
210 // aBB(i,j).bBB(k,l) gHi gHj gHk gHl
211 static TenseurBBBB Prod_tensoriel(const TenseurBB & aBB, const TenseurBB & bBB);
212 //fonctions static définissant le produit tensoriel barre de deux tenseurs
213 // aBB(i,k).bBB(j,l) gHi gHj gHk gHl
214 static TenseurBBBB Prod_tensoriel_barre(const TenseurBB & aBB, const TenseurBB & bBB);
215
216 // test
217 bool operator == ( const TenseurBBBB &) const ;
218 bool operator != ( const TenseurBBBB &) const;
219
220 // conversion de type
221 operator TenseurBBBB & (void)
222 {cout << " appel de la conversion de type\n";
223   return *this;};
224
225 // Retourne la composante i,j,k,l du tenseur
226 // acces en écriture,
227 // possible uniquement si c'est un seul type de produit tensoriel
228 // si c'est un mixte alors il y a erreur
229 void change (int i, int j, int k, int l, double& val);
230
231 // Retourne la composante i,j,k,l du tenseur
232 // acces en lecture seule
233 double operator () (int i, int j, int k, int l) const ;
234
235 // acces en lecture aux tenseurs de base

```



```

235     const TenseurBB & Z_1() const {return *Z1;};
236     const TenseurBB & Z_2() const {return *Z2;};
237     const TenseurBB & W_1() const {return *W1;};
238     const TenseurBB & W_2() const {return *W2;};
239     const TenseurBB * Z_1P() const {return Z1;};
240     const TenseurBB * Z_2P() const {return Z2;};
241     const TenseurBB * W_1P() const {return W1;};
242     const TenseurBB * W_2P() const {return W2;};
243
244     // variables
245
246     protected :
247
248     TenseurBB * Z1,* Z2,* W1,* W2; // les quatres tenseurs de construction
249     // &Z1 tensoriel &Z2 + &W1 {tensoriel}_{barre} &W2
250     // Z1 != NULL : le tenseur contient &Z1 tensoriel &Z2, sinon Z1=Z2= NULL
251     // W1 != NULL : le tenseur contient &W1 {tensoriel}_{barre} &W2, sinon W1=W2=NULL
252
253     // sortie d'un message standard
254     // dim = dimension du tenseur argument
255     void Message(int dim, string mes) const ;
256
257 };
258
259 //-----
260 //          cas des composantes mixtes BBHH
261 //-----
262
263 class TenseurBBHH
264 { friend TenseurBBHH & operator * (double r, const TenseurBBHH & t)
265   { return ( t*r); };
266 // produit contracte à gauche avec un tenseur d'ordre 2
267 // différent à droite
268 friend TenseurHH& operator && ( const TenseurHH & F , const TenseurBBHH & T);
269
270 // surcharge de l'operator de lecture
271 friend istream & operator » (istream &, TenseurBBHH &);
272 // surcharge de l'operator d'écriture
273 friend ostream & operator « (ostream &, const TenseurBBHH &);
274
275 public :
276 // Constructeur
277 TenseurBBHH() ; // par défaut
278 // construction d'un tenseur par produit tensoriel de tenseur
279 // si le pointeur Zil==NULL => on ne tiend pas compte de Zil et Zi2
280 // sinon on fait le produit tensoriel de Zil et Zi2
281 // si le pointeur Wil== NULL => on ne tiend pas compte de Wil et Wi2
282 // sinon on ajoute le produit tensoriel barre de Wil et Wi2
283 // le résultat est la somme des Zi et Wi éventuelles
284 // en composantes: Z1(i,j).Z2(k,l) + W1(i,k).W2(j,l)
285 TenseurBBHH(const TenseurBB * Zil,const TenseurHH * Zi2,const TenseurBH * Wil,const TenseurBH * Wi2)
286 ;
287 // constructeur de copie
288 TenseurBBHH (const TenseurBBHH &);
289 // DESTRUCTEUR :
290 ~TenseurBBHH() ;
291 // libération du la place mémoire
292 // cas = 0 : on libère tout
293 // cas = 1 : on libère les Zi
294 // cas = 2 : on libère les Wi
295 void Libere(int cas);
296
297 // METHODES PUBLIQUES :
298
299 int Dimension() const; // retourne la dimension du tenseur
300 // operations
301 TenseurBBHH operator + ( const TenseurBBHH &) const ;
302 void operator += ( const TenseurBBHH &);
303 TenseurBBHH operator - () const ; // oppose du tenseur
304 TenseurBBHH operator - ( const TenseurBBHH &) const ;
305 void operator -= ( const TenseurBBHH &);
306 TenseurBBHH& operator = ( const TenseurBBHH &);
307 TenseurBBHH operator * (const double &) const ;
308 void operator *= ( const double &);
309 TenseurBBHH operator / ( const double &) const ;
310 void operator /= ( const double &);
311
312 // produit contracte à droite avec un tenseur du second ordre
313 // différent à gauche !!
314 TenseurBB& operator && ( const TenseurBB &) const ;
315 //fonctions static définissant le produit tensoriel normal de deux tenseurs
316 // aBB(i,j).bHH(k,l) gHi gHj gBk gBl
317 static TenseurBBHH Prod_tensoriel(const TenseurBB & aBB, const TenseurHH & bHH);
318 //fonctions static définissant le produit tensoriel barre de deux tenseurs
319 // aBH(i,k).bBH(j,l) gHi gHj gBk gBl
320 static TenseurBBHH Prod_tensoriel_barre(const TenseurBH & aBH, const TenseurBH & bBH);

```

```

321 // test
322 bool operator == ( const TenseurBBHH & const ;
323 bool operator != ( const TenseurBBHH & const ;
324
325 // conversion de type
326 operator TenseurBBHH & (void)
327 {cout << " appel de la conversion de type\n";
328 return *this;};
329
330 // Retourne la composante i,j,k,l du tenseur
331 // acces en ecriture,
332 // possible uniquement si c'est un seul type de produit tensoriel
333 // si c'est un mixte alors il y a erreur
334 void change (int i, int j, int k, int l, double& val);
335
336 // Retourne la composante i,j,k,l du tenseur
337 // acces en lecture seule
338 double operator () (int i, int j, int k, int l) const ;
339
340 // acces en lecture aux tenseurs de base
341 const TenseurBB & Z_1() const {return *Z1;};
342 const TenseurHH & Z_2() const {return *Z2;};
343 const TenseurBH & W_1() const {return *W1;};
344 const TenseurBH & W_2() const {return *W2;};
345 const TenseurBB * Z_1P() const {return Z1;};
346 const TenseurHH * Z_2P() const {return Z2;};
347 const TenseurBH * W_1P() const {return W1;};
348 const TenseurBH * W_2P() const {return W2;};
349
350 // variables
351
352 protected :
353
354 TenseurBB * Z1; // les tenseurs de construction
355 TenseurHH * Z2; // "
356 TenseurBH * W1,* W2; // "
357
358 // &Z1 tensoriel &Z2 + &W1 {tensoriel}_{barre} &W2
359 // Z1 != NULL : le tenseur contient &Z1 tensoriel &Z2, sinon Z1=Z2= NULL
360 // W1 != NULL : le tenseur contient &W1 {tensoriel}_{barre} &W2, sinon W1=W2=NULL
361
362 // sortie d'un message standard
363 // dim = dimension du tenseur argument
364 void Message(int dim, string mes) const ;
365
366 };
367
368 //-----
369 // cas des composantes mixtes HHBB
370 //-----
371 class TenseurHHBB
372 { friend TenseurHHBB & operator * (double r, const TenseurHHBB & t)
373 { return ( t*r); };
374 // produit contracte à gauche avec un tenseur d'ordre 2
375 // différent à droite
376 friend TenseurBB& operator && ( const TenseurBB & F , const TenseurHHBB & T);
377
378 // surcharge de l'operator de lecture
379 friend istream & operator » (istream &, TenseurHHBB &);
380 // surcharge de l'operator d'ecriture
381 friend ostream & operator « (ostream &, const TenseurHHBB &);
382
383 public :
384 // Constructeur
385 TenseurHHBB() ; // par défaut
386 // construction d'un tenseur par produit tensoriel de tenseur
387 // si le pointeur Zil==NULL => on ne tiend pas compte de Zil et Zi2
388 // sinon on fait le produit tensoriel de Zil et Zi2
389 // si le pointeur Wil== NULL => on ne tiend pas compte de Wil et Wi2
390 // sinon on ajoute le produit tensoriel barre de Wil et Wi2
391 // le résultat est la somme des Zi et Wi éventuelles
392 // en composantes: Z1(i,j).Z2(k,l) + W1(i,k).W2(j,l)
393 TenseurHHBB(const TenseurHH * Zil,const TenseurBB * Zi2,const TenseurHB * Wil,const TenseurHB * Wi2)
;
394 // constructeur de copie
395 TenseurHHBB (const TenseurHHBB &);
396 // DESTRUCTEUR :
397 ~TenseurHHBB() ;
398 // libération du la place mémoire
399 // cas = 0 : on libère tout
400 // cas = 1 : on libère les Zi
401 // cas = 2 : on libère les Wi
402 void Libere(int cas);
403
404 // METHODES PUBLIQUES :
405
406 int Dimension() const; // retourne la dimension du tenseur

```

```

407 // operations
408 TenseurHHBB operator + ( const TenseurHHBB & ) const ;
409 void operator += ( const TenseurHHBB & );
410 TenseurHHBB operator - ( ) const ; // oppose du tenseur
411 TenseurHHBB operator - ( const TenseurHHBB & ) const ;
412 void operator -= ( const TenseurHHBB & );
413 TenseurHHBB& operator = ( const TenseurHHBB & );
414 TenseurHHBB operator * ( const double & ) const ;
415 void operator *= ( const double & );
416 TenseurHHBB operator / ( const double & ) const ;
417 void operator /= ( const double & );
418
419 // produit contracte à droite avec un tenseur du second ordre
420 // différent à gauche !!
421 TenseurHH& operator && ( const TenseurHH & ) const ;
422 //fonctions static définissant le produit tensoriel normal de deux tenseurs
423 // aHH(i,j).bBB(k,l) gBi gBj gHk gHl
424 static TenseurHHBB Prod_tensoriel(const TenseurHH & aHH, const TenseurBB & bBB);
425 //fonctions static définissant le produit tensoriel barre de deux tenseurs
426 // aHB(i,k).bHB(j,l) gBi gBj gHk gHl
427 static TenseurHHBB Prod_tensoriel_barre(const TenseurHB & aHB, const TenseurHB & bHB);
428
429 // test
430 bool operator == ( const TenseurHHBB & ) const ;
431 bool operator != ( const TenseurHHBB & ) const;
432
433 // conversion de type
434 operator TenseurHHBB & (void)
435 {cout << " appel de la conversion de type\n";
436  return *this;};
437
438 // Retourne la composante i,j,k,l du tenseur
439 // acces en ecriture,
440 // possible uniquement si c'est un seul type de produit tensoriel
441 // si c'est un mixte alors il y a erreur
442 void change (int i, int j, int k, int l, double& val);
443
444 // Retourne la composante i,j,k,l du tenseur
445 // acces en lecture seule
446 double operator () (int i, int j, int k, int l) const ;
447
448 // acces en lecture aux tenseurs de base
449 const TenseurHH & Z_1() const {return *Z1;};
450 const TenseurBB & Z_2() const {return *Z2;};
451 const TenseurHB & W_1() const {return *W1;};
452 const TenseurHB & W_2() const {return *W2;};
453 const TenseurHH * Z_1P() const {return Z1;};
454 const TenseurBB * Z_2P() const {return Z2;};
455 const TenseurHB * W_1P() const {return W1;};
456 const TenseurHB * W_2P() const {return W2;};
457
458 // variables
459
460 protected :
461
462 TenseurHH * Z1; // les tenseurs de construction
463 TenseurBB * Z2; // "
464 TenseurHB * W1,* W2; // "
465 // &Z1 tensoriel &Z2 + &W1 {tensoriel}_{barre} &W2
466 // Z1 != NULL : le tenseur contient &Z1 tensoriel &Z2, sinon Z1=Z2= NULL
467 // W1 != NULL : le tenseur contient &W1 {tensoriel}_{barre} &W2, sinon W1=W2=NULL
468
469 // sortie d'un message standard
470 // dim = dimension du tenseur argument
471 void Message(int dim, string mes) const ;
472
473 };
474
475
476 #ifndef MISE_AU_POINT
477 #include "TenseurO4_1.cp"
478 #include "TenseurO4_2.cp"
479 #define TenseurO4_H_deja_inclus
480 #endif
481
482 #endif

```

## 7.470 TenseurQ-1.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.

```

```

7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 * UNIVERSITE DE BRETAGNE SUD (UBS) --- ENSIBS DE LORIENT *
34 *****/
35 * LIMATB- Equipe DE GENIE MECANIQUE ET MATERIAUX (EG2M) *
36 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
37 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
38 *****/
39 * DATE: 14/06/2015 *
40 * $ *
41 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
42 * phone 0297874573, fax : 0297874588 *
43 * $ *
44 * PROJET: Herezh++ *
45 * $ *
46 *****/
47 * BUT: Definition des classes derivees tenseur du 4ieme ordre *
48 * de dimensionl. Ici de nombreuses fonctions ne sont pas *
49 * pas disponible du à la forme particulière de stockage *
50 * par contre c'est économique en stockage. *
51 * Stockage: (ijkl) = avec l valeurs *
52 * en fait tenseur symétrique sur ij et sur kl tel que: *
53 *  $A(i,j,k,l) = A(j,i,k,l) = A(i,j,l,k) = A(j,i,l,k)$  *
54 * $ *
55 * ***** *
56 * VERIFICATION: *
57 * *
58 * ! date ! auteur ! but ! *
59 * ----- *
60 * ! ! ! ! *
61 * $ *
62 * ***** *
63 * MODIFICATIONS: *
64 * ! date ! auteur ! but ! *
65 * ----- *
66 * $ *
67 *****/
68 #ifndef TENSEURQ1_H
69 #define TENSEURQ1_H
70
71 #include <iostream>
72 #include "TenseurQ.h"
73 #include "PtTabRel.h"
74 #include "Tableau2_T.h"
75
76 //-----
77 // cas des composantes 4 fois contravariantes 1HHHH
78 //-----
79 class TenseurBBHH;
80 class TenseurHHBB;
81
82
83 class Tenseur1HHHH : public TenseurHHHH
84 { // surcharge de l'operator de lecture
85 friend istream & operator » (istream &, Tenseur1HHHH &);
86 // surcharge de l'operator d'écriture
87 friend ostream & operator « (ostream &, const Tenseur1HHHH &);
88
89 public :
90 // Constructeur
91 Tenseur1HHHH() ; // par défaut
92 // initialisation de toutes les composantes a une meme valeur val

```

```

93     Tenseur1HHHH(const double val);
94     // initialisation à partir d'un produit tensoriel avec 2 cas
95     // booleen = true : produit tensoriel normal
96     //          *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
97     // booleen = false : produit tensoriel barre, qui conserve les symétries
98     // *this(i,j,k,l)
99     // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
100    Tenseur1HHHH(bool normal, const TenseurHH & aHH, const TenseurHH & bHH);
101
102    // DESTRUCTEUR :
103    ~Tenseur1HHHH();
104    // constructeur a partir d'une instance non differentiee
105    // dans le cas d'un tenseur quelconque, celui-ci
106    // est converti à condition que les symétries existent sinon erreur en debug
107    // opération longue dans ce cas !
108    Tenseur1HHHH(const TenseurHHHH &);
109    // constructeur de copie
110    Tenseur1HHHH(const Tenseur1HHHH &);
111
112    // METHODES PUBLIQUES :
113    //(2) virtuelles
114    // initialise toutes les composantes à val
115    void Inita(double val);
116    // operations
117    TenseurHHHH & operator + (const TenseurHHHH &) const;
118    void operator += (const TenseurHHHH &);
119    TenseurHHHH & operator - () const; // oppose du tenseur
120    TenseurHHHH & operator - (const TenseurHHHH &) const;
121    void operator -= (const TenseurHHHH &);
122    TenseurHHHH & operator = (const TenseurHHHH &);
123    TenseurHHHH & operator = (const Tenseur1HHHH & B);
124    { return this->operator=((TenseurHHHH &) B); };
125    TenseurHHHH & operator * (const double &) const;
126    void operator *= (const double &);
127    TenseurHHHH & operator / (const double &) const;
128    void operator /= (const double &);
129
130    // produit contracte à droite avec un tenseur du second ordre
131    // différent à gauche !!
132    TenseurHH& operator && (const TenseurBB &) const;
133    //fonctions définissant le produit tensoriel normal de deux tenseurs
134    // *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
135    static TenseurHHHH & Prod_tensoriel(const TenseurHH & aHH, const TenseurHH & bHH);
136    //fonctions définissant le produit tensoriel barre de deux tenseurs
137    // concernant les symétries !!
138    // *this(i,j,k,l)
139    // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
140    static TenseurHHHH & Prod_tensoriel_barre(const TenseurHH & aHH, const TenseurHH & bHH);
141
142    // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
143    // les 2 premiers indices sont échangés avec les deux derniers indices
144    TenseurHHHH & Transposelet2avec3et4() const;
145
146    // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
147    // plusZero = true: les données manquantes sont mises à 0
148    // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
149    // des données possibles
150    void Affectation_trans_dimension(const TenseurHHHH & B,bool plusZero);
151
152    // transférer un tenseur général accessible en indice, dans un tenseur 9 Tenseur1HHHH
153    // il n'y a pas de symétrisation !, seule certaines composantes sont prises en compte
154    void TransfertDunTenseurGeneral(const TenseurHHHH & aHHHH);
155
156    // ---- manipulation d'indice ----
157    // on baisse les deux premiers indices -> création d'un tenseur TenseurBBHH
158    TenseurBBHH& Baisse2premiersIndices();
159    // on baisse les deux derniers indices -> création d'un tenseur TenseurHHBB
160    TenseurHHBB& Baisse2derniersIndices();
161
162    // calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
163    // en argument : A -> une reference sur le tenseur resultat qui peut avoir une dimension
164    // différente du tenseur courant suivant que la dimension absolue et la dimension locale
165    // sont égales ou différentes, retour d'une reference sur A
166    TenseurHHHH & Baselocale(TenseurHHHH & A,const BaseH & gi) const;
167    // changement des composantes du tenseur, retour donc dans la même variance
168    // en argument : A -> une reference sur le tenseur resultat
169    // retour d'une reference sur A
170    // A = A^{ijkl} g_i rond g_j rond g_k rond g_l = A^{efgh} gp_i rond gpp_j rond g_k rond gp_l
171    // g_i = beta_i^j gp_j --> A^{efgh} = A^{ijkl} beta_i^e beta_j^f beta_k^g beta_l^h
172    TenseurHHHH & ChangeBase(TenseurHHHH & A,const BaseB & gi) const;
173
174    // test
175    int operator == (const TenseurHHHH &) const;
176
177    // change la composante i,j,k,l du tenseur
178    // acces en ecriture,
179    void Change(int i, int j, int k, int l,const double& val);

```

```

180 // en cumul : équivalent de +=
181 void ChangePlus (int i, int j, int k, int l, const double& val);
182
183 // Retourne la composante i,j,k,l du tenseur
184 // acces en lecture seule
185 double operator () (int i, int j, int k, int l) const ;
186
187 // calcul du maximum en valeur absolu des composantes du tenseur
188 double MaxiComposante() const;
189
190 // lecture et écriture de données
191 istream & Lecture(istream & entree);
192 ostream & Ecriture(ostream & sort) const ;
193
194 protected :
195 // allocator dans la liste de data
196 listdoubleIter ipointe;
197 // --- gestion de changement d'index ----
198 class ChangementIndex
199 { public:
200   ChangementIndex();
201   // passage pour les index de la forme vecteur à la forme i,j
202   Tableau <int> idx_i,idx_j;
203   // passage pour les index de la forme i,j à la forme vecteur
204   Tableau2 <int> odVect;
205 };
206 public :
207   static const ChangementIndex cdex1HHHH;
208
209   // fonction pour le produit contracté à gauche
210   TenseurHH& Prod_gauche( const TenseurBB & F) const;
211
212 };
213 //
214 //-----
215 //          cas des composantes 4 fois covariantes
216 //-----
217 class TenseurlBBBB : public TenseurBBBB
218 { // surcharge de l'operator de lecture
219   friend istream & operator » (istream &, TenseurlBBBB &);
220   // surcharge de l'operator d'écriture
221   friend ostream & operator « (ostream &, const TenseurlBBBB &);
222
223 public :
224   // Constructeur
225   TenseurlBBBB() ; // par défaut
226   // initialisation de toutes les composantes a une meme valeur val
227   TenseurlBBBB(const double val);
228   // initialisation à partir d'un produit tensoriel avec 2 cas
229   // booleen = true : produit tensoriel normal
230   //          *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
231   // booleen = false : produit tensoriel barre
232   //          *this(i,j,k,l)
233   //          = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
234   TenseurlBBBB(bool normal, const TenseurBB & aBB, const TenseurBB & bBB);
235
236   // DESTRUCTEUR :
237   ~TenseurlBBBB() ;
238   // constructeur a partir d'une instance non differenciee
239   // dans le cas d'un tenseur quelconque, celui-ci
240   // est converti à condition que les symétries existent sinon erreur en debug
241   // opération longue dans ce cas !
242   TenseurlBBBB (const TenseurBBBB &);
243   // constructeur de copie
244   TenseurlBBBB (const TenseurlBBBB &);
245
246   // METHODES PUBLIQUES :
247 //2) virtuelles
248 // initialise toutes les composantes à val
249 void Inita(double val) ;
250 // operations
251 TenseurBBBB & operator + ( const TenseurBBBB &) const ;
252 void operator += ( const TenseurBBBB &);
253 TenseurBBBB & operator - () const ; // oppose du tenseur
254 TenseurBBBB & operator - ( const TenseurBBBB &) const ;
255 void operator -= ( const TenseurBBBB &);
256 TenseurBBBB & operator = ( const TenseurBBBB &);
257 TenseurBBBB & operator = ( const TenseurlHHHH & B)
258 { return this->operator=((TenseurBBBB &) B); };
259 TenseurBBBB & operator * (const double &) const ;
260 void operator *= ( const double &);
261 TenseurBBBB & operator / ( const double &) const ;
262 void operator /= ( const double &);
263
264 // produit contracte à droite avec un tenseur du second ordre
265 // différent à gauche !!
266 TenseurBB& operator && ( const TenseurHH &) const ;

```

```

267 //fonctions définissant le produit tensoriel normal de deux tenseurs
268 // *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
269 static TenseurBBBB & Prod_tensoriel(const TenseurBB & aBB, const TenseurBB & bBB) ;
270 //fonctions définissant le produit tensoriel barre de deux tenseurs
271 // concernant les symétries !!
272 // *this(i,j,k,l)
273 // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
274 static TenseurBBBB & Prod_tensoriel_barre(const TenseurBB & aBB, const TenseurBB & bBB) ;
275
276 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
277 // les 2 premiers indices sont échangés avec les deux derniers indices
278 TenseurBBBB & Transposelet2avec3et4() const ;
279
280 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
281 // plusZero = true: les données manquantes sont mises à 0
282 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
283 // des données possibles
284 void Affectation_trans_dimension(const TenseurBBBB & B,bool plusZero);
285
286 // transférer un tenseur général accessible en indice, dans un tenseur l TenseurlHHHH
287 // il n'y a pas de symétrisation !, seule certaines composantes sont prises en compte
288 void TransfertDunTenseurGeneral(const TenseurBBBB & aBBBB);
289
290 // ---- manipulation d'indice ----
291 // on monte les deux premiers indices -> création d'un tenseur TenseurHHBB
292 TenseurHHBB& Monte2premiersIndices();
293 // on monte les deux derniers indices -> création d'un tenseur TenseurBBHH
294 TenseurBBHH& Monte2derniersIndices();
295 // on monte les 4 indices -> création d'un tenseur TenseurHHHH
296 TenseurHHHH& Monte4Indices();
297
298 // calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
299 // en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
300 // différente du tenseur courant suivant que la dimension absolue et la dimension locale
301 // sont égales ou différentes , retour d'une reference sur A
302 TenseurBBBB & Baselocale(TenseurBBBB & A,const BaseB & gi) const ;
303 // changement des composantes du tenseur, retour donc dans la même variance
304 // en argument : A -> une reference sur le tenseur résultat qui a la même dimension
305 // retour d'une reference sur A
306 // A = A_{ijkl} g^i rond g^j rond g^k rond g_l = A'_{efgh} gp^i rond gpp^j rond g^k rond gp^l
307 // g^i = gamma^i_j gp^j --> A'_{efgh} = A_{ijkl} gamma^i_e gamma^j_f gamma^k_g gamma^l_h
308 TenseurBBBB & ChangeBase(TenseurBBBB & A,const BaseH & gi) const;
309
310
311 // test
312 int operator == ( const TenseurBBBB &) const ;
313
314 // Retourne la composante i,j,k,l du tenseur
315 // acces en ecriture,
316 void Change (int i, int j, int k, int l,const double& val) ;
317 // en cumul : équivalent de +=
318 void ChangePlus (int i, int j, int k, int l,const double& val);
319
320 // Retourne la composante i,j,k,l du tenseur
321 // acces en lecture seule
322 double operator () (int i, int j, int k, int l) const ;
323
324 // calcul du maximum en valeur absolu des composantes du tenseur
325 double MaxiComposante() const;
326
327 // lecture et écriture de données
328 istream & Lecture(istream & entree);
329 ostream & Ecriture(ostream & sort) const ;
330
331 protected :
332 // allocator dans la liste de data
333 listdoubleIter ipointe;
334 // --- gestion de changement d'index ----
335 class ChangementIndex
336 { public:
337     ChangementIndex();
338     // passage pour les index de la forme vecteur à la forme i,j
339     Tableau <int> idx_i,idx_j;
340     // passage pour les index de la forme i,j à la forme vecteur
341     Tableau2 <int> odVect;
342 };
343 public :
344     static const ChangementIndex cdex1BBBB;
345
346     // fonction pour le poduit contracté à gauche
347     TenseurBB& Prod_gauche( const TenseurHH & F) const;
348 };
349
350
351 //
352 //-----
353 // cas des composantes mixte 1BBHH

```

```

354 //-----
355
356 class Tenseur1BBHH : public TenseurBBHH
357 { // surcharge de l'operator de lecture
358     friend istream & operator » (istream &, Tenseur1BBHH &);
359     // surcharge de l'operator d'écriture
360     friend ostream & operator « (ostream &, const Tenseur1BBHH &);
361
362     public :
363         // Constructeur
364         Tenseur1BBHH() ; // par défaut
365         // initialisation de toutes les composantes a une meme valeur val
366         Tenseur1BBHH(const double val);
367         // initialisation à partir d'un produit tensoriel
368         // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
369         Tenseur1BBHH(const TenseurBB & aBB, const TenseurHH & bHH);
370
371         // DESTRUCTEUR :
372         ~Tenseur1BBHH() ;
373         // constructeur a partir d'une instance non differenciee
374         // dans le cas d'un tenseur quelconque, celui-ci
375         // est converti à condition que les symétries existent sinon erreur en debug
376         // opération longue dans ce cas !
377         Tenseur1BBHH (const TenseurBBHH &);
378         // constructeur de copie
379         Tenseur1BBHH (const Tenseur1BBHH &);
380
381         // METHODES PUBLIQUES :
382         //(2)    virtuelles
383         // initialise toutes les composantes à val
384         void Inita(double val) ;
385         // operations
386         TenseurBBHH & operator + ( const TenseurBBHH &) const ;
387         void operator += ( const TenseurBBHH &);
388         TenseurBBHH & operator - ( ) const ; // oppose du tenseur
389         TenseurBBHH & operator - ( const TenseurBBHH &) const ;
390         void operator -= ( const TenseurBBHH &);
391         TenseurBBHH & operator = ( const TenseurBBHH &);
392         TenseurBBHH & operator * (const double &) const ;
393         void operator *= ( const double &);
394         TenseurBBHH & operator / ( const double &) const ;
395         void operator /= ( const double &);
396
397         // produit contracte à droite avec un tenseur du second ordre
398         // différent à gauche !!
399         TenseurBB& operator && ( const TenseurBB &) const ;
400         //fonctions définissant le produit tensoriel normal de deux tenseurs
401         // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
402         static TenseurBBHH & Prod_tensoriel(const TenseurBB & aBB, const TenseurHH & bHH) ;
403
404         // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
405         // les 2 premiers indices sont échangés avec les deux derniers indices
406         TenseurHHBB & Transposelet2avec3et4() const ;
407
408         // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
409         // plusZero = true: les données manquantes sont mises à 0
410         // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
411         // des données possibles
412         void Affectation_trans_dimension(const TenseurBBHH & B,bool plusZero);
413
414         // test
415         int operator == ( const TenseurBBHH &) const ;
416
417         // Retourne la composante i,j,k,l du tenseur
418         // acces en écriture,
419         void Change (int i, int j, int k, int l,const double& val) ;
420         // en cumul : équivalent de +=
421         void ChangePlus (int i, int j, int k, int l,const double& val);
422
423         // Retourne la composante i,j,k,l du tenseur
424         // acces en lecture seule
425         double operator () (int i, int j, int k, int l) const ;
426
427         // calcul du maximum en valeur absolu des composantes du tenseur
428         double MaxiComposante() const;
429
430         // lecture et écriture de données
431         istream & Lecture(istream & entree);
432         ostream & Ecriture(ostream & sort) const ;
433
434     protected :
435         // allocator dans la liste de data
436         listdoublelIter ipointe;
437         // --- gestion de changement d'index ----
438         class ChangementIndex
439         { public:
440             ChangementIndex();

```



```

441         // passage pour les index de la forme vecteur à la forme i,j
442         Tableau <int> idx_i,idx_j;
443         // passage pour les index de la forme i,j à la forme vecteur
444         Tableau2 <int> odVect;
445     };
446 public :
447     static const ChangementIndex  cdex1BBHH;
448
449     // fonction pour le produit contracté à gauche
450     TenseurHH& Prod_gauche( const TenseurHH & F) const;
451 };
452 //
453 //-----
454 //          cas des composantes mixte 2HHBB
455 //-----
456
457 class Tenseur1HHBB : public TenseurHHBB
458 { // surcharge de l'operator de lecture
459     friend istream & operator » (istream &, Tenseur1HHBB &);
460     // surcharge de l'operator d'écriture
461     friend ostream & operator « (ostream &, const Tenseur1HHBB &);
462
463 public :
464     // Constructeur
465     Tenseur1HHBB() ; // par défaut
466     // initialisation de toutes les composantes a une meme valeur val
467     Tenseur1HHBB(const double val);
468     // initialisation à partir d'un produit tensoriel avec 1 cas
469     // booleen = true : produit tensoriel normal
470     //          *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
471     Tenseur1HHBB(const TenseurHH & aHH, const TenseurBB & bBB);
472
473     // DESTRUCTEUR :
474     ~Tenseur1HHBB() ;
475     // constructeur a partir d'une instance non differenciee
476     // dans le cas d'un tenseur quelconque, celui-ci
477     // est converti à condition que les symétries existent sinon erreur en debug
478     // opération longue dans ce cas !
479     Tenseur1HHBB (const TenseurHHBB &);
480     // constructeur de copie
481     Tenseur1HHBB (const Tenseur1HHBB &);
482
483     // METHODES PUBLIQUES :
484     //2) virtuelles
485     // initialise toutes les composantes à val
486     void Inita(double val) ;
487     // operations
488     TenseurHHBB & operator + ( const TenseurHHBB &) const ;
489     void operator += ( const TenseurHHBB &);
490     TenseurHHBB & operator - () const ; // oppose du tenseur
491     TenseurHHBB & operator - ( const TenseurHHBB &) const ;
492     void operator -= ( const TenseurHHBB &);
493     TenseurHHBB & operator = ( const TenseurHHBB &);
494     TenseurHHBB & operator * (const double &) const ;
495     void operator *= ( const double &);
496     TenseurHHBB & operator / ( const double &) const ;
497     void operator /= ( const double &);
498
499     // produit contracte à droite avec un tenseur du second ordre
500     // différent à gauche !!
501     TenseurHH& operator && ( const TenseurHH &) const ;
502     //fonctions définissant le produit tensoriel normal de deux tenseurs
503     // *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
504     static TenseurHHBB & Prod_tensoriel(const TenseurHH & aHH, const TenseurBB & bBB) ;
505
506     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
507     // les 2 premiers indices sont échangés avec les deux derniers indices
508     TenseurBBHH & Transposelet2avec3et4() const ;
509
510     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
511     // plusZero = true: les données manquantes sont mises à 0
512     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
513     // des données possibles
514     void Affectation_trans_dimension(const TenseurHHBB & B,bool plusZero);
515
516     // test
517     int operator == ( const TenseurHHBB &) const ;
518
519     // Retourne la composante i,j,k,l du tenseur
520     // acces en écriture,
521     void Change (int i, int j, int k, int l,const double& val) ;
522     // en cumul : équivalent de +=
523     void ChangePlus (int i, int j, int k, int l,const double& val);
524
525     // Retourne la composante i,j,k,l du tenseur
526     // acces en lecture seule
527     double operator () (int i, int j, int k, int l) const ;

```

```

528
529 // calcul du maximum en valeur absolu des composantes du tenseur
530 double MaxiComposante() const;
531
532 // lecture et écriture de données
533 istream & Lecture(istream & entree);
534 ostream & Ecriture(ostream & sort) const ;
535
536 protected :
537 // allocator dans la liste de data
538 listdoubleIter ipointe;
539 // --- gestion de changement d'index ----
540 class ChangementIndex
541 { public:
542   ChangementIndex();
543   // passage pour les index de la forme vecteur à la forme i,j
544   Tableau <int> idx_i,idx_j;
545   // passage pour les index de la forme i,j à la forme vecteur
546   Tableau2 <int> odVect;
547 };
548 public :
549   static const ChangementIndex cdex1HHBB;
550
551 // fonction pour le produit contracté à gauche
552 TenseurBB& Prod_gauche( const TenseurBB & F) const;
553 };
554
555 #ifndef MISE_AU_POINT
556 #include "TenseurQ1-1.cc"
557 #include "TenseurQ1-2.cc"
558 #define TenseurQ1_H_deja_inclus
559 #endif
560
561
562 #endif

```

## 7.471 TenseurQ-2.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 * UNIVERSITE DE BRETAGNE SUD (UBS) --- ENSIBS DE LORIENT *
34 *****/
35 * LHMATB- Equipe DE GENIE MECANIQUE ET MATERIAUX (EG2M) *
36 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
37 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
38 *****/
39 * DATE: 09/06/2015 *
40 * * $ *
41 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
42 * phone 0297874573, fax : 0297874588 *
43 * * $ *
44 * PROJET: Herezh++ *
45 * * $ *
46 *****/
47 * BUT: Definition des classes derivees tenseur du 4ieme ordre *

```

```

48 *          de dimension2. Ici de nombreuses fonctions ne sont pas *
49 *          pas disponible du à la forme particulière de stockage *
50 *          par contre c'est économique en stockage. *
51 *          Stockage: (ijkl) = avec 9 valeurs *
52 *          en fait tenseur symétrique sur ij et sur kl tel que: *
53 *          A(i,j,k,l) = A(j,i,k,l) = A(i,j,l,k) = A(j,i,l,k) *
54 *          $ *
55 *          ***** *
56 *          VERIFICATION: *
57 *          *
58 *          ! date ! auteur ! but ! *
59 *          ----- *
60 *          ! ! ! ! *
61 *          $ *
62 *          ***** *
63 *          MODIFICATIONS: *
64 *          ! date ! auteur ! but ! *
65 *          ----- *
66 *          $ *
67 *          *****/
68 #ifndef TENSEURQ2_H
69 #define TENSEURQ2_H
70
71 #include <iostream>
72 #include "TenseurQ.h"
73 #include "PtTabRel.h"
74 #include "Tableau2_T.h"
75 #include "Tenseur2.h"
76
77 //-----
78 //          cas des composantes 4 fois contravariantes 2HHHH
79 //-----
80 class TenseurBBHH;
81 class TenseurHHBB;
82
83
84 class Tenseur2HHHH : public TenseurHHHH
85 { // surcharge de l'operator de lecture
86   friend istream & operator >> (istream &, Tenseur2HHHH &);
87   // surcharge de l'operator d'écriture
88   friend ostream & operator << (ostream &, const Tenseur2HHHH &);
89
90 public :
91   // Constructeur
92   Tenseur2HHHH() ; // par défaut
93   // initialisation de toutes les composantes a une meme valeur val
94   Tenseur2HHHH(const double val);
95   // initialisation à partir d'un produit tensoriel avec 2 cas
96   // booleen = true : produit tensoriel normal
97   //          *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
98   // booleen = false : produit tensoriel barre, qui conserve les symétries
99   // *this(i,j,k,l)
100  // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
101  Tenseur2HHHH(bool normal, const TenseurHH & aHH, const TenseurHH & bHH);
102
103  // DESTRUCTEUR :
104  ~Tenseur2HHHH() ;
105  // constructeur a partir d'une instance non differenciée
106  // dans le cas d'un tenseur quelconque, celui-ci
107  // est converti à condition que les symétries existent sinon erreur en debug
108  // opération longue dans ce cas !
109  Tenseur2HHHH (const TenseurHHHH &);
110  // constructeur de copie
111  Tenseur2HHHH (const Tenseur2HHHH &);
112
113  // METHODES PUBLIQUES :
114  //2) virtuelles
115  // initialise toutes les composantes à val
116  void Inita(double val) ;
117  // operations
118  TenseurHHHH & operator + ( const TenseurHHHH & ) const ;
119  void operator += ( const TenseurHHHH & );
120  TenseurHHHH & operator - ( ) const ; // oppose du tenseur
121  TenseurHHHH & operator - ( const TenseurHHHH & ) const ;
122  void operator -= ( const TenseurHHHH & );
123  TenseurHHHH & operator = ( const TenseurHHHH & );
124  TenseurHHHH & operator = ( const Tenseur2HHHH & B )
125  { return this->operator=((TenseurHHHH &) B); };
126  TenseurHHHH & operator * (const double &) const ;
127  void operator *= ( const double & );
128  TenseurHHHH & operator / ( const double &) const ;
129  void operator /= ( const double & );
130
131  // produit contracte à droite avec un tenseur du second ordre
132  // différent à gauche !!
133  TenseurHH& operator && ( const TenseurBB & ) const ;

```

```

134 //fonctions définissant le produit tensoriel normal de deux tenseurs
135 // *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
136 static TenseurHHHH & Prod_tensoriel(const TenseurHH & aHH, const TenseurHH & bHH) ;
137 //fonctions définissant le produit tensoriel barre de deux tenseurs
138 // conservant les symétries !!
139 // *this(i,j,k,l)
140 // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
141 static TenseurHHHH & Prod_tensoriel_barre(const TenseurHH & aHH, const TenseurHH & bHH) ;
142
143 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
144 // les 2 premiers indices sont échangés avec les deux derniers indices
145 TenseurHHHH & Transposelet2avec3et4() const ;
146
147 // il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
148 // --> variation par rapport aux composantes covariantes d'un tenseur (ex: composantes eps_ij)
149
150 // d sigma^ij / d eps_kl = d gamma^i_{.a} / d eps_kl . sigma^ab . gamma^j_{.b}
151 // + gamma^i_{.a} . d sigma^ab / d eps_kl . gamma^j_{.b}
152 // + gamma^i^_{.a} . sigma^ab . d gamma^j_{.b} / d eps_kl
153
154 // connaissant sa variation dans la base actuelle
155 // var_tensHHHH : en entrée: la variation du tenseur dans la base initiale qu'on appelle g_i
156 // ex: var_tensHHHH(ijkl) = d A^ij / d eps_kl
157 // : en sortie: la variation du tenseur dans la base finale qu'on appelle gp_i
158 // gamma : en entrée gpH(i) = gamma(i,j) * gH(j)
159 // var_gamma : en entrée : la variation de gamma
160 // ex: var_gamma(i,j,k,l) = d gamma^i_{.j} / d eps_kl
161 // tensHH : le tenseur dont on cherche la variation
162 /// -- pour mémoire ---
163 // changement de base (cf. théorie) : la matrice beta est telle que:
164 // gpB(i) = beta(i,j) * gB(j) <=> gp_i = beta_i^j * g_j
165 // et la matrice gamma telle que:
166 // gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
167 // gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
168 // c-a-d= gp^i = gamma^i_j * g^j
169 // rappel des différentes relations entre beta et gamma
170 // [beta]^{-1} = [gamma]^T ; [beta]^{-1T} = [gamma]
171 // [beta] = [gamma]^{-1T} ; [beta]^T = [gamma]^{-1}
172 // changement de base pour un tenseur en deux fois covariants:
173 // [Ap^kl] = [gamma] * [A^ij] * [gamma]^T
174
175 static TenseurHHHH & Var_tenseur_dans_nouvelle_base
176 (const Mat_pleine& gamma, Tenseur2HHHH& var_tensHHHH, const Tableau2 <Tenseur2HH>&
var_gamma
177 ,const Tenseur2HH& tensHH);
178
179
180 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
181 // plusZero = true: les données manquantes sont mises à 0
182 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
183 // des données possibles
184 virtual void Affectation_trans_dimension(const TenseurHHHH & B, bool plusZero);
185
186 // transférer un tenseur général de même dimension accessible en indice, dans un tenseur 9
Tenseur2HHHH
187 // il n'y a pas de symétrisation !, seule certaines composantes sont prises en compte
188 void TransfertDunTenseurGeneral(const TenseurHHHH & aHHHH);
189
190 // ---- manipulation d'indice ----
191 // on baisse les deux premiers indices -> création d'un tenseur TenseurBBHH
192 TenseurBBHH& Baisse2premiersIndices();
193 // on baisse les deux derniers indices -> création d'un tenseur TenseurHHBB
194 TenseurHHBB& Baisse2derniersIndices();
195
196 // calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
197 // en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
198 // différente du tenseur courant suivant que la dimension absolue et la dimension locale
199 // sont égales ou différentes, retour d'une reference sur A
200 TenseurHHHH & Baselocale(TenseurHHHH & A, const BaseH & gi) const ;
201 // changement des composantes du tenseur, retour donc dans la même variance
202 // en argument : A -> une reference sur le tenseur résultat qui a la même dimension
203 // retour d'une reference sur A
204 // A = A^{ijkl} g_i rond g_j rond g_k rond g_l = A^{efgh} gp_i rond gpp_j rond g_k rond gp_l
205 // g_i = beta_i^j gp_j --> A^{efgh} = A^{ijkl} beta_i^e beta_j^f beta_k^g beta_l^h
206 TenseurHHHH & ChangeBase(TenseurHHHH & A, const BaseB & gi) const;
207
208 // test
209 int operator == ( const TenseurHHHH & ) const ;
210
211 // change la composante i,j,k,l du tenseur
212 // acces en ecriture,
213 void Change (int i, int j, int k, int l, const double& val);
214 // en cumul : équivalent de +=
215 void ChangePlus (int i, int j, int k, int l, const double& val);
216
217 // Retourne la composante i,j,k,l du tenseur
218 // acces en lecture seule

```

```

219     double operator () (int i, int j, int k, int l) const ;
220
221     // calcul du maximum en valeur absolu des composantes du tenseur
222     double MaxiComposante() const;
223
224     // lecture et écriture de données
225     istream & Lecture(istream & entree);
226     ostream & Ecriture(ostream & sort) const ;
227
228     // affichage sous forme de tableau bidim
229     void Affiche_bidim(ostream & sort) const ;
230
231     protected :
232     // allocator dans la liste de data
233     listdouble9Iter ipointe;
234     // --- gestion de changement d'index ----
235     class ChangementIndex
236     { public:
237         ChangementIndex();
238         // passage pour les index de la forme vecteur à la forme i,j
239         Tableau <int> idx_i,idx_j;
240         // passage pour les index de la forme i,j à la forme vecteur
241         Tableau2 <int> odVect;
242     };
243     public :
244     static const ChangementIndex cdex2HHHH;
245
246     // fonction pour le produit contracté à gauche
247     TenseurHH& Prod_gauche( const TenseurBB & F) const;
248
249 };
250 //
251 //-----
252 //          cas des composantes 4 fois covariantes
253 //-----
254 class Tenseur2BBBB : public TenseurBBBB
255 { // surcharge de l'operator de lecture
256     friend istream & operator » (istream &, Tenseur2BBBB &);
257     // surcharge de l'operator d'écriture
258     friend ostream & operator « (ostream &, const Tenseur2BBBB &);
259
260     public :
261         // Constructeur
262         Tenseur2BBBB() ; // par défaut
263         // initialisation de toutes les composantes a une meme valeur val
264         Tenseur2BBBB(const double val);
265         // initialisation à partir d'un produit tensoriel avec 2 cas
266         // booleen = true : produit tensoriel normal
267         //          *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
268         // booleen = false : produit tensoriel barre
269         //          *this(i,j,k,l)
270         //          = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
271         Tenseur2BBBB(bool normal, const TenseurBB & aBB, const TenseurBB & bBB);
272
273         // DESTRUCTEUR :
274         ~Tenseur2BBBB() ;
275         // constructeur a partir d'une instance non differentiee
276         // dans le cas d'un tenseur quelconque, celui-ci
277         // est converti à condition que les symétries existent sinon erreur en debug
278         // opération longue dans ce cas !
279         Tenseur2BBBB (const TenseurBBBB &);
280         // constructeur de copie
281         Tenseur2BBBB (const Tenseur2BBBB &);
282
283         // METHODES PUBLIQUES :
284         //2) virtuelles
285         // initialise toutes les composantes à val
286         void Inita(double val) ;
287         // operations
288         TenseurBBBB & operator + ( const TenseurBBBB &) const ;
289         void operator += ( const TenseurBBBB &);
290         TenseurBBBB & operator - () const ; // oppose du tenseur
291         TenseurBBBB & operator - ( const TenseurBBBB &) const ;
292         void operator -= ( const TenseurBBBB &);
293         TenseurBBBB & operator = ( const TenseurBBBB &);
294         TenseurBBBB & operator = ( const Tenseur2HHHH & B)
295         { return this->operator=((TenseurBBBB &) B); };
296         TenseurBBBB & operator * (const double &) const ;
297         void operator *= ( const double &);
298         TenseurBBBB & operator / ( const double &) const ;
299         void operator /= ( const double &);
300
301         // produit contracte à droite avec un tenseur du second ordre
302         // différent à gauche !!
303         TenseurBB& operator && ( const TenseurHH &) const ;
304         //fonctions définissant le produit tensoriel normal de deux tenseurs
305         // *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl

```

```

306 static TenseurBBBB & Prod_tensoriel(const TenseurBB & aBB, const TenseurBB & bBB) ;
307 //fonctions définissant le produit tensoriel barre de deux tenseurs
308 // concernant les symétries !!
309 // *this(i,j,k,l)
310 // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
311 static TenseurBBBB & Prod_tensoriel_barre(const TenseurBB & aBB, const TenseurBB & bBB) ;
312
313 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
314 // les 2 premiers indices sont échangés avec les deux derniers indices
315 TenseurBBBB & Transposelet2avec3et4() const ;
316
317 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
318 // plusZero = true: les données manquantes sont mises à 0
319 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
320 // des données possibles
321 virtual void Affectation_trans_dimension(const TenseurBBBB & B,bool plusZero);
322
323 // transférer un tenseur général de même dimension accessible en indice, dans un tenseur 9
Tenseur2HHHH
324 // il n'y a pas de symétrisation !, seule certaines composantes sont prises en compte
325 void TransfertDunTenseurGeneral(const TenseurBBBB & aBBBB);
326
327 // ---- manipulation d'indice ----
328 // on monte les deux premiers indices -> création d'un tenseur TenseurHHBB
329 TenseurHHBB& Monte2premiersIndices();
330 // on monte les deux derniers indices -> création d'un tenseur TenseurBBHH
331 TenseurBBHH& Monte2derniersIndices();
332 // on monte les 4 indices -> création d'un tenseur TenseurHHHH
333 TenseurHHHH& Monte4Indices();
334
335 // calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
336 // en argument : A -> une reference sur le tenseur résultat qui peut avoir une dimension
337 // différente du tenseur courant suivant que la dimension absolue et la dimension locale
338 // sont égales ou différentes , retour d'une reference sur A
339 TenseurBBBB & Baselocale(TenseurBBBB & A,const BaseB & gi) const ;
340 // changement des composantes du tenseur, retour donc dans la même variance
341 // en argument : A -> une reference sur le tenseur résultat qui a la même dimension
342 // retour d'une reference sur A
343 // A = A_(ijkl) g^i rond g^j rond g^k rond g_l = A'_(efgh) gp^i rond gpp^j rond g^k rond gp^l
344 // g^i = gamma^i_j gp^j --> A'_(efgh) = A_(ijkl) gamma^i_e gamma^j_f gamma^k_g gamma^l_h
345 TenseurBBBB & ChangeBase(TenseurBBBB & A,const BaseH & gi) const;
346
347
348 // test
349 int operator == ( const TenseurBBBB &) const ;
350
351 // Retourne la composante i,j,k,l du tenseur
352 // acces en ecriture,
353 void Change (int i, int j, int k, int l,const double& val) ;
354 // en cumul : équivalent de +=
355 void ChangePlus (int i, int j, int k, int l,const double& val);
356
357 // Retourne la composante i,j,k,l du tenseur
358 // acces en lecture seule
359 double operator () (int i, int j, int k, int l) const ;
360
361 // calcul du maximum en valeur absolu des composantes du tenseur
362 double MaxiComposante() const;
363
364 // lecture et écriture de données
365 istream & Lecture(istream & entree);
366 ostream & Ecriture(ostream & sort) const ;
367
368 // affichage sous forme de tableau bidim
369 void Affiche_bidim(ostream & sort) const ;
370
371 protected :
372 // allocator dans la liste de data
373 listdouble9Iter ipointe;
374 // --- gestion de changement d'index ----
375 class ChangementIndex
376 { public:
377     ChangementIndex();
378     // passage pour les index de la forme vecteur à la forme i,j
379     Tableau <int> idx_i,idx_j;
380     // passage pour les index de la forme i,j à la forme vecteur
381     Tableau2 <int> odVect;
382 };
383 public :
384     static const ChangementIndex cdex2BBBB;
385
386     // fonction pour le poduit contracté à gauche
387     TenseurBB& Prod_gauche( const TenseurHH & F) const;
388
389 };
390
391 //

```

```

392 //-----
393 //          cas des composantes mixte 2BBHH
394 //-----
395
396 class Tenseur2BBHH : public TenseurBBHH
397 { // surcharge de l'operator de lecture
398   friend istream & operator » (istream &, Tenseur2BBHH &);
399   // surcharge de l'operator d'écriture
400   friend ostream & operator « (ostream &, const Tenseur2BBHH &);
401
402 public :
403   // Constructeur
404   Tenseur2BBHH() ; // par défaut
405   // initialisation de toutes les composantes a une meme valeur val
406   Tenseur2BBHH(const double val);
407   // initialisation à partir d'un produit tensoriel
408   // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
409   Tenseur2BBHH(const TenseurBB & aBB, const TenseurHH & bHH);
410
411   // DESTRUCTEUR :
412   ~Tenseur2BBHH() ;
413   // constructeur a partir d'une instance non differenciee
414   // dans le cas d'un tenseur quelconque, celui-ci
415   // est converti à condition que les symétries existent sinon erreur en debug
416   // opération longue dans ce cas !
417   Tenseur2BBHH (const TenseurBBHH &);
418   // constructeur de copie
419   Tenseur2BBHH (const Tenseur2BBHH &);
420
421   // METHODES PUBLIQUES :
422   //2) virtuelles
423   // initialise toutes les composantes à val
424   void Init(double val) ;
425   // operations
426   TenseurBBHH & operator + ( const TenseurBBHH &) const ;
427   void operator += ( const TenseurBBHH &);
428   TenseurBBHH & operator - () const ; // oppose du tenseur
429   TenseurBBHH & operator - ( const TenseurBBHH &) const ;
430   void operator -= ( const TenseurBBHH &);
431   TenseurBBHH & operator = ( const TenseurBBHH &);
432   TenseurBBHH & operator * (const double &) const ;
433   void operator *= ( const double &);
434   TenseurBBHH & operator / ( const double &) const ;
435   void operator /= ( const double &);
436
437   // produit contracte à droite avec un tenseur du second ordre
438   // différent à gauche !!
439   TenseurBB& operator && ( const TenseurBB &) const ;
440   //fonctions définissant le produit tensoriel normal de deux tenseurs
441   // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
442   static TenseurBBHH & Prod_tensoriel(const TenseurBB & aBB, const TenseurHH & bHH) ;
443
444   // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
445   // les 2 premiers indices sont échangés avec les deux derniers indices
446   TenseurHHBB & Transposelet2avec3et4() const ;
447
448   // il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
449   // --> variation par rapport aux composantes covariantes d'un tenseur (ex: composantes eps_ij)
450
451   //  $d \sigma_{ij} / d \epsilon_{kl} = d \beta_{i\{a} / d \epsilon_{kl} . \sigma_{ab} . \beta_{j\}^{b}$ 
452   //  $+ \beta_{i\{a} . d \sigma_{ab} / d \epsilon_{kl} . \beta_{j\}^{b}$ 
453   //  $+ \beta_{i\{a} . \sigma_{ab} . d \beta_{j\}^{b} / d \epsilon_{kl}$ 
454
455   // connaissant sa variation dans la base actuelle
456   // var_tensBBHH : en entrée: la variation du tenseur dans la base initiale qu'on appelle g^i
457   // ex: var_tensBBHH(ijkl) = d A_ij / d eps_kl
458   //      : en sortie: la variation du tenseur dans la base finale qu'on appelle gp^i
459   // beta      : en entrée gpB(i) = beta(i,j) * gB(j)
460   // var_beta  : en entrée : la variation de beta
461   // ex: var_beta(i,j,k,l) = d beta_i^{.j} / d eps_kl
462   // tensBB    : le tenseur dont on cherche la variation
463   /// -- pour mémoire ---
464   // changement de base (cf. théorie) : la matrice beta est telle que:
465   // gpB(i) = beta(i,j) * gB(j) <=> gp_i = beta_i^j * g_j
466   // et la matrice gamma telle que:
467   // gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
468   // gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
469   // c-a-d= gp^i = gamma^i_j * g^j
470   // rappel des différentes relations entre beta et gamma
471   // [beta]^{-1} = [gamma]^T ; [beta]^{-1T} = [gamma]
472   // [beta] = [gamma]^{-1T} ; [beta]^T = [gamma]^{-1}
473   // changement de base pour un tenseur en deux fois covariants:
474   // [Ap_kl] = [beta] * [A_ij] * [beta]^T
475
476   static TenseurBBHH & Var_tenseur_dans_nouvelle_base
477   (const Mat_pleine& beta, Tenseur2BBHH& var_tensBBHH, const Tableau2 <Tenseur2HH>& var_beta
478   ,const Tenseur2BB& tensBB);

```

```

479
480
481 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
482 // plusZero = true: les données manquantes sont mises à 0
483 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
484 // des données possibles
485 virtual void Affectation_trans_dimension(const TenseurBBHH & B,bool plusZero) ;
486
487 // test
488 int operator == ( const TenseurBBHH &) const ;
489
490 // Retourne la composante i,j,k,l du tenseur
491 // acces en ecriture,
492 void Change (int i, int j, int k, int l,const double& val) ;
493 // en cumul : équivalent de +=
494 void ChangePlus (int i, int j, int k, int l,const double& val);
495
496 // Retourne la composante i,j,k,l du tenseur
497 // acces en lecture seule
498 double operator () (int i, int j, int k, int l) const ;
499
500 // calcul du maximum en valeur absolu des composantes du tenseur
501 double MaxiComposante() const;
502
503 // lecture et écriture de données
504 istream & Lecture(istream & entree);
505 ostream & Ecriture(ostream & sort) const ;
506
507 // affichage sous forme de tableau bidim
508 void Affiche_bidim(ostream & sort) const ;
509
510 protected :
511 // allocator dans la liste de data
512 listdouble9Iter ipointe;
513 // --- gestion de changement d'index ----
514 class ChangementIndex
515 { public:
516     ChangementIndex();
517     // passage pour les index de la forme vecteur à la forme i,j
518     Tableau <int> idx_i,idx_j;
519     // passage pour les index de la forme i,j à la forme vecteur
520     Tableau2 <int> odVect;
521 };
522 public :
523     static const ChangementIndex cdex2BBHH;
524
525     // fonction pour le produit contracté à gauche
526     TenseurHH& Prod_gauche( const TenseurHH & F) const;
527 };
528 //
529 //-----
530 //          cas des composantes mixte 2HHBB
531 //-----
532
533 class Tenseur2HHBB : public TenseurHHBB
534 { // surcharge de l'operator de lecture
535     friend istream & operator » (istream &, Tenseur2HHBB &);
536     // surcharge de l'operator d'écriture
537     friend ostream & operator « (ostream &, const Tenseur2HHBB &);
538
539     public :
540         // Constructeur
541         Tenseur2HHBB() ; // par défaut
542         // initialisation de toutes les composantes a une meme valeur val
543         Tenseur2HHBB(const double val);
544         // initialisation à partir d'un produit tensoriel avec 1 cas
545         // booleen = true : produit tensoriel normal
546         //          *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
547         Tenseur2HHBB(const TenseurHH & aHH, const TenseurBB & bBB);
548
549         // DESTRUCTEUR :
550         ~Tenseur2HHBB() ;
551         // constructeur a partir d'une instance non differenciee
552         // dans le cas d'un tenseur quelconque, celui-ci
553         // est converti à condition que les symétries existent sinon erreur en debug
554         // opération longue dans ce cas !
555         Tenseur2HHBB (const TenseurHHBB &);
556         // constructeur de copie
557         Tenseur2HHBB (const Tenseur2HHBB &);
558
559         // METHODES PUBLIQUES :
560         //2) virtuelles
561         // initialise toutes les composantes à val
562         void Inita(double val) ;
563         // operations
564         TenseurHHBB & operator + ( const TenseurHHBB &) const ;
565         void operator += ( const TenseurHHBB &);

```



```

566 TenseurHHBB & operator - () const ; // oppose du tenseur
567 TenseurHHBB & operator - ( const TenseurHHBB &) const ;
568 void operator -= ( const TenseurHHBB &);
569 TenseurHHBB & operator = ( const TenseurHHBB &);
570 TenseurHHBB & operator * (const double &) const ;
571 void operator *= ( const double &);
572 TenseurHHBB & operator / ( const double &) const ;
573 void operator /= ( const double &);
574
575 // produit contracte à droite avec un tenseur du second ordre
576 // différent à gauche !!
577 TenseurHH& operator && ( const TenseurHH &) const ;
578 //fonctions définissant le produit tensoriel normal de deux tenseurs
579 // *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
580 static TenseurHHBB & Prod_tensoriel(const TenseurHH & aHH, const TenseurBB & bBB) ;
581
582 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
583 // les 2 premiers indices sont échangés avec les deux derniers indices
584 TenseurBBHH & Transposelet2avec3et4() const ;
585
586 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
587 // plusZero = true: les données manquantes sont mises à 0
588 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
589 // des données possibles
590 virtual void Affectation_trans_dimension(const TenseurHHBB & B,bool plusZero);
591
592 // test
593 int operator == ( const TenseurHHBB &) const ;
594
595 // Retourne la composante i,j,k,l du tenseur
596 // acces en ecriture,
597 void Change (int i, int j, int k, int l,const double& val) ;
598 // en cumul : équivalent de +=
599 void ChangePlus (int i, int j, int k, int l,const double& val);
600
601 // Retourne la composante i,j,k,l du tenseur
602 // acces en lecture seule
603 double operator () (int i, int j, int k, int l) const ;
604
605 // calcul du maximum en valeur absolu des composantes du tenseur
606 double MaxiComposante() const;
607
608 // lecture et écriture de données
609 istream & Lecture(istream & entree);
610 ostream & Ecriture(ostream & sort) const ;
611
612 // affichage sous forme de tableau bidim
613 void Affiche_bidim(ostream & sort) const ;
614
615 protected :
616 // allocator dans la liste de data
617 listdouble9Iter ipointe;
618 // --- gestion de changement d'index ----
619 class ChangementIndex
620 { public:
621     ChangementIndex();
622     // passage pour les index de la forme vecteur à la forme i,j
623     Tableau <int> idx_i,idx_j;
624     // passage pour les index de la forme i,j à la forme vecteur
625     Tableau2 <int> odVect;
626 };
627 public :
628     static const ChangementIndex cdex2HHBB;
629
630     // fonction pour le produit contracté à gauche
631     TenseurBB& Prod_gauche( const TenseurBB & F) const;
632 };
633
634 #ifndef MISE_AU_POINT
635 #include "TenseurQ2-1.cc"
636 #include "TenseurQ2-2.cc"
637 #define TenseurQ2_H_deja_inclus
638 #endif
639
640
641 #endif

```

## 7.472 TenseurQ-3.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.

```

```

7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *          LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)          *
34 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex      *
35 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
36 *****/
37 *      DATE:          3/5/2002                                          *
38 *                                                            $      *
39 *      AUTEUR:        G RIO (mailto:gerard.rio@univ-ubs.fr)           *
40 *                    Tel 0297874571 fax : 02.97.87.45.72            *
41 *                                                            $      *
42 *      PROJET:        Herezh++                                         *
43 *                                                            $      *
44 *****/
45 *      BUT:          Definition des classes derivees tenseur du 4ieme ordre *
46 *                  de dimension3. Ici de nombreuses fonctions ne sont pas *
47 *                  pas disponible du à la forme particulière de stockage *
48 *                  par contre c'est économique en stockage.          *
49 *                  Stockage: (ijkl) = avec 36 valeurs                *
50 *                  en fait tenseur symétrique sur ij et sur kl tel que: *
51 *                  A(i,j,k,l) = A(j,i,k,l) = A(i,j,l,k) = A(j,i,l,k) *
52 *                  typiquement le produit tensoriel de deux tenseurs symétriques : *
53 *                  A(i,j,k,l) = B(i,j) * C(k,l)                       *
54 *                  Chaque tenseur symétrique comporte 6 composantes *
55 *                  B(1,1)->1, B(2,2)->2, B(3,3)->3, B(2,1)->4, B(3,2)->5, B(3,1)->6 *
56 *                  du coup A est rangé dans un tableau 6x6            *
57 *                                                            $      *
58 *          *****/ *
59 *
60 *      VERIFICATION: *
61 *
62 *      ! date ! auteur ! but ! *
63 *      ----- *
64 *      ! ! ! ! *
65 *      $ *
66 *          *****/ *
67 *
68 *      MODIFICATIONS: *
69 *      ! date ! auteur ! but ! *
70 *      ----- *
71 *      $ *
72 *****/
73 #ifndef TENSEURQ3_H
74 #define TENSEURQ3_H
75
76 #include <iostream>
77 #include "TenseurQ.h"
78 #include "PtTabRel.h"
79 #include "Tenseur3.h"
80 #include "Tableau2_T.h"
81
82 //-----
83 //          cas des composantes 4 fois contravariantes 3HHHH
84 //-----
85 class TenseurBBHH;
86 class TenseurHHBB;
87
88 class Tenseur3HHHH : public TenseurHHHH
89 { // surcharge de l'operator de lecture
90     friend istream & operator >> (istream &, Tenseur3HHHH &);
91     // surcharge de l'operator d'écriture
92     friend ostream & operator << (ostream &, const Tenseur3HHHH &);
93 }

```

```

94 public :
95 // Constructeur
96 Tenseur3HHHH() ; // par défaut
97 // initialisation de toutes les composantes a une meme valeur val
98 Tenseur3HHHH(const double val);
99 // initialisation à partir d'un produit tensoriel avec 2 cas
100 // booleen = true : produit tensoriel normal
101 // *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
102 // booleen = false : produit tensoriel barre, qui conserve les symétries
103 // *this(i,j,k,l)
104 // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
105 Tenseur3HHHH(bool normal, const TenseurHH & aHH, const TenseurHH & bHH);
106
107 // DESTRUCTEUR :
108 ~Tenseur3HHHH() ;
109 // constructeur a partir d'une instance non differentiee
110 // dans le cas d'un tenseur quelconque, celui-ci
111 // est converti à condition que les symétries existent sinon erreur en debug
112 // opération longue dans ce cas !
113 Tenseur3HHHH (const TenseurHHHH &);
114 // constructeur de copie
115 Tenseur3HHHH (const Tenseur3HHHH &);
116
117 // METHODES PUBLIQUES :
118 //2) virtuelles
119 // initialise toutes les composantes à val
120 void Init(double val) ;
121 // operations
122 TenseurHHHH & operator + ( const TenseurHHHH & ) const ;
123 void operator += ( const TenseurHHHH & );
124 TenseurHHHH & operator - ( ) const ; // oppose du tenseur
125 TenseurHHHH & operator - ( const TenseurHHHH & ) const ;
126 void operator -= ( const TenseurHHHH & );
127 TenseurHHHH & operator = ( const TenseurHHHH & );
128 TenseurHHHH & operator = ( const Tenseur3HHHH & B );
129 { return this->operator=((TenseurHHHH &) B); };
130 TenseurHHHH & operator * (const double & ) const ;
131 void operator *= ( const double & );
132 TenseurHHHH & operator / ( const double & ) const ;
133 void operator /= ( const double & );
134
135 // produit contracte à droite avec un tenseur du second ordre
136 // différent à gauche !!
137 // A(i,j,k,l)*B(l,k)=A..B
138 // on commence par contracter l'indice du milieu puis externe
139 TenseurHH& operator && ( const TenseurBB & ) const ;
140 // contraction verticale: A(i,j,k,l)*B(k,l)=A:B
141 TenseurHH& ContractionVerticale( const TenseurBB & ) const ;
142
143 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
144 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
145 // A(i,j,k,l)*B(l,k,o,p)=A...B
146 TenseurHHHH& operator && ( const TenseurBBHH & ) const ;
147 TenseurHHBB& operator && ( const TenseurBBBB & ) const ;
148
149 //fonctions définissant le produit tensoriel normal de deux tenseurs
150 // *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
151 static TenseurHHHH & Prod_tensoriel(const TenseurHH & aHH, const TenseurHH & bHH) ;
152 //fonctions définissant le produit tensoriel croisé de deux tenseurs
153 // *this=aHH(i,k).bHH(j,l) gBi gBj gBk gBl
154 static TenseurHHHH & Prod_tensoriel_croise(const TenseurHH & aHH, const TenseurHH & bHH) ;
155 //fonctions définissant le produit tensoriel croisé de deux tenseurs
156 // *this=aHH(i,l).bHH(j,k) gBi gBj gBk gBl
157 static TenseurHHHH & Prod_tensoriel_croise_croise(const TenseurHH & aHH, const TenseurHH & bHH) ;
158
159 //fonctions définissant le produit tensoriel barre de deux tenseurs
160 // conservant les symétries !!
161 // *this(i,j,k,l)
162 // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
163 static TenseurHHHH & Prod_tensoriel_barre(const TenseurHH & aHH, const TenseurHH & bHH) ;
164
165 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
166 // les 2 premiers indices sont échangés avec les deux derniers indices
167 TenseurHHHH & Transposelet2avec3et4() const ;
168
169 // il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
170 // --> variation par rapport aux composantes covariantes d'un tenseur (ex: composantes eps_ij)
171
172 // d sigma^ij / d eps_kl = d gamma^i_{.a} / d eps_kl . sigma^ab . gamma^j_{.b}
173 // + gamma^i_{.a} . d sigma^ab / d eps_kl . gamma^j_{.b}
174 // + gamma^i^_{.a} . sigma^ab . d gamma^j_{.b} / d eps_kl
175
176 // connaissant sa variation dans la base actuelle
177 // var_tensHHHH : en entrée: la variation du tenseur dans la base initiale qu'on appelle g_i
178 // ex: var_tensHHHH(ijkl) = d A^ij / d eps_kl
179 // : en sortie: la variation du tenseur dans la base finale qu'on appelle gp_i
180 // gamma : en entrée gpH(i) = gamma(i,j) * gH(j)

```

```

181 // var_gamma : en entrée : la variation de gamma
182 // ex: var_gamma(i,j,k,l) = d gamma^i_{.j} / d eps_kl
183 // tensHH : le tenseur dont on cherche la variation
184 /// -- pour mémoire ---
185 // changement de base (cf. théorie) : la matrice beta est telle que:
186 // gpB(i) = beta(i,j) * gB(j) <=> gp_i = beta_i^j * g_j
187 // et la matrice gamma telle que:
188 // gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
189 // gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
190 // c-a-d= gp^i = gamma^i_j * g^j
191 // rappel des différentes relations entre beta et gamma
192 // [beta]^{-1} = [gamma]^T ; [beta]^{-1T} = [gamma]
193 // [beta] = [gamma]^{-1T} ; [beta]^T = [gamma]^{-1}
194 // changement de base pour un tenseur en deux fois covariants:
195 // [Ap^kl] = [gamma] * [A^ij] * [gamma]^T
196
197 static TenseurHHHH & Var_tenseur_dans_nouvelle_base
198 (const Mat_pleine& gamma, Tenseur3HHHH& var_tensHHHH, const Tableau2 <Tenseur3HH>&
var_gamma
199 ,const Tenseur3HH& tensHH);
200
201
202 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
203 // plusZero = true: les données manquantes sont mises à 0
204 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
205 // des données possibles
206 virtual void Affectation_trans_dimension(const TenseurHHHH & B, bool plusZero);
207
208 // transférer un tenseur général de même dimension accessible en indice, dans un tenseur 36
Tenseur3HHHH
209 // il n'y a pas de symétrisation !, seule certaines composantes sont prises en compte
210 void TransfertDunTenseurGeneral(const TenseurHHHH & aHHHH);
211
212 // ---- manipulation d'indice ----
213 // on baisse les deux premiers indices -> création d'un tenseur TenseurBBHH
214 TenseurBBHH& Baisse2premiersIndices();
215 // on baisse les deux derniers indices -> création d'un tenseur TenseurHHBB
216 TenseurHHBB& Baisse2derniersIndices();
217
218 // calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
219 // en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension
220 // différente du tenseur courant suivant que la dimension absolue et la dimension locale
221 // sont égales ou différentes , retour d'une référence sur A
222 TenseurHHHH & Baselocale(TenseurHHHH & A, const BaseH & gi) const ;
223
224 // changement des composantes du tenseur, retour donc dans la même variance
225 // en argument : A -> une référence sur le tenseur résultat qui a la même dimension
226 // retour d'une référence sur A
227 // A = A^{ijkl} g_i rond g_j rond g_k rond g_l = A^{efgh} gp_i rond gpp_j rond g_k rond gp_l
228 // g_i = beta_i^j gp_j --> A^{ijkl} beta_i^e beta_j^f beta_k^g beta_l^h
229 TenseurHHHH & ChangeBase(TenseurHHHH & A, const BaseB & gi) const;
230
231 // test
232 int operator == ( const TenseurHHHH &) const ;
233
234 // change la composante i,j,k,l du tenseur
235 // acces en ecriture,
236 void Change (int i, int j, int k, int l, const double& val) ;
237 // en cumul : équivalent de +=
238 void ChangePlus (int i, int j, int k, int l, const double& val);
239
240 // Retourne la composante i,j,k,l du tenseur
241 // acces en lecture seule
242 double operator () (int i, int j, int k, int l) const ;
243
244 // calcul du maximum en valeur absolu des composantes du tenseur
245 double MaxiComposante() const;
246
247 // lecture et écriture de données
248 istream & Lecture(istream & entree);
249 ostream & Ecriture(ostream & sort) const ;
250
251 // affichage sous forme de tableau bidim
252 void Affiche_bidim(ostream & sort) const ;
253
254 protected :
255 // allocator dans la liste de data
256 listdouble36Iter ipointe;
257 // --- gestion de changement d'index ----
258 class ChangementIndex
259 { public:
260 ChangementIndex();
261 // passage pour les index de la forme vecteur à la forme i,j
262 Tableau <int> idx_i, idx_j;
263 // passage pour les index de la forme i,j à la forme vecteur
264 Tableau2 <int> odVect;
265 };

```

```

266 public :
267     static const ChangementIndex cdex3HHHH;
268
269     // fonction pour le produit contracté à gauche
270     TenseurHH& Prod_gauche( const TenseurBB & F) const;
271
272 };
273 //
274 //-----
275 //          cas des composantes 4 fois covariantes
276 //-----
277 class Tenseur3BBBB : public TenseurBBBB
278 { // surcharge de l'operator de lecture
279     friend istream & operator » (istream &, Tenseur3BBBB &);
280     // surcharge de l'operator d'écriture
281     friend ostream & operator « (ostream &, const Tenseur3BBBB &);
282
283 public :
284     // Constructeur
285     Tenseur3BBBB() ; // par défaut
286     // initialisation de toutes les composantes a une meme valeur val
287     Tenseur3BBBB(const double val);
288     // initialisation à partir d'un produit tensoriel avec 2 cas
289     // booleen = true : produit tensoriel normal
290     //                               *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
291     // booleen = false : produit tensoriel barre
292     // *this(i,j,k,l)
293     // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
294     Tenseur3BBBB(bool normal, const TenseurBB & aBB, const TenseurBB & bBB);
295
296     // DESTRUCTEUR :
297     ~Tenseur3BBBB() ;
298     // constructeur a partir d'une instance non differentiee
299     // dans le cas d'un tenseur quelconque, celui-ci
300     // est converti à condition que les symétries existent sinon erreur en debug
301     // opération longue dans ce cas !
302     Tenseur3BBBB (const TenseurBBBB &);
303     // constructeur de copie
304     Tenseur3BBBB (const Tenseur3BBBB &);
305
306     // METHODES PUBLIQUES :
307 //2)    virtuelles
308     // initialise toutes les composantes à val
309     void Inita(double val) ;
310     // operations
311     TenseurBBBB & operator + ( const TenseurBBBB &) const ;
312     void operator += ( const TenseurBBBB &);
313     TenseurBBBB & operator - () const ; // oppose du tenseur
314     TenseurBBBB & operator - ( const TenseurBBBB &) const ;
315     void operator -= ( const TenseurBBBB &);
316     TenseurBBBB & operator = ( const TenseurBBBB &);
317     TenseurBBBB & operator = ( const Tenseur3HHHH & B)
318     { return this->operator=(TenseurBBBB &) B; };
319     TenseurBBBB & operator * (const double &) const ;
320     void operator *= ( const double &);
321     TenseurBBBB & operator / ( const double &) const ;
322     void operator /= ( const double &);
323
324     // produit contracte à droite avec un tenseur du second ordre
325     // différent à gauche !!
326     // A(i,j,k,l)*B(l,k)=A..B
327     // on commence par contracter l'indice du milieu puis externe
328     TenseurBB& operator && ( const TenseurHH &) const ;
329     // contraction verticale: A(i,j,k,l)*B(k,l)=A:B
330     TenseurBB& ContractionVerticale( const TenseurHH &) const ;
331
332     // produit deux fois contracte à droite avec un tenseur du quatrième ordre
333     // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
334     // A(i,j,k,l)*B(l,k,o,p)=A...B
335     TenseurBBBB& operator && ( const TenseurHHBB &) const ;
336     TenseurBBHH& operator && ( const TenseurHHHH &) const ;
337
338     //fonctions définissant le produit tensoriel normal de deux tenseurs
339     // *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
340     static TenseurBBBB & Prod_tensoriel(const TenseurBB & aBB, const TenseurBB & bBB) ;
341     //fonctions définissant le produit tensoriel barre de deux tenseurs
342     // concernant les symétries !!
343     // *this(i,j,k,l)
344     // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
345     static TenseurBBBB & Prod_tensoriel_barre(const TenseurBB & aBB, const TenseurBB & bBB) ;
346
347     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
348     // les 2 premiers indices sont échangés avec les deux derniers indices
349     TenseurBBBB & Transposelet2avec3et4() const ;
350
351     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
352     // plusZero = true: les données manquantes sont mises à 0

```

```

353 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
354 // des données possibles
355 virtual void Affectation_trans_dimension(const TenseurBBBB & B, bool plusZero);
356
357 // transférer un tenseur général de même dimension accessible en indice, dans un tenseur 36
Tenseur3HHHH
358 // il n'y a pas de symétrisation !, seule certaines composantes sont prises en compte
359 void TransfertDunTenseurGeneral(const TenseurBBBB & aBBBB);
360
361 // ---- manipulation d'indice ----
362 // on monte les deux premiers indices -> création d'un tenseur TenseurHHBB
363 TenseurHHBB& Monte2premiersIndices();
364 // on monte les deux derniers indices -> création d'un tenseur TenseurBBHH
365 TenseurBBHH& Monte2derniersIndices();
366 // on monte les 4 indices -> création d'un tenseur TenseurHHHH
367 TenseurHHHH& Monte4Indices();
368
369 // calcul des composantes locales du tenseur considéré s'exprimé dans une base absolue
370 // en argument : A -> une référence sur le tenseur résultat qui peut avoir une dimension
371 // différente du tenseur courant suivant que la dimension absolue et la dimension locale
372 // sont égales ou différentes , retour d'une référence sur A
373 TenseurBBBB & Baselocale(TenseurBBBB & A, const BaseB & gi) const ;
374 // changement des composantes du tenseur, retour donc dans la même variance
375 // en argument : A -> une référence sur le tenseur résultat qui a la même dimension
376 // retour d'une référence sur A
377 //  $A = A_{(ijkl)} g^i \text{ rond } g^j \text{ rond } g^k \text{ rond } g_l = A'_{(efgh)} g^{p^i} \text{ rond } g^{q^j} \text{ rond } g^k \text{ rond } g^{p^l}$ 
378 //  $g^i = \gamma^{a^i}_{i_j} g^{p^j} \rightarrow A'_{(efgh)} = A_{(ijkl)} \gamma^{a^i}_{i_e} \gamma^{a^j}_{j_f} \gamma^{a^k}_{k_g} \gamma^{a^l}_{l_h}$ 
379 TenseurBBBB & ChangeBase(TenseurBBBB & A, const BaseH & gi) const;
380
381
382 // test
383 int operator == ( const TenseurBBBB & ) const ;
384
385 // Retourne la composante i,j,k,l du tenseur
386 // acces en ecriture,
387 void Change (int i, int j, int k, int l, const double& val) ;
388 // en cumul : équivalent de +=
389 void ChangePlus (int i, int j, int k, int l, const double& val);
390
391 // Retourne la composante i,j,k,l du tenseur
392 // acces en lecture seule
393 double operator () (int i, int j, int k, int l) const ;
394
395 // calcul du maximum en valeur absolu des composantes du tenseur
396 double MaxiComposante() const;
397
398 // lecture et écriture de données
399 istream & Lecture(istream & entree);
400 ostream & Ecriture(ostream & sort) const ;
401
402 // affichage sous forme de tableau bidim
403 void Affiche_bidim(ostream & sort) const ;
404
405 protected :
406 // allocator dans la liste de data
407 listdouble36Iter ipointe;
408 // --- gestion de changement d'index ----
409 class ChangementIndex
410 { public:
411     ChangementIndex();
412     // passage pour les index de la forme vecteur à la forme i, j
413     Tableau <int> idx_i, idx_j;
414     // passage pour les index de la forme i, j à la forme vecteur
415     Tableau2 <int> odVect;
416 };
417 public :
418     static const ChangementIndex cdex3BBBB;
419
420     // fonction pour le produit contracté à gauche
421     TenseurBB& Prod_gauche( const TenseurHH & F) const;
422
423 };
424
425 //
426 //-----
427 //          cas des composantes mixte 3BBHH
428 //-----
429
430 class Tenseur3BBHH : public TenseurBBHH
431 { // surcharge de l'operator de lecture
432     friend istream & operator » (istream &, Tenseur3BBHH &);
433     // surcharge de l'operator d'ecriture
434     friend ostream & operator « (ostream &, const Tenseur3BBHH &);
435
436     public :
437         // Constructeur
438         Tenseur3BBHH() ; // par défaut

```

```

439 // initialisation de toutes les composantes a une meme valeur val
440 Tenseur3BBHH(const double val);
441 // initialisation à partir d'un produit tensoriel
442 // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
443 Tenseur3BBHH(const TenseurBB & aBB, const TenseurHH & bHH);
444
445 // DESTRUCTEUR :
446 ~Tenseur3BBHH() ;
447 // constructeur a partir d'une instance non differentiee
448 // dans le cas d'un tenseur quelconque, celui-ci
449 // est converti à condition que les symétries existent sinon erreur en debug
450 // opération longue dans ce cas !
451 Tenseur3BBHH (const TenseurBBHH &);
452 // constructeur de copie
453 Tenseur3BBHH (const Tenseur3BBHH &);
454
455 // METHODES PUBLIQUES :
456 //2) virtuelles
457 // initialise toutes les composantes à val
458 void Inita(double val) ;
459 // operations
460 TenseurBBHH & operator + ( const TenseurBBHH & ) const ;
461 void operator += ( const TenseurBBHH & );
462 TenseurBBHH & operator - ( ) const ; // oppose du tenseur
463 TenseurBBHH & operator - ( const TenseurBBHH & ) const ;
464 void operator -= ( const TenseurBBHH & );
465 TenseurBBHH & operator = ( const TenseurBBHH & );
466 TenseurBBHH & operator * (const double &) const ;
467 void operator *= ( const double & );
468 TenseurBBHH & operator / ( const double &) const ;
469 void operator /= ( const double & );
470
471 // produit contracte à droite avec un tenseur du second ordre
472 // différent à gauche !!
473 // A(i,j,k,l)*B(l,k)=A..B
474 // on commence par contracter l'indice du milieu puis externe
475 TenseurBB& operator && ( const TenseurBB & ) const ;
476 // contraction verticale: A(i,j,k,l)*B(k,l)=A:B
477 TenseurBB& ContractionVerticale( const TenseurBB & ) const ;
478
479 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
480 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
481 // A(i,j,k,l)*B(l,k,o,p)=A...B
482 TenseurBBBB& operator && ( const TenseurBBBB & ) const ;
483 TenseurBBHH& operator && ( const TenseurBBHH & ) const ;
484
485 //fonctions définissant le produit tensoriel normal de deux tenseurs
486 // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
487 static TenseurBBHH & Prod_tensoriel(const TenseurBB & aBB, const TenseurHH & bHH) ;
488
489 //fonctions définissant le produit tensoriel barre de deux tenseurs
490 // conservant les symétries !!
491 // *this(i,j,k,l)
492 // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
493 static TenseurBBHH & Prod_tensoriel_barre(const TenseurBB & aBB, const TenseurHH & bHH) ;
494
495 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
496 // les 2 premiers indices sont échangés avec les deux derniers indices
497 TenseurHHBB & Transposelet2avec3et4() const ;
498
499 // il s'agit ici de calculer la variation d'un tenseur dans une nouvelle base
500 // --> variation par rapport aux composantes covariantes d'un tenseur (ex: composantes eps_ij)
501
502 // d sigma_ij / d eps_kl = d beta_i^{.a} / d eps_kl . sigma_ab . beta_j^{.b}
503 // + beta_i^{.a} . d sigma_ab / d eps_kl . beta_j^{.b}
504 // + beta_i^{.a} . sigma_ab . d beta_j^{.b} / d eps_kl
505
506 // connaissant sa variation dans la base actuelle
507 // var_tensBBHH : en entrée: la variation du tenseur dans la base initiale qu'on appelle g^i
508 // ex: var_tensBBHH(ijkl) = d A_ij / d eps_kl
509 // : en sortie: la variation du tenseur dans la base finale qu'on appelle gp^i
510 // beta : en entrée gpB(i) = beta(i,j) * gB(j)
511 // var_beta : en entrée : la variation de beta
512 // ex: var_beta(i,j,k,l) = d beta_i^{.j} / d eps_kl
513 // tensBB : le tenseur dont on cherche la variation
514 // -- pour mémoire ---
515 // changement de base (cf. théorie) : la matrice beta est telle que:
516 // gpB(i) = beta(i,j) * gB(j) <=> gp_i = beta_i^j * g_j
517 // et la matrice gamma telle que:
518 // gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
519 // gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
520 // c-a-d= gp^i = gamma^i_j * g^j
521 // rappel des différentes relations entre beta et gamma
522 // [beta]^{-1} = [gamma]^T ; [beta]^{-1T} = [gamma]
523 // [beta] = [gamma]^{-1T} ; [beta]^T = [gamma]^{-1}
524 // changement de base pour un tenseur en deux fois covariants:
525 // [Ap_kl] = [beta] * [A_ij] * [beta]^T

```

```

526
527 static TenseurBBHH & Var_tenseur_dans_nouvelle_base
528     (const Mat_pleine& beta, Tenseur3BBHH& var_tensBBHH, const Tableau2 <Tenseur3HH>& var_beta
529     ,const Tenseur3BB& tensBB);
530
531 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
532 // plusZero = true: les données manquantes sont mises à 0
533 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
534 // des données possibles
535 virtual void Affectation_trans_dimension(const TenseurBBHH & B, bool plusZero);
536
537 // test
538 int operator == ( const TenseurBBHH &) const ;
539
540 // Retourne la composante i,j,k,l du tenseur
541 // acces en ecriture,
542 void Change (int i, int j, int k, int l, const double& val) ;
543 // en cumul : équivalent de +=
544 void ChangePlus (int i, int j, int k, int l, const double& val);
545
546 // Retourne la composante i,j,k,l du tenseur
547 // acces en lecture seule
548 double operator () (int i, int j, int k, int l) const ;
549
550 // calcul du maximum en valeur absolu des composantes du tenseur
551 double MaxiComposante() const;
552
553 // lecture et écriture de données
554 istream & Lecture(istream & entree);
555 ostream & Ecriture(ostream & sort) const ;
556
557 // affichage sous forme de tableau bidim
558 void Affiche_bidim(ostream & sort) const ;
559
560 protected :
561 // allocator dans la liste de data
562 listdouble36Iter ipointe;
563 // --- gestion de changement d'index ----
564 class ChangementIndex
565     { public:
566         ChangementIndex();
567         // passage pour les index de la forme vecteur à la forme i,j
568         Tableau <int> idx_i, idx_j;
569         // passage pour les index de la forme i,j à la forme vecteur
570         Tableau2 <int> odVect;
571     };
572 public :
573     static const ChangementIndex cdex3BBHH;
574
575     // fonction pour le produit contracté à gauche
576     TenseurHH& Prod_gauche( const TenseurHH & F) const;
577 };
578 //
579 //-----
580 //          cas des composantes mixte 3HHBB
581 //-----
582
583 class Tenseur3HHBB : public TenseurHHBB
584 { // surcharge de l'operator de lecture
585     friend istream & operator » (istream &, Tenseur3HHBB &);
586     // surcharge de l'operator d'écriture
587     friend ostream & operator « (ostream &, const Tenseur3HHBB &);
588
589     public :
590         // Constructeur
591         Tenseur3HHBB() ; // par défaut
592         // initialisation de toutes les composantes a une meme valeur val
593         Tenseur3HHBB(const double val);
594         // initialisation à partir d'un produit tensoriel avec 1 cas
595         // booleen = true : produit tensoriel normal
596         //          *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
597         Tenseur3HHBB(const TenseurHH & aHH, const TenseurBB & bBB);
598
599         // DESTRUCTEUR :
600         ~Tenseur3HHBB() ;
601         // constructeur a partir d'une instance non differenciee
602         // dans le cas d'un tenseur quelconque, celui-ci
603         // est converti à condition que les symétries existent sinon erreur en debug
604         // opération longue dans ce cas !
605         Tenseur3HHBB (const TenseurHHBB &);
606         // constructeur de copie
607         Tenseur3HHBB (const Tenseur3HHBB &);
608
609         // METHODES PUBLIQUES :
610         //2) virtuelles
611         // initialise toutes les composantes à val
612         void Init(double val) ;

```



```

613 // operations
614 TenseurHHBB & operator + ( const TenseurHHBB & ) const ;
615 void operator += ( const TenseurHHBB & );
616 TenseurHHBB & operator - ( ) const ; // oppose du tenseur
617 TenseurHHBB & operator - ( const TenseurHHBB & ) const ;
618 void operator -= ( const TenseurHHBB & );
619 TenseurHHBB & operator = ( const TenseurHHBB & );
620 TenseurHHBB & operator * ( const double & ) const ;
621 void operator *= ( const double & );
622 TenseurHHBB & operator / ( const double & ) const ;
623 void operator /= ( const double & );
624
625 // produit contracte à droite avec un tenseur du second ordre
626 // différent à gauche !!
627 // A(i,j,k,l)*B(l,k)=A..B
628 // on commence par contracter l'indice du milieu puis externe
629 TenseurHH& operator && ( const TenseurHH & ) const ;
630 // contraction verticale: A(i,j,k,l)*B(k,l)=A:B
631 TenseurHH& ContractionVerticale( const TenseurHH & ) const ;
632
633 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
634 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
635 // A(i,j,k,l)*B(l,k,o,p)=A...B
636 TenseurHHHH& operator && ( const TenseurHHHH & ) const ;
637 TenseurHHBB& operator && ( const TenseurHHBB & ) const ;
638
639 //fonctions définissant le produit tensoriel normal de deux tenseurs
640 // *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
641 static TenseurHHBB & Prod_tensoriel(const TenseurHH & aHH, const TenseurBB & bBB) ;
642
643 //fonctions définissant le produit tensoriel barre de deux tenseurs
644 // concernant les symétries !!
645 // *this(i,j,k,l)
646 // = 1/4 * (a(i,k).b(j,l) + a(j,k).b(i,l) + a(i,l).b(j,k) + a(j,l).b(i,k))
647 static TenseurHHBB & Prod_tensoriel_barre(const TenseurHH & aHH, const TenseurBB & bBB) ;
648
649 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
650 // les 2 premiers indices sont échangés avec les deux derniers indices
651 TenseurBBHH & Transposelet2avec3et4() const ;
652
653 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
654 // plusZero = true: les données manquantes sont mises à 0
655 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
656 // des données possibles
657 virtual void Affectation_trans_dimension(const TenseurHHBB & B,bool plusZero);
658
659 // test
660 int operator == ( const TenseurHHBB & ) const ;
661
662 // Retourne la composante i,j,k,l du tenseur
663 // acces en ecriture,
664 void Change (int i, int j, int k, int l,const double& val) ;
665 // en cumul : équivalent de +=
666 void ChangePlus (int i, int j, int k, int l,const double& val);
667
668 // Retourne la composante i,j,k,l du tenseur
669 // acces en lecture seule
670 double operator () (int i, int j, int k, int l) const ;
671
672 // calcul du maximum en valeur absolu des composantes du tenseur
673 double MaxiComposante() const;
674
675 // lecture et écriture de données
676 istream & Lecture(istream & entree);
677 ostream & Ecriture(ostream & sort) const ;
678
679 // affichage sous forme de tableau bidim
680 void Affiche_bidim(ostream & sort) const ;
681
682 protected :
683 // allocator dans la liste de data
684 listdouble36Iter ipointe;
685 // --- gestion de changement d'index ----
686 class ChangementIndex
687 { public:
688     ChangementIndex();
689     // passage pour les index de la forme vecteur à la forme i,j
690     Tableau <int> idx_i,idx_j;
691     // passage pour les index de la forme i,j à la forme vecteur
692     Tableau2 <int> odVect;
693 };
694 public :
695     static const ChangementIndex cdex3HHBB;
696
697     // fonction pour le produit contracté à gauche
698     TenseurBB& Prod_gauche( const TenseurBB & F) const;
699 };

```

```

700
701 #ifndef MISE_AU_POINT
702 #include "TenseurQ3-1.cc"
703 #include "TenseurQ3-2.cc"
704 #define TenseurQ3_H_deja_inclus
705 #endif
706
707
708 #endif

```

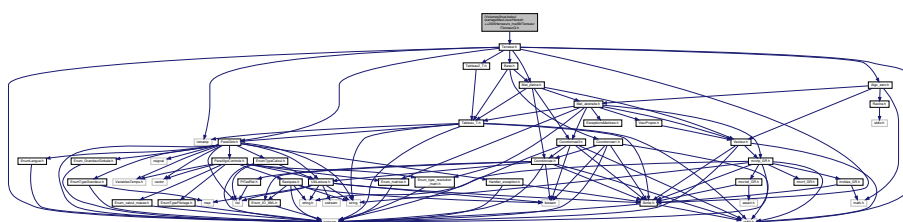
## 7.473 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/tenseurs\_mai99/Tenseur/TenseurQ.h

```

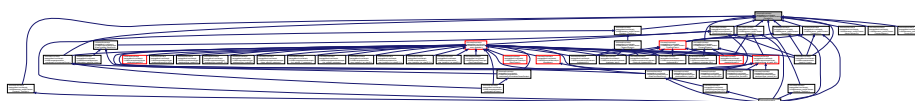
#include "Tenseur.h"
#include "TenseurQ.cc"

```

Graphe des dépendances par inclusion de TenseurQ.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Classes

- class [TenseurHHHH](#)  
*cas des composantes 4 fois contravariantes*
- class [LesMaillonsHHHH](#)  
*def d'un maillon de liste chaine pour memoriser les differents tenseurs intermediaires*
- class [TenseurBBBB](#)  
*cas des composantes 4 fois covariantes*
- class [LesMaillonsBBBB](#)  
*def d'un maillon de liste chaine pour memoriser les differents tenseurs intermediaires*
- class [TenseurHHBB](#)  
*cas des composantes mixte HHBB*
- class [LesMaillonsHHBB](#)  
*def d'un maillon de liste chaine pour memoriser les differents tenseurs intermediaires*
- class [TenseurBBHH](#)  
*cas des composantes mixte BBHH*
- class [LesMaillonsBBHH](#)  
*def d'un maillon de liste chaine pour memoriser les differents tenseurs intermediaires*
- class [TenseurHBHB](#)  
*cas des composantes 2 fois mixtes HBHB*
- class [LesMaillonsHBHB](#)  
*def d'un maillon de liste chaine pour memoriser les differents tenseurs intermediaires*
- class [TenseurBHBH](#)  
*cas des composantes 2 fois mixtes BHBH*
- class [LesMaillonsBHBH](#)  
*def d'un maillon de liste chaine pour memoriser les differents tenseurs intermediaires*

- class [TenseurHBBH](#)  
cas des composantes 2 fois mixtes HBBH
- class [LesMaillonsHBBH](#)  
def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
- class [TenseurBHHB](#)  
cas des composantes 2 fois mixtes BHHB
- class [LesMaillonsBHHB](#)  
def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires

## Fonctions

- void [LibereTenseurQ](#) ()  
Résolution du probleme de gestion de la definition de tenseurs supplementaires lors d'ecriture de grandes expressions. Il est necessaire de liberer l'espace apres les calculs.

### 7.473.1 Description détaillée

Définition des tenseurs d'ordre 4: méthodes strictement virtuelles

### 7.473.2 Documentation des fonctions

#### 7.473.2.1 LibereTenseurQ()

```
void LibereTenseurQ ( )
```

Résolution du probleme de gestion de la definition de tenseurs supplementaires lors d'ecriture de grandes expressions. Il est necessaire de liberer l'espace apres les calculs.

### 7.473.3 la fonction libereTenseur libere tout l'espace de tous les types de tenseurs

## 7.474 TenseurQ.h

[Aller à la documentation de ce fichier.](#)

```
1 /** \file TenseurQ.h
2 * Définition des tenseurs d'ordre 4: méthodes strictement virtuelles
3 */
4
5
6 // This file is part of the Herezh++ application.
7 //
8 // The finite element software Herezh++ is dedicated to the field
9 // of mechanics for large transformations of solid structures.
10 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
11 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
12 //
13 // Herezh++ is distributed under GPL 3 license ou ultérieure.
14 //
15 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
16 // AUTHOR : Gérard Rio
17 // E-MAIL : gerardrio56@free.fr
18 //
19 // This program is free software: you can redistribute it and/or modify
20 // it under the terms of the GNU General Public License as published by
21 // the Free Software Foundation, either version 3 of the License,
22 // or (at your option) any later version.
23 //
24 // This program is distributed in the hope that it will be useful,
25 // but WITHOUT ANY WARRANTY; without even the implied warranty
26 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
27 // See the GNU General Public License for more details.
28 //
29 // You should have received a copy of the GNU General Public License
30 // along with this program. If not, see <https://www.gnu.org/licenses/>.
31 //
32 // For more information, please consult: <https://herezh.irdl.fr/>.
33
34
35
36 /*****
```

```

37 *      DATE:          2/5/2002                *
38 *                                                    $ *
39 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr) *
40 *                                                    $ *
41 *      PROJET:        Herezh++                *
42 *                                                    $ *
43 *****
44 *      BUT:           Définir les tenseurs d'ordre 4 de différentes composantes.*
45 *                   La classe est virtuelle pure. *
46 *                   Elle se décline en fonction de la dimension du problème *
47 *                   L'objectif principal est de surcharger les différentes *
48 *                   opérations classiques. *
49 *                                                    $ *
50 *                   ***** *
51 *****/
52
53 #ifndef TENSEURQ_H
54 #define TENSEURQ_H
55
56 #include "Tenseur.h"
57
58
59 /** @defgroup Les_classes_tenseurs_virtuelles_ordre4
60 *
61 *      BUT:           Définir les tenseurs d'ordre 4 de différentes composantes.
62 *                   Les classes sont virtuelles pures.
63 *                   Elles se déclinent en fonction de la dimension du problème.
64 *                   L'objectif principal est de surcharger les différentes opérations
65 *                   classiques.
66 *
67 * \author           Gérard Rio
68 * \version          1.0
69 * \date             2/5/2002
70 * \brief            Définition des classes virtuelles pures de type Tenseur d'ordre 4, en coordonnées avec
                    différentes variances.
71 *
72 */
73
74
75
76 class TenseurBBBB; // def anticipée
77 class TenseurHHBB; // pour l'utilisation dans les produits contractés
78 class TenseurBBHH; //
79
80 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre4
81 /// @{
82 //-----
83 ///          cas des composantes 4 fois contravariantes
84 //-----
85 /// \author      Gérard Rio
86 /// \version     1.0
87 /// \date        2/5/2002
88 class TenseurHHHH
89 {
90     friend TenseurHHHH & operator * (double r, const TenseurHHHH & t)
91     { return (t*r); };
92     // produit contracté à gauche avec un tenseur d'ordre 2
93     // différent de à droite
94     friend TenseurHH & operator && ( const TenseurBB & F , const TenseurHHHH & T)
95     { return T.Prod_gauche(F); }
96
97 public :
98     // DESTRUCTEUR :
99     virtual ~TenseurHHHH() {};
100
101     // METHODES PUBLIQUES :
102 //1) non virtuelles
103     int Dimension() const { return dimension; }; // retourne la dimension du tenseur
104 //2) virtuelles
105     // initialise toutes les composantes à val
106     virtual void InitA(double val) = 0;
107     // opérations
108     virtual TenseurHHHH & operator + ( const TenseurHHHH & ) const = 0;
109     virtual void operator += ( const TenseurHHHH & ) = 0;
110     virtual TenseurHHHH & operator - ( ) const = 0; // oppose du tenseur
111     virtual TenseurHHHH & operator - ( const TenseurHHHH & ) const = 0;
112     virtual void operator -= ( const TenseurHHHH & ) = 0;
113     virtual TenseurHHHH & operator = ( const TenseurHHHH & ) = 0;
114     virtual TenseurHHHH & operator * (const double & ) const = 0 ;
115     virtual void operator *= ( const double & ) = 0;
116     virtual TenseurHHHH & operator / ( const double & ) const = 0;
117     virtual void operator /= ( const double & ) = 0;
118
119     // produit contracté à droite avec un tenseur du second ordre
120     // (différent de à gauche !!)
121     virtual TenseurHH & operator && ( const TenseurBB & ) const = 0 ;
122

```

```

123 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
124 // les 2 premiers indices sont échangés avec les deux derniers indices
125 virtual TenseurHHHH & Transposelet2avec3et4() const = 0;
126
127 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
128 // plusZero = true: les données manquantes sont mises à 0
129 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
130 // des données possibles
131 virtual void Affectation_trans_dimension(const TenseurHHHH & B,bool plusZero) = 0;
132
133 // test
134 virtual int operator == ( const TenseurHHHH &) const = 0;
135 int operator != ( const TenseurHHHH & a) const
136     { return !(*this == a);};
137
138 // Retourne la composante i,j,k,l du tenseur
139 // acces en ecriture,
140 virtual void Change (int i, int j, int k, int l,const double& val) =0;
141 // en cumul : équivalent de +=
142 virtual void ChangePlus (int i, int j, int k, int l,const double& val) =0;
143
144 // Retourne la composante i,j,k,l du tenseur
145 // acces en lecture seule
146 virtual double operator () (int i, int j, int k, int l) const =0;
147
148 // calcul du maximum en valeur absolu des composantes du tenseur
149 virtual double MaxiComposante() const = 0;
150
151 // // conversion de type
152 // operator TenseurHHHH & (void)
153 //     {cout << " appel de la conversion de type TenseurHHHH\n";
154 //     return *this;};
155
156 // lecture et écriture de données
157 virtual istream & Lecture(istream & entree) = 0;
158 virtual ostream & Ecriture(ostream & sort) const = 0;
159
160 // protected :
161
162 // variables
163 int dimension; // dimension du tenseur
164 double * t; // pointeur des données
165
166 protected :
167
168 // sortie d'un message standard
169 // dim = dimension du tenseur argument
170 void Message(int dim, string mes) const ;
171
172 // fonction pour le produit contracté à gauche
173 virtual TenseurHH & Prod_gauche( const TenseurBB & F) const = 0;
174
175 };
176 /// @} // end of group
177
178
179 /** @defgroup Les_classes_Maillons_tenseurs_ordre4
180 *
181 * BUT: On gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsXXXXX"
182 * qui stocke les pointeurs sur les tenseurs intermédiaire d'ordre 4
183 *
184 * \author Gérard Rio
185 * \version 1.0
186 * \date 2/5/2002
187 * \brief Définition des classes qui stockent les pointeurs sur les tenseurs intermédiaire d'ordre
188 * 4.
189 */
190
191 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsHHHHH"
192 // qui stocke les pointeurs sur les tenseurs intermédiaire
193
194 class PtTenseurHHHH; // def de la liste chainee des tenseurs intermediaires
195
196 /// @addtogroup Les_classes_Maillons_tenseurs_ordre4
197 /// @{
198
199 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
200 class LesMaillonsHHHH
201 { public :
202     ///liberation de la place occupee par des tenseurs crees pour les
203     /// operations intermediaires
204     static void Libere();
205     /// enregistrement de l'ajout d'un tenseur
206     static void NouveauMaillon(const TenseurHHHH *); // enregistrement de l'ajout d'un tenseur
207
208     /// dernier maillon

```

```

209     static PtTenseurHHHH * maille ;// dernier maillon
210     #ifdef MISE_AU_POINT
211         // nombre de maillon courant sauvegarde
212         static int nbmailHHHH; // nombre de maillon courant sauvegarde
213     #endif
214 };
215 /// @} // end of group
216
217 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre4
218 /// @{
219 //-----
220 //          cas des composantes 4 fois covariantes
221 //-----
222 /// \author   Gérard Rio
223 /// \version  1.0
224 /// \date    2/5/2002
225 class TenseurBBBB
226 { friend TenseurBBBB & operator * (double r, const TenseurBBBB & t)
227   { return ( t*r); };
228   // produit contracte à gauche avec un tenseur d'ordre 2
229   // différent de à droite
230   friend TenseurBB& operator && ( const TenseurHH & F , const TenseurBBBB & T)
231   { return T.Prod_gauche(F);}
232
233 public :
234     // DESTRUCTEUR :
235     virtual ~TenseurBBBB() {};
236
237     // METHODES PUBLIQUES :
238     //1) non virtuelles
239     int Dimension() const{ return dimension;}; // retourne la dimension du tenseur
240     //2) virtuelles
241     // initialise toutes les composantes à val
242     virtual void Inita(double val) = 0;
243     // operations
244     virtual TenseurBBBB & operator + ( const TenseurBBBB &) const = 0;
245     virtual void operator += ( const TenseurBBBB &) = 0;
246     virtual TenseurBBBB & operator - () const = 0; // oppose du tenseur
247     virtual TenseurBBBB & operator - ( const TenseurBBBB &) const = 0;
248     virtual void operator -= ( const TenseurBBBB &) = 0;
249     virtual TenseurBBBB & operator = ( const TenseurBBBB &) = 0;
250     virtual TenseurBBBB & operator * (const double &) const = 0 ;
251     virtual void operator *= ( const double &) = 0;
252     virtual TenseurBBBB & operator / ( const double &) const = 0;
253     virtual void operator /= ( const double &) = 0;
254
255     // produit contracte à droite avec un tenseur du second ordre
256     // (différent de à gauche !!)
257     virtual TenseurBB& operator && ( const TenseurHH & ) const = 0 ;
258
259     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
260     // les 2 premiers indices sont échangés avec les deux derniers indices
261     virtual TenseurBBBB & Transposelet2avec3et4() const = 0;
262
263     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
264     // plusZero = true: les données manquantes sont mises à 0
265     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
266     // des données possibles
267     virtual void Affectation_trans_dimension(const TenseurBBBB & B,bool plusZero) = 0;
268
269     // test
270     virtual int operator == ( const TenseurBBBB &) const = 0;
271     int operator != ( const TenseurBBBB & a) const
272     { return !(*this == a);};
273
274     // Retourne la composante i,j,k,l du tenseur
275     // acces en ecriture,
276     virtual void Change (int i, int j, int k, int l,const double& val) =0;
277     // en cumul : équivalent de +=
278     virtual void ChangePlus (int i, int j, int k, int l,const double& val) =0;
279
280     // Retourne la composante i,j,k,l du tenseur
281     // acces en lecture seule
282     virtual double operator () (int i, int j, int k, int l) const =0;
283
284     // calcul du maximum en valeur absolu des composantes du tenseur
285     virtual double MaxiComposante() const = 0;
286 //
287 //     // conversion de type
288 //     operator TenseurBBBB & (void)
289 //     {cout << " appel de la conversion de type TenseurBBBB\n";
290 //     return *this;};
291
292     // lecture et écriture de données
293     virtual istream & Lecture(istream & entree) = 0;
294     virtual ostream & Ecriture(ostream & sort) const = 0;
295

```

```

296 // protected :
297
298 // variables
299 int dimension; // dimension du tenseur
300 double * t; // pointeur des données
301
302 protected :
303
304 // sortie d'un message standard
305 // dim = dimension du tenseur argument
306 void Message(int dim, string mes) const ;
307
308 // fonction pour le produit contracté à gauche
309 virtual TenseurBB& Prod_gauche( const TenseurHH & F) const = 0;
310 };
311 /// @} // end of group
312
313 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsBBBB"
314 // qui stocke les pointeurs sur les tenseurs intermédiaire
315
316 class PtTenseurBBBB; // def de la liste chainee des tenseurs intermediaires
317
318 /// @addtogroup Les_classes_Maillons_tenseurs_ordre4
319 /// @{
320
321 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
322 class LesMaillonsBBBB
323 { public :
324     ///liberation de la place occupee par des tenseurs crees pour les
325     /// operations intermediaires
326     static void Libere();
327     /// enregistrement de l'ajout d'un tenseur
328     static void NouveauMaillon(const TenseurBBBB *); // enregistrement de l'ajout d'un tenseur
329
330     /// dernier maillon
331     static PtTenseurBBBB * maille ;// dernier maillon
332     #ifndef MISE_AU_POINT
333     /// nombre de maillon courant sauvegarde
334     static int nbmailBBBB; // nombre de maillon courant sauvegarde
335     #endif
336 };
337 /// @} // end of group
338
339 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre4
340 /// @{
341 //-----
342 /// cas des composantes mixte HHBB
343 //-----
344 /// \author Gérard Rio
345 /// \version 1.0
346 /// \date 2/5/2002
347 class TenseurHHBB
348 { friend TenseurHHBB & operator * (double r, const TenseurHHBB & t)
349     { return ( t*r); };
350     // produit contracte à gauche avec un tenseur d'ordre 2
351     // différent de à droite
352     friend TenseurBB & operator && ( const TenseurBB & F , const TenseurHHBB & T)
353     { return T.Prod_gauche(F); }
354
355 public :
356     // DESTRUCTEUR :
357     virtual ~TenseurHHBB() {};
358
359     // METHODES PUBLIQUES :
360     //1) non virtuelles
361     int Dimension() const{ return dimension;}; // retourne la dimension du tenseur
362     //2) virtuelles
363     // initialise toutes les composantes à val
364     virtual void Inita(double val) = 0;
365     // operations
366     virtual TenseurHHBB & operator + ( const TenseurHHBB &) const = 0;
367     virtual void operator += ( const TenseurHHBB &) = 0;
368     virtual TenseurHHBB & operator - () const = 0; // oppose du tenseur
369     virtual TenseurHHBB & operator - ( const TenseurHHBB &) const = 0;
370     virtual void operator -= ( const TenseurHHBB &) = 0;
371     virtual TenseurHHBB & operator = ( const TenseurHHBB &) = 0;
372     virtual TenseurHHBB & operator * (const double &) const = 0 ;
373     virtual void operator *= ( const double &) = 0;
374     virtual TenseurHHBB & operator / ( const double &) const = 0;
375     virtual void operator /= ( const double &) = 0;
376
377     // produit contracte à droite avec un tenseur du second ordre
378     // (différent de à gauche !!)
379     virtual TenseurHH& operator && ( const TenseurHH & ) const = 0 ;
380
381     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
382     // les 2 premiers indices sont échangés avec les deux derniers indices

```

```

383     virtual TenseurBBHH & Transposelet2avec3et4() const = 0;
384
385     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
386     // plusZero = true: les données manquantes sont mises à 0
387     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
388     // des données possibles
389     virtual void Affectation_trans_dimension(const TenseurHHBB & B,bool plusZero) = 0;
390
391     // test
392     virtual int operator == ( const TenseurHHBB &) const = 0;
393     int operator != ( const TenseurHHBB & a) const
394     { return !(*this == a);};
395
396     // Retourne la composante i,j,k,l du tenseur
397     // acces en ecriture,
398     virtual void Change (int i, int j, int k, int l,const double& val) =0;
399     // en cumul : équivalent de +=
400     virtual void ChangePlus (int i, int j, int k, int l,const double& val) =0;
401
402     // Retourne la composante i,j,k,l du tenseur
403     // acces en lecture seule
404     virtual double operator () (int i, int j, int k, int l) const =0;
405
406     // calcul du maximum en valeur absolu des composantes du tenseur
407     virtual double MaxiComposante() const = 0;
408
409     // // conversion de type
410     // operator TenseurHHBB & (void)
411     //     {cout << " appel de la conversion de type TenseurHHBB\n";
412     //     return *this;};
413
414     // lecture et écriture de données
415     virtual istream & Lecture(istream & entree) = 0;
416     virtual ostream & Ecriture(ostream & sort) const = 0;
417
418     // protected :
419
420     // variables
421     int dimension; // dimension du tenseur
422     double * t; // pointeur des données
423
424     protected :
425
426     // sortie d'un message standard
427     // dim = dimension du tenseur argument
428     void Message(int dim, string mes) const ;
429
430     // fonction pour le produit contracté à gauche
431     virtual TenseurBB& Prod_gauche( const TenseurBB & F) const = 0;
432 };
433 /// @} // end of group
434
435 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsHHBB"
436 // qui stocke les pointeurs sur les tenseurs intermédiaire
437
438 class PtTenseurHHBB; // def de la liste chainee des tenseurs intermediaires
439 /// @addtogroup Les_classes_Maillons_tenseurs_ordre4
440 /// @{
441
442 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
443 class LesMaillonsHHBB
444 { public :
445     ///liberation de la place occupee par des tenseurs crees pour les
446     /// operations intermediaires
447     static void Libere();
448     /// enregistrement de l'ajout d'un tenseur
449     static void NouveauMaillon(const TenseurHHBB *); // enregistrement de l'ajout d'un tenseur
450
451     /// dernier maillon
452     static PtTenseurHHBB * maille ;// dernier maillon
453     #ifndef MISE_AU_POINT
454         /// nombre de maillon courant sauvegarde
455         static int nbmailHHBB; // nombre de maillon courant sauvegarde
456     #endif
457 };
458 /// @} // end of group
459
460 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre4
461 /// @{
462 //-----
463 // cas des composantes mixte BBHH
464 //-----
465 /// \author Gérard Rio
466 /// \version 1.0
467 /// \date 2/5/2002
468 class TenseurBBHH
469 { friend TenseurBBHH & operator * (double r, const TenseurBBHH & t)

```



```

470     { return ( t*r); };
471 // produit contracte à gauche avec un tenseur d'ordre 2
472 // différent de à droite
473 friend TenseurHH & operator && ( const TenseurHH & F , const TenseurBBHH & T)
474     { return T.Prod_gauche(F);}
475
476 public :
477 // DESTRUCTEUR :
478 virtual ~TenseurBBHH() {};
479
480 // METHODES PUBLIQUES :
481 //1) non virtuelles
482 int Dimension() const{ return dimension;}; // retourne la dimension du tenseur
483 //2) virtuelles
484 // initialise toutes les composantes à val
485 virtual void Inita(double val) = 0;
486 // operations
487 virtual TenseurBBHH & operator + ( const TenseurBBHH &) const = 0;
488 virtual void operator += ( const TenseurBBHH &) = 0;
489 virtual TenseurBBHH & operator - ( const TenseurBBHH &) const = 0; // oppose du tenseur
490 virtual TenseurBBHH & operator - ( const TenseurBBHH &) const = 0;
491 virtual void operator -= ( const TenseurBBHH &) = 0;
492 virtual TenseurBBHH & operator = ( const TenseurBBHH &) = 0;
493 virtual TenseurBBHH & operator * (const double &) const = 0 ;
494 virtual void operator *= ( const double &) = 0;
495 virtual TenseurBBHH & operator / ( const double &) const = 0;
496 virtual void operator /= ( const double &) = 0;
497
498 // produit contracte à droite avec un tenseur du second ordre
499 // (différent de à gauche !!)
500 virtual TenseurBB& operator && ( const TenseurBB & ) const = 0 ;
501
502 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
503 // les 2 premiers indices sont échangés avec les deux derniers indices
504 virtual TenseurHHBB & Transposelet2avec3et4() const = 0;
505
506 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
507 // plusZero = true: les données manquantes sont mises à 0
508 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
509 // des données possibles
510 virtual void Affectation_trans_dimension(const TenseurBBHH & B,bool plusZero) = 0;
511
512 // test
513 virtual int operator == ( const TenseurBBHH &) const = 0;
514 int operator != ( const TenseurBBHH & a) const
515     { return !(*this == a);};
516
517 // Retourne la composante i,j,k,l du tenseur
518 // acces en ecriture,
519 virtual void Change (int i, int j, int k, int l,const double& val) =0;
520 // en cumul : équivalent de +=
521 virtual void ChangePlus (int i, int j, int k, int l,const double& val) =0;
522
523 // Retourne la composante i,j,k,l du tenseur
524 // acces en lecture seule
525 virtual double operator () (int i, int j, int k, int l) const =0;
526
527 // calcul du maximum en valeur absolu des composantes du tenseur
528 virtual double MaxiComposante() const = 0;
529
530 // // conversion de type
531 // operator TenseurBBHH & (void)
532 // {cout << " appel de la conversion de type TenseurBBHH\n";
533 // return *this;};
534
535 // lecture et écriture de données
536 virtual istream & Lecture(istream & entree) = 0;
537 virtual ostream & Ecriture(ostream & sort) const = 0;
538
539 // protected :
540
541 // variables
542 int dimension; // dimension du tenseur
543 double * t; // pointeur des données
544
545 protected :
546
547 // sortie d'un message standard
548 // dim = dimension du tenseur argument
549 void Message(int dim, string mes) const ;
550
551 // fonction pour le produit contracté à gauche
552 virtual TenseurHH& Prod_gauche( const TenseurHH & F) const = 0;
553 };
554 /// @} // end of group
555
556 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsBBHH"

```

```

557 // qui stocke les pointeurs sur les tenseurs intermédiaire
558
559 class PtTenseurBBHH; // def de la liste chainee des tenseurs intermediaires
560 /// @addtogroup Les_classes_Maillons_tenseurs_ordre4
561 /// @{
562
563 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
564 class LesMaillonsBBHH
565 { public :
566     ///liberation de la place occupee par des tenseurs crees pour les
567     /// operations intermediaires
568     static void Libere();
569     /// enregistrement de l'ajout d'un tenseur
570     static void NouveauMaillon(const TenseurBBHH *); // enregistrement de l'ajout d'un tenseur
571
572     /// dernier maillon
573     static PtTenseurBBHH * maille ;// dernier maillon
574 #ifndef MISE_AU_POINT
575     /// nombre de maillon courant sauvegarde
576     static int nbmailBBHH; // nombre de maillon courant sauvegarde
577 #endif
578 };
579 /// @} // end of group
580
581 class TenseurBBHH;
582
583 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre4
584 /// @{
585 //-----
586 ///          cas des composantes 2 fois mixtes HBHB
587 //-----
588 /// \author   Gérard Rio
589 /// \version  1.0
590 /// \date     2/5/2002
591 class TenseurHBHB
592 { friend TenseurHBHB & operator * (double r, const TenseurHBHB & t)
593   { return ( t*r); };
594   // produit contracte à gauche avec un tenseur d'ordre 2
595   // différent de à droite resultat(k,l)=F(i,j) T(j,l,k,l)
596   friend TenseurHB & operator && ( const TenseurHB & F , const TenseurHBHB & T)
597   { return T.Prod_gauche(F);}
598
599 public :
600     // DESTRUCTEUR :
601     virtual ~TenseurHBHB() {};
602
603     // METHODES PUBLIQUES :
604     //1) non virtuelles
605     int Dimension() const{ return dimension;}; // retourne la dimension du tenseur
606     //2) virtuelles
607     // initialise toutes les composantes à val
608     virtual void Inita(double val) = 0;
609     // operations
610     virtual TenseurHBHB & operator + ( const TenseurHBHB &) const = 0;
611     virtual void operator += ( const TenseurHBHB &) = 0;
612     virtual TenseurHBHB & operator - () const = 0; // oppose du tenseur
613     virtual TenseurHBHB & operator - ( const TenseurHBHB &) const = 0;
614     virtual void operator -= ( const TenseurHBHB &) = 0;
615     virtual TenseurHBHB & operator = ( const TenseurHBHB &) = 0;
616     virtual TenseurHBHB & operator * (const double &) const = 0 ;
617     virtual void operator *= ( const double &) = 0;
618     virtual TenseurHBHB & operator / ( const double &) const = 0;
619     virtual void operator /= ( const double &) = 0;
620
621     // produit contracte à droite avec un tenseur du second ordre
622     // (différent de à gauche !!)
623     virtual TenseurHB& operator && ( const TenseurHB & ) const = 0 ;
624
625     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
626     // les 2 premiers indices sont échangés avec les deux derniers indices
627     virtual TenseurBBHH & Transposelet2avec3et4() const = 0;
628
629     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
630     // plusZero = true: les données manquantes sont mises à 0
631     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
632     // des données possibles
633     virtual void Affectation_trans_dimension(const TenseurHBHB & B,bool plusZero) = 0;
634
635     // test
636     virtual int operator == ( const TenseurHBHB &) const = 0;
637     int operator != ( const TenseurHBHB & a) const
638     { return !(*this == a);};
639
640     // Retourne la composante i,j,k,l du tenseur
641     // acces en ecriture,
642     virtual void Change (int i, int j, int k, int l,const double& val) =0;
643     // en cumul : équivalent de +=

```

```

644     virtual void ChangePlus (int i, int j, int k, int l, const double& val) =0;
645
646     // Retourne la composante i,j,k,l du tenseur
647     // acces en lecture seule
648     virtual double operator () (int i, int j, int k, int l) const =0;
649
650     // calcul du maximum en valeur absolu des composantes du tenseur
651     virtual double MaxiComposante() const = 0;
652
653     // // conversion de type
654     // operator TenseurHBHB & (void)
655     // {cout << " appel de la conversion de type TenseurHBHB\n";
656     //     return *this;};
657
658     // lecture et écriture de données
659     virtual istream & Lecture(istream & entree) = 0;
660     virtual ostream & Ecriture(ostream & sort) const = 0;
661
662     // protected :
663
664     // variables
665     int dimension; // dimension du tenseur
666     double * t; // pointeur des données
667
668     protected :
669
670     // sortie d'un message standard
671     // dim = dimension du tenseur argument
672     void Message(int dim, string mes) const ;
673
674     // fonction pour le produit contracté à gauche
675     virtual TenseurHB & Prod_gauche( const TenseurHB & F) const = 0;
676 };
677 /// @} // end of group
678
679 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsHBHB"
680 // qui stocke les pointeurs sur les tenseurs intermédiaire
681
682 class PtTenseurHBHB; // def de la liste chaineé des tenseurs intermediaires
683 /// @addtogroup Les_classes_Maillons_tenseurs_ordre4
684 /// @{
685
686 /// def d'un maillon de liste chaineé pour memoriser les differents tenseurs intermediaires
687 class LesMaillonsHBHB
688 { public :
689     ///liberation de la place occupee par des tenseurs creés pour les
690     /// operations intermediaires
691     static void Libere();
692     /// enregistrement de l'ajout d'un tenseur
693     static void NouveauMaillon(const TenseurHBHB *); // enregistrement de l'ajout d'un tenseur
694
695     /// dernier maillon
696     static PtTenseurHBHB * maille ;// dernier maillon
697 #ifdef MISE_AU_POINT
698     /// nombre de maillon courant sauvegarde
699     static int nbmailHBHB; // nombre de maillon courant sauvegarde
700 #endif
701 };
702 /// @} // end of group
703
704 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre4
705 /// @{
706 //-----
707 ///          cas des composantes 2 fois mixtes BHBH
708 //-----
709 /// \author   Gérard Rio
710 /// \version  1.0
711 /// \date     2/5/2002
712 class TenseurBHBH
713 { friend TenseurBHBH & operator * (double r, const TenseurBHBH & t)
714   { return ( t*r); };
715   // produit contracte à gauche avec un tenseur d'ordre 2
716   // différent de à droite resultat(k,l)=F(i,j) T(j,i,k,l)
717   friend TenseurBH & operator && ( const TenseurBH & F , const TenseurBHBH & T)
718   { return T.Prod_gauche(F);}
719
720 public :
721     // DESTRUCTEUR :
722     virtual ~TenseurBHBH() {};
723
724     // METHODES PUBLIQUES :
725     //1) non virtuelles
726     int Dimension() const{ return dimension;}; // retourne la dimension du tenseur
727     //2) virtuelles
728     // initialise toutes les composantes à val
729     virtual void InitA(double val) = 0;
730     // operations

```

```

731 virtual TenseurBHHB & operator + ( const TenseurBHHB & ) const = 0;
732 virtual void operator += ( const TenseurBHHB & ) = 0;
733 virtual TenseurBHHB & operator - ( ) const = 0; // oppose du tenseur
734 virtual TenseurBHHB & operator - ( const TenseurBHHB & ) const = 0;
735 virtual void operator -= ( const TenseurBHHB & ) = 0;
736 virtual TenseurBHHB & operator = ( const TenseurBHHB & ) = 0;
737 virtual TenseurBHHB & operator * (const double & ) const = 0 ;
738 virtual void operator *= ( const double & ) = 0;
739 virtual TenseurBHHB & operator / ( const double & ) const = 0;
740 virtual void operator /= ( const double & ) = 0;
741
742 // produit contracte à droite avec un tenseur du second ordre
743 // (différent de à gauche !!)
744 virtual TenseurBH & operator && ( const TenseurBH & ) const = 0 ;
745
746 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
747 // les 2 premiers indices sont échangés avec les deux derniers indices
748 virtual TenseurHBHHB & Transposelet2avec3et4() const = 0;
749
750 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
751 // plusZero = true: les données manquantes sont mises à 0
752 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
753 // des données possibles
754 virtual void Affectation_trans_dimension(const TenseurBHHB & B,bool plusZero) = 0;
755
756 // test
757 virtual int operator == ( const TenseurBHHB & ) const = 0;
758 int operator != ( const TenseurBHHB & a ) const
759     { return !(*this == a);};
760
761 // Retourne la composante i,j,k,l du tenseur
762 // acces en ecriture,
763 virtual void Change (int i, int j, int k, int l,const double& val) =0;
764 // en cumul : équivalent de +=
765 virtual void ChangePlus (int i, int j, int k, int l,const double& val) =0;
766
767 // Retourne la composante i,j,k,l du tenseur
768 // acces en lecture seule
769 virtual double operator () (int i, int j, int k, int l) const =0;
770
771 // calcul du maximum en valeur absolu des composantes du tenseur
772 virtual double MaxiComposante() const = 0;
773
774 // // conversion de type
775 // operator TenseurBHHB & (void)
776 // {cout << " appel de la conversion de type TenseurBHHB\n";
777 // return *this;};
778
779 // lecture et écriture de données
780 virtual istream & Lecture(istream & entree) = 0;
781 virtual ostream & Ecriture(ostream & sort) const = 0;
782
783 // protected :
784
785 // variables
786 int dimension; // dimension du tenseur
787 double * t; // pointeur des données
788
789 protected :
790
791 // sortie d'un message standard
792 // dim = dimension du tenseur argument
793 void Message(int dim, string mes) const ;
794
795 // fonction pour le produit contracté à gauche
796 virtual TenseurBH & Prod_gauche( const TenseurBH & F) const = 0;
797 };
798 /// @} // end of group
799
800 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsBHHB"
801 // qui stocke les pointeurs sur les tenseurs intermédiaire
802
803 class PtTenseurBHHB; // def de la liste chainee des tenseurs intermediaires
804 /// @addtogroup Les_classes_Maillons_tenseurs_ordre4
805 /// @{
806
807 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
808 class LesMaillonsBHHB
809 { public :
810     ///liberation de la place occupee par des tenseurs crees pour les
811     /// operations intermediaires
812     static void Libere();
813     /// enregistrement de l'ajout d'un tenseur
814     static void NouveauMaillon(const TenseurBHHB *); // enregistrement de l'ajout d'un tenseur
815
816     /// dernier maillon
817     static PtTenseurBHHB * maille ;// dernier maillon

```

```

818 #ifndef MISE_AU_POINT
819     // nombre de maillon courant sauvegarde
820     static int nbmailHBBH; // nombre de maillon courant sauvegarde
821 #endif
822 };
823 /// @} // end of group
824
825 class TenseurBHHB;
826
827 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre4
828 /// @{
829 //-----
830 ///          cas des composantes 2 fois mixtes HBBH
831 //-----
832 /// \author   Gérard Rio
833 /// \version  1.0
834 /// \date    2/5/2002
835 class TenseurHBBH
836 { friend TenseurHBBH & operator * (double r, const TenseurHBBH & t)
837   { return ( t*r); };
838   // produit contracte à gauche avec un tenseur d'ordre 2
839   // différent de à droite resultat(k,l)=F(i,j) T(j,i,k,l)
840   friend TenseurBH & operator && ( const TenseurHB & F , const TenseurHBBH & T)
841   { return T.Prod_gauche(F);}
842
843 public :
844     // DESTRUCTEUR :
845     virtual ~TenseurHBBH() {};
846
847     // METHODES PUBLIQUES :
848     //1) non virtuelles
849     int Dimension() const{ return dimension;}; // retourne la dimension du tenseur
850     //2) virtuelles
851     // initialise toutes les composantes à val
852     virtual void Inita(double val) = 0;
853     // operations
854     virtual TenseurHBBH & operator + ( const TenseurHBBH & ) const = 0;
855     virtual void operator += ( const TenseurHBBH & ) = 0;
856     virtual TenseurHBBH & operator - ( ) const = 0; // oppose du tenseur
857     virtual TenseurHBBH & operator - ( const TenseurHBBH & ) const = 0;
858     virtual void operator -= ( const TenseurHBBH & ) = 0;
859     virtual TenseurHBBH & operator = ( const TenseurHBBH & ) = 0;
860     virtual TenseurHBBH & operator * (const double & ) const = 0 ;
861     virtual void operator *= ( const double & ) = 0;
862     virtual TenseurHBBH & operator / ( const double & ) const = 0;
863     virtual void operator /= ( const double & ) = 0;
864
865     // produit contracte à droite avec un tenseur du second ordre
866     // (différent de à gauche !!)
867     virtual TenseurHB& operator && ( const TenseurBH & ) const = 0 ;
868
869     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
870     // les 2 premiers indices sont échangés avec les deux derniers indices
871     virtual TenseurBHHB & Transposelet2avec3et4() const = 0;
872
873     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
874     // plusZero = true: les données manquantes sont mises à 0
875     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
876     // des données possibles
877     virtual void Affectation_trans_dimension(const TenseurHBBH & B,bool plusZero) = 0;
878
879     // test
880     virtual int operator == ( const TenseurHBBH & ) const = 0;
881     int operator != ( const TenseurHBBH & a) const
882     { return !(*this == a);};
883
884     // Retourne la composante i,j,k,l du tenseur
885     // acces en ecriture,
886     virtual void Change (int i, int j, int k, int l,const double& val) =0;
887     // en cumul : équivalent de +=
888     virtual void ChangePlus (int i, int j, int k, int l,const double& val) =0;
889
890     // Retourne la composante i,j,k,l du tenseur
891     // acces en lecture seule
892     virtual double operator () (int i, int j, int k, int l) const =0;
893
894     // calcul du maximum en valeur absolu des composantes du tenseur
895     virtual double MaxiComposante() const = 0;
896
897     // conversion de type
898     operator TenseurHBBH & (void)
899     {cout << " appel de la conversion de type TenseurHBBH\n";
900     return *this;};
901
902     // lecture et écriture de données
903     virtual istream & Lecture(istream & entree) = 0;
904     virtual ostream & Ecriture(ostream & sort) const = 0;

```

```

905
906 // protected :
907
908 // variables
909 int dimension; // dimension du tenseur
910 double * t; // pointeur des données
911
912 protected :
913
914 // sortie d'un message standard
915 // dim = dimension du tenseur argument
916 void Message(int dim, string mes) const ;
917
918 // fonction pour le produit contracté à gauche
919 virtual TenseurBH& Prod_gauche( const TenseurHB & F) const = 0;
920 };
921 /// @} // end of group
922
923 // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsHBBH"
924 // qui stocke les pointeurs sur les tenseurs intermédiaire
925
926 class PtTenseurHBBH; // def de la liste chainee des tenseurs intermediaires
927 /// @addtogroup Les_classes_Maillons_tenseurs_ordre4
928 /// @{
929
930 /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
931 class LesMaillonsHBBH
932 { public :
933     ///liberation de la place occupee par des tenseurs crees pour les
934     /// operations intermediaires
935     static void Libere();
936     /// enregistrement de l'ajout d'un tenseur
937     static void NouveauMaillon(const TenseurHBBH *); // enregistrement de l'ajout d'un tenseur
938
939     /// dernier maillon
940     static PtTenseurHBBH * maille ;// dernier maillon
941 #ifndef MISE_AU_POINT
942     /// nombre de maillon courant sauvegarde
943     static int nbmailHBBH; // nombre de maillon courant sauvegarde
944 #endif
945 };
946 /// @} // end of group
947
948 /// @addtogroup Les_classes_tenseurs_virtuelles_ordre4
949 /// @{
950 //-----
951 // cas des composantes 2 fois mixtes BHHB
952 //-----
953 /// \author Gérard Rio
954 /// \version 1.0
955 /// \date 2/5/2002
956 class TenseurBHHB
957 { friend TenseurBHHB & operator * (double r, const TenseurBHHB & t)
958     { return ( t*r); } ;
959     // produit contracte à gauche avec un tenseur d'ordre 2
960     // différent de à droite resultat(k,l)=F(i,j) T(j,i,k,l)
961     friend TenseurHB & operator && ( const TenseurBH & F , const TenseurBHHB & T)
962     { return T.Prod_gauche(F);}
963
964 public :
965     // DESTRUCTEUR :
966     virtual ~TenseurBHHB() {};
967
968     // METHODES PUBLIQUES :
969     //1) non virtuelles
970     int Dimension() const{ return dimension;}; // retourne la dimension du tenseur
971     //2) virtuelles
972     // initialise toutes les composantes à val
973     virtual void Inita(double val) = 0;
974     // operations
975     virtual TenseurBHHB & operator + ( const TenseurBHHB &) const = 0;
976     virtual void operator += ( const TenseurBHHB &) = 0;
977     virtual TenseurBHHB & operator - () const = 0; // oppose du tenseur
978     virtual TenseurBHHB & operator - ( const TenseurBHHB &) const = 0;
979     virtual void operator -= ( const TenseurBHHB &) = 0;
980     virtual TenseurBHHB & operator = ( const TenseurBHHB &) = 0;
981     virtual TenseurBHHB & operator * (const double &) const = 0 ;
982     virtual void operator *= ( const double &) = 0;
983     virtual TenseurBHHB & operator / ( const double &) const = 0;
984     virtual void operator /= ( const double &) = 0;
985
986     // produit contracte à droite avec un tenseur du second ordre
987     // (différent de à gauche !!)
988     virtual TenseurBH& operator && ( const TenseurHB & ) const = 0 ;
989
990     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
991     // les 2 premiers indices sont échangés avec les deux derniers indices

```

```

992     virtual TenseurHBBH & Transposelet2avec3et4() const = 0;
993
994     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
995     // plusZero = true: les données manquantes sont mises à 0
996     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
997     // des données possibles
998     virtual void Affectation_trans_dimension(const TenseurBHHB & B,bool plusZero) = 0;
999
1000    // test
1001    virtual int operator == ( const TenseurBHHB &) const = 0;
1002    int operator != ( const TenseurBHHB & a) const
1003    { return !(*this == a);};
1004
1005    // Retourne la composante i,j,k,l du tenseur
1006    // acces en ecriture,
1007    virtual void Change (int i, int j, int k, int l,const double& val) =0;
1008    // en cumul : équivalent de +=
1009    virtual void ChangePlus (int i, int j, int k, int l,const double& val) =0;
1010
1011    // Retourne la composante i,j,k,l du tenseur
1012    // acces en lecture seule
1013    virtual double operator () (int i, int j, int k, int l) const =0;
1014
1015    // calcul du maximum en valeur absolu des composantes du tenseur
1016    virtual double MaxiComposante() const = 0;
1017
1018    // // conversion de type
1019    // operator TenseurBHHB & (void)
1020    // {cout << " appel de la conversion de type TenseurBHHB\n";
1021    // return *this;};
1022
1023    // lecture et écriture de données
1024    virtual istream & Lecture(istream & entree) = 0;
1025    virtual ostream & Ecriture(ostream & sort) const = 0;
1026
1027    // protected :
1028
1029    // variables
1030    int dimension; // dimension du tenseur
1031    double * t; // pointeur des données
1032
1033    protected :
1034
1035    // sortie d'un message standard
1036    // dim = dimension du tenseur argument
1037    void Message(int dim, string mes) const ;
1038
1039    // fonction pour le produit contracté à gauche
1040    virtual TenseurHB& Prod_gauche( const TenseurBH & F) const = 0;
1041    };
1042    /// @} // end of group
1043
1044    // on gère les tenseurs intermédiaires au travers d'une classe "LesMaillonsBHHB"
1045    // qui stocke les pointeurs sur les tenseurs intermédiaire
1046
1047    class PtTenseurBHHB; // def de la liste chainee des tenseurs intermediaires
1048    /// @addtogroup Les_classes_Maillons_tenseurs_ordre4
1049    /// @{
1050
1051    /// def d'un maillon de liste chainee pour memoriser les differents tenseurs intermediaires
1052    class LesMaillonsBHHB
1053    { public :
1054        ///liberation de la place occupee par des tenseurs crees pour les
1055        /// operations intermediaires
1056        static void Libere();
1057        /// enregistrement de l'ajout d'un tenseur
1058        static void NouveauMaillon(const TenseurBHHB *); // enregistrement de l'ajout d'un tenseur
1059
1060        /// dernier maillon
1061        static PtTenseurBHHB * maille ;// dernier maillon
1062        #ifndef MISE_AU_POINT
1063            /// nombre de maillon courant sauvegarde
1064            static int nbmailBHHB; // nombre de maillon courant sauvegarde
1065        #endif
1066    };
1067    /// @} // end of group
1068
1069    //=====
1070    /// Résolution du probleme de gestion de la definition de tenseurs supplementaires lors
1071    /// d'écriture de grandes expressions. Il est necessaire de liberer l'espace
1072    /// apres les calculs
1073    /// la fonction libereTenseur libere tout l'espace de tous les types de tenseurs
1074    //=====
1075
1076    void LibereTenseurQ() ;
1077
1078

```

```

1079 #ifndef MISE_AU_POINT
1080 #include "TenseurQ.cc"
1081 #define TenseurQ_H_deja_inclus
1082 #endif
1083
1084 #endif

```

## 7.475 TenseurQ1gene.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *          LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)          *
34 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex        *
35 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
36 *****/
37 *      DATE:          29/2/2004                                          *
38 *                                                              $      *
39 *      AUTEUR:        G RIO (mailto:gerard.rio@univ-ubs.fr)            *
40 *                    Tel 0297874571 fax : 02.97.87.45.72              *
41 *                                                              $      *
42 *      PROJET:        Herezh++                                          *
43 *                                                              $      *
44 *****/
45 *      BUT:           Definition d'une classe derivatee de tenseur du 4ieme ordre *
46 *                    de dimensionl, il s'agit ici d'une classe générale, sans *
47 *                    particularités: c-a-d 1 composantes. (car en 1D)    *
48 *                                                              $      *
49 *      *****                                                    *
50 *      *                                                              *
51 *      VERIFICATION:                                             *
52 *      ! date ! auteur ! but ! *
53 *      ----- *
54 *      ! ! ! ! ! *
55 *      ! ! ! ! ! $ *
56 *      *****                                                    *
57 *      MODIFICATIONS:                                             *
58 *      ! date ! auteur ! but ! *
59 *      ----- *
60 *      ! ! ! ! ! $ *
61 *****/
62 #ifndef TENSEURQ1GENE_H
63 #define TENSEURQ1GENE_H
64
65 #include <iostream>
66 #include "Tenseur1_TroisSym.h"
67
68 // Du au fait qu'en une dimension, la classe générale est identique aux classes particulières
69 // on utilise des équivalences
70 //-----
71 //          cas des composantes 4 fois contravariantes 3HHHH
72 //-----
73
74 typedef TenseurQ1_troisSym_HHHH TenseurQ1geneHHHH ;
75

```



```

76 //-----
77 //          cas des composantes 4 fois covariantes
78 //-----
79
80 typedef TenseurQ1_troisSym_BBBB TenseurQ1geneBBBB ;
81
82 //-----
83 //          cas des composantes mixte inverse BHBH
84 //-----
85
86 class TenseurQ1geneBHBH : public TenseurBHBH
87 { // surcharge de l'operator de lecture
88     friend istream & operator > (istream &, TenseurQ1geneBHBH &);
89     // surcharge de l'operator d'écriture
90     friend ostream & operator < (ostream &, const TenseurQ1geneBHBH &);
91
92 public :
93     // Constructeur
94     TenseurQ1geneBHBH() ; // par défaut
95     // initialisation de toutes les composantes (ici une seule) a une meme valeur val
96     TenseurQ1geneBHBH(const double& x1111);
97
98     // DESTRUCTEUR :
99     ~TenseurQ1geneBHBH() ;
100    // constructeur a partir d'une instance non differenciee
101    TenseurQ1geneBHBH (const TenseurBHBH &);
102    // constructeur de copie
103    TenseurQ1geneBHBH (const TenseurQ1geneBHBH &);
104
105    // METHODES PUBLIQUES :
106    //2) virtuelles
107    // initialise toutes les composantes à val
108    void Init(double val) ;
109    // operations
110    TenseurBHBH & operator + ( const TenseurBHBH &) const ;
111    void operator += ( const TenseurBHBH &);
112    TenseurBHBH & operator - () const ; // oppose du tenseur
113    TenseurBHBH & operator - ( const TenseurBHBH &) const ;
114    void operator -= ( const TenseurBHBH &);
115    TenseurBHBH & operator = ( const TenseurBHBH &);
116    TenseurBHBH & operator * (const double &) const ;
117    void operator *= ( const double &);
118    TenseurBHBH & operator / ( const double &) const ;
119    void operator /= ( const double &);
120
121    // produit deux fois contracte à droite avec un tenseur du second ordre
122    // (différent de à gauche !!) def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
123    TenseurBH& operator && ( const TenseurBH &) const ;
124
125    // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
126    // les 2 premiers indices sont échangés avec les deux derniers indices
127    // ici en fait c'est le même tenseur grace aux symétries constitutives
128    TenseurHBHB & Transposelet2avec3et4() const ;
129
130    // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
131    // plusZero = true: les données manquantes sont mises à 0
132    // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
133    // des données possibles
134    virtual void Affectation_trans_dimension(const TenseurBHBH & B,bool plusZero) ;
135
136    // test
137    int operator == ( const TenseurBHBH &) const ;
138
139    // change la composante i,j,k,l du tenseur
140    // acces en écriture,
141    void Change (int i, int j, int k, int l,const double& val) ;
142    // en cumul : équivalent de +=
143    void ChangePlus (int i, int j, int k, int l,const double& val);
144
145    // Retourne la composante i,j,k,l du tenseur
146    // acces en lecture seule
147    double operator () (int i, int j, int k, int l) const ;
148
149    // calcul du maximum en valeur absolu des composantes du tenseur
150    double MaxiComposante() const;
151
152    // lecture et écriture de données
153    istream & Lecture(istream & entree);
154    ostream & Ecriture(ostream & sort) const ;
155
156 protected :
157     // allocator dans la liste de data
158     listdoubleIter ipointe;
159
160     // fonction pour le produit contracté à gauche
161     TenseurBH& Prod_gauche( const TenseurBH & F) const ;
162 };

```

```

163
164 //-----
165 //          cas des composantes mixte inverse HBHB
166 //-----
167
168 class TenseurQlgeneHBHB : public TenseurHBHB
169 { // surcharge de l'operator de lecture
170   friend istream & operator » (istream &, TenseurQlgeneHBHB &);
171   // surcharge de l'operator d'écriture
172   friend ostream & operator « (ostream &, const TenseurQlgeneHBHB &);
173
174   public :
175     // Constructeur
176     TenseurQlgeneHBHB() ; // par défaut
177     // initialisation de toutes les composantes (ici une seule) a une meme valeur val
178     TenseurQlgeneHBHB(const double& x1111);
179
180     // DESTRUCTEUR :
181     ~TenseurQlgeneHBHB() ;
182     // constructeur a partir d'une instance non differenciee
183     TenseurQlgeneHBHB (const TenseurHBHB &);
184     // constructeur de copie
185     TenseurQlgeneHBHB (const TenseurQlgeneHBHB &);
186
187     // METHODES PUBLIQUES :
188     //2)   virtuelles
189     // initialise toutes les composantes à val
190     void InitA(double val) ;
191     // operations
192     TenseurHBHB & operator + ( const TenseurHBHB &) const ;
193     void operator += ( const TenseurHBHB &);
194     TenseurHBHB & operator - ( const TenseurHBHB &) const ; // oppose du tenseur
195     TenseurHBHB & operator - ( const TenseurHBHB &) const ;
196     void operator -= ( const TenseurHBHB &);
197     TenseurHBHB & operator = ( const TenseurHBHB &);
198     TenseurHBHB & operator * (const double &) const ;
199     void operator *= ( const double &);
200     TenseurHBHB & operator / ( const double &) const ;
201     void operator /= ( const double &);
202
203     // produit deux fois contracte à droite avec un tenseur du second ordre
204     // (différent de à gauche !!) def dans Tenseur.Q.h à l'aide de la fonction privée Prod_gauche
205     TenseurHB& operator && ( const TenseurHB &) const ;
206
207     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
208     // les 2 premiers indices sont échangés avec les deux derniers indices
209     // ici en fait c'est le même tenseur grace aux symétries constitutives
210     TenseurBHBH & Transposelet2avec3et4() const ;
211
212     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
213     // plusZero = true: les données manquantes sont mises à 0
214     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
215     // des données possibles
216     virtual void Affectation_trans_dimension(const TenseurHBHB & B,bool plusZero);
217
218     // test
219     int operator == ( const TenseurHBHB &) const ;
220
221     // change la composante i,j,k,l du tenseur
222     // acces en écriture,
223     void Change (int i, int j, int k, int l,const double& val) ;
224     // en cumul : équivalent de +=
225     void ChangePlus (int i, int j, int k, int l,const double& val);
226
227     // Retourne la composante i,j,k,l du tenseur
228     // acces en lecture seule
229     double operator () (int i, int j, int k, int l) const ;
230
231     // calcul du maximum en valeur absolu des composantes du tenseur
232     double MaxiComposante() const;
233
234     // lecture et écriture de données
235     istream & Lecture(istream & entree);
236     ostream & Ecriture(ostream & sort) const ;
237
238   protected :
239     // allocator dans la liste de data
240     listdoubleIter ipointe;
241
242     // fonction pour le produit contracté à gauche
243     TenseurHB& Prod_gauche( const TenseurHB & F) const ;
244   };
245
246 //-----
247 //          cas des composantes mixte inverse BHBH
248 //-----
249

```

```

250 class TenseurQ1geneBHHB : public TenseurBHHB
251 { // surcharge de l'operator de lecture
252   friend istream & operator » (istream &, TenseurQ1geneBHHB &);
253   // surcharge de l'operator d'écriture
254   friend ostream & operator « (ostream &, const TenseurQ1geneBHHB &);
255
256 public :
257   // Constructeur
258   TenseurQ1geneBHHB() ; // par défaut
259   // initialisation de toutes les composantes (ici une seule) a une meme valeur val
260   TenseurQ1geneBHHB(const double& x1111);
261
262   // DESTRUCTEUR :
263   ~TenseurQ1geneBHHB() ;
264   // constructeur a partir d'une instance non differenciee
265   TenseurQ1geneBHHB (const TenseurBHHB &);
266   // constructeur de copie
267   TenseurQ1geneBHHB (const TenseurQ1geneBHHB &);
268
269   // METHODES PUBLIQUES :
270   //2) virtuelles
271   // initialise toutes les composantes à val
272   void Inita(double val) ;
273   // operations
274   TenseurBHHB & operator + ( const TenseurBHHB &) const ;
275   void operator += ( const TenseurBHHB &);
276   TenseurBHHB & operator - () const ; // oppose du tenseur
277   TenseurBHHB & operator - ( const TenseurBHHB &) const ;
278   void operator -= ( const TenseurBHHB &);
279   TenseurBHHB & operator = ( const TenseurBHHB &);
280   TenseurBHHB & operator * (const double &) const ;
281   void operator *= ( const double &);
282   TenseurBHHB & operator / ( const double &) const ;
283   void operator /= ( const double &);
284
285   // produit deux fois contracte à droite avec un tenseur du second ordre
286   // (différent de à gauche !!) def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
287   TenseurBH& operator && ( const TenseurHB &) const ;
288
289   // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
290   // les 2 premiers indices sont échangés avec les deux derniers indices
291   // ici en fait c'est le même tenseur grace aux symétries constitutives
292   TenseurHBBH & Transposelet2avec3et4() const ;
293
294   // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
295   // plusZero = true: les données manquantes sont mises à 0
296   // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
297   // des données possibles
298   virtual void Affectation_trans_dimension(const TenseurBHHB & B,bool plusZero) ;
299
300   // test
301   int operator == ( const TenseurBHHB &) const ;
302
303   // change la composante i,j,k,l du tenseur
304   // acces en écriture,
305   void Change (int i, int j, int k, int l,const double& val) ;
306   // en cumul : équivalent de +=
307   void ChangePlus (int i, int j, int k, int l,const double& val);
308
309   // Retourne la composante i,j,k,l du tenseur
310   // acces en lecture seule
311   double operator () (int i, int j, int k, int l) const ;
312
313   // calcul du maximum en valeur absolu des composantes du tenseur
314   double MaxiComposante() const;
315
316   // lecture et écriture de données
317   istream & Lecture(istream & entree);
318   ostream & Ecriture(ostream & sort) const ;
319
320 protected :
321   // allocator dans la liste de data
322   listdoubleIter ipointe;
323
324   // fonction pour le produit contracté à gauche
325   TenseurHB& Prod_gauche( const TenseurBH & F) const ;
326 };
327
328 //-----
329 //          cas des composantes 2 fois mixtes HBBH
330 //-----
331 class TenseurQ1geneHBBH : public TenseurHBBH
332 { // surcharge de l'operator de lecture
333   friend istream & operator » (istream &, TenseurQ1geneHBBH &);
334   // surcharge de l'operator d'écriture
335   friend ostream & operator « (ostream &, const TenseurQ1geneHBBH &);
336

```

```

337 public :
338     // Constructeur
339     TenseurQlgeneHBBH() ; // par défaut
340     // initialisation de toutes les composantes (ici une seule) a une meme valeur val
341     TenseurQlgeneHBBH(const double& x1111);
342
343     // DESTRUCTEUR :
344     ~TenseurQlgeneHBBH() ;
345     // constructeur a partir d'une instance non differenciee
346     TenseurQlgeneHBBH (const TenseurHBBH &);
347     // constructeur de copie
348     TenseurQlgeneHBBH (const TenseurQlgeneHBBH &);
349
350     // METHODES PUBLIQUES :
351     //2) virtuelles
352     // operations
353     // initialise toutes les composantes à val
354     void Init(double val) ;
355     TenseurHBBH & operator + ( const TenseurHBBH & ) const ;
356     void operator += ( const TenseurHBBH &);
357     TenseurHBBH & operator - ( ) const ; // oppose du tenseur
358     TenseurHBBH & operator - ( const TenseurHBBH & ) const ;
359     void operator -= ( const TenseurHBBH &);
360     TenseurHBBH & operator = ( const TenseurHBBH &);
361     TenseurHBBH & operator * (const double & ) const ;
362     void operator *= ( const double &);
363     TenseurHBBH & operator / ( const double & ) const ;
364     void operator /= ( const double &);
365
366     // produit deux fois contracte à droite avec un tenseur du second ordre
367     // (différent de à gauche !!) def dans Tenseur.Q.h à l'aide de la fonction privée Prod_gauche
368     TenseurHB& operator && ( const TenseurBH & ) const ;
369
370     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
371     // les 2 premiers indices sont échangés avec les deux derniers indices
372     // ici en fait c'est le même tenseur grace aux symétries constitutives
373     TenseurBHHB & Transposelet2avec3et4() const ;
374
375     // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
376     // plusZero = true: les données manquantes sont mises à 0
377     // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
378     // des données possibles
379     virtual void Affectation_trans_dimension(const TenseurHBBH & B,bool plusZero) ;
380
381     // test
382     int operator == ( const TenseurHBBH & ) const ;
383
384     // change la composante i,j,k,l du tenseur
385     // acces en ecriture,
386     void Change (int i, int j, int k, int l,const double& val) ;
387     // en cumul : équivalent de +=
388     void ChangePlus (int i, int j, int k, int l,const double& val);
389
390     // Retourne la composante i,j,k,l du tenseur
391     // acces en lecture seule
392     double operator () (int i, int j, int k, int l) const ;
393
394     // calcul du maximum en valeur absolu des composantes du tenseur
395     double MaxiComposante() const;
396
397     // lecture et écriture de données
398     istream & Lecture(istream & entree);
399     ostream & Ecriture(ostream & sort) const ;
400
401 protected :
402     // allocator dans la liste de data
403     listdoubleIter ipointe;
404
405     // fonction pour le produit contracté à gauche
406     TenseurBH& Prod_gauche( const TenseurHB & F) const ;
407 };
408
409
410 #ifndef MISE_AU_POINT
411 #include "TenseurQlgene.cc"
412 #define TenseurQlgene_H_deja_inclus
413 #endif
414
415 #endif

```

## 7.476 TenseurQ2gene.h

```

1
2
3 // This file is part of the Herezh++ application.

```

```

4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *          UNIVERSITE DE BRETAGNE SUD (UBS) --- LORIENT          *
34 *****/
35 *          IRDL - Equipe DE GENIE MECANIQUE ET MATERIAUX (EG2M) *
36 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
37 * tel. 02.97.32.82.47          http://IRDL.fr          *
38 *****/
39 *          DATE:          16/12/2019          *
40 *          *          *          *          *          *          *
41 *          AUTEUR:          G RIO (mailto:gerard.rio@univ-ubs.fr) *
42 *          phone 02.97.32.82.47          *
43 *          *          *          *          *          *          *
44 *          PROJET:          Herezh++          *
45 *          *          *          *          *          *          *
46 *****/
47 *          BUT:          Definition d'une classe derivee de tenseur du 4ieme ordre *
48 *          de dimension2, il s'agit ici d'une classe générale, sans *
49 *          particularités: c-a-d 16 composantes.          *
50 *          *          *          *          *          *          *
51 *          *****          *
52 *          VERIFICATION:          *
53 *          *          *          *          *          *          *
54 *          ! date ! auteur ! but          !          *
55 *          -----          *
56 *          !          !          !          !          *
57 *          *          *          *          *          *          *
58 *          *****          *
59 *          MODIFICATIONS:          *
60 *          ! date ! auteur ! but          !          *
61 *          -----          *
62 *          *          *          *          *          *          *
63 *****/
64 #ifndef TENSEURQ2GENE_H
65 #define TENSEURQ2GENE_H
66
67 #include <iostream>
68 #include "TenseurQ.h"
69 #include "PtTabRel.h"
70 #include "Tableau2_T.h"
71 #include "Tenseur2.h"
72
73 /*****
74 // pour l'instant on n'utilise que des tenseurs d'ordre deux symétriques
75 // à chaque fois qu'apparaît un tenseurs du second ordre
76 // si besoin est on améliora
77 ///////////////////////////////////////////////////////////////////
78
79 //-----
80 //          cas des composantes 4 fois contravariantes 2HHHH
81 //-----
82 class TenseurQ2geneHHHH : public TenseurHHHH
83 { // surcharge de l'operator de lecture
84     friend istream & operator > (istream &, TenseurQ2geneHHHH &);
85     // surcharge de l'operator d'écriture
86     friend ostream & operator < (ostream &, const TenseurQ2geneHHHH &);
87
88     public :
89         // Constructeur

```

```

90  TenseurQ2geneHHHH() ; // par défaut
91  // initialisation de toutes les composantes a une meme valeur val
92  TenseurQ2geneHHHH(const double val);
93  // initialisation à partir d'un produit tensoriel avec 3 cas
94  // cas = 1      : produit tensoriel normal
95  //              *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
96  // cas = 2      : produit tensoriel barre
97  //              *this=aHH(i,k).bHH(j,l) gBi gBj gBk gBl
98  // cas = 3      : produit tensoriel under barre
99  //              *this=aHH(i,l).bHH(j,k) gBi gBj gBk gBl
100 TenseurQ2geneHHHH(int cas, const TenseurHH & aHH, const TenseurHH & bHH);
101 TenseurQ2geneHHHH(int cas, const Tenseur2HH & aHH, const Tenseur2HH & bHH);
102
103 // DESTRUCTEUR :
104 ~TenseurQ2geneHHHH() ;
105 // constructeur a partir d'une instance non differenciee
106 TenseurQ2geneHHHH (const TenseurHHHH &);
107 // constructeur de copie
108 TenseurQ2geneHHHH (const TenseurQ2geneHHHH &);
109
110 // METHODES PUBLIQUES :
111 //2) virtuelles
112 // initialise toutes les composantes à val
113 void Init(double val) ;
114 // operations
115 TenseurHHHH & operator + ( const TenseurHHHH & ) const ;
116 void operator += ( const TenseurHHHH & );
117 TenseurHHHH & operator - () const ; // oppose du tenseur
118 TenseurHHHH & operator - ( const TenseurHHHH & ) const ;
119 void operator -= ( const TenseurHHHH & );
120 TenseurHHHH & operator = ( const TenseurHHHH & );
121 TenseurHHHH & operator = ( const TenseurQ2geneHHHH & B )
122 { return this->operator=((TenseurHHHH & ) B); };
123 TenseurHHHH & operator * (const double & ) const ;
124 void operator *= ( const double & );
125 TenseurHHHH & operator / ( const double & ) const ;
126 void operator /= ( const double & );
127
128 // produit deux fois contracte à droite avec un tenseur du second ordre
129 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
130 TenseurHH& operator && ( const TenseurBB & ) const ;
131 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
132 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
133 TenseurHHHH& operator && ( const TenseurBBHH & ) const ;
134 TenseurHHBB& operator && ( const TenseurBBBB & ) const ;
135
136 //fonctions définissant le produit tensoriel normal de deux tenseurs
137 // *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
138 static TenseurHHHH & Prod_tensoriel(const TenseurHH & aHH, const TenseurHH & bHH) ;
139 //fonctions définissant le produit tensoriel barre de deux tenseurs
140 // *this=aHH(i,k).bHH(j,l) gBi gBj gBk gBl
141 static TenseurHHHH & Prod_tensoriel_barre(const TenseurHH & aHH, const TenseurHH & bHH) ;
142 //fonctions définissant le produit tensoriel under_barre de deux tenseurs
143 // *this=aHH(i,l).bHH(j,k) gBi gBj gBk gBl
144 static TenseurHHHH & Prod_tensoriel_under_barre(const TenseurHH & aHH, const TenseurHH & bHH) ;
145
146 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
147 // les 2 premiers indices sont échangés avec les deux derniers indices
148 TenseurHHHH & Transposelet2avec3et4() const ;
149
150 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
151 // plusZero = true: les données manquantes sont mises à 0
152 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
153 // des données possibles
154 void Affectation_trans_dimension(const TenseurHHHH & B,bool plusZero);
155
156 // création d'un tenseur symétrique / au deux premiers indices et / au deux derniers indices
157 // B(i,i,i,i) = A(i,i,i,i); B(i,j,k,k) = 1/2(A(i,j,k,k)+ A(j,i,k,k)); si 2 premiers indices
différents
158 // B(i,i,k,l) = 1/2(A(i,i,k,l)+ A(j,i,l,k)); si 2 derniers indices différents
159 // B(i,j,k,l) = 1/4(A(i,j,k,l)+ A(j,i,k,l) + A(i,j,l,k)+ A(j,i,l,k)); si tous les indices différents
160 TenseurHHHH & Symetriselet2_3et4() const;
161
162 // test
163 int operator == ( const TenseurHHHH & ) const ;
164
165 // change la composante i,j,k,l du tenseur
166 // acces en ecriture,
167 void Change (int i, int j, int k, int l,const double& val) ;
168 // en cumul : équivalent de +=
169 void ChangePlus (int i, int j, int k, int l,const double& val);
170
171 // Retourne la composante i,j,k,l du tenseur
172 // acces en lecture seule
173 double operator () (int i, int j, int k, int l) const ;
174
175 // calcul du maximum en valeur absolu des composantes du tenseur

```

```

176     double MaxiComposante() const;
177
178     // lecture et écriture de données
179     istream & Lecture(istream & entree);
180     ostream & Ecriture(ostream & sort) const ;
181
182     protected :
183     // allocator dans la liste de data
184     listdouble16Iter ipointe;
185
186     // fonction pour le produit contracté à gauche
187     TenseurHH& Prod_gauche( const TenseurBB & F) const;
188     TenseurBBHH& Prod_gauche( const TenseurBBBB & F) const;
189     TenseurHHHH& Prod_gauche( const TenseurHHBB & F) const;
190 };
191 //
192 //-----
193 //          cas des composantes 4 fois covariantes
194 //-----
195 class TenseurQ2geneBBBB : public TenseurBBBB
196 { // surcharge de l'operator de lecture
197     friend istream & operator » (istream &, TenseurQ2geneBBBB &);
198     // surcharge de l'operator d'écriture
199     friend ostream & operator « (ostream &, const TenseurQ2geneBBBB &);
200
201     public :
202     // Constructeur
203     TenseurQ2geneBBBB() ; // par défaut
204     // initialisation de toutes les composantes a une meme valeur val
205     TenseurQ2geneBBBB(const double val);
206     // initialisation à partir d'un produit tensoriel avec 3 cas
207     // cas = 1      : produit tensoriel normal
208     //              *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
209     // cas = 2      : produit tensoriel barre
210     //              *this=aBB(i,k).bBB(j,l) gHi gHj gHk gHl
211     // cas = 3      : produit tensoriel under barre
212     //              *this=aBB(i,l).bBB(j,k) gHi gHj gHk gHl
213     TenseurQ2geneBBBB(int cas, const TenseurBB & aBB, const TenseurBB & bBB);
214     TenseurQ2geneBBBB(int cas, const Tenseur2BB & aBB, const Tenseur2BB & bBB);
215
216     // DESTRUCTEUR :
217     ~TenseurQ2geneBBBB() ;
218     // constructeur a partir d'une instance non differenciee
219     TenseurQ2geneBBBB (const TenseurBBBB &);
220     // constructeur de copie
221     TenseurQ2geneBBBB (const TenseurQ2geneBBBB &);
222
223     // METHODES PUBLIQUES :
224     //2) virtuelles
225     // initialise toutes les composantes à val
226     void Inita(double val) ;
227     // operations
228     TenseurBBBB & operator + ( const TenseurBBBB &) const ;
229     void operator += ( const TenseurBBBB &);
230     TenseurBBBB & operator - () const ; // oppose du tenseur
231     TenseurBBBB & operator - ( const TenseurBBBB &) const ;
232     void operator -= ( const TenseurBBBB &);
233     TenseurBBBB & operator = ( const TenseurBBBB &);
234     TenseurBBBB & operator = ( const TenseurQ2geneBBBB & B)
235     { return this->operator=((TenseurBBBB &) B); };
236     TenseurBBBB & operator * (const double &) const ;
237     void operator *= ( const double &);
238     TenseurBBBB & operator / ( const double &) const ;
239     void operator /= ( const double &);
240
241     // produit deux fois contracte à droite avec un tenseur du second ordre
242     // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
243     TenseurBB& operator && ( const TenseurHH &) const ;
244     // produit deux fois contracte à droite avec un tenseur du quatrième ordre
245     // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
246     TenseurBBBB& operator && ( const TenseurHHBB &) const ;
247     TenseurBBHH& operator && ( const TenseurHHHH &) const ;
248
249     //fonctions définissant le produit tensoriel normal de deux tenseurs
250     // *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
251     static TenseurBBBB & Prod_tensoriel(const TenseurBB & aBB, const TenseurBB & bBB) ;
252     //fonctions définissant le produit tensoriel barre de deux tenseurs
253     // *this=aBB(i,k).bBB(j,l) gHi gHj gHk gHl
254     static TenseurBBBB & Prod_tensoriel_barre(const TenseurBB & aBB, const TenseurBB & bBB) ;
255     //fonctions définissant le produit tensoriel under_barre de deux tenseurs
256     // *this=aBB(i,l).bBB(j,k) gHi gHj gHk gHl
257     static TenseurBBBB & Prod_tensoriel_under_barre(const TenseurBB & aBB, const TenseurBB & bBB) ;
258
259     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
260     // les 2 premiers indices sont échangés avec les deux derniers indices
261     TenseurBBBB & Transposelet2avec3et4() const ;
262

```

```

263 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
264 // plusZero = true: les données manquantes sont mises à 0
265 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
266 // des données possibles
267 void Affectation_trans_dimension(const TenseurBBBB & B, bool plusZero);
268
269 // création d'un tenseur symétrique / au deux premiers indices et / au deux derniers indices
270 // B(i,i,i,i) = A(i,i,i,i); B(i,j,k,k) = 1/2(A(i,j,k,k) + A(j,i,k,k)); si 2 premiers indices
différents
271 // B(i,i,k,l) = 1/2(A(i,i,k,l) + A(j,i,l,k)); si 2 derniers indices différents
272 // B(i,j,k,l) = 1/4(A(i,j,k,l) + A(j,i,k,l) + A(i,j,l,k) + A(j,i,l,k)); si tous les indices différents
273 TenseurBBBB & Symetriselet2_3et4() const;
274
275 // test
276 int operator == ( const TenseurBBBB & ) const ;
277
278 // Retourne la composante i,j,k,l du tenseur
279 // acces en ecriture,
280 void Change (int i, int j, int k, int l, const double& val) ;
281 // en cumul : équivalent de +=
282 void ChangePlus (int i, int j, int k, int l, const double& val);
283
284 // Retourne la composante i,j,k,l du tenseur
285 // acces en lecture seule
286 double operator () (int i, int j, int k, int l) const ;
287
288 // calcul du maximum en valeur absolu des composantes du tenseur
289 double MaxiComposante() const;
290
291 // lecture et écriture de données
292 istream & Lecture(istream & entree);
293 ostream & Ecriture(ostream & sort) const ;
294
295 protected :
296 // allocator dans la liste de data
297 listdoublel6Iter ipointe;
298
299 // fonction pour le produit contracté à gauche
300 TenseurBB& Prod_gauche( const TenseurHH & F) const;
301 TenseurHHBB& Prod_gauche( const TenseurHHHH & F) const;
302 TenseurBBBB& Prod_gauche( const TenseurBBHH & F) const;
303 };
304
305 //
306 //-----
307 // cas des composantes mixte 2BBHH
308 //-----
309
310 class TenseurQ2geneBBHH : public TenseurBBHH
311 { // surcharge de l'operateur de lecture
312 friend istream & operator » (istream &, TenseurQ2geneBBHH &);
313 // surcharge de l'operateur d'ecriture
314 friend ostream & operator « (ostream &, const TenseurQ2geneBBHH &);
315
316 public :
317 // Constructeur
318 TenseurQ2geneBBHH() ; // par défaut
319 // initialisation de toutes les composantes a une meme valeur val
320 TenseurQ2geneBBHH(const double val);
321 // initialisation à partir d'un produit tensoriel normal
322 // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
323 TenseurQ2geneBBHH(const TenseurBB & aBB, const TenseurHH & bHH);
324 TenseurQ2geneBBHH(const Tenseur2BB & aBB, const Tenseur2HH & bHH);
325
326 // DESTRUCTEUR :
327 ~TenseurQ2geneBBHH() ;
328 // constructeur a partir d'une instance non differenciee
329 TenseurQ2geneBBHH (const TenseurBBHH &);
330 // constructeur de copie
331 TenseurQ2geneBBHH (const TenseurQ2geneBBHH &);
332
333 // METHODES PUBLIQUES :
334 //2) virtuelles
335 // initialise toutes les composantes à val
336 void Inita(double val) ;
337 // operations
338 TenseurBBHH & operator + ( const TenseurBBHH & ) const ;
339 void operator += ( const TenseurBBHH & );
340 TenseurBBHH & operator - () const ; // oppose du tenseur
341 TenseurBBHH & operator - ( const TenseurBBHH & ) const ;
342 void operator -= ( const TenseurBBHH & );
343 TenseurBBHH & operator = ( const TenseurBBHH & );
344 TenseurBBHH & operator * (const double & ) const ;
345 void operator *= ( const double & );
346 TenseurBBHH & operator / ( const double & ) const ;
347 void operator /= ( const double & );
348

```



```

349 // produit deux fois contracte à droite avec un tenseur du second ordre
350 // diffèrent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
351 TenseurBB& operator && ( const TenseurBB & ) const ;
352 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
353 // diffèrent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
354 TenseurBBBB& operator && ( const TenseurBBBB & ) const ;
355 TenseurBBHH& operator && ( const TenseurBBHH & ) const ;
356
357 //fonctions définissant le produit tensoriel normal de deux tenseurs
358 // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
359 static TenseurBBHH & Prod_tensoriel(const TenseurBB & aBB, const TenseurHH & bHH) ;
360
361 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
362 // les 2 premiers indices sont échangés avec les deux derniers indices
363 TenseurHBBB & Transposelet2avec3et4() const ;
364
365 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
366 // plusZero = true: les données manquantes sont mises à 0
367 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
368 // des données possibles
369 void Affectation_trans_dimension(const TenseurBBHH & B, bool plusZero);
370
371 // test
372 int operator == ( const TenseurBBHH & ) const ;
373
374 // Retourne la composante i,j,k,l du tenseur
375 // acces en ecriture,
376 void Change (int i, int j, int k, int l, const double& val) ;
377 // en cumul : équivalent de +=
378 void ChangePlus (int i, int j, int k, int l, const double& val);
379
380 // Retourne la composante i,j,k,l du tenseur
381 // acces en lecture seule
382 double operator () (int i, int j, int k, int l) const ;
383
384 // calcul du maximum en valeur absolu des composantes du tenseur
385 double MaxiComposante() const;
386
387 // lecture et écriture de données
388 istream & Lecture(istream & entree);
389 ostream & Ecriture(ostream & sort) const ;
390
391 protected :
392 // allocator dans la liste de data
393 listdoublel6Iter ipointe;
394
395 // fonction pour le produit contracté à gauche
396 TenseurHH& Prod_gauche( const TenseurHH & F) const;
397 TenseurBBHH& Prod_gauche( const TenseurBBHH & F) const;
398 TenseurHHHH& Prod_gauche( const TenseurHHHH & F) const;
399 };
400 //
401 //-----
402 //          cas des composantes mixte 2HHBB
403 //-----
404
405 class TenseurQ2geneHHBB : public TenseurHHBB
406 { // surcharge de l'operator de lecture
407   friend istream & operator » (istream &, TenseurQ2geneHHBB &);
408   // surcharge de l'operator d'ecriture
409   friend ostream & operator « (ostream &, const TenseurQ2geneHHBB &);
410
411 public :
412   // Constructeur
413   TenseurQ2geneHHBB() ; // par défaut
414   // initialisation de toutes les composantes a une meme valeur val
415   TenseurQ2geneHHBB(const double val);
416
417   // initialisation à partir d'un produit tensoriel normal
418   // *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
419   TenseurQ2geneHHBB(const TenseurHH & aHH, const TenseurBB & bBB);
420   TenseurQ2geneHHBB(const Tenseur2HH & aHH, const Tenseur2BB & bBB);
421
422   // DESTRUCTEUR :
423   ~TenseurQ2geneHHBB() ;
424   // constructeur a partir d'une instance non differenciee
425   TenseurQ2geneHHBB (const TenseurHHBB &);
426   // constructeur de copie
427   TenseurQ2geneHHBB (const TenseurQ2geneHHBB &);
428
429   // METHODES PUBLIQUES :
430   //2) virtuelles
431   // initialise toutes les composantes à val
432   void Inita(double val) ;
433   // operations
434   TenseurHHBB & operator + ( const TenseurHHBB & ) const ;
435   void operator += ( const TenseurHHBB & );

```

```

436 TenseurHHBB & operator - () const ; // oppose du tenseur
437 TenseurHHBB & operator - ( const TenseurHHBB & ) const ;
438 void operator -= ( const TenseurHHBB & );
439 TenseurHHBB & operator = ( const TenseurHHBB & );
440 TenseurHHBB & operator * (const double & ) const ;
441 void operator *= ( const double & );
442 TenseurHHBB & operator / ( const double & ) const ;
443 void operator /= ( const double & );
444
445 // produit deux fois contracte à droite avec un tenseur du second ordre
446 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
447 TenseurHH & operator && ( const TenseurHH & ) const ;
448 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
449 // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
450 TenseurHHHH & operator && ( const TenseurHHHH & ) const ;
451 TenseurHHBB & operator && ( const TenseurHHBB & ) const ;
452
453 //fonctions définissant le produit tensoriel normal de deux tenseurs
454 // *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
455 static TenseurHHBB & Prod_tensoriel(const TenseurHH & aHH, const TenseurBB & bBB) ;
456
457 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
458 // les 2 premiers indices sont échangés avec les deux derniers indices
459 TenseurBBHH & Transposelet2avec3et4() const ;
460
461 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
462 // plusZero = true: les données manquantes sont mises à 0
463 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
464 // des données possibles
465 void Affectation_trans_dimension(const TenseurHHBB & B,bool plusZero);
466
467 // test
468 int operator == ( const TenseurHHBB & ) const ;
469
470 // Retourne la composante i,j,k,l du tenseur
471 // acces en ecriture,
472 void Change (int i, int j, int k, int l,const double& val) ;
473 // en cumul : équivalent de +=
474 void ChangePlus (int i, int j, int k, int l,const double& val);
475
476 // Retourne la composante i,j,k,l du tenseur
477 // acces en lecture seule
478 double operator () (int i, int j, int k, int l) const ;
479
480 // calcul du maximum en valeur absolu des composantes du tenseur
481 double MaxiComposante() const;
482
483 // lecture et écriture de données
484 istream & Lecture(istream & entree);
485 ostream & Ecriture(ostream & sort) const ;
486
487 protected :
488 // allocator dans la liste de data
489 listdoublel6Iter ipointe;
490
491 // fonction pour le produit contracté à gauche
492 TenseurBB & Prod_gauche( const TenseurBB & F) const;
493 TenseurHHBB & Prod_gauche( const TenseurHHBB & F) const;
494 TenseurBBBB & Prod_gauche( const TenseurBBBB & F) const;
495 };
496
497 #ifndef MISE_AU_POINT
498 #include "TenseurQ2gene-1.cc"
499 #include "TenseurQ2gene-2.cc"
500 #define TenseurQ2gene_H_deja_inclus
501 #endif
502
503
504 #endif

```

## 7.477 TenseurQ3.h.old.h

```

1 /*****
2 *          LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)          *
3 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex        *
4 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
5 *****/
6 *      DATE:          3/5/2002                                          *
7 *                                                            $          *
8 *      AUTEUR:       G RIO (mailto:gerard.rio@univ-ubs.fr)            *
9 *                                                            $          *
10 *      PROJETS:     Herezh++                                          *
11 *                                                            $          *
12 *
13 *****/

```

```

14 *      BUT:  Definition des classes derivees tenseur du 4ieme ordre      *
15 *          de dimension3.                                             *
16 *                                                                 $    *
17 *      ***** *
18 *      VERIFICATION: *
19 * *
20 *      ! date !  auteur !          but          ! *
21 *      ----- *
22 *      !          !          !          !          *
23 *                                                                 $    *
24 *      ***** *
25 *      MODIFICATIONS: *
26 *      ! date !  auteur !          but          ! *
27 *      ----- *
28 *                                                                 $    *
29 *      *****/
30 #ifndef TENSEURQ3_H
31 #define TENSEURQ3_H
32
33 #include <iostream>
34 #include "TenseurQ.h"
35 #include "PtTabRel.h"
36 # include "Tableau2_T.h"
37
38 //-----
39 //          cas des composantes 4 fois contravariantes 3HHHH
40 //-----
41 class Tenseur3HHHH : public TenseurHHHH
42 { // surcharge de l'operator de lecture
43   friend istream & operator >> (istream &, Tenseur3HHHH &);
44   // surcharge de l'operator d'écriture
45   friend ostream & operator << (ostream &, const Tenseur3HHHH &);
46
47 public :
48   // Constructeur
49   Tenseur3HHHH() ; // par défaut
50   // initialisation de toutes les composantes a une meme valeur val
51   Tenseur3HHHH(const double val);
52   // initialisation à partir d'un produit tensoriel avec 2 cas
53   // booleen = true : produit tensoriel normal
54   //          *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
55   // booleen = false : produit tensoriel barre
56   //          *this=aHH(i,k).bHH(j,l) gBi gBj gBk gBl
57   Tenseur3HHHH(bool normal, const TenseurHH & aHH, const TenseurHH & bHH);
58
59   // DESTRUCTEUR :
60   ~Tenseur3HHHH() ;
61   // constructeur a partir d'une instance non differenciee
62   Tenseur3HHHH (const TenseurHHHH &);
63   // constructeur de copie
64   Tenseur3HHHH (const Tenseur3HHHH &);
65
66   // METHODES PUBLIQUES :
67 //2)  virtuelles
68   // operations
69   TenseurHHHH & operator + ( const TenseurHHHH & ) const ;
70   void operator += ( const TenseurHHHH &);
71   TenseurHHHH & operator - ( ) const ; // oppose du tenseur
72   TenseurHHHH & operator - ( const TenseurHHHH & ) const ;
73   void operator -= ( const TenseurHHHH &);
74   TenseurHHHH & operator = ( const TenseurHHHH &);
75   TenseurHHHH & operator * (const double &) const ;
76   void operator *= ( const double &);
77   TenseurHHHH & operator / ( const double &) const ;
78   void operator /= ( const double &);
79
80   // produit contracte à droite avec un tenseur du second ordre
81   // différent à gauche !!
82   TenseurHH & operator && ( const TenseurBB & ) const ;
83   //fonctions définissant le produit tensoriel normal de deux tenseurs
84   // *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
85   static TenseurHHHH & Prod_tensoriel(const TenseurHH & aHH, const TenseurHH & bHH) ;
86   //fonctions définissant le produit tensoriel barre de deux tenseurs
87   // *this=aHH(i,k).bHH(j,l) gBi gBj gBk gBl
88   static TenseurHHHH & Prod_tensoriel_barre(const TenseurHH & aHH, const TenseurHH & bHH) ;
89
90   // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
91   // les 2 premiers indices sont échangés avec les deux derniers indices
92   TenseurHHHH & Transposelet2avec3et4() const ;
93
94   // test
95   int operator == ( const TenseurHHHH & ) const ;
96
97   // change la composante i,j,k,l du tenseur
98   // acces en écriture,
99   void change (int i, int j, int k, int l, double& val) ;

```

```

100
101 // Retourne la composante i,j,k,l du tenseur
102 // acces en lecture seule
103 double operator () (int i, int j, int k, int l) const ;
104
105 // calcul du maximum en valeur absolu des composantes du tenseur
106 double MaxiComposante() const;
107
108 // lecture et écriture de données
109 istream & Lecture(istream & entree);
110 ostream & Ecriture(ostream & sort);
111
112 protected :
113 // allocator dans la liste de data
114 listdouble36Iter ipointe;
115 // --- gestion de changement d'index ----
116 class ChangementIndex
117 { public:
118     ChangementIndex();
119     // passage pour les index de la forme vecteur à la forme i,j
120     Tableau <int> idx_i,idx_j;
121     // passage pour les index de la forme i,j à la forme vecteur
122     Tableau2 <int> odVect;
123 };
124 static ChangementIndex cdex3HHHH;
125
126 };
127 //
128 //-----
129 //          cas des composantes 4 fois covariantes
130 //-----
131 class Tenseur3BBBB : public TenseurBBBB
132 { // surcharge de l'operator de lecture
133     friend istream & operator » (istream &, Tenseur3BBBB &);
134     // surcharge de l'operator d'écriture
135     friend ostream & operator « (ostream &, const Tenseur3BBBB &);
136
137     public :
138     // Constructeur
139     Tenseur3BBBB() ; // par défaut
140     // initialisation de toutes les composantes a une meme valeur val
141     Tenseur3BBBB(const double val);
142     // initialisation à partir d'un produit tensoriel avec 2 cas
143     // booleen = true : produit tensoriel normal
144     //                *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
145     // booleen = false : produit tensoriel barre
146     //                *this=aBB(i,k).bBB(j,l) gHi gHj gHk gHl
147     Tenseur3BBBB(bool normal, const TenseurBB & aBB, const TenseurBB & bBB);
148
149     // DESTRUCTEUR :
150     ~Tenseur3BBBB() ;
151     // constructeur a partir d'une instance non differenciee
152     Tenseur3BBBB (const TenseurBBBB &);
153     // constructeur de copie
154     Tenseur3BBBB (const Tenseur3BBBB &);
155
156     // METHODES PUBLIQUES :
157     //2) virtuelles
158     // operations
159     TenseurBBBB & operator + ( const TenseurBBBB & ) const ;
160     void operator += ( const TenseurBBBB & );
161     TenseurBBBB & operator - ( ) const ; // oppose du tenseur
162     TenseurBBBB & operator - ( const TenseurBBBB & ) const ;
163     void operator -= ( const TenseurBBBB & );
164     TenseurBBBB & operator = ( const TenseurBBBB & );
165     TenseurBBBB & operator * (const double & ) const ;
166     void operator *= ( const double & );
167     TenseurBBBB & operator / ( const double & ) const ;
168     void operator /= ( const double & );
169
170     // produit contracte à droite avec un tenseur du second ordre
171     // différent à gauche !!
172     TenseurBB& operator && ( const TenseurHH & ) const ;
173     //fonctions définissant le produit tensoriel normal de deux tenseurs
174     // *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
175     static TenseurBBBB & Prod_tensoriel(const TenseurBB & aBB, const TenseurBB & bBB) ;
176     //fonctions définissant le produit tensoriel barre de deux tenseurs
177     // *this=aBB(i,k).bBB(j,l) gHi gHj gHk gHl
178     static TenseurBBBB & Prod_tensoriel_barre(const TenseurBB & aBB, const TenseurBB & bBB) ;
179
180     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
181     // les 2 premiers indices sont échangés avec les deux derniers indices
182     TenseurBBBB & Transposelet2avec3et4() const ;
183
184     // test
185     int operator == ( const TenseurBBBB & ) const ;
186

```

```

187 // Retourne la composante i,j,k,l du tenseur
188 // acces en ecriture,
189 void change (int i, int j, int k, int l, double& val) ;
190
191 // Retourne la composante i,j,k,l du tenseur
192 // acces en lecture seule
193 double operator () (int i, int j, int k, int l) const ;
194
195 // calcul du maximum en valeur absolu des composantes du tenseur
196 double MaxiComposante() const;
197
198 // lecture et écriture de données
199 istream & Lecture(istream & entree);
200 ostream & Ecriture(ostream & sort);
201
202 protected :
203 // allocator dans la liste de data
204 listdouble36Iter ipointe;
205 // --- gestion de changement d'index ----
206 class ChangementIndex
207 { public:
208     ChangementIndex();
209     // passage pour les index de la forme vecteur à la forme i,j
210     Tableau <int> idx_i,idx_j;
211     // passage pour les index de la forme i,j à la forme vecteur
212     Tableau2 <int> odVect;
213 };
214 static ChangementIndex cdex3BBBB;
215
216 };
217
218 //
219 //-----
220 //          cas des composantes mixte 3BBHH
221 //-----
222
223 class Tenseur3BBHH : public TenseurBBHH
224 { // surcharge de l'operateur de lecture
225     friend istream & operator » (istream &, Tenseur3BBHH &);
226     // surcharge de l'operateur d'ecriture
227     friend ostream & operator « (ostream &, const Tenseur3BBHH &);
228
229     public :
230     // Constructeur
231     Tenseur3BBHH() ; // par défaut
232     // initialisation de toutes les composantes a une meme valeur val
233     Tenseur3BBHH(const double val);
234     // initialisation à partir d'un produit tensoriel
235     //          *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
236     Tenseur3BBHH(const TenseurBB & aBB, const TenseurHH & bHH);
237
238     // DESTRUCTEUR :
239     ~Tenseur3BBHH() ;
240     // constructeur a partir d'une instance non differenciee
241     Tenseur3BBHH (const TenseurBBHH &);
242     // constructeur de copie
243     Tenseur3BBHH (const Tenseur3BBHH &);
244
245     // METHODES PUBLIQUES :
246     //2) virtuelles
247     // operations
248     TenseurBBHH & operator + ( const TenseurBBHH &) const ;
249     void operator += ( const TenseurBBHH &);
250     TenseurBBHH & operator - () const ; // oppose du tenseur
251     TenseurBBHH & operator - ( const TenseurBBHH &) const ;
252     void operator -= ( const TenseurBBHH &);
253     TenseurBBHH & operator = ( const TenseurBBHH &);
254     TenseurBBHH & operator * (const double &) const ;
255     void operator *= ( const double &);
256     TenseurBBHH & operator / ( const double &) const ;
257     void operator /= ( const double &);
258
259     // produit contracte à droite avec un tenseur du second ordre
260     // différent à gauche !!
261     TenseurBB& operator && ( const TenseurBB &) const ;
262     //fonctions définissant le produit tensoriel normal de deux tenseurs
263     // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
264     static TenseurBBHH & Prod_tensoriel(const TenseurBB & aBB, const TenseurHH & bHH) ;
265
266     // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
267     // les 2 premiers indices sont échangés avec les deux derniers indices
268     TenseurHHBB & Transposelet2avec3et4() const ;
269
270     // test
271     int operator == ( const TenseurBBHH &) const ;
272
273     // Retourne la composante i,j,k,l du tenseur

```

```

274 // acces en ecriture,
275 void change (int i, int j, int k, int l, double& val) ;
276
277 // Retourne la composante i,j,k,l du tenseur
278 // acces en lecture seule
279 double operator () (int i, int j, int k, int l) const ;
280
281 // calcul du maximum en valeur absolu des composantes du tenseur
282 double MaxiComposante() const;
283
284 // lecture et écriture de données
285 istream & Lecture(istream & entree);
286 ostream & Ecriture(ostream & sort);
287
288 protected :
289 // allocator dans la liste de data
290 listdouble36Iter ipointe;
291 // --- gestion de changement d'index ----
292 class ChangementIndex
293 { public:
294   ChangementIndex();
295   // passage pour les index de la forme vecteur à la forme i,j
296   Tableau<int> idx_i,idx_j;
297   // passage pour les index de la forme i,j à la forme vecteur
298   Tableau2<int> odVect;
299 };
300 static ChangementIndex cdexBBHH;
301
302 };
303 //
304 //-----
305 //          cas des composantes mixte 3HHBB
306 //-----
307
308 class Tenseur3HHBB : public TenseurHHBB
309 { // surcharge de l'operator de lecture
310   friend istream & operator » (istream &, Tenseur3HHBB &);
311   // surcharge de l'operator d'ecriture
312   friend ostream & operator « (ostream &, const Tenseur3HHBB &);
313
314 public :
315   // Constructeur
316   Tenseur3HHBB() ; // par défaut
317   // initialisation de toutes les composantes a une meme valeur val
318   Tenseur3HHBB(const double val);
319   // initialisation à partir d'un produit tensoriel avec 1 cas
320   // booleen = true : produit tensoriel normal
321   //          *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
322   Tenseur3HHBB(const TenseurHH & aHH, const TenseurBB & bBB);
323
324   // DESTRUCTEUR :
325   ~Tenseur3HHBB() ;
326   // constructeur a partir d'une instance non differenciee
327   Tenseur3HHBB (const TenseurHHBB &);
328   // constructeur de copie
329   Tenseur3HHBB (const Tenseur3HHBB &);
330
331   // METHODES PUBLIQUES :
332   //2) virtuelles
333   // operations
334   TenseurHHBB & operator + ( const TenseurHHBB &) const ;
335   void operator += ( const TenseurHHBB &);
336   TenseurHHBB & operator - () const ; // oppose du tenseur
337   TenseurHHBB & operator - ( const TenseurHHBB &) const ;
338   void operator -= ( const TenseurHHBB &);
339   TenseurHHBB & operator = ( const TenseurHHBB &);
340   TenseurHHBB & operator * (const double &) const ;
341   void operator *= ( const double &);
342   TenseurHHBB & operator / ( const double &) const ;
343   void operator /= ( const double &);
344
345   // produit contracte à droite avec un tenseur du second ordre
346   // différent à gauche !!
347   TenseurHH & operator && ( const TenseurHH &) const ;
348   //fonctions définissant le produit tensoriel normal de deux tenseurs
349   // *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
350   static TenseurHHBB & Prod_tensoriel(const TenseurHH & aHH, const TenseurBB & bBB) ;
351
352   // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
353   // les 2 premiers indices sont échangés avec les deux derniers indices
354   TenseurBBHH & Transposelet2avec3et4() const ;
355
356   // test
357   int operator == ( const TenseurHHBB &) const ;
358
359   // Retourne la composante i,j,k,l du tenseur
360   // acces en ecriture,

```

```

361 void change (int i, int j, int k, int l, double& val) ;
362
363 // Retourne la composante i,j,k,l du tenseur
364 // acces en lecture seule
365 double operator () (int i, int j, int k, int l) const ;
366
367 // calcul du maximum en valeur absolu des composantes du tenseur
368 double MaxiComposante() const;
369
370 // lecture et écriture de données
371 istream & Lecture(istream & entree);
372 ostream & Ecriture(ostream & sort);
373
374 protected :
375 // allocator dans la liste de data
376 listdouble36Iter ipointe;
377 // --- gestion de changement d'index ----
378 class ChangementIndex
379 { public:
380     ChangementIndex();
381     // passage pour les index de la forme vecteur à la forme i,j
382     Tableau <int> idx_i,idx_j;
383     // passage pour les index de la forme i,j à la forme vecteur
384     Tableau2 <int> odVect;
385 };
386 static ChangementIndex cdexHHBB;
387
388 };
389
390 #ifndef MISE_AU_POINT
391 #include "TenseurQ3-1.cc"
392 #include "TenseurQ3-2.cc"
393 #define TenseurQ3_H_deja_inclus
394 #endif
395
396
397 #endif

```

## 7.478 TenseurQ3gene.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 * LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M)
34 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex
35 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr
36 *****/
37 * DATE: 3/5/2002
38 *
39 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr)
40 * Tel 0297874571 fax : 02.97.87.45.72
41 *
42 * PROJET: Herezh++
43 *
44 *****/
45 * BUT: Definition d'une classe derivee de tenseur du 4ieme ordre *

```

```

46 *           de dimension3, il s'agit ici d'une classe générale, sans *
47 *           particularités: c-a-d 81 composantes.                    *
48 *                                                                 *
49 *           ***** *
50 *           VERIFICATION:                                           *
51 *                                                                 *
52 *           ! date ! auteur ! but !                                *
53 *           -----!-----!-----!-----!-----!-----!----- *
54 *           ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! *
55 *                                                                 *
56 *           ***** *
57 *           MODIFICATIONS:                                           *
58 *           ! date ! auteur ! but !                                *
59 *           -----!-----!-----!-----!-----!-----!----- *
60 *                                                                 *
61 *           *****/
62 #ifndef TenseurQ3GENE_H
63 #define TenseurQ3GENE_H
64
65 #include <iostream>
66 #include "TenseurQ.h"
67 #include "PtTabRel.h"
68 #include "Tableau2_T.h"
69 #include "Tenseur3.h"
70
71 //*****
72 // pour l'instant on n'utilise que des tenseurs d'ordre deux symétriques
73 // à chaque fois qu'apparaît un tenseurs du second ordre
74 // si besoin est on améliora
75 ///////////////////////////////////////////////////////////////////
76
77 //-----
78 // cas des composantes 4 fois contravariantes 3HHHH
79 //-----
80 class TenseurQ3geneHHHH : public TenseurHHHH
81 { // surcharge de l'operator de lecture
82   friend istream & operator > (istream &, TenseurQ3geneHHHH &);
83   // surcharge de l'operator d'écriture
84   friend ostream & operator < (ostream &, const TenseurQ3geneHHHH &);
85
86 public :
87   // Constructeur
88   TenseurQ3geneHHHH() ; // par défaut
89   // initialisation de toutes les composantes a une meme valeur val
90   TenseurQ3geneHHHH(const double val);
91   // initialisation à partir d'un produit tensoriel avec 3 cas
92   // cas = 1 : produit tensoriel normal
93   // *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
94   // cas = 2 : produit tensoriel barre
95   // *this=aHH(i,k).bHH(j,l) gBi gBj gBk gBl
96   // cas = 3 : produit tensoriel under barre
97   // *this=aHH(i,l).bHH(j,k) gBi gBj gBk gBl
98   TenseurQ3geneHHHH(int cas, const TenseurHH & aHH, const TenseurHH & bHH);
99   TenseurQ3geneHHHH(int cas, const Tenseur3HH & aHH, const Tenseur3HH & bHH);
100
101   // DESTRUCTEUR :
102   ~TenseurQ3geneHHHH() ;
103   // constructeur a partir d'une instance non differenciee
104   TenseurQ3geneHHHH (const TenseurHHHH &);
105   // constructeur de copie
106   TenseurQ3geneHHHH (const TenseurQ3geneHHHH &);
107
108   // METHODES PUBLIQUES :
109   //2) virtuelles
110   // initialise toutes les composantes à val
111   void Inita(double val) ;
112   // operations
113   TenseurHHHH & operator + ( const TenseurHHHH & ) const ;
114   void operator += ( const TenseurHHHH & );
115   TenseurHHHH & operator - ( ) const ; // oppose du tenseur
116   TenseurHHHH & operator - ( const TenseurHHHH & ) const ;
117   void operator -= ( const TenseurHHHH & );
118   TenseurHHHH & operator = ( const TenseurHHHH & );
119   TenseurHHHH & operator = ( const TenseurQ3geneHHHH & B)
120   { return this->operator=((TenseurHHHH &) B); };
121   TenseurHHHH & operator * (const double &) const ;
122   void operator *= ( const double & );
123   TenseurHHHH & operator / ( const double &) const ;
124   void operator /= ( const double & );
125
126   // produit deux fois contracte à droite avec un tenseur du second ordre
127   // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
128   TenseurHH& operator && ( const TenseurBB & ) const ;
129   // produit deux fois contracte à droite avec un tenseur du quatrième ordre
130   // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
131   TenseurHHHH& operator && ( const TenseurBBHH & ) const ;

```



```

132 TenseurHHBB& operator && ( const TenseurBBBB & ) const ;
133
134 //fonctions définissant le produit tensoriel normal de deux tenseurs
135 // *this=aHH(i,j).bHH(k,l) gBi gBj gBk gBl
136 static TenseurHHHH & Prod_tensoriel(const TenseurHH & aHH, const TenseurHH & bHH) ;
137 //fonctions définissant le produit tensoriel barre de deux tenseurs
138 // *this=aHH(i,k).bHH(j,l) gBi gBj gBk gBl
139 static TenseurHHHH & Prod_tensoriel_barre(const TenseurHH & aHH, const TenseurHH & bHH) ;
140 //fonctions définissant le produit tensoriel under_barre de deux tenseurs
141 // *this=aHH(i,l).bHH(j,k) gBi gBj gBk gBl
142 static TenseurHHHH & Prod_tensoriel_under_barre(const TenseurHH & aHH, const TenseurHH & bHH) ;
143
144 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
145 // les 2 premiers indices sont échangés avec les deux derniers indices
146 TenseurHHHH & Transposelet2avec3et4() const ;
147
148 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
149 // plusZero = true: les données manquantes sont mises à 0
150 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
151 // des données possibles
152 void Affectation_trans_dimension(const TenseurHHHH & B, bool plusZero);
153
154 // création d'un tenseur symétrique / au deux premiers indices et / au deux derniers indices
155 // B(i,i,i,i) = A(i,i,i,i); B(i,j,k,k) = 1/2(A(i,j,k,k)+ A(j,i,k,k)); si 2 premiers indices
différents
156 // B(i,i,k,l) = 1/2(A(i,i,k,l)+ A(j,i,l,k)); si 2 derniers indices différents
157 // B(i,j,k,l) = 1/4(A(i,j,k,l)+ A(j,i,k,l) + A(i,j,l,k)+ A(j,i,l,k)); si tous les indices différents
158 TenseurHHHH & Symetriselet2_3et4() const;
159
160 // test
161 int operator == ( const TenseurHHHH & ) const ;
162
163 // change la composante i,j,k,l du tenseur
164 // acces en ecriture,
165 void Change (int i, int j, int k, int l, const double& val) ;
166 // en cumul : équivalent de +=
167 void ChangePlus (int i, int j, int k, int l, const double& val);
168
169 // Retourne la composante i,j,k,l du tenseur
170 // acces en lecture seule
171 double operator () (int i, int j, int k, int l) const ;
172
173 // calcul du maximum en valeur absolu des composantes du tenseur
174 double MaxiComposante() const;
175
176 // lecture et écriture de données
177 istream & Lecture(istream & entree);
178 ostream & Ecriture(ostream & sort) const ;
179
180 protected :
181 // allocator dans la liste de data
182 listdouble81Iter ipointe;
183
184 // fonction pour le produit contracté à gauche
185 TenseurHH& Prod_gauche( const TenseurBB & F) const;
186 TenseurBBHH& Prod_gauche( const TenseurBBBB & F) const;
187 TenseurHHHH& Prod_gauche( const TenseurHHBB & F) const;
188 };
189 //
190 //-----
191 // cas des composantes 4 fois covariantes
192 //-----
193 class TenseurQ3geneBBBB : public TenseurBBBB
194 { // surcharge de l'operator de lecture
195 friend istream & operator » (istream &, TenseurQ3geneBBBB &);
196 // surcharge de l'operator d'écriture
197 friend ostream & operator « (ostream &, const TenseurQ3geneBBBB &);
198
199 public :
200 // Constructeur
201 TenseurQ3geneBBBB() ; // par défaut
202 // initialisation de toutes les composantes a une meme valeur val
203 TenseurQ3geneBBBB(const double val);
204 // initialisation à partir d'un produit tensoriel avec 3 cas
205 // cas = 1 : produit tensoriel normal
206 // *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
207 // cas = 2 : produit tensoriel barre
208 // *this=aBB(i,k).bBB(j,l) gHi gHj gHk gHl
209 // cas = 3 : produit tensoriel under barre
210 // *this=aBB(i,l).bBB(j,k) gHi gHj gHk gHl
211 TenseurQ3geneBBBB(int cas, const TenseurBB & aBB, const TenseurBB & bBB);
212 TenseurQ3geneBBBB(int cas, const Tenseur3BB & aBB, const Tenseur3BB & bBB);
213
214 // DESTRUCTEUR :
215 ~TenseurQ3geneBBBB() ;
216 // constructeur a partir d'une instance non differenciee
217 TenseurQ3geneBBBB (const TenseurBBBB &);

```

```

218 // constructeur de copie
219 TenseurQ3geneBBBB (const TenseurQ3geneBBBB &);
220
221 // METHODES PUBLIQUES :
222 //2) virtuelles
223 // initialise toutes les composantes à val
224 void InitA(double val) ;
225 // operations
226 TenseurBBBB & operator + ( const TenseurBBBB & ) const ;
227 void operator += ( const TenseurBBBB & );
228 TenseurBBBB & operator - ( const TenseurBBBB & ) const ; // oppose du tenseur
229 TenseurBBBB & operator - ( const TenseurBBBB & ) const ;
230 void operator -= ( const TenseurBBBB & );
231 TenseurBBBB & operator = ( const TenseurBBBB & );
232 TenseurBBBB & operator = ( const TenseurQ3geneBBBB & B)
233 { return this->operator=(TenseurBBBB & B); };
234 TenseurBBBB & operator * (const double & ) const ;
235 void operator *= ( const double & );
236 TenseurBBBB & operator / ( const double & ) const ;
237 void operator /= ( const double & );
238
239 // produit deux fois contracte à droite avec un tenseur du second ordre
240 // diffèrent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
241 TenseurBB& operator && ( const TenseurHH & ) const ;
242 // produit deux fois contracte à droite avec un tenseur du quatrième ordre
243 // diffèrent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
244 TenseurBBBB& operator && ( const TenseurHHBB & ) const ;
245 TenseurBBHH& operator && ( const TenseurHHHH & ) const ;
246
247 //fonctions définissant le produit tensoriel normal de deux tenseurs
248 // *this=aBB(i,j).bBB(k,l) gHi gHj gHk gHl
249 static TenseurBBBB & Prod_tensoriel(const TenseurBB & aBB, const TenseurBB & bBB) ;
250 //fonctions définissant le produit tensoriel barre de deux tenseurs
251 // *this=aBB(i,k).bBB(j,l) gHi gHj gHk gHl
252 static TenseurBBBB & Prod_tensoriel_barre(const TenseurBB & aBB, const TenseurBB & bBB) ;
253 //fonctions définissant le produit tensoriel under_barre de deux tenseurs
254 // *this=aBB(i,l).bBB(j,k) gHi gHj gHk gHl
255 static TenseurBBBB & Prod_tensoriel_under_barre(const TenseurBB & aBB, const TenseurBB & bBB) ;
256
257 // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
258 // les 2 premiers indices sont échangés avec les deux derniers indices
259 TenseurBBBB & Transposelet2avec3et4() const ;
260
261 // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
262 // plusZero = true: les données manquantes sont mises à 0
263 // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
264 // des données possibles
265 void Affectation_trans_dimension(const TenseurBBBB & B, bool plusZero);
266
267 // création d'un tenseur symétrique / au deux premiers indices et / au deux derniers indices
268 // B(i,i,i,i) = A(i,i,i,i); B(i,j,k,k) = 1/2(A(i,j,k,k) + A(j,i,k,k)); si 2 premiers indices
différents
269 // B(i,i,k,l) = 1/2(A(i,i,k,l) + A(j,i,l,k)); si 2 derniers indices différents
270 // B(i,j,k,l) = 1/4(A(i,j,k,l) + A(j,i,k,l) + A(i,j,l,k) + A(j,i,l,k)); si tous les indices différents
271 TenseurBBBB & Symetriselet2_3et4() const;
272
273 // test
274 int operator == ( const TenseurBBBB & ) const ;
275
276 // Retourne la composante i,j,k,l du tenseur
277 // acces en ecriture,
278 void Change (int i, int j, int k, int l, const double& val) ;
279 // en cumul : équivalent de +=
280 void ChangePlus (int i, int j, int k, int l, const double& val);
281
282 // Retourne la composante i,j,k,l du tenseur
283 // acces en lecture seule
284 double operator () (int i, int j, int k, int l) const ;
285
286 // calcul du maximum en valeur absolu des composantes du tenseur
287 double MaxiComposante() const;
288
289 // lecture et écriture de données
290 istream & Lecture(istream & entree);
291 ostream & Ecriture(ostream & sort) const ;
292
293 protected :
294 // allocator dans la liste de data
295 listdouble8lIter ipointe;
296
297 // fonction pour le poduit contracté à gauche
298 TenseurBB& Prod_gauche( const TenseurHH & F) const;
299 TenseurHHBB& Prod_gauche( const TenseurHHHH & F) const;
300 TenseurBBBB& Prod_gauche( const TenseurBBHH & F) const;
301 };
302
303 //

```

```

304 //-----
305 //          cas des composantes mixte 3BBHH
306 //-----
307
308 class TenseurQ3geneBBHH : public TenseurBBHH
309 { // surcharge de l'operator de lecture
310   friend istream & operator » (istream &, TenseurQ3geneBBHH &);
311   // surcharge de l'operator d'écriture
312   friend ostream & operator « (ostream &, const TenseurQ3geneBBHH &);
313
314 public :
315   // Constructeur
316   TenseurQ3geneBBHH() ; // par défaut
317   // initialisation de toutes les composantes a une meme valeur val
318   TenseurQ3geneBBHH(const double val);
319   // initialisation à partir d'un produit tensoriel normal
320   // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
321   TenseurQ3geneBBHH(const TenseurBB & aBB, const TenseurHH & bHH);
322   TenseurQ3geneBBHH(const Tenseur3BB & aBB, const Tenseur3HH & bHH);
323
324   // DESTRUCTEUR :
325   ~TenseurQ3geneBBHH() ;
326   // constructeur a partir d'une instance non differenciee
327   TenseurQ3geneBBHH (const TenseurBBHH &);
328   // constructeur de copie
329   TenseurQ3geneBBHH (const TenseurQ3geneBBHH &);
330
331   // METHODES PUBLIQUES :
332   //2) virtuelles
333   // initialise toutes les composantes à val
334   void Inita(double val) ;
335   // operations
336   TenseurBBHH & operator + ( const TenseurBBHH & ) const ;
337   void operator += ( const TenseurBBHH &);
338   TenseurBBHH & operator - ( ) const ; // oppose du tenseur
339   TenseurBBHH & operator - ( const TenseurBBHH & ) const ;
340   void operator -= ( const TenseurBBHH &);
341   TenseurBBHH & operator = ( const TenseurBBHH &);
342   TenseurBBHH & operator * (const double & ) const ;
343   void operator *= ( const double &);
344   TenseurBBHH & operator / ( const double & ) const ;
345   void operator /= ( const double &);
346
347   // produit deux fois contracte à droite avec un tenseur du second ordre
348   // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
349   TenseurBB & operator && ( const TenseurBB & ) const ;
350   // produit deux fois contracte à droite avec un tenseur du quatrième ordre
351   // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
352   TenseurBBBB & operator && ( const TenseurBBBB & ) const ;
353   TenseurBBHH & operator && ( const TenseurBBHH & ) const ;
354
355   //fonctions définissant le produit tensoriel normal de deux tenseurs
356   // *this=aBB(i,j).bHH(k,l) gHi gHj gBk gBl
357   static TenseurBBHH & Prod_tensoriel(const TenseurBB & aBB, const TenseurHH & bHH) ;
358
359   // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
360   // les 2 premiers indices sont échangés avec les deux derniers indices
361   TenseurHHBB & Transposelet2avec3et4() const ;
362
363   // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
364   // plusZero = true: les données manquantes sont mises à 0
365   // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
366   // des données possibles
367   void Affectation_trans_dimension(const TenseurBBHH & B,bool plusZero);
368
369   // test
370   int operator == ( const TenseurBBHH & ) const ;
371
372   // Retourne la composante i,j,k,l du tenseur
373   // acces en écriture,
374   void Change (int i, int j, int k, int l,const double& val) ;
375   // en cumul : équivalent de +=
376   void ChangePlus (int i, int j, int k, int l,const double& val);
377
378   // Retourne la composante i,j,k,l du tenseur
379   // acces en lecture seule
380   double operator () (int i, int j, int k, int l) const ;
381
382   // calcul du maximum en valeur absolu des composantes du tenseur
383   double MaxiComposante() const;
384
385   // lecture et écriture de données
386   istream & Lecture(istream & entree);
387   ostream & Ecriture(ostream & sort) const ;
388
389 protected :
390   // allocator dans la liste de data

```

```

391     listdouble81Iter ipointe;
392
393     // fonction pour le produit contracté à gauche
394     TenseurHH& Prod_gauche( const TenseurHH & F) const;
395     TenseurBBHH& Prod_gauche( const TenseurBBHH & F) const;
396     TenseurHHHH& Prod_gauche( const TenseurHHHH & F) const;
397 };
398 //
399 //-----
400 //         cas des composantes mixte 3HHBB
401 //-----
402
403 class TenseurQ3geneHHBB : public TenseurHHBB
404 { // surcharge de l'operator de lecture
405     friend istream & operator » (istream &, TenseurQ3geneHHBB &);
406     // surcharge de l'operator d'écriture
407     friend ostream & operator « (ostream &, const TenseurQ3geneHHBB &);
408
409     public :
410         // Constructeur
411         TenseurQ3geneHHBB () ; // par défaut
412         // initialisation de toutes les composantes a une meme valeur val
413         TenseurQ3geneHHBB(const double val);
414
415         // initialisation à partir d'un produit tensoriel normal
416         // *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
417         TenseurQ3geneHHBB(const TenseurHH & aHH, const TenseurBB & bBB);
418         TenseurQ3geneHHBB(const Tenseur3HH & aHH, const Tenseur3BB & bBB);
419
420         // DESTRUCTEUR :
421         ~TenseurQ3geneHHBB() ;
422         // constructeur a partir d'une instance non differenciee
423         TenseurQ3geneHHBB (const TenseurHHBB &);
424         // constructeur de copie
425         TenseurQ3geneHHBB (const TenseurQ3geneHHBB &);
426
427         // METHODES PUBLIQUES :
428         //2) virtuelles
429         // initialise toutes les composantes à val
430         void Inita(double val) ;
431         // operations
432         TenseurHHBB & operator + ( const TenseurHHBB &) const ;
433         void operator += ( const TenseurHHBB &);
434         TenseurHHBB & operator - ( const TenseurHHBB &) const ; // oppose du tenseur
435         TenseurHHBB & operator - ( const TenseurHHBB &) const ;
436         void operator -= ( const TenseurHHBB &);
437         TenseurHHBB & operator = ( const TenseurHHBB &);
438         TenseurHHBB & operator * (const double &) const ;
439         void operator *= ( const double &);
440         TenseurHHBB & operator / ( const double &) const ;
441         void operator /= ( const double &);
442
443         // produit deux fois contracte à droite avec un tenseur du second ordre
444         // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
445         TenseurHH& operator && ( const TenseurHH &) const ;
446         // produit deux fois contracte à droite avec un tenseur du quatrième ordre
447         // différent à gauche !! def dans TenseurQ.h à l'aide de la fonction privée Prod_gauche
448         TenseurHHHH& operator && ( const TenseurHHHH &) const ;
449         TenseurHHBB& operator && ( const TenseurHHBB &) const ;
450
451         //fonctions définissant le produit tensoriel normal de deux tenseurs
452         // *this=aHH(i,j).bBB(k,l) gBi gBj gHk gHl
453         static TenseurHHBB & Prod_tensoriel(const TenseurHH & aHH, const TenseurBB & bBB) ;
454
455         // ATTENTION creation d'un tenseur transpose qui est supprime par Libere
456         // les 2 premiers indices sont échangés avec les deux derniers indices
457         TenseurBBHH & Transposelet2avec3et4() const ;
458
459         // affectation de B dans this, plusZero = false: les données manquantes sont inchangées,
460         // plusZero = true: les données manquantes sont mises à 0
461         // si au contraire la dimension de B est plus grande que *this, il y a uniquement affectation
462         // des données possibles
463         void Affectation_trans_dimension(const TenseurHHBB & B,bool plusZero);
464
465         // test
466         int operator == ( const TenseurHHBB &) const ;
467
468         // Retourne la composante i,j,k,l du tenseur
469         // acces en écriture,
470         void Change (int i, int j, int k, int l,const double& val) ;
471         // en cumul : équivalent de +=
472         void ChangePlus (int i, int j, int k, int l,const double& val);
473
474         // Retourne la composante i,j,k,l du tenseur
475         // acces en lecture seule
476         double operator () (int i, int j, int k, int l) const ;
477

```

```

478 // calcul du maximum en valeur absolu des composantes du tenseur
479 double MaxiComposante() const;
480
481 // lecture et écriture de données
482 istream & Lecture(istream & entree);
483 ostream & Ecriture(ostream & sort) const ;
484
485 protected :
486 // allocator dans la liste de data
487 listdouble81Iter ipointe;
488
489 // fonction pour le produit contracté à gauche
490 TenseurBB& Prod_gauche( const TenseurBB & F) const;
491 TenseurHHBB& Prod_gauche( const TenseurHHBB & F) const;
492 TenseurBBBB& Prod_gauche( const TenseurBBBB & F) const;
493 };
494
495 #ifndef MISE_AU_POINT
496 #include "TenseurQ3gene-1.cc"
497 #include "TenseurQ3gene-2.cc"
498 #define TenseurQ3gene_H_deja_inclus
499 #endif
500
501
502 #endif

```

## 7.479 TypeConsTens.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 #ifndef TYPECONSTENS_H
33 #define TYPECONSTENS_H
34
35 #include "Tenseur.h"
36 #include "Tenseur1.h"
37 #include "Tenseur2.h"
38 #include "Tenseur3.h"
39 #include "TenseurQ-3.h"
40 #include "TenseurQ-2.h"
41 #include "TenseurQ-1.h"
42
43 // la definition exacte de ces variables globales est faite dans le fichier TypeConsTensPrin.h
44 // qui n'est inclus qu'une seule fois dans le fichier du programme principal
45
46 // constantes dont la dimensions est connu
47
48 // ---- tenseurs d'ordre 2 ----
49 extern Tenseur1HH IdHH1, ZeroHH1; // def du tenseur identite et du tenseur nul
50 extern Tenseur1BB IdBB1, ZeroBB1; // def du tenseur identite et du tenseur nul
51 extern Tenseur1HB IdHB1, ZeroHB1; // def du tenseur identite et du tenseur nul
52 extern Tenseur1BH IdBH1, ZeroBH1; // def du tenseur identite et du tenseur nul
53
54 extern Tenseur2HH IdHH2, ZeroHH2; // def du tenseur identite et du tenseur nul
55 extern Tenseur2BB IdBB2, ZeroBB2; // def du tenseur identite et du tenseur nul
56 extern Tenseur2HB IdHB2, ZeroHB2; // def du tenseur identite et du tenseur nul
57 extern Tenseur2BH IdBH2, ZeroBH2; // def du tenseur identite et du tenseur nul

```

```

58
59 extern Tenseur3HH IdHH3,ZeroHH3; // def du tenseur identite et du tenseur nul
60 extern Tenseur3BB IdBB3,ZeroBB3; // def du tenseur identite et du tenseur nul
61 extern Tenseur3HB IdHB3,ZeroHB3; // def du tenseur identite et du tenseur nul
62 extern Tenseur3BH IdBH3,ZeroBH3; // def du tenseur identite et du tenseur nul
63
64 // ---- tenseurs d'ordre 4 (36 coeff) ----
65 extern Tenseur3HHHH IdHHHH3; // def du tenseur identite (delta^{ij} delta_{kl})
66 extern Tenseur3BBBB IdBBBB3; // def du tenseur identite (delta_{ij} delta_{kl})
67 extern Tenseur3HHBB IdHHBB3; // def du tenseur identite (delta^{ij} delta_{kl})
68 extern Tenseur3BBHH IdBBHH3; // def du tenseur identite (delta_{ij} delta_{kl})
69
70 extern Tenseur3HHHH PIdHHHH3; // def du tenseur identite (delta^{ik} delta^{jl})
71 extern Tenseur3BBBB PIdBBBB3; // def du tenseur identite (delta_{ik} delta_{jl})
72
73 // ---- tenseurs d'ordre 4 (9 coeff) ----
74 extern Tenseur2HHHH IdHHHH2; // def du tenseur identite (delta^{ij} delta_{kl})
75 extern Tenseur2BBBB IdBBBB2; // def du tenseur identite (delta_{ij} delta_{kl})
76 extern Tenseur2HHBB IdHHBB2; // def du tenseur identite (delta^{ij} delta_{kl})
77 extern Tenseur2BBHH IdBBHH2; // def du tenseur identite (delta_{ij} delta_{kl})
78
79 extern Tenseur2HHHH PIdHHHH2; // def du tenseur identite (delta^{ik} delta^{jl})
80 extern Tenseur2BBBB PIdBBBB2; // def du tenseur identite (delta_{ik} delta_{jl})
81
82 // ---- tenseurs d'ordre 4 (1 coeff) ----
83 extern Tenseur1HHHH IdHHHH1; // def du tenseur identite (delta^{ij} delta_{kl})
84 extern Tenseur1BBBB IdBBBB1; // def du tenseur identite (delta_{ij} delta_{kl})
85 extern Tenseur1HHBB IdHHBB1; // def du tenseur identite (delta^{ij} delta_{kl})
86 extern Tenseur1BBHH IdBBHH1; // def du tenseur identite (delta_{ij} delta_{kl})
87
88 extern Tenseur1HHHH PIdHHHH1; // def du tenseur identite (delta^{ik} delta^{jl})
89 extern Tenseur1BBBB PIdBBBB1; // def du tenseur identite (delta_{ik} delta_{jl})
90
91 // constantes dont la dimensions est celle du pb
92 // defini au debut du pb avec la fonction : ConstantesTenseur(int dim)
93
94 extern TenseurHH * IdHH,* ZeroHH; // def du tenseur identite et du tenseur nul
95 extern TenseurBB * IdBB,* ZeroBB; // def du tenseur identite et du tenseur nul
96 extern TenseurHB * IdHB,* ZeroHB; // def du tenseur identite et du tenseur nul
97 extern TenseurBH * IdBH,* ZeroBH; // def du tenseur identite et du tenseur nul
98
99 // tableau des constantes, fonction de la dimension
100 extern Tableau <TenseurHB *> Id_dim_HB; // tenseur identite fonction de la dimension:
101
102
103 #endif

```

## 7.480 TypeConstensPrinc.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty
23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 // dans le fichier principal
33 // constantes dont la dimensions est connu
34
35 // ---- tenseurs d'ordre 2 ----
36 Tenseur1HH IdHH1(1.),ZeroHH1(0.); // def du tenseur identite et du tenseur nul

```

```

37 Tenseur1BB IdBB1(1.),ZeroBB1(0.); // def du tenseur identite et du tenseur nul
38 Tenseur1HB IdHB1(1.),ZeroHB1(0.); // def du tenseur identite et du tenseur nul
39 Tenseur1BH IdBH1(1.),ZeroBH1(0.); // def du tenseur identite et du tenseur nul
40
41 Tenseur2HH IdHH2(1.,1.,0.) ,ZeroHH2(0.); // def du tenseur identite et du tenseur nul
42 Tenseur2BB IdBB2(1.,1.,0.) ,ZeroBB2(0.); // def du tenseur identite et du tenseur nul
43 Tenseur2HB IdHB2(1.,1.,0.,0.),ZeroHB2(0.); // def du tenseur identite et du tenseur nul
44 Tenseur2BH IdBH2(1.,1.,0.,0.),ZeroBH2(0.); // def du tenseur identite et du tenseur nul
45
46 Tenseur3HH IdHH3(1.,1.,1.,0.,0.,0.) ,ZeroHH3(0.); // def du tenseur identite et du tenseur nul
47 Tenseur3BB IdBB3(1.,1.,1.,0.,0.,0.) ,ZeroBB3(0.); // def du tenseur identite et du tenseur nul
48 Tenseur3HB IdHB3(1.,0.,0.,0.,1.,0.,0.,1.),ZeroHB3(0.); // def du tenseur identite et du tenseur nul
49 Tenseur3BH IdBH3(1.,0.,0.,0.,1.,0.,0.,1.),ZeroBH3(0.); // def du tenseur identite et du tenseur nul
50
51 // ---- tenseurs d'ordre 4 (36 coeff) ----
52 Tenseur3HHHH IdHHHH3(true,IdHH3,IdHH3); // def du tenseur identite (delta^{ij} delta^{kl})
53 Tenseur3BBBB IdBBBB3(true,IdBB3,IdBB3); // def du tenseur identite (delta_{ij} delta_{kl})
54 Tenseur3HHBB IdHHBB3(IdHH3,IdBB3); // def du tenseur identite (delta^{ij} delta_{kl})
55 Tenseur3BBHH IdBBHH3(IdBB3,IdHH3); // def du tenseur identite (delta_{ij} delta^{kl})
56
57 Tenseur3HHHH PIIdHHHH3(false,IdHH3,IdHH3); // def du tenseur identite (delta^{ik} delta^{jl})
58 Tenseur3BBBB PIIdBBBB3(false,IdBB3,IdBB3); // def du tenseur identite (delta_{ik} delta_{jl})
59
60 // ---- tenseurs d'ordre 4 (9 coeff) ----
61 Tenseur2HHHH IdHHHH2(true,IdHH2,IdHH2); // def du tenseur identite (delta^{ij} delta^{kl})
62 Tenseur2BBBB IdBBBB2(true,IdBB2,IdBB2); // def du tenseur identite (delta_{ij} delta_{kl})
63 Tenseur2HHBB IdHHBB2(IdHH2,IdBB2); // def du tenseur identite (delta^{ij} delta_{kl})
64 Tenseur2BBHH IdBBHH2(IdBB2,IdHH2); // def du tenseur identite (delta_{ij} delta^{kl})
65
66 Tenseur2HHHH PIIdHHHH2(false,IdHH2,IdHH2); // def du tenseur identite (delta^{ik} delta^{jl})
67 Tenseur2BBBB PIIdBBBB2(false,IdBB2,IdBB2); // def du tenseur identite (delta_{ik} delta_{jl})
68
69 // ----- tenseurs d'ordre 4 (1 coeff) -----
70 Tenseur1HHHH IdHHHH1(true,IdHH1,IdHH1); // def du tenseur identite (delta^{ij} delta^{kl})
71 Tenseur1BBBB IdBBBB1(true,IdBB1,IdBB1); // def du tenseur identite (delta_{ij} delta_{kl})
72 Tenseur1HHBB IdHHBB1(IdHH1,IdBB1); // def du tenseur identite (delta^{ij} delta_{kl})
73 Tenseur1BBHH IdBBHH1(IdBB1,IdHH1); // def du tenseur identite (delta_{ij} delta^{kl})
74
75 Tenseur1HHHH PIIdHHHH1(false,IdHH1,IdHH1); // def du tenseur identite (delta^{ik} delta^{jl})
76 Tenseur1BBBB PIIdBBBB1(false,IdBB1,IdBB1); // def du tenseur identite (delta_{ik} delta_{jl})
77
78
79 // constantes dont la dimensions est celle du pb
80 // defini au debut du pb avec la fonction : ConstantesTenseur(int dim)
81
82 TenseurHH * IdHH,* ZeroHH; // def du tenseur identite
83 TenseurBB * IdBB,* ZeroBB; // def du tenseur identite
84 TenseurHB * IdHB,* ZeroHB; // def du tenseur identite
85 TenseurBH * IdBH,* ZeroBH; // def du tenseur identite
86 Tableau <TenseurHB *> Id_dim_HB(3); // tenseur identite fonction de la dimension:
87 // Id_dim_HB(1) = (TenseurHB*) &IdHB1;
88 // Id_dim_HB(2) = (TenseurHB*) &IdHB2;
89 // Id_dim_HB(3) = (TenseurHB*) &IdHB3;

```

## 7.481 Vecteur.h

```

1 // FICHER : Vecteur.h
2 // CLASSE : Vecteur
3
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.

```

```

30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33
34 /*****
35 *   DATE:      23/01/97
36 *
37 *   AUTEUR:    G RIO
38 *
39 *   PROJET:    Herezh++
40 *
41 *****/
42 *
43 // La classe Vecteur permet de declarer des vecteurs d'une longueur predefinie par
44 // une allocation dynamique de memoire. Les composantes d'un vecteur de cette classe
45 // sont de type double.
46 // correspond au vecteur absolu ( par opposition avec des vecteurs avec des coordonnees
47 // covariantes ou contravariantes
48 *
49 *
50 *   $ *
51 *   MODIFICATIONS:
52 *   ! date !   auteur !   but
53 *   -----
54 *   $ *
55 *****/
56
57
58
59 #ifndef VECTEURS__H
60 #define VECTEURS__H
61
62 // #include "Debug.h"
63
64 #include <iostream>
65 #include <stdlib.h>
66 #include "Sortie.h"
67
68 #include "mvvtp_GR.h" // classe template MV++
69
70
71
72 /** @defgroup Les_classes_Vecteur
73 *
74 // Les classes de type "vecteur" permettent de declarer des vecteurs d'une longueur predefinie par
75 // une allocation dynamique de memoire. Les composantes d'un vecteur
76 // sont de type double.
77 // correspond au vecteur absolu ( par opposition avec des vecteurs avec des coordonnees
78 // covariantes ou contravariantes
79 * \author   Gérard Rio
80 * \version  1.0
81 * \date    23/01/97
82 * \brief   Définition des classes de type "vecteur"
83 *
84 */
85
86
87
88 /// @addtogroup Les_classes_Vecteur
89 /// @{
90 ///
91
92 class Coordonnee; // declare a la fin du fichier, car coordonnees utilise aussi vecteur
93 class CoordonneeH; // declare a la fin du fichier, car coordonnees utilise aussi vecteur
94 class CoordonneeB; // declare a la fin du fichier, car coordonnees utilise aussi vecteur
95
96 /// La classe Vecteur permet de declarer des vecteurs d'une longueur predefinie par
97 /// une allocation dynamique de memoire. Les composantes d'un vecteur de cette classe
98 /// sont de type double.
99 /// Correspond à un vecteur absolu ( par opposition avec des vecteurs avec des coordonnees
100 /// covariantes ou contravariantes
101
102 class Vecteur
103 {
104
105     /// utilisation directe du pointeur de double par la classe MatLapack
106     friend class MatLapack;
107
108     public :
109
110     /// Surcharge de l'operateur * : multiplication entre un scalaire et un vecteur
111     friend Vecteur operator* (double val, const Vecteur& vec);
112     /// surcharge de l'operateur de lecture
113     /// les informations sont le type puis la taille puis les datas
114     friend istream & operator » (istream &, Vecteur &);
115     /// surcharge de l'operateur d'écriture non formatée
116     /// les informations sont le type puis la taille puis les datas séparées par

```



```

117 // un espace
118 friend ostream & operator << (ostream &, const Vecteur &);
119
120 // spécifiquement à la classe MV_Vector<double>
121 // friend MV_Vector<double> * Nouveau_MV_Vector_double(Vecteur & b);
122 // friend const MV_Vector<double> * Nouveau_MV_Vector_double_const(const Vecteur & b);
123
124 // CONSTRUCTEURS :
125
126 // Constructeur par défaut
127 Vecteur ();
128 // Constructeur fonction de la taille du vecteur
129 // initialisation à zéro par défaut
130 Vecteur (int n);
131 // Constructeur fonction de la taille du vecteur et d'une
132 // valeur d'initialisation pour les composantes
133 Vecteur (int n,double val);
134 // Constructeur fonction d'un tableau de composantes de type T
135 // aucun test n'est fait concernant la taille disponible de vec
136 // routine dangereuse !!!!!
137 Vecteur (int n,double* vec);
138 // Constructeur fonction d'un Point
139 // trop dangereux, car en fait il y a des conversions implicites dans des calculs par exemple, qui se
    passent
140 // sans qu'on le voit explicitement
141 //il vaut mieux utiliser la conversion explicite (même si ça parait un peu plus long, mais je crois que
    l'on gagne au final
142 // Vecteur (const Coordonnee& a);
143 // Vecteur (const CoordonneeH& a);
144 // Vecteur (const CoordonneeB& a);
145
146 // constructeur fonction d'un MV_Vector : oui car c'est un cas particulier (différent des
    coordonnées)
147 Vecteur (const MV_Vector <double>& a);
148 // Constructeur de copie
149 Vecteur (const Vecteur& vec);
150
151
152 // DESTRUCTEUR :
153 ~Vecteur ();
154
155
156 // METHODES :
157
158 // Retourne la taille du vecteur
159 int Taille () const;
160 // Affichage des composantes du vecteur
161 void Affiche () const ;
162 // Affichage des composantes du vecteur avec paramètres de contrôle
163 void Affiche (int min_i,int max_i,int pas_i, int nbdiggit) const ;
164 // affichage à l'écran après une demande interactive du vecteur
165 // entete: une chaine explicative, a afficher en entête
166 void Affichage_ecran(string entete) const;
167
168 // Change la taille du vecteur (la nouvelle taille est n)
169 // si la nouvelle taille est inférieur, on garde les valeurs existantes du vecteur
170 // si la nouvelle taille est supérieur, on garde les valeurs existantes et les
171 // nouvelles valeurs sont mises à 0
172 void Change_taille (int n);
173 // Change la taille du vecteur avec initialisation de tous le nouveau vecteur
174 void Change_taille (int n,const double& val_init);
175 // Permet de desallouer l'ensemble des elements du vecteur
176 void Libere();
177 // Calcul du maximum en valeur absolu des composantes du vecteur
178 double Max_val_abs () const ;
179 // Calcul du maximum en valeur absolu des composantes du vecteur
180 // ramene également l'indice de tableau du maximum
181 double Max_val_abs (int& i) const ;
182 // Calcul du maximum en valeur absolu des composantes du vecteur
183 // mais ramène la grandeur signée (avec son signe)
184 double Max_val_abs_signe() const ;
185 // Calcul du maximum en valeur absolu des composantes du vecteur
186 // mais ramène la grandeur signée (avec son signe)
187 // ramene également l'indice de tableau du maximum
188 double Max_val_abs_signe(int& i) const ;
189 // calcul, récupération et affichage éventuelle
190 // des mini, maxi, et en valeur absolue la moyenne des composantes du vecteur
191 // en retour: le min, le max et la moyenne en valeur absolue
192 Coordonnee MinMaxMoy(bool affiche) const;
193
194 // Calcul de la norme euclidienne des composantes du vecteur
195 double Norme () const ;
196 // norme le vecteur
197 Vecteur& Normer () ;
198 // Calcul du produit scalaire entre deux vecteurs
199 double Prod_scal (const Vecteur& vec) const ;
200 // somme de tous les composantes du vecteur

```

```

201     double Somme() const ;
202     /// somme de tous les valeurs absolues des composantes du vecteur
203     double SommeAbs() const ;
204     /// moyenne de tous les composantes du vecteur
205     double Moyenne() const ;
206     /// Surcharge de l'operateur + : addition entre deux vecteurs
207     Vecteur operator+ (const Vecteur& vec) const ;
208     /// Surcharge de l'operateur - : soustraction entre deux vecteurs
209     Vecteur operator- (const Vecteur& vec) const ;
210     /// Surcharge de l'operateur - : oppose d'un vecteur
211     Vecteur operator- () const;
212     /// Surcharge de l'operateur +=
213     void operator+= (const Vecteur& vec);
214     /// Surcharge de l'operateur -=
215     void operator-= (const Vecteur& vec);
216     /// Surcharge de l'operateur *= : multiplication d'un scalaire par un vecteur
217     void operator*= (double val);
218     /// Surcharge de l'operateur == : test d'egalite entre deux vecteurs
219     int operator== (const Vecteur& vec) const ;
220     /// Surcharge de l'operateur !=
221     /// Renvoie 1 si les deux vecteurs ne sont pas egaux
222     /// Renvoie 0 sinon
223     int operator!= (const Vecteur& vec) const ;
224     /// Surcharge de l'operateur = : realise l'egalite de deux vecteurs
225     Vecteur& operator= (const Vecteur& vec);
226     // Surcharge de l'operateur = : affectation a partir d'un point
227 // trop dangereux, il vaut mieux utiliser la conversion explicite
228 // Vecteur& operator= ( const Coordonnee& point);
229 // Vecteur& Egale_Coordonnee( const Coordonnee& point);
230     /// affectation avec un MV_Vecteur
231     Vecteur& operator= ( const MV_Vector<double> & A);
232     /// Surcharge de l'operateur * : multiplication d'un vecteur par un scalaire
233     Vecteur operator* (double val) const ;
234     /// Surcharge de l'operateur / : division des composantes d'un vecteur par un scalaire
235     Vecteur operator/ ( const double val) const ;
236     /// Surcharge de l'operateur /= : division des composantes d'un vecteur par un scalaire
237     void operator/= ( const double val);
238     /// Surcharge de l'operateur * : multiplication entre deux vecteurs
239     /// Permet de realiser le produit scalaire entre deux vecteurs
240     double operator* (const Vecteur& vec) const ;
241     /// Retourne la ieme composante du vecteur : acces en lecture uniquement
242     double operator() (int i) const;
243     /// Retourne la ieme composante du vecteur : acces en lecture et en ecriture
244     double& operator() (int i);
245     /// Surcharge de l'operateur ! : renvoie la norme d'un vecteur
246     double operator! () const ;
247     /// mise a zero d'un vecteur
248     void Zero();
249     /// initialise toutes les composantes à val
250     void Inita(double val);
251     /// calcul le produit P(i) = (*this)(i) * A(i); ramène P qui est passé en paramètre
252     Vecteur& Prod_ii(const Vecteur& A, Vecteur& P) const;
253     /// -- création explicite de coordonnees équivalentes
254     Coordonnee Coordo() const;
255     /// -- création explicite de coordonnees équivalentes
256     CoordonneeH CoordoH() const;
257     /// -- création explicite de coordonnees équivalentes
258     CoordonneeB CoordoB() const;
259
260     /// affectation à une partie d'un vecteur qui doit avoir une taille donc plus grande
261     /// (*this)(i) = vec(i+indice-1); i= 1 à la taille de (*this)
262     /// ramène *this
263     Vecteur& Egale_une_partie_de(const Vecteur& vec,int indice);
264     /// affectation d'une partie de (*this) au vecteur passé en paramètre
265     /// (*this)(i+indice-1) = vec(i); i= 1 à la taille de vec
266     /// ramène *this
267     Vecteur& Une_partie_egale(int indice, const Vecteur& vec);
268
269     /// methode pour la compatibilite avec des classes derivees
270     Vecteur & Vect();
271     /// ramène un vecteur MV_Vector <double> qui utilise la même place
272     /// mémoire que this ce qui permet d'utiliser le type MV_Vector
273     MV_Vector <double> MV_vecteur_double();
274     const MV_Vector <double> MV_vecteur_double() const ;
275     /// spécifiquement à la classe MV_Vector<double>
276     /// ramène un pointeur de nouveau vecteur MV_Vector <double> qui utilise la même place
277     /// mémoire que this ce qui permet d'utiliser le type MV_Vector
278     MV_Vector<double> * Nouveau_MV_Vector_double();
279     /// idem en const
280     const MV_Vector<double> * Nouveau_MV_Vector_double_const() const;
281
282     protected :
283
284
285     int taille; // taille du vecteur
286     double* v; // pointeur sur les composantes du vecteur
287

```

```

288
289 // fonction protégé pour l'accès directe au pointeur, utilisé
290 // par MV_Vector<double> pour créer un vecteur à la même place
291 double * Pointeur_vect() {return v;}
292
293 };
294
295 /// @} // end of group
296
297 /// @addtogroup Les_classes_Vecteur
298 /// @{
299 ///
300
301
302 /// définition d'une classe de jonction entre les Vecteurs et les MV_Vecteurs
303 class MV_Vecteur : public MV_Vector<double>, public Vecteur
304 {
305
306 // Pour l'instant il y a des possibilités de pb due à des possibles
307 // non synchronisation entre des modifications des deux parties
308 public :
309 /// par défaut
310 MV_Vecteur () { }; // défaut
311
312 /// utilise la même place mémoire que vec
313 MV_Vecteur (const Vecteur& vec) :
314 Vecteur(vec)
315 { p_ = v ; dim_ = taille ; ref_=1;};
316 /// de copie
317 MV_Vecteur (const MV_Vector<double>& vec) :
318 MV_Vector<double>(vec)
319 { taille = dim_; v = p_;ref_=1;};
320
321 /// Change la taille du vecteur (la nouvelle taille est n)
322 void Change_taille (int )
323 { cout << "\n erreur !! fonction pour l'instant interdite "
324 << " dans le cas d'objet de type MV_Vecteur "
325 << "\n MV_Vecteur::Change_taille (int n)";
326 Sortie(1);
327 };
328 /// Permet de desallouer l'ensemble des elements du vecteur
329 void Libere()
330 { cout << "\n erreur !! fonction pour l'instant interdite "
331 << " dans le cas d'objet de type MV_Vecteur "
332 << "\n MV_Vecteur::Libere()";
333 Sortie(1);
334 };
335
336
337 };
338 /// @} // end of group
339
340 #ifndef MISE_AU_POINT
341 #include "Vecteur.cc"
342 #define VECTEUR_H_deja_inclus
343 #endif
344
345 //#include "Coordonnee.h"
346
347 #endif

```

## 7.482 VeurPropre.h

```

1
2
3 // This file is part of the Herezh++ application.
4 //
5 // The finite element software Herezh++ is dedicated to the field
6 // of mechanics for large transformations of solid structures.
7 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
8 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
9 //
10 // Herezh++ is distributed under GPL 3 license ou ultérieure.
11 //
12 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
13 // AUTHOR : Gérard Rio
14 // E-MAIL : gerardrio56@free.fr
15 //
16 // This program is free software: you can redistribute it and/or modify
17 // it under the terms of the GNU General Public License as published by
18 // the Free Software Foundation, either version 3 of the License,
19 // or (at your option) any later version.
20 //
21 // This program is distributed in the hope that it will be useful,
22 // but WITHOUT ANY WARRANTY; without even the implied warranty

```

```

23 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
24 // See the GNU General Public License for more details.
25 //
26 // You should have received a copy of the GNU General Public License
27 // along with this program. If not, see <https://www.gnu.org/licenses/>.
28 //
29 // For more information, please consult: <https://herezh.irdl.fr/>.
30
31
32 /*****
33 *      UNIVERSITE DE BRETAGNE SUD  --- I.U.P/I.U.T. DE LORIENT  *
34 *****/
35 *      LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX  *
36 *      Tel 97.80.80.60  *
37 *      Centre de Genie Industriel 56520 GUIDEL-PLAGES  *
38 *****/
39 *      DATE:      14/05/98  *
40 *      $  *
41 *      AUTEUR:    G RIO  *
42 *      $  *
43 *      PROJET:    Herezh++  *
44 *      $  *
45 *****/
46 *      BUT: Stockage d'une valeur propre et d'un vecteur propre éventuel *
47 *      $  *
48 *      *****/
49 *      VERIFICATION:  *
50 *      *
51 *      ! date ! auteur ! but ! *
52 *      ----- *
53 *      ! ! ! ! ! *
54 *      $  *
55 *      *****/
56 *      MODIFICATIONS:  *
57 *      ! date ! auteur ! but ! *
58 *      ----- *
59 *      $  *
60 *****/
61 #ifndef VEURPROPRE_H
62 #define VEURPROPRE_H
63
64 #include "Vecteur.h"
65
66 class VeurPropre
67 {
68 // surcharge de l'operator de lecture
69 // les informations sont le type puis les datas
70 friend istream & operator » (istream &, VeurPropre &);
71 // surcharge de l'operator d'écriture non formatée
72 // les informations sont le type puis les datas séparées par
73 // un espace
74 friend ostream & operator « (ostream &, const VeurPropre &);
75
76 public :
77 // CONSTRUCTEURS :
78 // définition d'une instance qui ne contient qu'une valeur propre
79 VeurPropre();
80 // définition d'une instance qui contient une valeur propre et
81 // un vecteur propre de dimension n
82 VeurPropre(int n);
83 // définition d'une instance à partir d'une valeur et d'un vecteur propre
84 VeurPropre(double val, Vecteur v );
85
86 // constructeur de copie
87 VeurPropre(const VeurPropre& a);
88
89 // DESTRUCTEUR :
90 ~VeurPropre();
91
92 // surcharge de l'affectation
93 VeurPropre& operator= (const VeurPropre& a);
94
95 // METHODES PUBLIQUES :
96 // retourne la valeur propre
97 inline double& Val() { return val;};
98 // retourne le vecteur propre
99 inline Vecteur& Pv() {return *pv;};
100 // Affichage de la valeur propre et du vecteur correspondant.
101 void Affiche () const ;
102
103 private :
104 // VARIABLES PROTEGEES :
105 double val ; // la valeur propre
106 Vecteur * pv; // contenant un vecteur propre éventuel
107
108

```

```

109 // METHODES PROTEGEES :
110
111 };
112
113 #endif

```

## 7.483 Basiques.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      14/03/2003
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 * *****/
37 *   BUT:      conteneurs ultra basiques, permettant entre autres
38 *            la différenciation des types.
39 *
40 *   *****
41 *
42 * *****/
43 #ifndef BASIQUES_H
44 #define BASIQUES_H
45
46 #include <iostream>
47 #include <fstream>
48 #include <string.h>
49 #include <string>
50 #include "Enum_IO_XML.h"
51 using namespace std;
52
53 /** @defgroup Les_conteneurs_ultra_basiques
54 *
55 *   BUT:      conteneurs ultra basiques, permettant entre autres
56 *            la différenciation des types.
57 *
58 *
59 *   \author   Gérard Rio
60 *   \version  1.0
61 *   \date    14/03/2003
62 *   \brief   Listes des conteneurs ultra basiques
63 *
64 */
65
66 /// @addtogroup Les_conteneurs_ultra_basiques
67 /// @{
68 ///
69
70 /// cas de 2 entiers
71 class DeuxEntiers
72 {
73 public :
74     // VARIABLES PUBLIQUES :
75     int un,deux;
76     // gestion schéma XML
77     static short int impre_schem_XML;

```

```

78
79 // CONSTRUCTEURS :
80 DeuxEntiers() : un(0),deux(0) {};
81 DeuxEntiers(int u, int v) : un(u),deux(v) {};
82 DeuxEntiers(const DeuxEntiers & de) : un(de.un),deux(de.deux) {};
83
84 // DESTRUCTEUR :
85 ~DeuxEntiers() {};
86
87 // METHODES PUBLIQUES :
88 // surcharge de l'affectation
89 DeuxEntiers& operator= (const DeuxEntiers& de)
90 { un = de.un; deux = de.deux; return (*this);};
91 // surcharge de l'operator de lecture
92 friend istream & operator » (istream & ent, DeuxEntiers & de)
93 { ent » de.un » de.deux; return ent;};
94 // surcharge de l'operator d'écriture
95 friend ostream & operator « (ostream & sort , const DeuxEntiers & de)
96 { sort « de.un « " " « de.deux « " "; return sort;};
97 // surcharge de lecture en XML
98 istream & LectXML_DeuxEntiers(istream & ent);
99 // surcharge d'écriture en XML
100 ostream & EcritXML_DeuxEntiers(ostream & sort);
101 //Surcharge d'opérateurs logiques
102 bool operator == (const DeuxEntiers& a) const
103 { if (( un==a.un) && (deux==a.deux)) return true; else return false;};
104 bool operator != (const DeuxEntiers& a) const { return !(*this == a);};
105 // !*!*!* surcharge de l'opérateur de comparaison : ici elle se fait uniquement
106 // sur le deuxième entier !*!*!*!
107 bool operator > (const DeuxEntiers& a) const { return (this->deux > a.deux);};
108 bool operator >= (const DeuxEntiers& a) const { return (this->deux >= a.deux);};
109 bool operator < (const DeuxEntiers& a) const { return (this->deux < a.deux);};
110 bool operator <= (const DeuxEntiers& a) const { return (this->deux <= a.deux);};
111 // sortie du schemaXML: en fonction de enu
112 void SchemaXML_DeuxEntiers(ofstream& sort,const Enum_IO_XML enu) const ;
113 };
114 /// @} // end of group
115
116 /// @addtogroup Les_conteneurs_ultra_basiques
117 /// @{
118 ///
119
120 /// cas de 2 double
121 class DeuxDoubles
122 {
123 public :
124 // VARIABLES PUBLIQUES :
125 double un,deux;
126 // gestion schéma XML
127 static short int impre_schem_XML;
128
129 // CONSTRUCTEURS :
130 DeuxDoubles() : un(0.),deux(0.) {};
131 DeuxDoubles(double u, double v) : un(u),deux(v) {};
132 DeuxDoubles(const DeuxDoubles & de) : un(de.un),deux(de.deux) {};
133
134 // DESTRUCTEUR :
135 ~DeuxDoubles() {};
136
137 // METHODES PUBLIQUES :
138 // surcharge de l'affectation
139 DeuxDoubles& operator= (const DeuxDoubles& de)
140 { un = de.un; deux = de.deux; return (*this);};
141 // surcharge de l'operator de lecture
142 friend istream & operator » (istream & ent, DeuxDoubles & de)
143 { ent » de.un » de.deux; return ent;};
144 // surcharge de l'operator d'écriture
145 friend ostream & operator « (ostream & sort , const DeuxDoubles & de)
146 { sort « de.un « " " « de.deux « " "; return sort;};
147 // surcharge de lecture en XML
148 istream & LectXML_DeuxDoubles(istream & ent);
149 // surcharge d'écriture en XML
150 ostream & EcritXML_DeuxDoubles(ostream & sort);
151 //Surcharge d'opérateurs logiques
152 bool operator == (const DeuxDoubles& a) const
153 { if (( un==a.un) && (deux==a.deux)) return true; else return false;};
154 bool operator != ( const DeuxDoubles& a) const { return !(*this == a);};
155 // sortie du schemaXML: en fonction de enu
156 void SchemaXML_DeuxDoubles(ofstream& sort,const Enum_IO_XML enu) const ;
157 };
158 /// @} // end of group
159
160 /// @addtogroup Les_conteneurs_ultra_basiques
161 /// @{
162 ///
163
164 /// cas d' 1 entier et 1 double

```

```

165 class Entier_et_Double
166 {
167     public :
168         // VARIABLES PUBLIQUES :
169         double x;
170         int n;
171         // gestion schéma XML
172         static short int impre_schem_XML;
173
174         // CONSTRUCTEURS :
175         Entier_et_Double() : x(0),n(0.) {};
176         Entier_et_Double(int u, double v) : n(u),x(v) {};
177         Entier_et_Double(const Entier_et_Double & de) : x(de.x),n(de.n) {};
178
179         // DESTRUCTEUR :
180         ~Entier_et_Double() {};
181
182         // METHODES PUBLIQUES :
183         // surcharge de l'affectation
184         Entier_et_Double& operator= (const Entier_et_Double& de)
185         { x = de.x; n = de.n; return (*this);};
186         // surcharge de l'opérateur de lecture
187         friend istream & operator » (istream & ent, Entier_et_Double & de)
188         { ent » de.n » de.x; return ent;};
189         // surcharge de l'opérateur d'écriture
190         friend ostream & operator « (ostream & sort , const Entier_et_Double & de)
191         { sort « de.n « " " « de.x « " "; return sort;};
192         // surcharge de lecture en XML
193         istream & LectXML_Entier_et_Double(istream & ent);
194         // surcharge d'écriture en XML
195         ostream & EcritXML_Entier_et_Double(ostream & sort);
196         //Surcharge d'opérateurs logiques
197         bool operator == (const Entier_et_Double& a) const
198         { if ((n==a.n) && (x==a.x)) return true; else return false;};
199         bool operator != (const Entier_et_Double& a) const { return !(*this == a);};
200         // sortie du schemaXML: en fonction de enu
201         void SchemaXML_Entier_et_Double(ofstream& sort,const Enum_IO_XML enu) const ;
202         // surcharge de l'opérateur de comparaison : ici elle se fait uniquement sur le double
203         bool operator > (const Entier_et_Double& a) const { return (this->x > a.x);};
204         bool operator >= (const Entier_et_Double& a) const { return (this->x >= a.x);};
205         bool operator < (const Entier_et_Double& a) const { return (this->x < a.x);};
206         bool operator <= (const Entier_et_Double& a) const { return (this->x <= a.x);};
207     };
208     /// @} // end of group
209
210     /// @addtogroup Les_conteneurs_ultra_basiques
211     /// @{
212     ///
213
214     /// cas de deux String
215     class Deux_String
216     {
217     public :
218         // VARIABLES PUBLIQUES :
219         string nom1,nom2;
220         // gestion schéma XML
221         static short int impre_schem_XML;
222
223         // CONSTRUCTEURS :
224         Deux_String() : nom1(),nom2() {};
225         Deux_String(const string& n1, const string& n2) : nom1(n1),nom2(n2) {};
226         Deux_String(const Deux_String & de) : nom1(de.nom1),nom2(de.nom2) {};
227
228         // DESTRUCTEUR :
229         ~Deux_String() {};
230
231         // METHODES PUBLIQUES :
232         // surcharge de l'affectation
233         Deux_String& operator= (const Deux_String& de)
234         { nom1 = de.nom1; nom2 = de.nom2; return (*this);};
235         // surcharge de l'opérateur de lecture
236         friend istream & operator » (istream & ent, Deux_String & de)
237         { ent » de.nom1 » de.nom2; return ent;};
238         // surcharge de l'opérateur d'écriture
239         friend ostream & operator « (ostream & sort , const Deux_String & de)
240         { sort « de.nom1 « " " « de.nom2 « " "; return sort;};
241         // surcharge de lecture en XML
242         istream & LectXML_Deux_String(istream & ent);
243         // surcharge d'écriture en XML
244         ostream & EcritXML_Deux_String(ostream & sort);
245         //Surcharge d'opérateurs logiques
246         bool operator == (const Deux_String& a) const
247         { if ((nom1 == a.nom1) && (nom2 == a.nom2)) return true; else return false;};
248         bool operator != (const Deux_String& a) const { return !(*this == a);};
249         // sortie du schemaXML: en fonction de enu
250         void SchemaXML_Deux_String(ofstream& sort,const Enum_IO_XML enu) const ;
251         // surcharge de l'opérateur de comparaison : ici elle se fait uniquement sur le premier nom = nom1

```

```

252     bool operator > (const Deux_String& a) const { return (this->nom1 > a.nom1);};
253     bool operator >= (const Deux_String& a) const { return (this->nom1 >= a.nom1);};
254     bool operator < (const Deux_String& a) const { return (this->nom1 < a.nom1);};
255     bool operator <= (const Deux_String& a) const { return (this->nom1 <= a.nom1);};
256 };
257 /// @} // end of group
258
259 /// @addtogroup Les_conteneurs_ultra_basiques
260 /// @{
261 ///
262
263 /// cas d'un string et un entier
264 class String_et_entier
265 {
266     public :
267         // VARIABLES PUBLIQUES :
268         string nom;
269         int n;
270         // gestion schéma XML
271         static short int impre_schem_XML;
272
273         // CONSTRUCTEURS :
274         String_et_entier() : nom(),n() {};
275         String_et_entier(const string& nomm, int nn) : nom(nomm),n(nn) {};
276         String_et_entier(const String_et_entier & de) : nom(de.nom),n(de.n) {};
277
278         // DESTRUCTEUR :
279         ~String_et_entier() {};
280
281         // METHODES PUBLIQUES :
282         // surcharge de l'affectation
283         String_et_entier& operator= (const String_et_entier& de)
284         { nom = de.nom; n = de.n; return (*this);};
285         // surcharge de l'operator de lecture
286         friend istream & operator » (istream & ent, String_et_entier & de)
287         { ent » de.nom » de.n; return ent;};
288         // surcharge de l'operator d'écriture
289         friend ostream & operator « (ostream & sort , const String_et_entier & de)
290         { sort « de.nom « " " « de.n « " "; return sort;};
291         // surcharge de lecture en XML
292         istream & LectXML_String_et_entier(istream & ent);
293         // surcharge d'écriture en XML
294         ostream & EcritXML_String_et_entier(ostream & sort);
295         //Surcharge d'opérateurs logiques
296         bool operator == (const String_et_entier& a) const
297         { if (( nom == a.nom) && (n == a.n)) return true; else return false;};
298         bool operator != (const String_et_entier& a) const { return !(*this == a);};
299         // sortie du schemaXML: en fonction de enu
300         void SchemaXML_String_et_entier(ofstream& sort,const Enum_IO_XML enu) const ;
301 };
302 /// @} // end of group
303
304 /// @addtogroup Les_conteneurs_ultra_basiques
305 /// @{
306 ///
307
308 /// cas de trois String
309 class Trois_String
310 {
311     public :
312         // VARIABLES PUBLIQUES :
313         string nom1,nom2,nom3;
314         // gestion schéma XML
315         static short int impre_schem_XML;
316
317         // CONSTRUCTEURS :
318         Trois_String() : nom1(),nom2(),nom3() {};
319         Trois_String(const string& n1, const string& n2,const string& n3) : nom1(n1),nom2(n2),nom3(n3) {};
320         Trois_String(const Trois_String & de) : nom1(de.nom1),nom2(de.nom2),nom3(de.nom3) {};
321
322         // DESTRUCTEUR :
323         ~Trois_String() {};
324
325         // METHODES PUBLIQUES :
326         // surcharge de l'affectation
327         Trois_String& operator= (const Trois_String& de)
328         { nom1 = de.nom1; nom2 = de.nom2; nom3 = de.nom3; return (*this);};
329         // surcharge de l'operator de lecture
330         friend istream & operator » (istream & ent, Trois_String & de)
331         { ent » de.nom1 » de.nom2 » de.nom3; return ent;};
332         // surcharge de l'operator d'écriture
333         friend ostream & operator « (ostream & sort , const Trois_String & de)
334         { sort « de.nom1 « " " « de.nom2 « " " « de.nom3 « " "; return sort;};
335         // surcharge de lecture en XML
336         istream & LectXML_Trois_String(istream & ent);
337         // surcharge d'écriture en XML
338         ostream & EcritXML_Trois_String(ostream & sort);

```



```

339 //Surcharge d'opérateurs logiques
340 bool operator == (const Trois_String& a) const
341 { if (( nom1 == a.nom1) && (nom2 == a.nom2)&& (nom3 == a.nom3)) return true; else return false;};
342 bool operator != (const Trois_String& a) const { return !(*this == a);};
343 // sortie du schemaXML: en fonction de enu
344 void SchemaXML_Trois_String(ofstream& sort,const Enum_IO_XML enu) const ;
345 // surcharge de l'opérateur de comparaison : ici elle se fait uniquement sur le premier nom = nom1
346 bool operator > (const Trois_String& a) const { return (this->nom1 > a.nom1);};
347 bool operator >= (const Trois_String& a) const { return (this->nom1 >= a.nom1);};
348 bool operator < (const Trois_String& a) const { return (this->nom1 < a.nom1);};
349 bool operator <= (const Trois_String& a) const { return (this->nom1 <= a.nom1);};
350 };
351 /// @} // end of group
352
353 /// @addtogroup Les_conteneurs_ultra_basiques
354 /// @{
355 ///
356
357 /// cas de 4 string et un entier
358 class quatre_string_un_entier
359 {
360 public :
361 // VARIABLES PUBLIQUES :
362 string nom1,nom2,nom3,nom4;
363 int n;
364 // gestion schéma XML
365 static short int impre_schem_XML;
366
367 // CONSTRUCTEURS :
368 quatre_string_un_entier() : nom1(""),nom2(""),nom3(""),nom4(""),n(0) {};
369 quatre_string_un_entier(const string& n1, const string& n2,const string& n3,const string& n4,const
int& nn) : nom1(n1),nom2(n2),nom3(n3),nom4(n4),n(nn) {};
370 quatre_string_un_entier(const quatre_string_un_entier & de) :
371 nom1(de.nom1),nom2(de.nom2),nom3(de.nom3),nom4(de.nom4),n(de.n) {};
372
373 // DESTRUCTEUR :
374 ~quatre_string_un_entier() {};
375
376 // METHODES PUBLIQUES :
377 // surcharge de l'affectation
378 quatre_string_un_entier& operator= (const quatre_string_un_entier& de)
379 { nom1 = de.nom1; nom2 = de.nom2; nom3 = de.nom3;nom4 = de.nom4;n=de.n;
380 return (*this);};
381 // surcharge de l'opérateur de lecture
382 friend istream & operator » (istream & ent, quatre_string_un_entier & de)
383 { ent » de.nom1 » de.nom2 » de.nom3» de.nom4» de.n; return ent;};
384 // surcharge de l'opérateur d'écriture
385 friend ostream & operator « (ostream & sort , const quatre_string_un_entier & de)
386 { sort « de.nom1 « " " « de.nom2 « " "« de.nom3 « " "« de.nom4 « " "« de.n « " "; return sort;};
387 // surcharge de lecture en XML
388 istream & LectXML_quatre_string_un_entier(istream & ent);
389 // surcharge d'écriture en XML
390 ostream & EcritXML_quatre_string_un_entier(ostream & sort);
391 //Surcharge d'opérateurs logiques
392 bool operator == (const quatre_string_un_entier& a) const
393 { if (( nom1 == a.nom1) && (nom2 == a.nom2)&& (nom3 == a.nom3)&& (nom4 == a.nom4)&& (n == a.n))
394 return true; else return false;};
395 bool operator != (const quatre_string_un_entier& a) const { return !(*this == a);};
396 // sortie du schemaXML: en fonction de enu
397 void SchemaXML_quatre_string_un_entier(ofstream& sort,const Enum_IO_XML enu) const ;
398 // surcharge de l'opérateur de comparaison : ici elle se fait par décroissance
399 // est uniquement intéressante pour classer,
400 bool operator > (const quatre_string_un_entier& a) const
401 { if (this->nom1 > a.nom1)
402 {return true;}
403 else if (this->nom1 < a.nom1)
404 {return false;}
405 else // cas d'une égalité
406 // on passe au second string
407 { if (this->nom2 > a.nom2)
408 {return true;}
409 else if (this->nom2 < a.nom2)
410 {return false;}
411 else // cas d'une égalité
412 // on passe au troisième string
413 { if (this->nom3 > a.nom3)
414 {return true;}
415 else if (this->nom3 < a.nom3)
416 {return false;}
417 else // cas d'une égalité
418 // on passe au quatrième string
419 { if (this->nom4 > a.nom4)
420 {return true;}
421 else if (this->nom4 < a.nom4)
422 {return false;}
423 else // cas d'une égalité
424 // on passe à l'entier

```

```

425         { if (this->n > a.n)
426             {return true;}
427         else if (this->n < a.n)
428             {return false;}
429         else // cas d'une égalité
430             {return false;; // car totalement identique donc non supérieur
431         }
432     }
433 }
434 }
435 };
436 bool operator >= (const quatre_string_un_entier& a) const
437 { if (this->nom1 > a.nom1)
438     {return true;}
439   else if (this->nom1 < a.nom1)
440     {return false;}
441   else // cas d'une égalité
442     // on passe au second string
443     { if (this->nom2 > a.nom2)
444         {return true;}
445       else if (this->nom2 < a.nom2)
446         {return false;}
447       else // cas d'une égalité
448         // on passe au troisième string
449         { if (this->nom3 > a.nom3)
450             {return true;}
451           else if (this->nom3 < a.nom3)
452             {return false;}
453           else // cas d'une égalité
454             // on passe au quatrième string
455             { if (this->nom4 > a.nom4)
456                 {return true;}
457               else if (this->nom4 < a.nom4)
458                 {return false;}
459               else // cas d'une égalité
460                 // on passe à l'entier
461                 { if (this->n > a.n)
462                     {return true;}
463                   else if (this->n < a.n)
464                     {return false;}
465                   else // cas d'une égalité
466                     {return true;; // car totalement identique
467                 }
468             }
469         }
470     }
471 };
472
473 bool operator < (const quatre_string_un_entier& a) const
474 { if (this->nom1 < a.nom1)
475     {return true;}
476   else if (this->nom1 > a.nom1)
477     {return false;}
478   else // cas d'une égalité
479     // on passe au second string
480     { if (this->nom2 < a.nom2)
481         {return true;}
482       else if (this->nom2 > a.nom2)
483         {return false;}
484       else // cas d'une égalité
485         // on passe au troisième string
486         { if (this->nom3 < a.nom3)
487             {return true;}
488           else if (this->nom3 > a.nom3)
489             {return false;}
490           else // cas d'une égalité
491             // on passe au quatrième string
492             { if (this->nom4 < a.nom4)
493                 {return true;}
494               else if (this->nom4 > a.nom4)
495                 {return false;}
496               else // cas d'une égalité
497                 // on passe à l'entier
498                 { if (this->n < a.n)
499                     {return true;}
500                   else if (this->n > a.n)
501                     {return false;}
502                   else // cas d'une égalité
503                     {return false;; // car totalement identique donc non inférieur
504                 }
505             }
506         }
507     }
508 };
509
510 bool operator <= (const quatre_string_un_entier& a) const
511 { if (this->nom1 < a.nom1)

```

```

512         {return true;}
513     else if (this->nom1 > a.nom1)
514         {return false;}
515     else // cas d'une égalité
516         // on passe au second string
517         { if (this->nom2 < a.nom2)
518             {return true;}
519             else if (this->nom2 > a.nom2)
520                 {return false;}
521             else // cas d'une égalité
522                 // on passe au troisième string
523                 { if (this->nom3 < a.nom3)
524                     {return true;}
525                     else if (this->nom3 > a.nom3)
526                         {return false;}
527                     else // cas d'une égalité
528                         // on passe au quatrième string
529                         { if (this->nom4 < a.nom4)
530                             {return true;}
531                             else if (this->nom4 > a.nom4)
532                                 {return false;}
533                             else // cas d'une égalité
534                                 // on passe à l'entier
535                                 { if (this->n < a.n)
536                                     {return true;}
537                                     else if (this->n > a.n)
538                                         {return false;}
539                                     else // cas d'une égalité
540                                         {return true;}; // car totalement identique
541                                 }
542                             }
543                         }
544                 }
545         };
546
547 };
548 /// @} // end of group
549
550 /// @addtogroup Les_conteneurs_ultra_basiques
551 /// @{
552 ///
553
554 /// cas de 3 entiers
555 class TroisEntiers
556 {
557     public :
558         // VARIABLES PUBLIQUES :
559         int un,deux,trois;
560         // gestion schéma XML
561         static short int impre_schem_XML;
562
563         // CONSTRUCTEURS :
564         TroisEntiers() : un(0),deux(0),trois(0) {};
565         TroisEntiers(int u, int v,int w) : un(u),deux(v),trois(w) {};
566         TroisEntiers(const TroisEntiers & de) : un(de.un),deux(de.deux),trois(de.trois) {};
567
568         // DESTRUCTEUR :
569         ~TroisEntiers() {};
570
571         // METHODES PUBLIQUES :
572         // surcharge de l'affectation
573         TroisEntiers& operator= (const TroisEntiers& de)
574         { un = de.un; deux = de.deux;trois = de.trois; return (*this);};
575         // surcharge de l'opérateur de lecture
576         friend istream & operator » (istream & ent, TroisEntiers & de)
577         { ent » de.un » de.deux » de.trois; return ent;};
578         // surcharge de l'opérateur d'écriture
579         friend ostream & operator « (ostream & sort , const TroisEntiers & de)
580         { sort « de.un « " " « de.deux « " "« de.trois; return sort;};
581         // surcharge de lecture en XML
582         istream & LectXML_TroisEntiers(istream & ent);
583         // surcharge d'écriture en XML
584         ostream & EcritXML_TroisEntiers(ostream & sort);
585         //Surcharge d'opérateurs logiques
586         bool operator == (const TroisEntiers& a) const
587         { if ((un==a.un) && (deux==a.deux) && (trois == a.trois)) return true; else return false;};
588         bool operator != (const TroisEntiers& a) const { return !(this == a);};
589         // surcharge de l'opérateur de comparaison
590         bool operator > (const TroisEntiers& a) const
591         { if (un > a.un)
592             {return true;}
593             else if (un < a.un)
594                 {return false;}
595             else // égalité, on test le suivant
596                 {if (deux > a.deux)
597                     {return true;}
598                     else if (deux < a.deux)

```

```

599         {return false;}
600     else // égalité, on test le suivant
601     {if (trois > a.trois)
602         {return true;}
603         else if (trois < a.trois)
604             {return false;}
605         else // égalité, on test le suivant
606             {return false;} // car c'est l'égalité parfaite donc pas supérieur
607     }
608 }
609 };
610 bool operator >= (const TroisEntiers& a) const
611 { if (un > a.un)
612     {return true;}
613     else if (un < a.un)
614         {return false;}
615     else // égalité, on test le suivant
616     {if (deux > a.deux)
617         {return true;}
618         else if (deux < a.deux)
619             {return false;}
620         else // égalité, on test le suivant
621             {if (trois > a.trois)
622                 {return true;}
623                 else if (trois < a.trois)
624                     {return false;}
625                 else // égalité, on test le suivant
626                     {return true;} // car c'est l'égalité parfaite
627             }
628     }
629 };
630 bool operator < (const TroisEntiers& a) const
631 { if (un < a.un)
632     {return true;}
633     else if (un > a.un)
634         {return false;}
635     else // égalité, on test le suivant
636     {if (deux < a.deux)
637         {return true;}
638         else if (deux > a.deux)
639             {return false;}
640         else // égalité, on test le suivant
641             {if (trois < a.trois)
642                 {return true;}
643                 else if (trois > a.trois)
644                     {return false;}
645                 else // égalité, on test le suivant
646                     {return false;} // car c'est l'égalité parfaite donc pas inférieur
647             }
648     }
649 };
650 bool operator <= (const TroisEntiers& a) const
651 { if (un < a.un)
652     {return true;}
653     else if (un > a.un)
654         {return false;}
655     else // égalité, on test le suivant
656     {if (deux < a.deux)
657         {return true;}
658         else if (deux > a.deux)
659             {return false;}
660         else // égalité, on test le suivant
661             {if (trois < a.trois)
662                 {return true;}
663                 else if (trois > a.trois)
664                     {return false;}
665                 else // égalité, on test le suivant
666                     {return true;} // car c'est l'égalité parfaite
667             }
668     }
669 };
670 // sortie du schemaXML: en fonction de enu
671 void SchemaXML_TroisEntiers(ofstream& sort,const Enum_IO_XML enu) const ;
672 };
673 /// @} // end of group
674
675 /// @addtogroup Les_conteneurs_ultra_basiques
676 /// @{
677 ///
678
679 /// cas de deux String et un entier
680 class Deux_String_un_entier
681 {
682 public :
683     // VARIABLES PUBLIQUES :
684     string nom1,nom2;
685     int n;

```

```

686 // gestion schéma XML
687 static short int impre_schem_XML;
688
689 // CONSTRUCTEURS :
690 Deux_String_un_entier() : nom1(""),nom2(""),n(0) {};
691 Deux_String_un_entier(const string& n1, const string& n2, const int& nn)
692     : nom1(n1),nom2(n2),n(nn) {};
693 Deux_String_un_entier(const Deux_String_un_entier & de) :
694     nom1(de.nom1),nom2(de.nom2),n(de.n) {};
695
696 // DESTRUCTEUR :
697 ~Deux_String_un_entier() {};
698
699 // METHODES PUBLIQUES :
700 // surcharge de l'affectation
701 Deux_String_un_entier& operator= (const Deux_String_un_entier& de)
702 { nom1 = de.nom1; nom2 = de.nom2; n=de.n;
703   return (*this);};
704 // surcharge de l'operateur de lecture
705 friend istream & operator >> (istream & ent, Deux_String_un_entier & de)
706 { ent >> de.nom1 >> de.nom2 >> de.n; return ent;};
707 // surcharge de l'operateur d'écriture
708 friend ostream & operator << (ostream & sort , const Deux_String_un_entier & de)
709 { sort << de.nom1 << " " << de.nom2 << " " << de.n << " "; return sort;};
710 // surcharge de lecture en XML
711 istream & LectXML_Deux_String_un_entier(istream & ent);
712 // surcharge d'écriture en XML
713 ostream & EcrivXML_Deux_String_un_entier(ostream & sort);
714 //Surcharge d'opérateurs logiques
715 bool operator == (const Deux_String_un_entier& a) const
716 { if (( nom1 == a.nom1) && (nom2 == a.nom2)&& (n == a.n))
717     return true; else return false;};
718 bool operator != (const Deux_String_un_entier& a) const { return !(*this == a);};
719 // sortie du schemaXML: en fonction de enu
720 void SchemaXML_Deux_String_un_entier(ofstream& sort,const Enum_IO_XML enu) const ;
721 // surcharge de l'opérateur de comparaison : ici elle se fait par décroissance
722 // est uniquement intéressante pour classer,
723 bool operator > (const Deux_String_un_entier& a) const
724 { if (this->nom1 > a.nom1)
725     {return true;}
726   else if (this->nom1 < a.nom1)
727     {return false;}
728   else // cas d'une égalité
729     // on passe au second string
730     { if (this->nom2 > a.nom2)
731         {return true;}
732       else if (this->nom2 < a.nom2)
733         {return false;}
734       else // cas d'une égalité
735         // on passe à l'entier
736         { if (this->n > a.n)
737             {return true;}
738           else if (this->n < a.n)
739             {return false;}
740           else // cas d'une égalité
741             {return false;}; // car totalement identique donc non supérieur
742         }
743     }
744 };
745 bool operator >= (const Deux_String_un_entier& a) const
746 { if (this->nom1 > a.nom1)
747     {return true;}
748   else if (this->nom1 < a.nom1)
749     {return false;}
750   else // cas d'une égalité
751     // on passe au second string
752     { if (this->nom2 > a.nom2)
753         {return true;}
754       else if (this->nom2 < a.nom2)
755         {return false;}
756       else // cas d'une égalité
757         // on passe à l'entier
758         { if (this->n > a.n)
759             {return true;}
760           else if (this->n < a.n)
761             {return false;}
762           else // cas d'une égalité
763             {return true;}; // car totalement identique
764         }
765     }
766 };
767
768 bool operator < (const Deux_String_un_entier& a) const
769 { if (this->nom1 < a.nom1)
770     {return true;}
771   else if (this->nom1 > a.nom1)
772     {return false;}

```

```

773     else // cas d'une égalité
774         // on passe au second string
775         { if (this->nom2 < a.nom2)
776             {return true;}
777             else if (this->nom2 > a.nom2)
778                 {return false;}
779             else // cas d'une égalité
780                 // on passe à l'entier
781                 { if (this->n < a.n)
782                     {return true;}
783                     else if (this->n > a.n)
784                         {return false;}
785                     else // cas d'une égalité
786                         {return false;}; // car totalement identique donc non inférieur
787                 }
788             }
789     };
790
791     bool operator <= (const Deux_String_un_entier& a) const
792     { if (this->nom1 < a.nom1)
793         {return true;}
794         else if (this->nom1 > a.nom1)
795             {return false;}
796         else // cas d'une égalité
797             // on passe au second string
798             { if (this->nom2 < a.nom2)
799                 {return true;}
800                 else if (this->nom2 > a.nom2)
801                     {return false;}
802                 else // cas d'une égalité
803                     // on passe à l'entier
804                     { if (this->n < a.n)
805                         {return true;}
806                         else if (this->n > a.n)
807                             {return false;}
808                         else // cas d'une égalité
809                             {return true;}; // car totalement identique
810                     }
811             }
812     };
813
814 };
815 /// @} // end of group
816
817 #endif

```

## 7.484 Ddl\_enum\_etendu-copie.h

```

1 /*****
2 *   UNIVERSITE DE BRETAGNE SUD (UBS) --- I.U.P/I.U.T. DE LORIENT *
3 *****/
4 *   LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M) *
5 *   Centre de Recherche Rue de Saint Maud - 56325 Lorient cedex *
6 *   tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
7 *****/
8 *   DATE:      19/01/2001 *
9 *   $ *
10 *   AUTEUR:    G RIO (mailto:gerard.rio@univ-ubs.fr) *
11 *             Tel 0297874571 fax : 02.97.87.45.72 *
12 *   $ *
13 *   PROJET:    Herezh++ *
14 *   $ *
15 *****/
16 *   BUT:  une classe qui permet de dfinir des identificateurs de *
17 *        grandeurs secondaires associes des Enum_ddl ddl de base.*
18 *        Ces grandeurs ne sont pas vraiment des ddl, mais ils s'en *
19 *        dduisent. Ainsi pour ne pas alourdir le type Enum ddl *
20 *        on cr cette classe associe. Elle est prvue en particu- *
21 *        lier pour grer les sorties de rsultats. *
22 *        les lments sont ordonns selon la rgle suivante : *
23 *        si c'est un Enum_ddl pur -> ordre de l'Enum *
24 *        si c'est un lment de tab_Dee -> ordre dans tab_Dee *
25 *        + maxi des Enum_ddl *
26 *        lorsqu'il s'agit d'un enum_ddl tout simple, dans ce cas *
27 *        Non_vide == true *
28 *   $ *
29 *   ***** *
30 *   VERIFICATION: *
31 *   ! date ! auteur ! but ! *
32 *   ----- *
33 *   ! ! ! ! *
34 *   $ *
35 *   ***** *
36 *

```

```

37 *      MODIFICATIONS:
38 *      ! date !      auteur !      but
39 *      -----
40 *
41 *      $
42 *      *****/
43 #ifndef DDL_ENUM_ETENDUE_H
44 #define DDL_ENUM_ETENDUE_H
45
46 #include "Enum_ddl.h"
47 #include <string.h>
48 #include "List_io.h"
49 #include <iomanip>
50 #include <algorithm>
51
52 // d'finition d'une classe pour l'initialisation du tableau tab_Dee
53 class Initialisation_tab_Dee { public: Initialisation_tab_Dee(); };
54
55 class Ddl_enum_etendu
56 {
57 // surcharge de l'operator de lecture
58 friend istream & operator » (istream & ent, Ddl_enum_etendu & a)
59 // { ent » a.nom » a.enu; return ent; };
60 // { string no; ent » no; a=Ddl_enum_etendu(no); return ent; };
61
62 // surcharge de l'operator d'ecriture
63 friend ostream & operator « (ostream & sort , const Ddl_enum_etendu & a)
64 // { sort « a.nom « " " « a.enu « " "; return sort; };
65 // { if (a.Nom_vide()) sort « " " « Nom_ddl(a.enu); else sort « " " « a.nom; return sort; };
66
67 // permission pour l'initialisation du tableau tab_Dee
68 friend class Initialisation_tab_Dee;
69
70 public :
71 // CONSTRUCTEURS :
72 // constructeur par dfaut
73 Ddl_enum_etendu(Enum_ddl en = NU_DDL, string no = "-");
74 // constructeur fonction d'un string
75 Ddl_enum_etendu( string no );
76 // constructeur fonction d'une chaine
77 Ddl_enum_etendu( char * no );
78 // constructeur de copie
79 Ddl_enum_etendu(const Ddl_enum_etendu& a) :
80 nom(a.nom), enu(a.enu), posi_nom(a.posi_nom) {};
81 // DESTRUCTEUR :
82 ~Ddl_enum_etendu() {};
83
84 // METHODES PUBLIQUES :
85
86 // surcharge d'affectation
87 Ddl_enum_etendu & operator = ( const Ddl_enum_etendu & a)
88 { nom = a.nom; enu = a.enu; posi_nom = a.posi_nom; return *this; };
89 // surcharge d'egalite
90 bool operator == ( const Ddl_enum_etendu & a) const
91 { if (posi_nom == a.posi_nom) return true; else return false; };
92 // surcharge de non egalite
93 bool operator != ( const Ddl_enum_etendu & a) const
94 { if (posi_nom != a.posi_nom) return true; else return false; };
95 // surcharge de comparaison
96 bool operator > (const Ddl_enum_etendu & a) const
97 { return (posi_nom > a.posi_nom); };
98 bool operator >= (const Ddl_enum_etendu & a) const
99 { return (posi_nom >= a.posi_nom); };
100 bool operator < (const Ddl_enum_etendu & a) const
101 { return (posi_nom < a.posi_nom); };
102 bool operator <= (const Ddl_enum_etendu & a) const
103 { return (posi_nom <= a.posi_nom); };
104
105 // rcup de l'numration
106 Enum_ddl Enum() const {return enu; };
107 // rcup du nom
108 string Nom() const {return nom; };
109 // rcup du nom gnrique (sans les indices de composantes)
110 string NomGenerique() const
111 {if (nom=="-"){return string(NomGeneric(enu));}
112 else {return string(nom.substr(1,nom.length()-2));};
113 };
114 // position donne un numro quivalent du type numr
115 int Position() const {return posi_nom; };
116 // modification du nom
117 // test pour savoir si le nom est vide
118 bool Nom_vide() const { if (nom == "-") return true; else return false; };
119 // retour le type de grandeur auquel appartient le ddl tendue
120 // par exemple : UY : appartient un vecteur
121 // SIG12 : un tenseur, TEMP : un scalaire
122 EnumTypeGrandeur TypeDeGrandeur() const;
123
124 // test pour savoir si le nom passer en paramtre est valide

```

```

124 // ramne vrai si no correspond un Enum_ddl ou
125 // s'il correspond un type driv : ex contrainte de mise est un type
126 // drive de contrainte
127 static bool VerifExistence(string no) ;
128 // rcupration d'un Ddl_enum_etendu correspondant un string
129 static Ddl_enum_etendu RecupDdl_enum_etendu(string to) ;
130 // transformation d'une liste d'Enum_ddl en Ddl_enum_etendu
131 static List_io <Ddl_enum_etendu> TransfoList_io(const List_io <Enum_ddl> & li) ;
132 // transformation d'un tableau d'numration en un tableau de Ddl_enum_etendu
133 static Tableau < Ddl_enum_etendu > TransfoTableau(const Tableau <Enum_ddl> & tab) ;
134 // test si un lment existe dans une liste donnee
135 static bool Existe_dans_la_liste
136 (const List_io <Ddl_enum_etendu> & lis, const Ddl_enum_etendu& dd) ;
137 // rcupration du nombre maximum de ddl tendu existant
138 static int NBmax_ddl_enum_etendue() {return taillTab+NbEnum_ddl();};
139
140 protected :
141 // VARIABLES PROTEGEES :
142 string nom;
143 Enum_ddl enu;
144 int posi_nom; // position du nom : quivalent un type numr
145
146 // on dfinie le tableau des Ddl_enum_etendu qui sont valides
147 // en variables globales
148 static Tableau < Ddl_enum_etendu > tab_Dee;
149 static Initialisation_tab_Dee init_tab_Dee;
150 static int taillTab; // taille du tableau tab_Dee
151
152 // METHODES PROTEGEES :
153
154 };
155
156 #endif

```

## 7.485 Ddl\_enum\_etendu.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 * DATE: 19/01/2001 *
31 * * $ *
32 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
33 * * $ *
34 * PROJET: Herezh++ *
35 * * $ *
36 *****/
37 * BUT: une classe qui permet de définir des identificateurs de *
38 * grandeurs secondaires associées à des Enum_ddl ddl de base.*
39 * Ces grandeurs ne sont pas vraiment des ddl, mais ils s'en *
40 * déduisent. Ainsi pour ne pas alourdir le type Enum ddl *
41 * on crée cette classe associée. Elle est prévue en particu- *
42 * lier pour gérer les sorties de résultats. *
43 * les éléments sont ordonnés selon la règle suivante : *
44 * si c'est un Enum_ddl pur -> ordre de l'Enum *
45 * si c'est un élément de tab_Dee -> ordre dans tab_Dee *
46 * + maxi des Enum_ddl *
47 * lorsqu'il s'agit d'un enum_ddl tout simple, dans ce cas *
48 * Non_vide == true *
49 * * $ *

```



```

50 *      *
51 *      *
52 *
53 #ifndef DDL_ENUM_ETENDUE_H
54 #define DDL_ENUM_ETENDUE_H
55 *
56 #include "Enum_ddl.h"
57 #include <string.h>
58 #include "List_io.h"
59 #include <iomanip>
60 #include <algorithm>
61 #include "Enum_TypeQuelconque.h"
62 *
63 /** @defgroup Group_Ddl_enum_etendu
64 *
65 *      BUT: une classe qui permet de définir des identificateurs de
66 *      grandeurs secondaires associées à des Enum_ddl ddl de base.
67 *      Ces grandeurs ne sont pas vraiment des ddl, mais ils s'en
68 *      déduisent. Ainsi pour ne pas alourdir le type Enum ddl
69 *      on crée cette classe associée. Elle est prévue en particu-
70 *      lier pour gérer les sorties de résultats.
71 *      les éléments sont ordonnés selon la règle suivante :
72 *      si c'est un Enum_ddl pur -> ordre de l'Enum
73 *      si c'est un élément de tab_Dee -> ordre dans tab_Dee
74 *      plus maxi des Enum_ddl
75 *      lorsqu'il s'agit d'un enum_ddl tout simple, dans ce cas
76 *      Non_vide == true
77 *
78 *
79 * \author   Gérard Rio
80 * \version  1.0
81 * \date    19/01/2001
82 * \brief   classe qui permet de définir des identificateurs de grandeurs secondaires associées à des
83 *          Enum_ddl ddl de base.
84 */
85 *
86 /// @addtogroup Group_Ddl_enum_etendu
87 /// @{
88 *
89 /// une classe secondaire: définition d'une classe pour l'initialisation du tableau tab_Dee
90 /// et de la map qui relie les string et les Ddl_enum_etendu
91 class Initialisation_tab_Dee { public: Initialisation_tab_Dee(); };
92 /// @} // end of group
93 *
94 /// @addtogroup Group_Ddl_enum_etendu
95 /// @{
96 *
97 /// la classe principale: Ddl_enum_etendu
98 class Ddl_enum_etendu
99 {
100 // surcharge de l'operator de lecture
101 friend istream & operator » (istream & ent, Ddl_enum_etendu & a)
102 //   { ent » a.nom » a.enu ; return ent; };
103   { string no; ent » no; a=Ddl_enum_etendu(no); return ent; };
104 *
105 // surcharge de l'operator d'écriture
106 friend ostream & operator « (ostream & sort , const Ddl_enum_etendu & a)
107 //   { sort « a.nom « " " « a.enu « " "; return sort; };
108   { if (a.Nom_vide()) sort « " " « Nom_ddl(a.enu); else sort « " " « a.nom; return sort; };
109 *
110 // permission pour l'initialisation du tableau tab_Dee
111 friend class Initialisation_tab_Dee;
112 *
113 public :
114 // CONSTRUCTEURS :
115 // constructeur par défaut
116 Ddl_enum_etendu(Enum_ddl en = NU_DDL, string no = "-");
117 // constructeur fonction d'un string
118 Ddl_enum_etendu( string no );
119 // constructeur fonction d'une chaîne
120 // Ddl_enum_etendu( char * no );
121 // constructeur de copie
122 Ddl_enum_etendu(const Ddl_enum_etendu& a) :
123   nom(a.nom), enu(a.enu), posi_nom(a.posi_nom) {};
124 // DESTRUCTEUR :
125 ~Ddl_enum_etendu() {};
126 *
127 // METHODES PUBLIQUES :
128 *
129 // surcharge d'affectation
130 Ddl_enum_etendu & operator = ( const Ddl_enum_etendu & a)
131   { nom = a.nom; enu = a.enu; posi_nom = a.posi_nom; return *this; };
132 // surcharge d'égalité
133 bool operator == ( const Ddl_enum_etendu & a) const
134   { if (posi_nom == a.posi_nom) return true; else return false; };
135 // surcharge de non égalité

```

```

136 bool operator != ( const Ddl_enum_etendu & a) const
137 { if (posi_nom != a.posi_nom) return true; else return false;};
138 // surcharge de comparaison
139 bool operator > (const Ddl_enum_etendu & a) const
140 { return (posi_nom > a.posi_nom);};
141 bool operator >= (const Ddl_enum_etendu & a) const
142 { return (posi_nom >= a.posi_nom);};
143 bool operator < (const Ddl_enum_etendu & a) const
144 { return (posi_nom < a.posi_nom);};
145 bool operator <= (const Ddl_enum_etendu & a) const
146 { return (posi_nom <= a.posi_nom);};
147
148 // récup de l'énumération
149 Enum_ddl Enum() const {return enu;};
150 // récup du nom
151 string Nom() const {return nom;};
152 // récup du nom générique (sans les indices de composantes)
153 string NomGenerique() const
154 {if (nom=="-"){return string(NomGeneric(enu));}
155  else {return string(nom.substr(1,nom.length()-2));}
156  };
157 // récup d'un nom, qui est soit le nom de l'enum soit le nom + complet
158 // s'il existe
159 string Nom_plein() const {return ((nom=="-")? Nom_ddl(enu) : nom );};
160 // position donne un numéro équivalent du type énuméré
161 int Position() const {return posi_nom;};
162 // modification du nom
163 // test pour savoir si le nom est vide
164 bool Nom_vide() const { if (nom == "-") return true; else return false; };
165 // retour le type de grandeur auquel appartient le ddl étendue
166 // par exemple : UY : appartient à un vecteur
167 // SIG12 : à un tenseur, TEMP : à un scalaire
168 EnumTypeGrandeur TypeDeGrandeur() const;
169
170 // test pour savoir si le nom passer en paramètre est valide
171 // ramène vrai si no correspond à un Enum_ddl ou
172 // s'il correspond à un type dérivé : ex contrainte de mise est un type
173 // dérivée de contrainte
174 static bool VerifExistence(string no) ;
175 // récupération d'un Ddl_enum_etendu correspondant à un string
176 static Ddl_enum_etendu RecupDdl_enum_etendu(string to) ;
177 // transformation d'une liste d'Enum_ddl en Ddl_enum_etendu
178 static List_io <Ddl_enum_etendu> TransfoList_io(const List_io <Enum_ddl> & li) ;
179 // transformation d'un tableau d'énumération en un tableau de Ddl_enum_etendu
180 static Tableau < Ddl_enum_etendu > TransfoTableau(const Tableau <Enum_ddl> & tab) ;
181 // test si un élément existe dans une liste donnée
182 static bool Existe_dans_la_liste
183 (const List_io <Ddl_enum_etendu> & lis, const Ddl_enum_etendu& dd) ;
184 // récupération du nombre maximum de ddl étendu existant
185 static int NBmax_ddl_enum_etendue() {return taillTab+NbEnum_ddl();};
186 // récup d'un tableau d' enum_ddl_etendue particulier : de force normale et tangentielle
187 // 2 éléments, le premier correspond à "reaction_normale" et le second à "reaction_tangentielle"
188 static const Tableau <Ddl_enum_etendu>& Tab_FN_FT() {return tab_FN_FT;};
189
190 // récupération du premier Ddl_enum_etendu du même type : i.e. la première composante
191 static Ddl_enum_etendu PremierDdlEnumEtenduFamille(const Ddl_enum_etendu& a);
192
193 // création et récupération d'une liste d'enum de grandeurs quelconques équivalentes
194 // sous forme de grandeurs évoluées, à la liste de Ddl_enum_etendu passée en argument
195 // et d'une liste de ddl_enum_etendu, pour les grandeurs qui n'ont pas d'équivalent
196 // retourne également true ou false suivant que toutes les grandeurs ont un équivalent ou pas
197 static void Equivalent_en_grandeur_quelconque(const list <Ddl_enum_etendu> & list_enu_etendu
198 , list <EnumTypeQuelconque> & list_enu_quelconque
199 , list <Ddl_enum_etendu> & list_enu_restant);
200 // idem mais pour une seule grandeur: ramène l'enum quelconque s'il existe sinon
201 // ramène RIEN_TYPEQUELCONQUE
202 static EnumTypeQuelconque Equivalent_en_grandeur_quelconque(const Ddl_enum_etendu & enu_etendu);
203
204 protected :
205 // VARIABLES PROTEGEES :
206 string nom;
207 Enum_ddl enu;
208 int posi_nom; // position du nom : équivalent à un type énuméré
209
210 // on définit le tableau des Ddl_enum_etendu qui sont valides
211 // en variables globales
212 static Tableau < Ddl_enum_etendu > tab_Dee;
213 static Initialisation_tab_Dee init_tab_Dee;
214 static int taillTab; // taille du tableau tab_Dee
215
216 // tableaux particuliers
217 static Tableau <Ddl_enum_etendu> tab_FN_FT; // le tableau des deux ddl etendus
218
219 // def de la map qui fait la liaison entre les string et les Ddl_enum_etendu
220 static map < string, int , std::less < string> > map_Ddl_enum_etendu;
221
222 // METHODES PROTEGEES :

```

```

223
224 };
225 ///< @} // end of group
226
227 #endif

```

## 7.486 Epai.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // FICHER : Epai.h
30 // CLASSE : Epai
31 /*****
32 *      DATE:      08/07/2008
33 *
34 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:    Herezh++
37 *
38 *
39 *      BUT:       Conteneur très basique pour les épaisseurs
40 *
41 *      *****
42 *
43 *      *****/
44
45 #ifndef EPAI_SIMPLE_H
46 #define EPAI_SIMPLE_H
47
48 #include <iostream>
49 #include <fstream>
50
51 //
52 //-----
53 //!      Conteneur très basique pour les épaisseurs
54 //-----
55
56      ///< \author      Gérard Rio
57      ///< \version    1.0
58      ///< \date      08/07/2008
59
60
61 class Epai
62 {public:
63     // VARIABLES PUBLIQUES :
64     double epaisseur0,epaisseur_t,epaisseur_tdt;
65     // CONSTRUCTEURS :
66     Epai(): epaisseur0(0.),epaisseur_t(0.),epaisseur_tdt(0.) {};
67     Epai(const double ep0,const double ept, const double eptdt) :
68         epaisseur0(ep0),epaisseur_t(ept),epaisseur_tdt(eptdt)
69         {};
70     Epai(const Epai & a) :
71         epaisseur0(a.epaisseur0),epaisseur_t(a.epaisseur_t)
72         ,epaisseur_tdt(a.epaisseur_tdt)
73         {};
74     // DESTRUCTEUR :
75     ~Epai() {};
76
77     // METHODES PUBLIQUES :

```

```

78 // changement de toutes les valeurs à une valeur déterminée
79 void Change_tout(const double& val) {epaisseur0=epaisseur_t=epaisseur_tdt=val;};
80 // surcharge de l'affectation
81 Epai& operator= (const Epai& a)
82 { epaisseur0 = a.epaisseur0; epaisseur_t = a.epaisseur_t;
83   epaisseur_tdt=a.epaisseur_tdt;
84   return (*this);
85 };
86 // surcharge de l'operator de lecture
87 friend istream & operator >> (istream & ent, Epai & de)
88 { ent >> de.epaisseur0 >> de.epaisseur_t
89   >> de.epaisseur_tdt;
90   return ent;
91 };
92 // surcharge de l'operator d'écriture
93 friend ostream & operator << (ostream & sort , const Epai & de)
94 { sort << " " << de.epaisseur0 << " " << de.epaisseur_t
95   << " " << de.epaisseur_tdt << " ";
96   return sort;
97 };
98
99 };
100
101 #endif

```

## 7.487 List\_io.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      23/01/97
31 *
32 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *   *****
37 *   BUT: création d'un conteneur liste stl, qui comporte en plus
38 *   une surcharge de lecture écriture.
39 *   Idem pour LaList.
40 *
41 *   *****
42 *   MODIFICATIONS:
43 *   ! date ! auteur ! but
44 *   -----
45 *   !24/8/2003! rio ! introduction de LaList
46 *
47 *   *****/
48 #ifndef LIST_IO_H
49 #define LIST_IO_H
50
51 #include <iostream>
52 #include <stdlib.h>
53 #include <list>
54 #include <iomanip>
55 #include "LaList.h"
56 #include "Sortie.h"
57 #include "ParaGlob.h"
58 using namespace std; //introduces namespace std

```

```

59
60 /** @defgroup Les_list_io
61 *
62 * BUT: création de conteneurs type listes stl, qui comportent en plus
63 * une surcharge de lecture écriture.
64 * Idem pour LaList.
65 *
66 *
67 * \author Gérard Rio
68 * \version 1.0
69 * \date 23/01/97
70 * \brief création de conteneurs type listes stl avec surcharge de lecture écriture
71 *
72 */
73
74
75 /// @addtogroup Les_list_io
76 /// @{
77 ///
78
79 //-----
80 //! List_io classe template identique à list de stl, avec en plus une lecture et écriture
81 //-----
82 /// \author Gérard Rio
83 /// \version 1.0
84 /// \date 23/01/97
85
86 template <class T>
87 class List_io : public LaLIST<T>
88 { // surcharge de l'operator de lecture
89 friend istream & operator >> (istream & entree, List_io<T> & )
90 { // du au pb de lecture de pointeur
91 cout << "\n erreur dans la lecture d'une list_io, la methode n'est pas utilisable en template "
92 << "\n friend istream & operator >> (... ";
93 Sortie(1);
94 return entree;
95 }
96 // --- information pour créer un programme de lecture cohérent avec l'écriture ----
97 // exemple de surcharge de l'operator de lecture
98 // friend istream & operator >> (istream & entree, List_io& a)
99 // { string nom; entree >> nom;
100 // #ifdef MISE_AU_POINT
101 // if (nom != "debut_List_IO=")
102 // { cout << "\n erreur dans la lecture d'une list_io, on ne trouve pas le mot cle:
debut_List_IO= "
103 // << "\n friend istream & operator >> (... ";
104 // Sortie(1);
105 // }
106 // #endif
107 // int taille; entree >> taille; // lecture de la taille
108 // T un_elem;
109 // for (int i=1; i<=taille; i++)
110 // { entree >> un_elem; // lecture
111 // a.push_back(un_elem); // enregistrement
112 // }
113 // return entree;
114 // }
115 // fin --- information pour créer un programme de lecture cohérent avec l'écriture ----
116
117 // surcharge de l'operator d'écriture
118 friend ostream & operator << (ostream & sort, const List_io<T> & a)
119 { // tout d'abord un indicateur donnant le type
120 sort << "\n debut_List_IO= " << "(taille= " << a.size() << " ) ";
121 typename List_io<T>::const_iterator iter_courant, iter_fin;
122 iter_fin = a.end();
123 for (iter_courant=a.begin(); iter_courant!=iter_fin; iter_courant++)
124 { sort << setprecision(ParaGlob::NbdigdoCA()) << (*iter_courant) ; sort << " "; }
125 sort << "\n";
126 return sort;
127 }
128
129 public:
130 // on définit également une opération de lecture et d'écriture pour
131 // les itérateurs, ce qui permettra de créer des tableaux d'iterateur
132 // surcharge de l'operator de lecture, a priori pas utile
133 friend istream & operator >> (istream & entree, typename LaLIST<T>::iterator& )
134 { cout << "erreur, cette surcharge ne doit pas être utilisée "
135 << "List_io::iterator, operator << \n";
136 Sortie(1);
137 return entree;
138 }
139 // surcharge de l'operator d'écriture, a priori pas utile
140 friend ostream & operator << (ostream & sort, const typename LaLIST<T>::iterator )
141 { cout << "erreur, cette surcharge ne doit pas être utilisée "
142 << "List_io::iterator, operator >> \n";
143 Sortie(1);
144 return sort;

```

```

145     }
146
147     // surcharge de l'operator de lecture, a priori pas utile
148     friend istream & operator » (istream & entree, typename LaLIST<T>::const_iterator& )
149     { cout << "erreur, cette surcharge ne doit pas être utilisée "
150       << "List_io::const_iterator, operator << \n";
151       Sortie(1);
152       return entree;
153     }
154     // surcharge de l'operator d'écriture, a priori pas utile
155     friend ostream & operator « (ostream & sort, const typename LaLIST<T>::const_iterator& )
156     { cout << "erreur, cette surcharge ne doit pas être utilisée "
157       << "List_io::const_iterator_io, operator » \n";
158       Sortie(1);
159       return sort;
160     }
161
162     public :
163         // CONSTRUCTEURS :
164         List_io (const List_io& li): list(li) {};
165         List_io (Allocator<list_node>::size_type n, const T& a): list(n,a) {};
166
167         // DESTRUCTEUR :
168         // METHODES PUBLIQUES :
169
170     private :
171         // VARIABLES PROTEGEES :
172         // CONSTRUCTEURS :
173         // DESTRUCTEUR :
174         // METHODES PROTEGEES :
175
176 };
177
178 /// @} // end of group
179
180 /// @addtogroup Les_list_io
181 /// @{
182 ///
183
184 //-----
185 ///!      LaLIST_io classe template identique à List_io: existe pour des raisons historiques et en
186           prévision de changements futurs éventuelles (?)
187 //-----
188 /// \author   Gérard Rio
189 /// \version  1.0
190 /// \date     24/8/2003
191
192 // idem pour LaList
193
194 template <class T>
195 class LaLIST_io : public LaLIST<T>
196 { // surcharge de l'operator de lecture
197   friend istream & operator » (istream & entree, LaLIST_io<T>& )
198   // l'opérateur de lecture n'est pas surchargeable car on n'a pas de
199   // critère d'arrêt de la lecture (on en faire mais pour l'instant pas
200   // utile
201   { cout << "erreur, cette surcharge ne doit pas être utilisée "
202     << "LaLIST_io, operator » \n";
203     Sortie(1);
204     return entree;
205   }
206   // surcharge de l'operator d'écriture
207   friend ostream & operator « (ostream & sort, const LaLIST_io<T>& a)
208   { // tout d'abord un indicateur donnant le type
209     sort << "\nLaLIST_IO= ";
210     typename LaLIST_io<T>::const_iterator iter_courant,iter_fin;
211     iter_fin = a.end();
212     for (iter_courant=a.begin();iter_courant!=iter_fin;iter_courant++)
213       { sort << setw (22) « (*iter_courant) ; sort « " "; }
214     sort << "\n";
215     return sort;
216   }
217
218   public:
219     // on définit également une opération de lecture et d'écriture pour
220     // les itérateurs, ce qui permettra de créer des tableaux d'iterator
221     // surcharge de l'operator de lecture, a priori pas utile
222     friend istream & operator » (istream & entree, typename LaLIST<T>::iterator& )
223     { cout << "erreur, cette surcharge ne doit pas être utilisée "
224       << "LaLIST_io::iterator, operator << \n";
225       Sortie(1);
226       return entree;
227     }
228     // surcharge de l'operator d'écriture, a priori pas utile
229     friend ostream & operator « (ostream & sort, const typename LaLIST<T>::iterator )
230     { cout << "erreur, cette surcharge ne doit pas être utilisée "
231       << "LaLIST_io::iterator, operator » \n";

```

```

231         Sortie(1);
232         return sort;
233     }
234
235     // surcharge de l'operator de lecture, a priori pas utile
236     friend istream & operator > (istream & entree, typename LaLIST<T>::const_iterator& )
237     { cout << "erreur, cette surcharge ne doit pas être utilisée "
238       << "LaLIST_io::const_iterator, operator << \n";
239       Sortie(1);
240       return entree;
241     }
242     // surcharge de l'operator d'écriture, a priori pas utile
243     friend ostream & operator << (ostream & sort, const typename LaLIST<T>::const_iterator& )
244     { cout << "erreur, cette surcharge ne doit pas être utilisée "
245       << "LaLIST_io::const_iterator_io, operator >> \n";
246       Sortie(1);
247       return sort;
248     }
249
250 public :
251     // CONSTRUCTEURS :
252     // DESTRUCTEUR :
253     // METHODES PUBLIQUES :
254
255 private :
256     // VARIABLES PROTEGEES :
257     // CONSTRUCTEURS :
258     // DESTRUCTEUR :
259     // METHODES PROTEGEES :
260
261 };
262 /// @} // end of group
263
264
265
266 #endif

```

## 7.488 Map\_io.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           04/01/2007
31 *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:         Herezh++
35 *
36 *   *****
37 *   BUT: création d'un conteneur map stl, qui comporte en plus
38 *   une surcharge de lecture écriture.
39 *
40 *   *****
41 *   *****/
42 #ifndef MAP_IO_H
43 #define MAP_IO_H
44
45 #include <iostream>
46 #include <stdlib.h>

```

```

47 #include <map>
48 #include <iomanip>
49 #include "Sortie.h"
50 #include "ParaGlob.h"
51 using namespace std; //introduces namespace std
52
53 -----
54 //!      Map_io  classe template de type map STL avec une lecture et écriture
55 -----
56 /// \author   Gérard Rio
57 /// \version  1.0
58 /// \date    04/01/2007
59
60 template <class T>
61 class Map_io : public map<T>
62 { // surcharge de l'operator de lecture
63     friend istream & operator >> (istream & entree, Map_io<T> & )
64     { // du au pb de lecture de pointeur
65         cout << "\n erreur dans la lecture d'une Map_io, la methode n'est pas utilisable en template "
66             << "\n friend istream & operator >> (... ";
67         Sortie(1);
68         return entree;
69     }
70     // --- information pour créer un programme de lecture cohérent avec l'écriture ----
71     // exemple de surcharge de l'operator de lecture
72     friend istream & operator >> (istream & entree, Map_io& a)
73     { string nom; entree >> nom;
74     #ifdef MISE_AU_POINT
75     if (nom != "debut_Map_IO=")
76     { cout << "\n erreur dans la lecture d'une Map_io, on ne trouve pas le mot cle: debut_Map_IO="
77     << "\n friend istream & operator >> (... ";
78     Sortie(1);
79     }
80     #endif
81     int taille;entree >> taille; // lecture de la taille
82     T un_elem;
83     for (int i=1;i<=taille;i++)
84     { entree >> un_elem; // lecture
85     a.push_back(un_elem); // enregistrement
86     }
87     return entree;
88     }
89     // fin --- information pour créer un programme de lecture cohérent avec l'écriture ----
90
91     // surcharge de l'operator d'écriture
92     friend ostream & operator << (ostream & sort, const Map_io<T> & a)
93     { // tout d'abord un indicateur donnant le type
94     sort << "\n debut_Map_IO= " << "(taille= " << a.size() << " ) ";
95     typename Map_io<T>::const_iterator iter_courant,iter_fin;
96     iter_fin = a.end();
97     for (iter_courant=a.begin();iter_courant!=iter_fin;iter_courant++)
98     { sort << setprecision(ParaGlob::NbdigdoCA()) << (*iter_courant) ; sort << " "; }
99     sort << "\n";
100    return sort;
101    }
102
103 public:
104     // on définit également une opération de lecture et d'écriture pour
105     // les itérateurs, ce qui permettra de créer des tableaux d'iterator
106     // surcharge de l'operator de lecture, a priori pas utile
107     friend istream & operator >> (istream & entree, typename map<T>::iterator& )
108     { cout << "erreur, cette surcharge ne doit pas être utilisée "
109         << "Map_io::iterator, operator << \n";
110     Sortie(1);
111     return entree;
112     }
113     // surcharge de l'operator d'écriture, a priori pas utile
114     friend ostream & operator << (ostream & sort, const typename map<T>::iterator )
115     { cout << "erreur, cette surcharge ne doit pas être utilisée "
116         << "Map_io::iterator, operator >> \n";
117     Sortie(1);
118     return sort;
119     }
120
121     // surcharge de l'operator de lecture, a priori pas utile
122     friend istream & operator >> (istream & entree, typename map<T>::const_iterator& )
123     { cout << "erreur, cette surcharge ne doit pas être utilisée "
124         << "Map_io::const_iterator, operator << \n";
125     Sortie(1);
126     return entree;
127     }
128     // surcharge de l'operator d'écriture, a priori pas utile
129     friend ostream & operator << (ostream & sort, const typename map<T>::const_iterator& )
130     { cout << "erreur, cette surcharge ne doit pas être utilisée "
131         << "Map_io::const_iterator_io, operator >> \n";
132     Sortie(1);

```



```

133         return sort;
134     }
135
136     public :
137         // CONSTRUCTEURS :
138
139         // DESTRUCTEUR :
140         // METHODES PUBLIQUES :
141
142     private :
143         // VARIABLES PROTEGEES :
144         // CONSTRUCTEURS :
145         // DESTRUCTEUR :
146         // METHODES PROTEGEES :
147
148 };
149
150
151 #endif

```

## 7.489 Nb\_assemb.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      23/01/97
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *
37 *   BUT: création d'un type simple, équivalent au type entier,
38 *   mais permettant des vérifications de type plus précise,
39 *   ceci pour les numéros d'assemblage.
40 *
41 *   *****
42 *****/
43
44 #ifndef NB_ASSEMB_H
45 #define NB_ASSEMB_H
46
47 #include <iostream>
48 #include <stdlib.h>
49 #include "Sortie.h"
50 #include <string>
51 #include <fstream>
52 //
53 //-----
54 //!   Nb_assemb:  description des numéros d'assemblage
55 //-----
56     /// \author   Gérard Rio
57     /// \version  1.0
58     /// \date    23/01/97
59
60 // description des numéros d'assemblage
61 class Nb_assemb
62 { public :
63     // surcharge de l'operator de lecture avec le type

```

```

64 friend istream & operator » (istream & ent, Nb_assemb & a)
65 { // lecture du type et vérification
66   std::string nomtype;
67   ent » nomtype;
68   if (nomtype != "num_assemb")
69     { Sortie(1);
70       return ent;
71     }
72   // lecture de la donnée
73   ent » a.n ;
74   return ent;
75 };
76 // surcharge de l'operator d'écriture
77 friend ostream & operator « (ostream & sort, const Nb_assemb & a)
78 { // écriture du type et de la donnée
79   sort « "num_assemb " « a.n « " ";
80   return sort;
81 };
82 // constructeurs
83 Nb_assemb(int nn = 0): n(nn) {}; // par défaut
84 Nb_assemb(const Nb_assemb& a) : n(a.n) {}; // de copie
85
86 // surcharge de l'opérateur d'affectation
87 Nb_assemb& operator= (const Nb_assemb& c)
88   { n = c.n; return (*this);}
89
90 //Surcharge d'opérateur logique
91 bool operator == (const Nb_assemb& a) const
92   { return (a.n == n); };
93 bool operator != (const Nb_assemb& a) const
94   { return (a.n != n); };
95
96
97 // donnée
98 int n;
99 };
100
101
102 #endif

```

## 7.490 PlusieursCoordonnees.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier PlusieursCoordonnees.h
30
31 /*****
32 *      DATE:      19/01/2001
33 *
34 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:    Herezh++
37 *
38 *****/
39 *      BUT:      classes relatives à des nombres fixes de coordonnées.
40 *
41 *      *****
42 *****/
43

```

```

44 /** @defgroup Les_classes_PlusieursCoordonnees
45 *
46 *      BUT:      classes relatives à des nombres fixes de coordonnées.
47 *
48 * \author      Gérard Rio
49 * \version    1.0
50 * \date      19/01/2001
51 * \brief      classes relatives à des nombres fixes de coordonnées.
52 *
53 */
54
55
56 #ifndef PLUSIEURS_COORDONNEES_H
57 #define PLUSIEURS_COORDONNEES_H
58
59 #include "Coordonnee.h"
60
61 /// @addtogroup Les_classes_PlusieursCoordonnees
62 /// @{///
63
64 //
65 //-----
66 //!      DeuxCoordonnees:      classe relative à 2 coordonnées
67 //-----
68      /// \author      Gérard Rio
69      /// \version    1.0
70      /// \date      19/01/2001
71 class DeuxCoordonnees
72 {
73 public :
74      // CONSTRUCTEURS :
75      DeuxCoordonnees() : col(),co2() {};
76      DeuxCoordonnees(const Coordonnee & coo1, const Coordonnee & coo2):
77      col(coo1),co2(coo2) {};
78      DeuxCoordonnees(const DeuxCoordonnees& deuxcoo):
79      col(deuxcoo.col),co2(deuxcoo.co2) {};
80      DeuxCoordonnees(int dima): col(dima),co2(dima) {};
81      // DESTRUCTEUR :
82      ~DeuxCoordonnees() {};
83      // METHODES PUBLIQUES :
84      DeuxCoordonnees& operator= (const DeuxCoordonnees& de)
85      { col = de.col; co2 = de.co2; return (*this);};
86      // acces aux coordonnées
87      Coordonnee& Premier() {return col;};
88      Coordonnee& Second() {return co2;};
89      Coordonnee Premier()const {return col;};
90      Coordonnee Second()const {return co2;};
91
92 private :
93      // VARIABLES PROTEGEES :
94      Coordonnee col,co2;
95
96 };
97 /// @} // end of group
98
99 #endif

```

## 7.491 Ponderation.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //

```

```

27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      04/07/2016
31 *
32 *   AUTEUR:    G RIO    (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *****/
37 *   BUT:
38 *   Def de classes conteneurs pour manipuler différentes pondérations
39 *
40 *   *****
41 *****/
42
43 #ifndef PONDERATION_H
44 #define PONDERATION_H
45
46
47 #include <string>
48 #include "Tableau_T.h"
49 #include "Enum_GrandeurGlobale.h"
50 #include "Fonction_nD.h"
51 #include "Ddl_enum_etendu.h"
52 #include "CourbelD.h"
53 #include "LesCourbes1D.h"
54 #include "LesFonctions_nD.h"
55 #include "Enum_TypeQuelconque.h"
56 #include "LesPtIntegMecaInterne.h"
57 #include "Noeud.h"
58 #include "Deformation.h"
59
60 /** @defgroup Les_classes_Ponderation
61 *
62 *   BUT:   Def de classes conteneurs pour manipuler différentes pondérations
63 *
64 *
65 * \author   Gérard Rio
66 * \version  1.0
67 * \date    04/07/2016
68 * \brief   Def de classes conteneurs pour manipuler différentes pondérations
69 *
70 */
71
72 /// @addtogroup Les_classes_Ponderation
73 /// @{
74 ///
75
76 // -----
77 // une classe de travail, qui permet d'utiliser une pondération qui dépend de plusieurs
78 // courbelD, chacune dépendant d'une grandeur consultable par un string
79 // -----
80
81     class Ponderation_Consultable
82     {
83     public:
84         Ponderation_Consultable (); // le constructeur par défaut
85         Ponderation_Consultable (const Ponderation_Consultable& a); // le constructeur de copie
86         ~Ponderation_Consultable (); // destructeur
87
88         // ----- affectation des fonctions à partir des noms sauvegardés
89         // intéressant quand cette affectation à lieu longtemps après la lecture
90         void Affectation_fonctions(LesCourbes1D& lesCourbes1D);
91
92         // --- acces aux données---
93         // à chaque grandeur est associé une fonction 1D
94         Tableau <string>& Type_grandeur_Consultable() {return type_grandeur_Consultable;};
95
96         // les fonctions 1D
97         Tableau <CourbelD*>& C_proport() {return c_proport;};
98
99         // le nom des fonctions, sert en particulier pour une définition
100        // différée avec la lecture: 1) lecture 2) affectation des courbes ... après
101        Tableau <string>& Tab_nom_fonction() {return tab_nom_fonction;};
102
103        //--- vérification que tout est ok pour le calcul -----
104        void Verif_complet() const;
105
106        //----- lecture écriture de restart -----
107        // cas donne le niveau de la récupération
108        // = 1 : on récupère tout
109        // = 2 : on récupère uniquement les données variables (supposées comme telles)
110        void Lecture_base_info(ifstream& ent,const int cas,LesCourbes1D& lesCourbes1D);
111        // cas donne le niveau de sauvegarde
112        // = 1 : on sauvegarde tout
113        // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
114        void Ecriture_base_info(ofstream& sort,const int cas);

```

```

114         // sortie du schemaXML: en fonction de enu
115         void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu);
116
117     protected:
118         Tableau <string> type_grandeur_Consultable; // à chaque grandeur est associé une
fonction 1D
119         Tableau <Courbe1D*> c_proport; // les fonctions 1D
120         Tableau <string> tab_nom_fonction; // le nom des fonctions, sert en particulier pour une
définition
121
122         // différée avec la lecture: 1) lecture 2) affectation des courbes ...
123     };
124     @} // end of group
125     @addtogroup Les_classes_Ponderation
126     @{
127
128     // -----
129     // une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble
130     // de grandeurs globales au travers d'une fonction nD
131     // -----
132
133     class Ponderation_GGlobal
134     { public:
135         Ponderation_GGlobal (); // le constructeur par défaut
136         Ponderation_GGlobal (const Ponderation_GGlobal& a); // le constructeur de copie
137         ~Ponderation_GGlobal (); // destructeur
138
139         // lecture sur le flot d'entrée
140         void LecturePonderation(const List_io <string>& grandeurs
141             , UtilLecture * entreePrinc,LesFonctions_nD& lesFonctionsnD);
142
143         // ----- affectation de la fonction à partir du nom sauvegardé
144         // intéressant quand cette affectation à lieu longtemps après la lecture
145         void Affectation_fonctions(LesFonctions_nD& lesFonctionsnD);
146
147         // --- acces aux données---
148         // le nom de la fonction
149         string& Nom_fonction() {return nom_fonction;}
150
151         // la fonction multidimensionnel pour l'utilisation
152         Fonction_nD* C_proport() {return c_proport;};
153         // la fonction multidimensionnel pour sont affectation
154         void Assigne_proport(Fonction_nD* pt) {c_proport = pt;};
155
156         //-- vérification que tout est ok pour le calcul -----
157         void Verif_complet() const;
158
159         // affichage
160         void Affiche();
161
162         //---- lecture écriture de restart ----
163         // cas donne le niveau de la récupération
164         // = 1 : on récupère tout
165         // = 2 : on récupère uniquement les données variables (supposées comme telles)
166         void Lecture_base_info(ifstream& ent,const int cas,LesFonctions_nD& lesFonctionsnD);
167         // idem mais sans définition de la courbe:
168         // correspond au cas de la sauvegarde avec san_courbe = true
169         void Lecture_base_info(ifstream& ent,const int cas);
170         // cas donne le niveau de sauvegarde
171         // = 1 : on sauvegarde tout
172         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
173         // sans_courbe: indique si on sauve oui ou non la courbe elle-même
174         // true: on sauve la courbe
175         // false : on sauve uniquement le nom de la courbe
176         void Ecriture_base_info(ofstream& sort,const int cas,bool sans_courbe=false);
177         // sortie du schemaXML: en fonction de enu
178         void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu);
179
180     protected:
181         string nom_fonction; // le nom de la fonction, sert en particulier pour une définition de la
fonction
182
183         // différée avec la lecture: 1) lecture 2) affectation de la fonction ...
184     };
185     @} // end of group
186
187     @addtogroup Les_classes_Ponderation
188     @{
189
190
191     // -----
192     // une classe de travail, qui permet d'utiliser une pondération qui dépend d'un ensemble
193     // de grandeurs quelconque au travers d'une fonction nD
194     // -----
195

```

```

196     class Ponderation_TypeQuelconque
197     { public:
198         Ponderation_TypeQuelconque (); // le constructeur par défaut
199         Ponderation_TypeQuelconque (const Ponderation_TypeQuelconque& a); // le constructeur de
copie
200         ~Ponderation_TypeQuelconque (); // destructeur
201
202         // lecture sur le flot d'entrée
203         // si grandeurs n'est pas vide, on vérifie que les grandeurs passées en paramètre
204         // sont bien des variables de la fonction (locale ou globale)
205         void LecturePonderation(const List_io <string>& grandeurs
206             , UtilLecture * entreePrinc, LesFonctions_nD& lesFonctionsnD);
207
208         // ----- affectation de la fonction à partir du nom sauvegardé
209         // intéressant quand cette affectation à lieu longtemps après la lecture
210         void Affectation_fonctions(LesFonctions_nD& lesFonctionsnD);
211
212         // --- acces aux données---
213         // le nom de la fonction
214         string& Nom_fonction() {return nom_fonction;};
215
216         // le type des grandeurs quelconques qui représentent les arguments de la fonction
217         // List_io <EnumTypeQuelconque>& Type_grandeur_Quelc() {return
type_grandeur_Quelconque;};
218
219         // les arguments pour l'appel de la fonction
220         // Tableau <double>& Tab_argument() {return tab_argument;};
221
222         // la fonction multidimensionnel pour l'utilisation
223         Fonction_nD* C_proport() {return c_proport;};
224         // la fonction multidimensionnel pour sont affectation
225         void Assigne_proport(Fonction_nD* pt) {c_proport = pt;};
226
227         //-- vérification que tout est ok pour le calcul -----
228         void Verif_complet() const;
229
230         //----- lecture écriture de restart -----
231         // cas donne le niveau de la récupération
232         // = 1 : on récupère tout
233         // = 2 : on récupère uniquement les données variables (supposées comme telles)
234         void Lecture_base_info(ifstream& ent, const int cas, LesFonctions_nD& lesFonctionsnD);
235         // idem mais sans définition de la fonction nD:
236         // correspond au cas de la sauvegarde avec san_courbe = true
237         void Lecture_base_info(ifstream& ent, const int cas);
238         // cas donne le niveau de sauvegarde
239         // = 1 : on sauvegarde tout
240         // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
241         // sans_fct: indique si on sauve oui ou non la fonction elle-même
242         // true: on sauve la fonction
243         // false : on sauve uniquement le nom de la fonction
244         void Ecriture_base_info(ofstream& sort, const int cas, bool sans_fct=false);
245         // sortie du schemaXML: en fonction de enu
246         void SchemaXML_Fonctions_nD(ofstream& sort, const Enum_IO_XML enu);
247
248         protected:
249         string nom_fonction; // le nom de la fonction, sert en particulier pour une définition de la
fonction
250
251         // différée avec la lecture: 1) lecture 2) affectation de la fonction ...
après
252         Fonction_nD* c_proport; // la fonction multidimensionnel
253
254         // des variables internes qui n'ont plus à être accessible en externe,
255         // c'est la fonction nD qui stocke les infos
256         List_io < EnumTypeQuelconque > type_grandeur_Quelconque; // les arguments de la
fonction
257         Tableau <double> tab_argument; //argument pour la fonction
258
259     };
260 /// @} // end of group
261
262 /// @addtogroup Les_classes_Ponderation
263 /// @{
264 ///
265
266 // -----
267 // une seconde classe de travail qui permet d'utiliser une pondération qui dépend de n
268 // CourbelD, chacune fonction d'un ddl étendu
269 // -----
270
271     class Ponderation
272     { public:
273         Ponderation (); // le constructeur par défaut
274         Ponderation (const Ponderation& a); // le constructeur de copie
275         ~Ponderation (); // destructeur
276
277         // ----- opération de lecture sur le flot d'entrée -----

```

```

278
279
280 // lecture d'une courbe: intéressant si on lit courbe par courbe
281 // le dimensionnement est mis à jour à chaque appel
282 void LectureDonneesPonderation_uneCourbe
283     (Ddl_enum_etendu ddl, UtilLecture * entreePrinc, LesCourbes1D& lesCourbes1D);
284
285 // lecture de l'ensemble de la pondération
286 void LectureDonneesPonderation
287     (const List_io <string> & list_id_ddl_etendu
288     , UtilLecture * entreePrinc, LesCourbes1D& lesCourbes1D);
289
290 // ----- affectation de la fonction à partir du nom sauvegardé
291 // intéressant quand cette affectation à lieu longtemps après la lecture
292 void Affectation_fonctions(LesCourbes1D& lesCourbes1D);
293
294 // --- acces aux données---
295
296 // à chaque grandeur est associé une fonction 1D
297 Tableau <Ddl_enum_etendu>& Type_grandeur() {return type_grandeur;};
298 const Tableau <Ddl_enum_etendu>& Const_Type_grandeur() const {return type_grandeur;};
299
300 // indique si la grandeur est définie aux noeuds ou au point d'integ
301 Tableau <bool>& Valeur_aux_noeuds() {return valeur_aux_noeuds;};
302 const Tableau <bool>& Const_Valeur_aux_noeuds() const {return valeur_aux_noeuds;};
303
304 // les fonctions 1D
305 Tableau <CourbelD*>& C_proport() {return c_proport;};
306 const Tableau <CourbelD*>& Const_C_proport() const {return c_proport;};
307
308 // le nom des fonctions, sert en particulier pour une définition
309 // différée avec la lecture: 1) lecture 2) affectation des courbes ... après
310 Tableau <string>& Tab_nom_fonction() {return tab_nom_fonction;};
311
312 //-- vérification que tout est ok pour le calcul -----
313 void Verif_complet() const;
314
315 // affichage
316 void Affiche();
317
318 // activation des données des noeuds et/ou elements nécessaires au fonctionnement
319 // exemple: mise en service des ddl de température aux noeuds
320 void Activation_donnees
321     (Tableau<Noeud *>& tabnoeud, bool dilatation, LesPtIntegMecaInterne& lesPtMecaInt);
322
323 // calcul global de la pondération, sous forme d'un produit
324 // NB: ar défaut ramène 1. si toutes les fonctions sont inactives
325 // sinon ramène le produit :
326 // 1) des fonctions de Ddl_enum_etendu accessibles
327 // soit au pti méca et ou soit par interpolation via def
328 // 2) d'une fonction éventuelle du temps
329 // 3) d'une fonction éventuelle de thermodonnee
330 double CalculPonderMultiplicatif(const PtIntegMecaInterne& ptintmeca
331     , const Deformation & def
332     , Enum_dure temps, const ThermoDonnee& dTP);
333
334 //----- lecture écriture de restart -----
335 // cas donne le niveau de la récupération
336 // = 1 : on récupère tout
337 // = 2 : on récupère uniquement les données variables (supposées comme telles)
338 void Lecture_base_info(ifstream& ent, const int cas, LesCourbes1D& lesCourbes1D);
339 // cas donne le niveau de sauvegarde
340 // = 1 : on sauvegarde tout
341 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
342 void Ecriture_base_info(ofstream& sort, const int cas);
343 // sortie du schemaXML: en fonction de enu
344 void SchemaXML_Fonctions_nD(ofstream& sort, const Enum_IO_XML enu);
345
346 protected:
347     Tableau <Ddl_enum_etendu> type_grandeur; // à chaque grandeur est associé une fonction
348     1D
349     Tableau <bool> valeur_aux_noeuds; // indique si la grandeur est définie aux noeuds ou
350     au point d'integ
351     Tableau <CourbelD*> c_proport; // les fonctions 1D
352     Tableau <string> tab_nom_fonction; // le nom des fonctions, sert en particulier pour une
353     définition
354     // différée avec la lecture: 1) lecture 2) affectation des courbes ...
355     après
356     };
357 /// @} // end of group
358
359 /// @addtogroup Les_classes_Ponderation
360 /// @{
361 ///
362 /// -----
363 /// une classe de travail qui permet d'utiliser une pondération qui dépend

```

```

361  /// du temps via une CourbelD
362  // -----
363
364      class Ponderation_temps
365  { public:
366          Ponderation_temps (); // le constructeur par défaut
367          Ponderation_temps (const Ponderation_temps& a); // le constructeur de copie
368          ~Ponderation_temps (); // destructeur
369
370          // lecture sur le flot d'entrée
371          void LectureDonneesPonderation (Utilecture * entreePrinc, LesCourbes1D& lesCourbes1D);
372
373          // ----- affectation de la fonction à partir du nom sauvegardé
374          // intéressant quand cette affectation à lieu longtemps après la lecture
375          void Affectation_fonctions (LesCourbes1D& lesCourbes1D);
376
377          // --- acces aux données---
378          // le nom de la fonction, sert en particulier pour une définition de la fonction
379          // différée avec la lecture: 1) lecture 2) affectation de la fonction ... après
380          string& Nom_fonction() {return nom_fonction;};
381
382          // la fonctions 1D, fonction du temps
383          CourbelD* C_proport() {return c_proport;};
384
385          //-- vérification que tout est ok pour le calcul -----
386          void Verif_complet() const;
387
388          // affichage
389          void Affiche();
390
391          // calcul de la pondération
392          // par défaut ramène 1. si la fonction du temps est inactive par exemple
393          // sinon ramène la valeur de la fonction du temps actuel
394          double CalculPonder();
395
396          //----- lecture écriture de restart -----
397          // cas donne le niveau de la récupération
398          // = 1 : on récupère tout
399          // = 2 : on récupère uniquement les données variables (supposées comme telles)
400          void Lecture_base_info(ifstream& ent, const int cas, LesCourbes1D& lesCourbes1D);
401          // idem mais sans définition de la courbe:
402          // correspond au cas de la sauvegarde avec sans_courbe = true
403          void Lecture_base_info(ifstream& ent, const int cas);
404          // cas donne le niveau de sauvegarde
405          // = 1 : on sauvegarde tout
406          // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
407          // sans_courbe: indique si on sauve oui ou non la courbe elle-même
408          // true: on sauve la courbe
409          // false : on sauve uniquement le nom de la courbe
410          void Ecriture_base_info(ofstream& sort, const int cas, bool sans_courbe=false);
411          // sortie du schemaXML: en fonction de enu
412          void SchemaXML_Fonctions_nD(ofstream& sort, const Enum_IO_XML enu);
413
414          protected:
415          string nom_fonction; // le nom de la fonction, sert en particulier pour une définition de la
fonction
416
417          // différée avec la lecture: 1) lecture 2) affectation de la fonction ...
après
417          CourbelD* c_proport; // la fonctions 1D, fonction du temps
418          };
419  /// @} // end of group
420
421
422 #endif

```

## 7.492 PtTabRel.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //

```



```

19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      23/01/97
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *****/
37 *   BUT:   Listes de petits tableaux de réels
38 *   la definition exacte des listes globales est faite dans le
39 *   fichier PtTabRel_Prin.h qui n'est inclus qu'une seule fois
40 *   dans le fichier du programme principal
41 *
42 *   *****
43 *****/
44
45
46 #ifndef PTTABREL_H
47 #define PTTABREL_H
48
49 // #include <boost/pool/poolfwd.hpp>
50 // #include </opt/local/include/boost/pool/poolfwd.hpp>
51 #include <list>
52
53 /** @defgroup Les_listes_de_petits_tableaux_de_reels
54 *
55 *   BUT:   Listes de petits tableaux de réels
56 *   la definition exacte des listes globales est faite dans le
57 *   fichier PtTabRel_Prin.h qui n'est inclus qu'une seule fois
58 *   dans le fichier du programme principal
59 *
60 *
61 *   \author   Gérard Rio
62 *   \version  1.0
63 *   \date    23/01/97
64 *   \brief   Listes de petits tableaux de réels
65 *
66 */
67
68 /** @defgroup Les_pointeurs_dans_listes_de_petits_tableaux
69 *
70 *   BUT:   def de pointeur qui donne la position d'un élément dans une liste
71 *
72 *   \author   Gérard Rio
73 *   \version  1.0
74 *   \date    23/01/97
75 *   \brief   def de pointeur qui donne la position d'un élément dans une liste
76 *
77 */
78
79 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
80 /// @{
81 ///
82
83 //-----
84 //!   ReelsPointe  classe virtuelle de laquelle dérive les classe contenant un pointeur de réel
85 //-----
86 /// \author   Gérard Rio
87 /// \version  1.0
88 /// \date    23/01/97
89
90 // ===== tout d'abord une classe virtuelle de laquelle dérive les classe contenant
91 // un pointeur de réel
92 class ReelsPointe
93 { public :
94   };
95
96 /// @} // end of group
97
98 //=====
99 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
100 /// @{
101 ///
102 /// cas des tableaux de 1 réel
103 class Reels1
104 { public :
105   // Données :

```

```

106     double donnees ;
107     };
108     /// @} // end of group
109
110     extern std::list < Reels1 > listdouble1;
111     typedef std::list < Reels1 >::iterator listdouble1Iter ;
112
113     /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
114     /// @{
115     ///
116     class Reel1Pointe : public ReelsPointe
117     { public :
118         // constructeur
119         Reel1Pointe () : ipointe()
120         { listdouble1.push_front(Reels1()); // allocation d'un maillon
121           ipointe = listdouble1.begin(); // mémorisation du maillon
122         }
123         // destructeur
124         ~ Reel1Pointe ()
125         { listdouble1.erase(ipointe); } // suppression de l'élément de la liste
126         // data
127         listdouble1Iter ipointe; // donne la position dans la liste du maillon
128     };
129     /// @} // end of group
130
131
132     //=====
133     /// @addtogroup Les_listes_de_petits_tableaux_de_reels
134     /// @{
135     ///
136     /// cas des tableaux de 2 réels
137     class Reels2
138     { public :
139         // Données :
140         double donnees [2];
141     };
142     /// @} // end of group
143
144     extern std::list <Reels2> listdouble2;
145     typedef std::list < Reels2 >::iterator listdouble2Iter ;
146
147     /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
148     /// @{
149     ///
150     /// cas des tableaux de 2 réels
151     class Reel2Pointe : public ReelsPointe
152     { public :
153         // constructeur
154         Reel2Pointe () : ipointe()
155         { listdouble2.push_front(Reels2()); // allocation d'un maillon
156           ipointe = listdouble2.begin(); // mémorisation du maillon
157         }
158         // destructeur
159         ~ Reel2Pointe ()
160         { listdouble2.erase(ipointe); } // suppression de l'élément de la liste
161         // data
162         listdouble2Iter ipointe; // donne la position dans la liste du maillon
163     };
164     /// @} // end of group
165
166
167     //=====
168     /// @addtogroup Les_listes_de_petits_tableaux_de_reels
169     /// @{
170     ///
171     /// cas des tableaux de trois réels
172     class Reels3
173     { public :
174         // Données :
175         double donnees [3];
176     };
177     /// @} // end of group
178
179     extern std::list <Reels3> listdouble3;
180     typedef std::list < Reels3 >::iterator listdouble3Iter ;
181
182     /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
183     /// @{
184     ///
185     /// cas des tableaux de trois réels
186     class Reel3Pointe : public ReelsPointe
187     { public :
188         // constructeur
189         Reel3Pointe () : ipointe()
190         { listdouble3.push_front(Reels3()); // allocation d'un maillon
191           ipointe = listdouble3.begin(); // mémorisation du maillon
192         }

```

```

193         // destructeur
194         ~ Reel3Pointe ()
195         { listdouble3.erase(ipointe); } ; // suppression de l'élément de la liste
196         // data
197         listdouble3Iter ipointe; // donne la position dans la liste du maillon
198     };
199 /// @} // end of group
200
201 //=====
202 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
203 /// @{
204 ///
205 /// cas des tableaux de quatre réels
206 class Reels4
207 { public :
208     /* Reels4 () { donnees [0] = 11; donnees [1] = 12; donnees [2] = 13; donnees [3] = 14; //
        initialisation
209         }; */
210
211     // Données :
212     double donnees [4];
213 };
214 /// @} // end of group
215
216 typedef std::list < Reels4 >::iterator listdouble4Iter ;
217 extern std::list <Reels4> listdouble4;
218
219 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
220 /// @{
221 ///
222 /// cas des tableaux de quatre réels
223 class Reel4Pointe : public ReelsPointe
224 { public :
225     // constructeur
226     Reel4Pointe () : ipointe()
227     { listdouble4.push_front(Reels4()); // allocation d'un maillon
228       ipointe = (listdouble4.begin()); // mémorisation du maillon
229     }
230     // destructeur
231     ~ Reel4Pointe ()
232     { listdouble4.erase( ipointe); } ; // suppression de l'élément de la liste
233     // data
234     listdouble4Iter ipointe; // donne la position dans la liste du maillon
235 };
236 /// @} // end of group
237
238 //=====
239 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
240 /// @{
241 ///
242 /// cas des tableaux de 5 réels
243 class Reels5
244 { public :
245     // Données :
246     double donnees [5];
247 };
248 /// @} // end of group
249
250 typedef std::list < Reels5 >::iterator listdouble5Iter ;
251 extern std::list <Reels5> listdouble5;
252
253 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
254 /// @{
255 ///
256 /// cas des tableaux de 5 réels
257 class Reel5Pointe : public ReelsPointe
258 { public :
259     // constructeur
260     Reel5Pointe () : ipointe()
261     { listdouble5.push_front(Reels5()); // allocation d'un maillon
262       ipointe = listdouble5.begin(); // mémorisation du maillon
263     }
264     // destructeur
265     ~ Reel5Pointe ()
266     { listdouble5.erase(ipointe); } ; // suppression de l'élément de la liste
267     // data
268     listdouble5Iter ipointe; // donne la position dans la liste du maillon
269 };
270 /// @} // end of group
271
272 //=====
273 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
274 /// @{
275 ///
276 /// cas des tableaux de 6 réels
277 class Reels6
278 { public :

```

```

279 // Données :
280 double donnees [6];
281 };
282 /// @} // end of group
283
284 extern std::list <Reels6> listdouble6;
285 typedef std::list < Reels6 >::iterator listdouble6Iter ;
286
287 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
288 /// @{
289 ///
290 /// cas des tableaux de 6 réels
291 class Reel6Pointe : public ReelsPointe
292 { public :
293     // constructeur
294     Reel6Pointe () : ipointe()
295     { listdouble6.push_front(Reels6()); // allocation d'un maillon
296       ipointe = listdouble6.begin(); // mémorisation du maillon
297     }
298     // destructeur
299     ~ Reel6Pointe ()
300     { listdouble6.erase(ipointe); } // suppression de l'élément de la liste
301     // data
302     listdouble6Iter ipointe; // donne la position dans la liste du maillon
303 };
304 /// @} // end of group
305
306 //=====
307 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
308 /// @{
309 ///
310 /// cas des tableaux de sept réels
311 class Reels7
312 { public :
313     // Données :
314     double donnees [7];
315 };
316 /// @} // end of group
317
318 typedef std::list < Reels7 >::iterator listdouble7Iter ;
319 extern std::list <Reels7> listdouble7;
320
321 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
322 /// @{
323 ///
324 /// cas des tableaux de sept réels
325 class Reel7Pointe : public ReelsPointe
326 { public :
327     // constructeur
328     Reel7Pointe () : ipointe()
329     { listdouble7.push_front(Reels7()); // allocation d'un maillon
330       ipointe = listdouble7.begin(); // mémorisation du maillon
331     }
332     // destructeur
333     ~ Reel7Pointe ()
334     { listdouble7.erase(ipointe); } // suppression de l'élément de la liste
335     // data
336     listdouble7Iter ipointe; // donne la position dans la liste du maillon
337 };
338 /// @} // end of group
339
340 //=====
341 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
342 /// @{
343 ///
344 /// cas des tableaux de 8 réels
345 class Reels8
346 { public :
347     // Données :
348     double donnees [8];
349 };
350 /// @} // end of group
351
352 typedef std::list < Reels8 >::iterator listdouble8Iter ;
353 extern std::list < Reels8 > listdouble8;
354
355 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
356 /// @{
357 ///
358 /// cas des tableaux de 8 réels
359 class Reel8Pointe : public ReelsPointe
360 { public :
361     // constructeur
362     Reel8Pointe () : ipointe()
363     { listdouble8.push_front(Reels8()); // allocation d'un maillon
364       ipointe = listdouble8.begin(); // mémorisation du maillon
365     }

```

```

366         // destructeur
367         ~ Reel8Pointe ()
368         { listdouble8.erase(ipointe); } ; // suppression de l'élément de la liste
369         // data
370         listdouble8Iter ipointe; // donne la position dans la liste du maillon
371     };
372 /// @} // end of group
373
374 //=====
375 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
376 /// @{
377 ///
378 /// cas des tableaux de 9 réels
379 class Reels9
380 { public :
381     // Données :
382     double donnees [9];
383 };
384 /// @} // end of group
385
386 /*class Mat_3x3 // une vision explicite des identifiants d'une matrice 3x3
387 { public :
388     // Données :
389     double M11, M12, M13, M21, M22, M23, M31, M32, M33;
390 };
391
392 // une union permettant d'avoir deux mêmes visions des tableaux à 9 réels
393 union Reels9_Mat_3x3 { Reels9* s9; Mat_3x3* M; }; */
394
395 extern std::list < Reels9 > listdouble9;
396 typedef std::list < Reels9 >::iterator listdouble9Iter ;
397
398 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
399 /// @{
400 ///
401 /// cas des tableaux de 9 réels
402 class Reel9Pointe : public ReelsPointe
403 { public :
404     // constructeur
405     Reel9Pointe () : ipointe()
406     { listdouble9.push_front(Reels9()); // allocation d'un maillon
407       ipointe = listdouble9.begin(); // mémorisation du maillon
408     }
409     // destructeur
410     ~ Reel9Pointe ()
411     { listdouble9.erase(ipointe); } ; // suppression de l'élément de la liste
412     // data
413     listdouble9Iter ipointe; // donne la position dans la liste du maillon
414 };
415 /// @} // end of group
416
417 //=====
418 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
419 /// @{
420 ///
421 /// cas des tableaux de 16 réels
422 class Reels16
423 { public :
424     // Données :
425     double donnees [16];
426 };
427 /// @} // end of group
428
429 extern std::list < Reels16 > listdouble16;
430 typedef std::list < Reels16 >::iterator listdouble16Iter ;
431
432 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
433 /// @{
434 ///
435 /// cas des tableaux de 16 réels
436 class Reel16Pointe : public ReelsPointe
437 { public :
438     // constructeur
439     Reel16Pointe () : ipointe()
440     { listdouble16.push_front(Reels16()); // allocation d'un maillon
441       ipointe = listdouble16.begin(); // mémorisation du maillon
442     }
443     // destructeur
444     ~ Reel16Pointe ()
445     { listdouble16.erase(ipointe); } ; // suppression de l'élément de la liste
446     // data
447     listdouble16Iter ipointe; // donne la position dans la liste du maillon
448 };
449 /// @} // end of group
450
451 //=====
452 /// @addtogroup Les_listes_de_petits_tableaux_de_reels

```

```

453 /// @{
454 ///
455 /// cas des tableaux de 21 réels
456 class Reels21
457 { public :
458     // Données :
459     double donnees [21];
460 };
461 /// @} // end of group
462
463 extern std::list < Reels21 > listdouble21;
464 typedef std::list < Reels21 >::iterator listdouble21Iter ;
465
466 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
467 /// @{
468 ///
469 /// cas des tableaux de 21 réels
470 class Reel21Pointe : public ReelsPointe
471 { public :
472     // constructeur
473     Reel21Pointe () : ipointe()
474     { listdouble21.push_front(Reels21()); // allocation d'un maillon
475       ipointe = listdouble21.begin(); // mémorisation du maillon
476     }
477     // destructeur
478     ~ Reel21Pointe ()
479     { listdouble21.erase(ipointe);} ; // suppression de l'élément de la liste
480     // data
481     listdouble21Iter ipointe; // donne la position dans la liste du maillon
482 };
483 /// @} // end of group
484
485 //=====
486 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
487 /// @{
488 ///
489 /// cas des tableaux de 36 réels
490 class Reels36
491 { public :
492     // Données :
493     double donnees [36];
494 };
495 /// @} // end of group
496
497 extern std::list < Reels36 > listdouble36;
498 typedef std::list < Reels36 >::iterator listdouble36Iter ;
499
500 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
501 /// @{
502 ///
503 /// cas des tableaux de 36 réels
504 class Reel36Pointe : public ReelsPointe
505 { public :
506     // constructeur
507     Reel36Pointe () : ipointe()
508     { listdouble36.push_front(Reels36()); // allocation d'un maillon
509       ipointe = listdouble36.begin(); // mémorisation du maillon
510     }
511     // destructeur
512     ~ Reel36Pointe ()
513     { listdouble36.erase(ipointe);} ; // suppression de l'élément de la liste
514     // data
515     listdouble36Iter ipointe; // donne la position dans la liste du maillon
516 };
517 /// @} // end of group
518
519 //=====
520 /// @addtogroup Les_listes_de_petits_tableaux_de_reels
521 /// @{
522 ///
523 /// cas des tableaux de 81 réels
524 class Reels81
525 { public :
526     // Données :
527     double donnees [81];
528 };
529 /// @} // end of group
530
531 extern std::list < Reels81 > listdouble81;
532 typedef std::list < Reels81 >::iterator listdouble81Iter ;
533
534 /// @addtogroup Les_pointeurs_dans_listes_de_petits_tableaux
535 /// @{
536 ///
537 /// cas des tableaux de 81 réels
538 class Reel81Pointe : public ReelsPointe
539 { public :

```

```

540     // constructeur
541     Reel81Pointe () : ipointe()
542     { listdouble81.push_front(Reels81()); // allocation d'un maillon
543       ipointe = listdouble81.begin(); // mémorisation du maillon
544     }
545     // destructeur
546     ~ Reel81Pointe ()
547     { listdouble81.erase(ipointe); } ; // suppression de l'élément de la liste
548     // data
549     listdouble81Iter ipointe; // donne la position dans la liste du maillon
550 };
551 /// @} // end of group
552
553 //=====
554
555
556 #endif

```

## 7.493 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/TypeBase/PtTabRel\_Princ.h

listes de petits tableaux de réels

### Variables

- std::list< [Reels1](#) > **listdouble1**  
*cas des tableaux de 1 réels*
- std::list< [Reels2](#) > **listdouble2**  
*cas des tableaux de 2 réels*
- std::list< [Reels3](#) > **listdouble3**  
*cas des tableaux de trois réels*
- std::list< [Reels4](#) > **listdouble4**  
*cas des tableaux de quatre réels*
- std::list< [Reels5](#) > **listdouble5**  
*cas des tableaux de 5 réels*
- std::list< [Reels6](#) > **listdouble6**  
*cas des tableaux de 6 réels*
- std::list< [Reels7](#) > **listdouble7**  
*cas des tableaux de 7 réels*
- std::list< [Reels8](#) > **listdouble8**  
*cas des tableaux de 8 réels*
- std::list< [Reels9](#) > **listdouble9**  
*cas des tableaux de 9 réels*
- std::list< [Reels16](#) > **listdouble16**  
*cas des tableaux de 16 réels*
- std::list< [Reels21](#) > **listdouble21**  
*cas des tableaux de 36 réels*
- std::list< [Reels36](#) > **listdouble36**  
*cas des tableaux de 36 réels*
- std::list< [Reels81](#) > **listdouble81**  
*cas des tableaux de 81 réels*

### 7.493.1 Description détaillée

listes de petits tableaux de réels

listes de petits tableaux de réels ce fichier initialise les listes générales. Il ne doit être inclus que dans le fichier qui contient le programme principal on n'inclus pas la def des class de base 1,2,3, ... reel, en général arrivée à l'inclusion elles sont déjà définis

## 7.494 PtTabRel\_Princ.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file PtTabRel_Princ.h
2   \brief listes de petits tableaux de réels
3
4   listes de petits tableaux de réels
5   ce fichier initialise les listes générales. Il ne doit être inclus que dans le fichier
6   qui contient le programme principal
7   on n'inclus pas la def des class de base 1,2,3, ... reel, en général arrivée à l'inclusion
8   elles sont déjà définis
9 */
10
11 // This file is part of the Herezh++ application.
12 //
13 // The finite element software Herezh++ is dedicated to the field
14 // of mechanics for large transformations of solid structures.
15 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
16 // INSTITUT DE RECHERCHE DUPIY DE LÔME (IRDL) <https://www.irdl.fr/>.
17 //
18 // Herezh++ is distributed under GPL 3 license ou ultérieure.
19 //
20 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
21 // AUTHOR : Gérard Rio
22 // E-MAIL : gerardrio56@free.fr
23 //
24 // This program is free software: you can redistribute it and/or modify
25 // it under the terms of the GNU General Public License as published by
26 // the Free Software Foundation, either version 3 of the License,
27 // or (at your option) any later version.
28 //
29 // This program is distributed in the hope that it will be useful,
30 // but WITHOUT ANY WARRANTY; without even the implied warranty
31 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
32 // See the GNU General Public License for more details.
33 //
34 // You should have received a copy of the GNU General Public License
35 // along with this program. If not, see <https://www.gnu.org/licenses/>.
36 //
37 // For more information, please consult: <https://herezh.irdl.fr/>.
38
39 // listes de petits tableaux de réels
40 // ce fichier initialise les listes générales. Il ne doit être inclus que dans le fichier
41 // qui contient le programme principal
42 // on n'inclus pas la def des class de base 1,2,3, ... reel, en général arrivée à l'inclusion
43 // elles sont déjà définis
44
45
46 //=====
47 /// cas des tableaux de 1 réels
48 std::list <Reels1> listdouble1;
49 //=====
50 /// cas des tableaux de 2 réels
51 std::list <Reels2> listdouble2;
52 //=====
53 /// cas des tableaux de trois réels
54 std::list <Reels3> listdouble3;
55 //=====
56 /// cas des tableaux de quatre réels
57 std::list <Reels4> listdouble4;
58 //=====
59 /// cas des tableaux de 5 réels
60 std::list <Reels5> listdouble5;
61 //=====
62 /// cas des tableaux de 6 réels
63 std::list <Reels6> listdouble6;
64 //=====
65 /// cas des tableaux de 7 réels
66 std::list <Reels7> listdouble7;
67 //=====
68 /// cas des tableaux de 8 réels
69 std::list <Reels8> listdouble8;
70 //=====
71 /// cas des tableaux de 9 réels
72 std::list <Reels9> listdouble9;
73 //=====
74 /// cas des tableaux de 16 réels
75 std::list <Reels16> listdouble16;
76 //=====
77 /// cas des tableaux de 36 réels
78 std::list <Reels21> listdouble21;
79 //=====
80 /// cas des tableaux de 36 réels
81 std::list <Reels36> listdouble36;
82 //=====
83 /// cas des tableaux de 81 réels
84 std::list <Reels81> listdouble81;
85 //=====
86
87

```



## 7.495 Section.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // FICHER : Sect.h
30 // CLASSE : Sect
31 /*****
32 *      DATE:      06/03/2023
33 *
34 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:    Herezh++
37 *
38 *
39 *      BUT:      Conteneur très basique pour les sections
40 *
41 *      *****
42 *
43 *****/
44
45 #ifndef SECT_SIMPLE_H
46 #define SECT_SIMPLE_H
47
48 #include <iostream>
49 #include <fstream>
50
51 //
52 //-----
53 //!      Conteneur très basique pour les sections
54 //-----
55
56      /// \author      Gérard Rio
57      /// \version    1.0
58      /// \date      06/03/2023
59
60 class Sect
61 {public:
62     // VARIABLES PUBLIQUES :
63     double section0,section_t,section_tdt;
64     // CONSTRUCTEURS :
65     Sect(): section0(0.),section_t(0.),section_tdt(0.) {};
66     Sect(const double ep0,const double ept, const double eptdt) :
67         section0(ep0),section_t(ept),section_tdt(eptdt)
68     {};
69     Sect(const Sect & a) :
70         section0(a.section0),section_t(a.section_t)
71         ,section_tdt(a.section_tdt)
72     {};
73     // DESTRUCTEUR :
74     ~Sect() {};
75
76     // METHODES PUBLIQUES :
77     // surcharge de l'affectation
78     Sect& operator= (const Sect& a)
79     { section0 = a.section0; section_t = a.section_t;
80       section_tdt=a.section_tdt;
81       return (*this);
82     };
83     // surcharge de l'operator de lecture
84     friend istream & operator » (istream & ent, Sect & de)
85     { ent » de.section0 » de.section_t

```

```

86         » de.section_tdt;
87         return ent;
88     };
89     // surcharge de l'operator d'écriture
90     friend ostream & operator << (ostream & sort , const Sect & de)
91     { sort << " " << de.section0 << " " << de.section_t
92       << " " << de.section_tdt << " ";
93       return sort;
94     };
95
96 };
97
98 #endif

```

## 7.496 Temps\_CPU\_HZpp.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           01/02/2016
31 *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:         Herezh++
35 *
36 *   *****
37 *   BUT:  une classe dédiée à la gestion des temps d'exécution
38 *   il s'agit ici uniquement du temps user
39 *
40 *   *****
41 *
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !           but
45 *   -----
46 *   !           !           !
47 *   *****
48 *
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but
51 *   -----
52 *
53 *   *****/
54 #ifndef TEMPS_CPU_HZPP_H
55 #define TEMPS_CPU_HZPP_H
56
57 #include <iomanip>
58 #include <string.h>
59 #include <iostream>
60 #include <stdlib.h>
61 #include "Sortie.h"
62 #include "ParaGlob.h"
63 #include "Enum_IO_XML.h"
64
65 // ne fonctionne que si on accepte Boost
66 #ifdef UTILISATION_DE_LA_LIBRAIRIE_BOOST
67     #include <boost/chrono/include.hpp>
68     #include <boost/system/error_code.hpp>
69     using namespace boost::chrono;

```

```

70 #ifndef BOOST_CHRONO_HAS_PROCESS_CLOCKS
71 #define BOOST_CHRONO_HAS_PROCESS_CLOCKS
72 #endif
73
74 // typedef boost::chrono::duration<long long, boost::micro> microseconds; // classique
75 // typedef boost::chrono::duration<long long, boost::nano> nanoseconds; // classique
76 // typedef boost::chrono::duration<double, boost::micro> flotant_microseconds; // ajout GR
77 #include <boost/chrono/round.hpp>
78
79 // format duration as [-]d/hh:mm:ss.cc
80 template <class CharT, class Traits, class Rep, class Period>
81 std::basic_ostream<CharT, Traits>&
82 display(std::basic_ostream<CharT, Traits>& os,
83         boost::chrono::duration<Rep, Period> d)
84 {
85     using namespace std;
86     using namespace boost;
87
88     typedef boost::chrono::duration<long long, boost::ratio<86400> > days;
89     typedef boost::chrono::duration<long long, boost::centi> centiseconds;
90
91     // if negative, print negative sign and negate
92     if (d < boost::chrono::duration<Rep, Period>(0))
93     {
94         d = -d;
95         os << '-';
96     }
97     // round d to nearest centiseconds, to even on tie
98     centiseconds cs = boost::chrono::duration_cast<centiseconds>(d);
99     if (d - cs > boost::chrono::milliseconds(5)
100         || (d - cs == boost::chrono::milliseconds(5) && cs.count() & 1))
101         ++cs;
102     // separate seconds from centiseconds
103     boost::chrono::seconds s = boost::chrono::duration_cast<boost::chrono::seconds>(cs);
104     cs -= s;
105     // separate minutes from seconds
106     boost::chrono::minutes m = boost::chrono::duration_cast<boost::chrono::minutes>(s);
107     s -= m;
108     // separate hours from minutes
109     boost::chrono::hours h = boost::chrono::duration_cast<boost::chrono::hours>(m);
110     m -= h;
111     // separate days from hours
112     days dy = boost::chrono::duration_cast<days>(h);
113     h -= dy;
114     // print d/hh:mm:ss.cc
115     os << dy.count() << '/';
116     if (h < boost::chrono::hours(10))
117         os << '0';
118     os << h.count() << ':';
119     if (m < boost::chrono::minutes(10))
120         os << '0';
121     os << m.count() << ':';
122     if (s < boost::chrono::seconds(10))
123         os << '0';
124     os << s.count() << '.';
125     if (cs < centiseconds(10))
126         os << '0';
127     os << cs.count();
128     return os;
129 }
130
131
132
133
134 #else // sinon en attendant on définit des types par défaut
135     typedef long hours;
136     typedef long minutes;
137     typedef long seconds;
138     typedef long long milliseconds;
139     typedef long long microseconds;
140     typedef long long nanoseconds;
141 #endif
142
143 //-----
144 //! Temps_CPU_HZpp une classe dédiée à la gestion des temps d'exécution, il s'agit ici uniquement
145 //-----
146 // du temps user
147 //-----
148 /// \author Gérard Rio
149 /// \version 1.0
150 /// \date 01/02/2016
151
152 class Temps_CPU_HZpp
153 {
154     // surcharge de l'operator de lecture
155     friend istream & operator >> (istream & ent, Temps_CPU_HZpp & a);
156
157     // surcharge de l'operator d'écriture

```

```

156     friend ostream & operator << (ostream & sort , const Temps_CPU_HZpp & a);
157
158 public :
159     // CONSTRUCTEURS :
160     Temps_CPU_HZpp(); // par défaut, initialise à défaut
161
162     // constructeur de copie
163     Temps_CPU_HZpp(const Temps_CPU_HZpp& tps):
164         temps_user(tps.temps_user), debut_temps_user(tps.debut_temps_user)
165         , comptage_en_cours(tps.comptage_en_cours){};
166
167     // DESTRUCTEUR :
168     ~Temps_CPU_HZpp(){};
169
170     // --- METHODES PUBLIQUES :
171     // mise en route du comptage
172     void Mise_en_route_du_comptage();
173     // arrêt du comptage et cumul
174     void Arrêt_du_comptage();
175     // indique si oui ou non, on est en phase de comptage
176     // permet de fermer un comptage en catastrophe,
177     bool Comptage_en_cours() const {return comptage_en_cours;};
178     // retour de la valeur actuelle du temps_cpu user en milliseconde
179     // long long Temps_CPU_User() const {return temps_user.count()/1000;};
180     // retour de la valeur actuelle du temps_cpu user en microseconde
181     long long Temps_CPU_User() const {return temps_user.count()/1000;};
182     // retour de la valeur actuelle du temps_cpu user en milliseconde
183     long long Temps_CPU_User_milli() const {return temps_user.count()/1000000;};
184
185     // affichage en microsecondes
186     // si niveau = 1 ou plus, on a plus d'info pour le debug par exemple
187     void Affiche(ostream & sort,int niveau = 0);
188     // affichage en hh:mn:s:ml
189     void Affiche_hh_mn_s_ml(ostream & sort);
190
191
192     // surcharge de l'affectation
193     Temps_CPU_HZpp& operator= (const Temps_CPU_HZpp& de)
194     { temps_user = de.temps_user; return (*this);};
195     // accumulation
196     Temps_CPU_HZpp& operator+= (const Temps_CPU_HZpp& de)
197     { temps_user += de.temps_user; return (*this);};
198     // diminution
199     Temps_CPU_HZpp& operator-= (const Temps_CPU_HZpp& de)
200     { temps_user -= de.temps_user; return (*this);};
201
202     // surcharge de lecture en XML
203     istream & LectXML_Temps_CPU_HZpp(istream & ent);
204     // surcharge d'écriture en XML
205     ostream & EcritXML_Temps_CPU_HZpp(ostream & sort);
206
207     // Surcharge d'opérateurs logiques: ne concerne que la durée reel
208     bool operator == (const Temps_CPU_HZpp& a) const
209     { if ( temps_user == a.temps_user) return true; else return false;};
210     bool operator != (const Temps_CPU_HZpp& a) const { return !( *this == a);};
211     // surcharge de l'opérateur de comparaison :
212     bool operator > (const Temps_CPU_HZpp& a) const { return (this->temps_user > a.temps_user);};
213     bool operator >= (const Temps_CPU_HZpp& a) const { return (this->temps_user >= a.temps_user);};
214     bool operator < (const Temps_CPU_HZpp& a) const { return (this->temps_user < a.temps_user);};
215     bool operator <= (const Temps_CPU_HZpp& a) const { return (this->temps_user <= a.temps_user);};
216
217     // sortie du schemaXML: en fonction de enu
218     void SchemaXML_Temps_CPU_HZpp(ofstream& sort,const Enum_IO_XML enu) const ;
219
220
221 private :
222     // VARIABLES PROTEGEES :
223
224     // def des variables du process
225     bool comptage_en_cours; // indique si oui ou non on est dans une phase de comptage
226     // microsecondes temps_user;
227     nanoseconds temps_user;
228     // def des variables de début: au lieu d'utiliser un time_point et une duration
229     // j'utilise directement des microsecondes car sinon tout est en nanoseconde et
230     // il faut redéfinir un compteur en microsecondes: c'est une solution possible qui a été faite
231     // dans Chrono_GR.h mais je ne suis pas sûr que 1) cela soit indispensable, 2) que cela soit pérenne
232     // car j'ai été obligé de récupérer du code de boost et de l'adapter du coup c'est une usine dont je
233     // ne
234     // suis pas sûr de bien tout maîtriser donc j'utilise une solution plus simple
235     // microsecondes debut_temps_user;
236     nanoseconds debut_temps_user;
237
238     // gestion schéma XML
239     static short int impre_schem_XML;
240 };
241

```

```
242 #endif
243
```

## 7.497 Temps\_CPU\_HZpp\_3.h

```
1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           01/02/2016
31 *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:         Herezh++
35 *
36 *   *****
37 *   BUT:  une classe dédiée à la gestion des temps d'exécution
38 *   ici on cumule 3 compteurs: user, système et réel
39 *
40 *   *****
41 *
42 *   *****/
43 #ifndef TEMPS_CPU_HZPP_3_H
44 #define TEMPS_CPU_HZPP_3_H
45
46 #include <iomanip>
47 #include <string.h>
48 #include <iostream>
49 #include <stdlib.h>
50 #include "Sortie.h"
51 #include "ParaGlob.h"
52 #include "Enum_IO_XML.h"
53 #include "Temps_CPU_HZpp.h"
54
55 // ne fonctionne que si on accepte Boost
56 #ifdef UTILISATION_DE_LA_LIBRAIRIE_BOOST
57 #include <boost/chrono/include.hpp>
58 #include <boost/system/error_code.hpp>
59 using namespace boost::chrono;
60 // typedef boost::chrono::duration<long long, boost::micro> microseconds; // classique
61 // typedef boost::chrono::duration<double, boost::micro> flotant_microseconds; // ajout GR
62 #include <boost/chrono/round.hpp>
63
64 #else // sinon en attendant on définit des types par défaut
65 typedef long hours;
66 typedef long minutes;
67 typedef long seconds;
68 typedef long long milliseconds;
69 typedef long long microseconds;
70 #endif
71
72 //-----
73 //! Temps_CPU_HZpp_3 une classe dédiée à la gestion des temps d'exécution, ici on cumule 3
74 //-----
75 /// \author Gérard Rio
76 /// \version 1.0
77 /// \date 01/02/2016
78
79 class Temps_CPU_HZpp_3
```

```

80 {
81     // surcharge de l'operator de lecture
82     friend istream & operator » (istream & ent, Temps_CPU_HZpp_3 & a);
83
84     // surcharge de l'operator d'écriture
85     friend ostream & operator « (ostream & sort, const Temps_CPU_HZpp_3 & a);
86
87 public :
88     // CONSTRUCTEURS :
89     Temps_CPU_HZpp_3(); // par défaut, initialise à défaut
90
91     // DESTRUCTEUR :
92     ~Temps_CPU_HZpp_3(){};
93
94     // --- METHODES PUBLIQUES :
95     // mise en route du comptage
96     void Mise_en_route_du_comptage();
97     // arrêt du comptage et cumul
98     void Arrêt_du_comptage();
99     // retour de la valeur actuelle du temps_cpu user en milliseconde
100    long long Temps_CPU_User() const {return temps_user.count()/1000;};
101    // idem temps_cpu system
102    long long Temps_CPU_System() const {return temps_system.count()/1000;};
103    // idem temps_cpu reel
104    long long Temps_CPU_Reel() const {return temps_reel.count()/1000;};
105
106    // affichage en microsecondes
107    // si niveau = 1 ou plus, on a plus d'info pour le debug par exemple
108    void Affiche(ostream & sort,int niveau = 0);
109    // affichage en hh:mn:s:ml
110    void Affiche_hh_mn_s_ml(ostream & sort);
111
112
113    // surcharge de l'affectation
114    Temps_CPU_HZpp_3& operator= (const Temps_CPU_HZpp_3& de)
115    { temps_user = de.temps_user; temps_reel = de.temps_reel;
116      temps_system = de.temps_system;
117      return (*this);};
118    // surcharge de lecture en XML
119    istream & LectXML_Temps_CPU_HZpp_3(istream & ent);
120    // surcharge d'écriture en XML
121    ostream & EcritXML_Temps_CPU_HZpp_3(ostream & sort);
122
123    // sortie du schemaXML: en fonction de enu
124    void SchemaXML_Temps_CPU_HZpp_3(ofstream& sort,const Enum_IO_XML enu) const ;
125
126
127 private :
128     // VARIABLES PROTEGEES :
129
130     // def des variables du process
131     microseconds temps_user;
132     microseconds temps_system;
133     microseconds temps_reel;
134     // def des variables de début: au lieu d'utiliser un time_point et une duration
135     // j'utilise directement des microsecondes car sinon tout est en nanoseconde et
136     // il faut redéfinir un compteur en microsecondes: c'est une solution possible qui a été faite
137     // dans Chrono_GR.h mais je ne suis pas sûr que 1) cela soit indispensable, 2) que cela soit pérenne
138     // car j'ai été obligé de récupérer du code de boost et de l'adapter du coup c'est une usine dont je
139     ne
140     // suis pas sûr de bien tout maîtriser donc j'utilise une solution plus simple
141     microseconds debut_temps_user;
142     microseconds debut_temps_system;
143     microseconds debut_temps_reel;
144
145     // gestion schéma XML
146     static short int impre_schem_XML;
147 };
148
149 #endif
150

```

## 7.498 TypeQuelconque.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)

```

```

11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      29/05/2004
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *   $
37 *
38 *   BUT:   une classe générique qui permet de définir
39 *          des identificateurs de grandeurs quelconque.
40 *          Ces identificateurs sont prévue en particulier pour gérer
41 *          les sorties de résultats.
42 *          Le type s'appuie sur un type énuméré pour les gestions
43 *          rapide de tableau ou de choix (via case).
44 *          Le type contient une grandeur pointée, mais ne crée pas
45 *          ni ne détruit la grandeur pointée ! Ainsi au niveau de
46 *          l'affectation, il ne gère que le changement de pointeur.
47 *
48 *   $
49 *
50 *   VERIFICATION:
51 *
52 *   ! date !   auteur !           but
53 *   -----
54 *   !           !           !           !
55 *   $
56 *
57 *   MODIFICATIONS:
58 *
59 *   ! date !   auteur !           but
60 *   -----
61 *   $
62 */
63 #ifndef TYPE_QUELCONQUE_H
64 #define TYPE_QUELCONQUE_H
65
66 #include <iomanip>
67 #include "List_io.h"
68 #include "Tableau_T.h"
69 #include <algorithm>
70 #include <string.h>
71
72 #include "EnumTypeGrandeur.h"
73 #include "Enum_TypeQuelconque.h"
74 #include "EnuTypeQuelParticulier.h"
75 #include "Enum_ddl.h"
76 #include "Mat_pleine.h"
77 #include "TypeQuelconque_enum_etendu.h"
78
79 /** @defgroup Group_TypeQuelconque
80 *
81 *   BUT:   une classe générique qui permet de définir
82 *          des identificateurs de grandeurs quelconque.
83 *          Ces identificateurs sont prévue en particulier pour gérer
84 *          les sorties de résultats.
85 *          Le type s'appuie sur un type énuméré pour les gestions
86 *          rapide de tableau ou de choix (via case).
87 *          Le type contient une grandeur pointée, mais ne crée pas
88 *          ni ne détruit la grandeur pointée ! Ainsi au niveau de
89 *          l'affectation, il ne gère que le changement de pointeur.
90 *
91 *   \author   Gérard Rio
92 *   \version  1.0
93 *   \date     29/05/2004
94 *   \brief    classe qui permet de définir des identificateurs de grandeurs quelconque.
95 */
96

```

```

97 /// @addtogroup Group_TypeQuelconque
98 /// @{
99
100 /// TypeQuelconque : def de la classe générique globale qui sert a gérer la grandeur
101 /// particulière qui est attachée via un pointeur
102
103 class TypeQuelconque
104 {
105     // surcharge de l'operator de lecture
106     friend istream & operator » (istream & ent, TypeQuelconque & a);
107
108     // surcharge de l'operator d'écriture
109     friend ostream & operator « (ostream & sort , const TypeQuelconque & a);
110
111     public :
112     //-----
113     /// définition de la classe virtuelle qui serait déclinée en une grandeur particulière
114     /// il s'agit ici de la classe conteneur
115     //-----
116     class Grandeur
117     { // surcharge de l'operator de lecture
118         friend istream & operator » (istream & ent, Grandeur & a)
119         { a.Lecture_grandeur(ent); return ent; };
120         // surcharge de l'operator d'écriture
121         friend ostream & operator « (ostream & sort , const Grandeur & a)
122         { a.Ecriture_grandeur(sort); return sort; };
123         public: virtual ~Grandeur(){};
124         virtual Grandeur* New_idem_grandeur() const =0; // ramène une grandeur identique, créée
125
126         virtual TypeQuelconque::Grandeur & operator = ( const TypeQuelconque::Grandeur & a) = 0; //
127         surcharge d'affectation (totale a priori)
128         // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
129         // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
130         même type
131         // affectation uniquement des données numériques (pas les pointeurs, pas les string)
132         virtual void Affectation_numerique( const TypeQuelconque::Grandeur & a) = 0;
133         // Surcharge de l'operator +=
134         virtual void operator+= (const Grandeur& c) = 0;
135         // Surcharge de l'operator -=
136         virtual void operator-= (const Grandeur& c) = 0;
137         // Surcharge de l'operator *=
138         virtual void operator*= (double val) = 0;
139         // Surcharge de l'operator /= : division par un scalaire
140         virtual void operator/= (double val) = 0;
141         // récup du nom de référence si diff de NULL
142         // c'est un nom qui permet de discriminer différentes grandeurs quelconque qui
143         // ont la même signature (mêmes énumérés) Cependant n'existe pas toujours
144         virtual const string* Nom_ref() const {return NULL;};
145         // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
146         // si pas possible, ramène 0
147         virtual double GrandeurNumOrdre(int num) const =0;
148         virtual int NbMaxiNumeroOrdre() const = 0; // récup du nb maxi de numéros d'ordres
149         virtual void Grandeur_brut(ostream & sort,int nbcar) const =0; // sortie de la grandeur brut
150
151         sur sort
152         // ramène le type de grandeur associée de la grandeur de base, car
153         // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
154         virtual EnumTypeGrandeur Type_grandeurAssocie() const =0;
155         // ramène le type de la structure: tableau, liste ext..
156         virtual EnumType2Niveau Type_structure_grandeurAssocie() const = 0;
157         // ramène le type de la grandeur particulière
158         virtual EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const = 0;
159         // change de repère de la grandeur si nécessaire
160         virtual void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma) =0;
161         // lecture
162         virtual istream & Lecture_grandeur(istream & ent) =0;
163         // écriture
164         virtual ostream & Ecriture_grandeur(ostream & sort) const =0;
165         // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
166         virtual void InitParDefaut() = 0;
167     };
168
169     // CONSTRUCTEURS :
170     // constructeur par défaut
171     // ensuite il faut utiliser ChangeGrandeur pour affecter une bonne valeur
172     TypeQuelconque();
173     // constructeur fonction d'un type donnée et d'une grandeur
174     // c'est le 1er constructeur officiel, qui sert
175     // posi_ddl: indique un ddl dont les points de calcul (ex: les pt d'integ) sont les même
176     // que la grandeur particulière
177     TypeQuelconque(TypeQuelconque_enum_etendu enuType,Enum_ddl posi_ddl,const TypeQuelconque::Grandeur&
178     pt_gr);
179     // le même constructeur mais à partir d'un énum
180     TypeQuelconque(EnumTypeQuelconque enuType,Enum_ddl posi_ddl,const TypeQuelconque::Grandeur& pt_gr);
181
182     // constructeur fonction d'un type donnée uniquement
183     // c'est le 2er constructeur officiel, qui sert
184     // ici il n'y a pas de grandeur associée, util uniquement pour gérer le type mais pas la grandeur

```



```

179     TypeQuelconque(TypeQuelconque_enum_etendu enuType);
180     // le même constructeur mais à partir d'un énum
181     TypeQuelconque(EnumTypeQuelconque enuType);
182
183 // -- ces deux constructeurs ont été imaginés, mais a priori ne sont pas utilisés ---
184 // // Constructeur fonction d'un type de donnée, et d'un énuméré permettant de construire la
    grandeur
185 // // celle-ci , si elle dépend de la dimension, est construite suivant la dimension courante
186 // // posi_ddl: indique un ddl dont les points de calcul (ex: les pt d'integ) sont les même
187 // // que la grandeur particulière
188 //     TypeQuelconque(TypeQuelconque_enum_etendu enuType, Enum_ddl posi_ddl, EnumTypeQuelParticulier
    enuGrandQuelParti);
189 // // le même constructeur mais à partir d'un énum
190 //     TypeQuelconque(EnumTypeQuelconque enuType, Enum_ddl posi_ddl, EnumTypeQuelParticulier
    enuGrandQuelParti);
191
192     // constructeur de copie
193     TypeQuelconque(const TypeQuelconque& a);
194     // DESTRUCTEUR :
195     ~TypeQuelconque();
196
197     // METHODES PUBLIQUES :
198
199     // changement d'une grandeur quelconque
200     // la grandeur initiale est transformée en la grandeur passée en paramètre
201     // utile après une construction par défaut
202     void ChangeGrandeur(const TypeQuelconque& a);
203
204     // surcharge d'affectation
205     // ne doit concerner que des grandeurs quelconques de même type
206     // il n'y a donc pas de changement automatique de type, volontairement pour éviter des erreurs
207     // et un long temps de traitement
208     TypeQuelconque & operator = ( const TypeQuelconque & a);
209
210     // surcharge d'égalité: ne concerne que les types, ne concerne pas la grandeur elle-même pointée !
211     bool operator == ( const TypeQuelconque & a) const { return (enuType == a.enuType);};
212     // surcharge de non égalité
213     bool operator != ( const TypeQuelconque & a) const { return (enuType != a.enuType);};
214     // surcharge de comparaison
215     bool operator > (const TypeQuelconque & a) const {return (enuType > a.enuType);};
216     bool operator >= (const TypeQuelconque & a) const {return (enuType > a.enuType);};
217     bool operator < (const TypeQuelconque & a) const {return (enuType < a.enuType);};
218     bool operator <= (const TypeQuelconque & a) const {return (enuType < a.enuType);};
219
220     // récup du type de grandeur associé
221     // il s'agit ici du type de base, car la grandeur peut être un type complexe comme un tableau
222     // de type de base ou une liste
223     EnumTypeGrandeur EnumTypeGrandeur() const {return enuType.TypeDeGrandeurTG();};
224     // récup du typeQuelconque
225     const TypeQuelconque_enum_etendu& EnumTypeQuelconque() const {return enuType;};
226     // écriture d'une chaîne de caractère composés de l'Enumtypequelconque et du string additionnel
227
228     // récup du type de calcul de la grandeur:
229     // = 1 indique si la grandeur est naturellement exprimé dans le repère globale
230     // = 0 indique que la grandeur est dans le repère naturelle (ou une combinaison)
231     int TypeExpressionGrandeur() const {return EnumTypeQuelconqueGlobale(enuType.EnumTQ());};
232     // sortie de la grandeur uniquement sur sort (sans info particulière c-a-d de typage)
233     void Grandeur_brut(ostream & sort, int nbcarr) const
    {Gestion_erreur();pt_grandeur->Grandeur_brut(sort,nbcarr);};
234     // ramène le ddl associé, qui correspond à un ddl éventuellement du même type, mais surtout
235     // qui est calculé au même point géométrique !
236     // si ce ddl n'existe pas => pas d'existence de la grandeur quelconque
237     Enum_ddl Enum() const {return enu_ddl_posi;};
238     // change de repère de la grandeur si nécessaire
239     void Change_repere(const Mat_pleine& beta0, const Mat_pleine& betat, const Mat_pleine& betatdt
    , const Mat_pleine& gamma0, const Mat_pleine& gammat, const Mat_pleine& gammatdt);
240
241     // indique si c'est un type numérique ou non
242     bool Type_numerique() const
    {return Type_grandeur_numerique(Type_de_grandeur_associee(enuType.EnumTQ()));};
243     // retourne le nombre maxi de numéros d'ordre (c'est à dire de numéros de classement
244     // des différents éléments de la grandeur, s'il y en a plusieurs
245     // pour un tableau, un vecteur une liste, c'est directement le numéro d'ordre
246     // pour des tenseurs, ou tableaux multidimensionnels c'est de dernier indice qui varie
247     // d'abord puis on remonte
248     int NbMaxiNumeroOrdre() const {Gestion_erreur();return pt_grandeur->NbMaxiNumeroOrdre();};
249     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
250     // si pas possible, ramène 0
251     double GrandeurNumOrdre(int num) const {Gestion_erreur();return
    pt_grandeur->GrandeurNumOrdre(num);};
252
253     // récupération de la grandeur en lecture/écriture pour un cast !!! attention il faut savoir ce que
    l'on fait
254     Grandeur* Grandeur_pointee() {return pt_grandeur;};
255     const Grandeur* Const_Grandeur_pointee() const {return pt_grandeur;};
256     // écriture de l'entête de la grandeur : TypeQuelconque_enum_etendu EnumType2Niveau EnumTypeGrandeur
    et nombre de grandeur
257     // numérique équivalente si c'est une grandeur numérique
258     void Ecriture_entete_grandeur(ostream & sort);

```

```

259 // --- lecture en 2 temps
260 // lecture des infos internes sauf de la grandeur
261 // retour de l'EnumTypeQuelParticulier de la grandeur
262 EnumTypeQuelParticulier Lecture_un (istream & ent);
263
264 // idem mais avec création du TypeQuelconque_enum_etendu interne
265 // s'il n'existe pas, il s'agit uniquement du nom de def, mais les infos
266 // associées: EnumTypeQuelconque et EnumTypeGrandeur sont lues et vérifiés
267 // le string, lui peut ne pas exister: intéressant lorsqu'il s'agit de grandeurs
268 // qui sont créés pendant le calcul et/ou en restart
269 EnumTypeQuelParticulier Lecture_un_avec_creation_TypeQuelconque_enum_etendu(istream & ent);
270
271 // gestion de l'activité du type quelconque
272 void Active() {actif=true;};
273 void Inactive() {actif = false;};
274 bool Activite() const {return actif;};
275
276 protected :
277 // VARIABLES PROTEGEES :
278 TypeQuelconque_enum_etendu enumType;
279 Enum_ddl enu_ddl_posi; // un enum d'un ddl qui est calculé au même point que le type quelconque
280 Grandeur * pt_grandeur; // le pointeur sur la grandeur correspondantes
281 bool actif; // un boolean qui permet une gestion du type: par exemple prise en compte ou non
282
283 // METHODES PROTEGEES :
284 // gestion de l'erreur dans le cas où il n'y a pas de grandeur associée
285 #ifdef MISE_AU_POINT
286 void Gestion_erreur() const ;
287 #endif
288 #ifndef MISE_AU_POINT
289 void Gestion_erreur() const {};
290 #endif
291
292 };
293
294 /// @} // end of group
295
296
297 #endif

```

## 7.499 TypeQuelconque\_enum\_etendu.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           03/12/2017                               *
31 *   *                                                       *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)     *
33 *   *                                                       *
34 *   PROJET:         Herezh++                                 *
35 *   *                                                       *
36 *   *                                                       *
37 *   BUT:   une classe qui permet de définir des identificateurs de *
38 *   grandeurs secondaires associés des EnumTypeQuelconque. *
39 *   Ces grandeurs ne sont pas vraiment des EnumTypeQuelconque, *
40 *   mais ils s'en déduisent. Ainsi pour ne pas alourdir l'enum *
41 *   de base on crée cette classe associée. Elle est prévue en particu- *
42 *   lier pour gérer les sorties de résultats. *
43 *   les éléments sont ordonnés selon la règle suivante : *

```

```

44 *          si c'est un EnumTypeQuelconque pur -> ordre de l'Enum      *
45 *          si c'est un élément de tab_Daa -> ordre dans tab_Daa      *
46 *          + maxi des EnumTypeQuelconque                             *
47 *          lorsqu'il s'agit d'un EnumTypeQuelconque tout simple,     *
48 *          dans ce cas Non_vider == true                             *
49 *          $                                                         *
50 *          //////////////////////////////////////////////////////////////////// *
51 *          $                                                         *
52 *          //////////////////////////////////////////////////////////////////// */
53 #ifndef TYPEQUELCONQUE_ENUM_ETENDUE_H
54 #define TYPEQUELCONQUE_ENUM_ETENDUE_H
55
56 #include "Enum_TypeQuelconque.h"
57 #include <string.h>
58 #include "List_io.h"
59 #include <iomanip>
60 #include <algorithm>
61
62 /** @defgroup Group_TypeQuelconque_enum_etendu Group_TypeQuelconque_enum_etendu
63 *
64 *      BUT: une classe qui permet de définir des identificateurs de
65 *      grandeurs secondaires associées des EnumTypeQuelconque.
66 *      Ces grandeurs ne sont pas vraiment des EnumTypeQuelconque,
67 *      mais ils s'en déduisent. Ainsi pour ne pas alourdir l'enum
68 *      de base on crée cette classe associée. Elle est prévue en particu-
69 *      lier pour gérer les sorties de résultats.
70 *      les éléments sont ordonnés selon la règle suivante :
71 *      si c'est un EnumTypeQuelconque pur -> ordre de l'Enum
72 *      si c'est un élément de tab_Daa -> ordre dans tab_Daa
73 *      + maxi des EnumTypeQuelconque
74 *      lorsqu'il s'agit d'un EnumTypeQuelconque tout simple,
75 *      dans ce cas Non_vider == true
76 *
77 *
78 * \author   Gérard Rio
79 * \version  1.0
80 * \date    03/12/2017
81 * \brief   classe qui permet de définir des identificateurs de grandeurs secondaires associées à des
82 *          EnumTypeQuelconque.
83 */
84
85 /// @addtogroup Group_TypeQuelconque_enum_etendu
86 /// @{
87
88 /// Initialisation_tab_Daa: définition d'une classe pour l'initialisation du tableau tab_Daa
89 class Initialisation_tab_Daa { public: Initialisation_tab_Daa(); };
90 /// @} // end of group
91
92 /// @addtogroup Group_TypeQuelconque_enum_etendu
93 /// @{
94
95 /// TypeQuelconque_enum_etendu: la classe qui gère les types quelconques étendus
96
97 class TypeQuelconque_enum_etendu
98 {
99 // surcharge de l'opérateur de lecture
100 friend istream & operator » (istream & ent, TypeQuelconque_enum_etendu & a)
101 { string no; ent » no; a=TypeQuelconque_enum_etendu(no);
102   ent » a.enu » a.type_grandeur;
103   return ent;};
104
105 // !!! voir aussi lecture éléments distinctes qui doit être cohérent
106
107 // surcharge de l'opérateur d'écriture
108 friend ostream & operator « (ostream & sort , const TypeQuelconque_enum_etendu & a)
109 { if (a.Non_vider()) sort « " " « NomTypeQuelconque(a.enu); else sort « " " « a.nom;
110   sort « " " « a.enu « " " « a.type_grandeur « " ";
111   return sort;};
112
113 // permission pour l'initialisation du tableau tab_Daa
114 friend class Initialisation_tab_Daa;
115
116 public :
117 // CONSTRUCTEURS :
118 // constructeur par défaut
119 TypeQuelconque_enum_etendu(EnumTypeQuelconque en = RIEN_TYPEQUELCONQUE, string no = "-");
120 // constructeur fonction d'un string
121 TypeQuelconque_enum_etendu( string no );
122 // constructeur fonction d'une chaîne
123 TypeQuelconque_enum_etendu( char * no );
124 // constructeur de copie
125 TypeQuelconque_enum_etendu(const TypeQuelconque_enum_etendu& a) :
126   nom(a.nom), enu(a.enu), posi_nom(a.posi_nom) ,type_grandeur(a.type_grandeur)
127   {};
128 // DESTRUCTEUR :
129 ~TypeQuelconque_enum_etendu() {};

```

```

130
131 // METHODES PUBLIQUES :
132
133 // ajout d'un nouveau typeQuelconque étendu
134 // s'il existait avant, il n'y a pas d'ajout
135 // ramène un exemplaire de la création
136 static TypeQuelconque_enum_etendu Ajout_un_TypeQuelconque_enum_etendu
137     (EnumTypeQuelconque enu, const string no, EnumTypeGrandeur type_grandeur);
138
139 // surcharge d'affectation
140 TypeQuelconque_enum_etendu & operator = ( const TypeQuelconque_enum_etendu & a)
141     { nom = a.nom; enu = a.enu; posi_nom = a.posi_nom; type_grandeur = a.type_grandeur;
142     return *this; };
143 // surcharge d'égalité
144 bool operator == ( const TypeQuelconque_enum_etendu & a) const
145     { if (posi_nom == a.posi_nom) return true; else return false; };
146 // surcharge de non égalité
147 bool operator != ( const TypeQuelconque_enum_etendu & a) const
148     { if (posi_nom != a.posi_nom) return true; else return false; };
149 // surcharge de comparaison
150 bool operator > (const TypeQuelconque_enum_etendu & a) const
151     { return (posi_nom > a.posi_nom); };
152 bool operator >= (const TypeQuelconque_enum_etendu & a) const
153     { return (posi_nom >= a.posi_nom); };
154 bool operator < (const TypeQuelconque_enum_etendu & a) const
155     { return (posi_nom < a.posi_nom); };
156 bool operator <= (const TypeQuelconque_enum_etendu & a) const
157     { return (posi_nom <= a.posi_nom); };
158
159 // récup de l'énumération
160 EnumTypeQuelconque EnumTQ() const {return enu; };
161 // récup du nom
162 string Nom() const {return nom; };
163 // récup du nom plein
164 string NomPlein() const
165     {if (nom=="-"){return NomTypeQuelconque(enu); }
166     else {return nom; };
167     };
168 // récup du nom générique (sans les indices de composantes)
169 string NomGenerique() const
170     {if (nom=="-"){return NomGeneriqueTypeQuelconque(enu); }
171     else {return string(nom.substr(1,nom.length()-2)); };
172     };
173 // position donne un numéro équivalent du type enum
174 int Position() const {return posi_nom; };
175 // modification du nom
176 // test pour savoir si le nom est vide
177 bool Nom_vide() const { if (nom == "-") return true; else return false; };
178 // retour le type de grandeur auquel appartient le type quelconque tendue
179 // par exemple : vecteur ou scalaire
180 EnumTypeGrandeur TypeDeGrandeurTG() const {return type_grandeur; };
181
182 // test pour savoir si le nom passer en paramètre est valide
183 // ramène vrai si no correspond à un EnumTypeQuelconque ou
184 // s'il correspond un type dérivé
185 static bool VerifExistence(string no) ;
186 // récupération d'un TypeQuelconque_enum_etendu correspondant un string
187 static TypeQuelconque_enum_etendu RecupTypeQuelconque_enum_etendu(string to) ;
188 // transformation d'une liste d'EnumTypeQuelconque en TypeQuelconque_enum_etendu
189 static List_io <TypeQuelconque_enum_etendu> TransfoList_io(const List_io <EnumTypeQuelconque> &
190     li) ;
191 // transformation d'un tableau d'énumération en un tableau de TypeQuelconque_enum_etendu
192 static Tableau < TypeQuelconque_enum_etendu > TransfoTableau(const Tableau <EnumTypeQuelconque> &
193     tab) ;
194 // test si un élément existe dans une liste donnée
195 static bool Existe_dans_la_liste
196     (const List_io <TypeQuelconque_enum_etendu> & lis, const TypeQuelconque_enum_etendu& dd) ;
197 // récupération du nombre maximum de ddl tendu existant
198 static int NBmax_TypeQuelconque_enum_etendue()
199     {return taillTab+ClassPourEnumTypeQuelconque::NombreEnumTypeQuelconque(); };
200 // lecture éléments distinctes: idem la surcharge de lecture mais sans création d'un
201 // TypeQuelconque_enum_etendu
202 // permet de récupérer les grandeurs pour crer un nouveau type étendu
203 // on ne le fait pas systématiquement, de manière à pouvoir contrôler les créations éventuelles
204 static void Lecture_element_distinctes(istream & ent, string& no, EnumTypeQuelconque& enu
205     , EnumTypeGrandeur& type_grandeur)
206     { ent » no » enu » type_grandeur; };
207 // lecture avec création éventuelle si le type n'existe pas
208 static TypeQuelconque_enum_etendu Lecture_avec_creation_eventuelle(istream & ent);
209
210 protected :
211 // VARIABLES PROTEGEES :
212 string nom;
213 EnumTypeQuelconque enu;
214 int posi_nom; // position du nom : équivalent à un type énuméré
215 EnumTypeGrandeur type_grandeur;

```

```

215
216 // on définit le tableau des TypeQuelconque_enum_etendu qui sont valides
217 // en variables globales
218 static Tableau < TypeQuelconque_enum_etendu > tab_Daa;
219 static Initialisation_tab_Daa init_tab_Daa;
220 static int taillTab; // taille du tableau tab_Daa
221
222 // METHODES PROTEGEES :
223
224 };
225 /// @} // end of group
226
227 #endif

```

## 7.500 TypeQuelconqueParticulier.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 * DATE: 30/05/2004 *
31 * * $ *
32 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
33 * * $ *
34 * PROJET: Herezh++ *
35 * * $ *
36 *****/
37 * BUT: classes qui permettent de définir des identificateurs de *
38 * grandeurs particulières héritants du TypeQuelconque. *
39 * Ces identificateurs sont prévue en particulier pour gérer *
40 * les sorties de résultats. *
41 * * $ *
42 * ***** *
43 * * $ *
44 *****/
45 #ifndef TYPE_QUELCONQUE_PARTICULIER_H
46 #define TYPE_QUELCONQUE_PARTICULIER_H
47
48 #include <string.h>
49 #include "EnumTypeGrandeur.h"
50
51 #include "List_io.h"
52 #include <iomanip>
53 #include <algorithm>
54 #include "TypeQuelconque.h"
55 #include "Tenseur.h"
56 #include "NevezTenseur.h"
57 #include "Tableau_T.h"
58 #include "Tableau2_T.h"
59 #include "Ddl_etendu.h"
60 #include "Fonction_nD.h"
61 #include "Base.h"
62
63
64 /** @defgroup Les_grandeurs_particulieres Les_grandeurs_particulieres
65 *
66 * BUT: classes qui permettent de définir des identificateurs de
67 * grandeurs particulières héritants du TypeQuelconque.
68 * Ces identificateurs sont prévue en particulier pour gérer
69 * les sorties de résultats.

```

```

70  *
71  *
72  * \author    Gérard Rio
73  * \version  1.0
74  * \date     30/05/2004
75  * \brief    Définition des grandeurs particulières.
76  *
77  */
78
79  //-----
80  //| définition des fonctions permettant la création de la grandeur particulière |
81  //| en lecture après la lecture d'un énuméré de type particulier |
82  //-----
83
84  TypeQuelconque::Grandeur* NevezGrandeurParticuliereEnLecture(EnuTypeQuelParticulier enu,istream & ent);
85
86
87
88  /// @addtogroup Les_grandeurs_particulieres
89  /// @{
90  ///
91
92  //-----
93  /// grandeur par défaut: TypeQuelconque::Grandeur::Grandeur_defaut|
94  //-----
95  class Grandeur_defaut : public TypeQuelconque::Grandeur
96  { // surcharge de l'opérateur de lecture
97    friend istream & operator » (istream & ent, Grandeur_defaut & a);
98    // surcharge de l'opérateur d'écriture
99    friend ostream & operator « (ostream & sort , const Grandeur_defaut & a) ;
100   public:
101     // constructeur par défaut
102     Grandeur_defaut() {};
103     ~Grandeur_defaut() {};
104     TypeQuelconque::Grandeur* New_idem_grandeur() const; // ramène une grandeur identique, créée
105     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
106     // ** opérations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
107     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
108     même type
109     // ***** ici toutes ces opérations sont illicites !!! *****
110     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
111     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
112     // Surcharge de l'opérateur +=
113     void operator+= (const Grandeur & c) ;
114     // Surcharge de l'opérateur -=
115     void operator-= (const Grandeur & c) ;
116     // Surcharge de l'opérateur *=
117     void operator*= (double val);
118     // Surcharge de l'opérateur /= : division par un scalaire
119     void operator/= (double val) ;
120     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
121     // si pas possible, ramène 0
122     double GrandeurNumOrdre(int num) const ;
123     int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
124     void Grandeur_brut(ostream & sort,int nbcars) const; // sortie de la grandeur brut sur sort
125     // ramène le type de grandeur associée de la grandeur de base, car
126     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
127     EnumTypeGrandeur Type_grandeurAssocie() const {return RIEN_TYPEGRANDEUR;};
128     // ramène le type de la structure: tableau, liste ext..
129     EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE;};
130     // ramène le type de la grandeur particulière
131     EnuTypeQuelParticulier Type_enumGrandeurParticuliere()
132     const{return RIEN_TYPE_QUELCONQUE_PARTICULIER;};
133     // change de repère de la grandeur si nécessaire
134     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma) {};
135     // lecture
136     istream & Lecture_grandeur(istream & ent) { ent » *this;return ent;};
137     // écriture
138     virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort « *this; return sort;};
139     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
140     void InitParDefaut();
141 };
142
143  /// @} // end of group
144
145  /// @addtogroup Les_grandeurs_particulieres
146  /// @{
147  ///
148  //-----
149  /// grandeur TenseurHH: TypeQuelconque::Grandeur::Grandeur_TenseurHH|
150  //-----
151  class Grandeur_TenseurHH : public TypeQuelconque::Grandeur
152  { friend class Tab_Grandeur_TenseurHH;friend class Tab2_Grandeur_TenseurHH;
153    // surcharge de l'opérateur de lecture
154    friend istream & operator » (istream & ent, Grandeur_TenseurHH & a);
155    // surcharge de l'opérateur d'écriture
156    friend ostream & operator « (ostream & sort , const Grandeur_TenseurHH & a);

```

```

156     public:
157     // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
158     // ptTens défini, donc -> erreur
159     Grandeur_TenseurHH();
160     // constructeur légal
161     Grandeur_TenseurHH(const TenseurHH& tens): ptTens(NevetzTenseurHH(tens)) {};
162     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
163     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
164     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
165     Grandeur_TenseurHH(istream & ent);
166     // constructeur de copie
167     Grandeur_TenseurHH(const Grandeur_TenseurHH& a): ptTens(NevetzTenseurHH(*a.ptTens)) {};
168     // destructeur
169     ~Grandeur_TenseurHH();
170     // ramène une grandeur identique, créée
171     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Grandeur_TenseurHH(*ptTens);};
172     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
173     // le constructeur adoc l'objet correcte et valide
174     void EcriturePourLectureAvecCreation(ostream & sort);
175     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
176     Grandeur & operator = ( const Grandeur_TenseurHH & a); // surcharge d'affectation
177     // ** opérations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
178     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
179     même type
180     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
181     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
182     // Surcharge de l'opérateur +=
183     void operator+= (const Grandeur& c) ;
184     void operator+= (const Grandeur_TenseurHH& aa) {(*ptTens) += (*aa.ptTens)};};
185     // Surcharge de l'opérateur -=
186     void operator-= (const Grandeur& c) ;
187     void operator-= (const Grandeur_TenseurHH& aa) {(*ptTens) -= (*aa.ptTens)};};
188     // Surcharge de l'opérateur *=
189     void operator*= (double val) {*ptTens *= val};};
190     // Surcharge de l'opérateur /= : division par un scalaire
191     void operator/= (double val) {*ptTens /= val};};
192     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
193     // si pas possible, ramène 0
194     double Grandeur_NumOrdre(int num) const ;
195     int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
196     void Grandeur_brut(ostream & sort,int nbcars) const ;
197     // ramène le type de grandeur associée de la grandeur de base, car
198     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
199     EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURHH};};
200     // ramène le type de la structure: tableau, liste ext..
201     EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE};};
202     // ramène le type de la grandeur particulière
203     EnuTypeQuelParticulier Type_enumGrandeurParticuliere() const{return PARTICULIER_TENSEURHH};};
204     // change de repère de la grandeur
205     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
206     // lecture
207     istream & Lecture_grandeur(istream & ent) { ent >> *this;return ent};};
208     // ecriture
209     virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << *this; return sort};};
210     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
211     void InitParDefaut() {ptTens->Inita(0.);};
212     // méthode spécifique d'accès à la grandeur
213     // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
214     TenseurHH* ConteneurTenseur() const {return ptTens};};
215     // la référence est un peu plus sur (il est toujours possible de faire un delete sur un
216     // pointeur de ref
217     // mais c'est plus difficile de le faire en se trompant ! enfin j'espère)
218     TenseurHH& RefConteneurTenseur() const {return *ptTens};};
219     protected:
220     TenseurHH* ptTens;
221     };
222     @} // end of group
223     @addtogroup Les_grandeurs_particulieres
224     @
225     @
226     @
227     //-----
228     /// grandeur TenseurBB: TypeQuelconque::Grandeur::Grandeur_TenseurBB|
229     //-----
230     class Grandeur_TenseurBB : public TypeQuelconque::Grandeur
231     { friend class Tab_Grandeur_TenseurBB;
232     // surcharge de l'opérateur de lecture
233     friend istream & operator >> (istream & ent, Grandeur_TenseurBB & a);
234     // surcharge de l'opérateur d'écriture
235     friend ostream & operator << (ostream & sort , const Grandeur_TenseurBB & a);
236     public:
237     // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
238     // ptTens défini, donc -> erreur
239     Grandeur_TenseurBB();
240     // constructeur légal

```

```

241     Grandeur_TenseurBB(const TenseurBB& tens): ptTens(NevetzTenseurBB(tens)) {};
242     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
243     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
244     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
245     Grandeur_TenseurBB(istream & ent);
246     // constructeur de copie
247     Grandeur_TenseurBB(const Grandeur_TenseurBB& a): ptTens(NevetzTenseurBB(*a.ptTens)) {};
248     // destructeur
249     ~Grandeur_TenseurBB();
250     // ramène une grandeur identique, créée
251     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Grandeur_TenseurBB(*ptTens)};
252     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
253     // le constructeur adoc l'objet correcte et valide
254     void EcriturePourLectureAvecCreation(ostream & sort);
255     Grandeur & operator = (const Grandeur & a); // surcharge d'affectation
256     Grandeur & operator = (const Grandeur_TenseurBB & a); // surcharge d'affectation
257     // ** opérations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
258     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
259     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
260     void Affectation_numerique(const TypeQuelconque::Grandeur & a);
261     // Surcharge de l'opérateur +=
262     void operator+= (const Grandeur & c);
263     void operator+= (const Grandeur_TenseurBB& aa) {(*ptTens) += (*aa.ptTens)};
264     // Surcharge de l'opérateur -=
265     void operator-= (const Grandeur & c);
266     void operator-= (const Grandeur_TenseurBB& aa) {(*ptTens) -= (*aa.ptTens)};
267     // Surcharge de l'opérateur *=
268     void operator*= (double val) {*ptTens *= val};
269     // Surcharge de l'opérateur /= : division par un scalaire
270     void operator/= (double val) {*ptTens /= val};
271     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
272     // si pas possible, ramène 0
273     double GrandeurNumOrdre(int num) const;
274     int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
275     void Grandeur_brut(ostream & sort,int nbcarr) const;
276     // ramène le type de grandeur associée de la grandeur de base, car
277     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
278     EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURBB};
279     // ramène le type de la structure: tableau, liste ext..
280     EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE};
281     // ramène le type de la grandeur particulière
282     EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const {return PARTICULIER_TENSEURBB};
283     // change de repère de la grandeur
284     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
285     // lecture
286     istream & Lecture_grandeur(istream & ent) { ent >> *this;return ent};
287     // écriture
288     virtual ostream & Ecriture_grandeur(ostream & sort) const { sort << *this; return sort};
289     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
290     void InitParDefaut() {ptTens->Inita(0.)};
291     // méthode spécifique d'accès à la grandeur
292     // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
293     TenseurBB* ConteneurTenseur() const {return ptTens};
294     // la référence est un peu plus sur (il est toujours possible de faire un delete sur un
pointeur de ref
295     // mais c'est plus difficile de le faire en se trompant ! enfin j'espère)
296     TenseurBB& RefConteneurTenseur() const {return *ptTens};
297
298     protected:
299     TenseurBB* ptTens;
300     };
301 /// @} // end of group
302
303 /// @addtogroup Les_grandeurs_particulieres
304 /// @{
305 ///
306
307 //-----
308 /// grandeur TenseurBH: TypeQuelconque::Grandeur::Grandeur_TenseurBH|
309 //-----
310     class Grandeur_TenseurBH : public TypeQuelconque::Grandeur
311     { friend class Tab_Grandeur_TenseurBH;
312     // surcharge de l'opérateur de lecture
313     friend istream & operator >> (istream & ent, Grandeur_TenseurBH & a);
314     // surcharge de l'opérateur d'écriture
315     friend ostream & operator << (ostream & sort, const Grandeur_TenseurBH & a);
316     public:
317     // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
318     // ptTens défini, donc -> erreur
319     Grandeur_TenseurBH();
320     // constructeur légal
321     Grandeur_TenseurBH(const TenseurBH& tens): ptTens(NevetzTenseurBH(tens)) {};
322     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
323     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
324     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
325     Grandeur_TenseurBH(istream & ent);

```



```

326 // constructeur de copie
327 Grandeur_TenseurBH(const Grandeur_TenseurBH& a): ptTens(NevetzTenseurBH(*a.ptTens)) {};
328 // destructeur
329 ~Grandeur_TenseurBH();
330 // ramène une grandeur identique, créée
331 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Grandeur_TenseurBH(*ptTens);};
332 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
333 // le constructeur adoc l'objet correcte et valide
334 void EcriturePourLectureAvecCreation(ostream & sort);
335 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
336 Grandeur & operator = ( const Grandeur_TenseurBH & a); // surcharge d'affectation
337 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
338 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
339 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
340 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
341 // Surcharge de l'opérateur +=
342 void operator+= (const Grandeur& c) ;
343 void operator+= (const Grandeur_TenseurBH& aa) {(*ptTens) += (*aa.ptTens)};};
344 // Surcharge de l'opérateur -=
345 void operator-= (const Grandeur& c) ;
346 void operator-= (const Grandeur_TenseurBH& aa) {(*ptTens) -= (*aa.ptTens)};};
347 // Surcharge de l'opérateur *=
348 void operator*= (double val) {*ptTens *= val};};
349 // Surcharge de l'opérateur /= : division par un scalaire
350 void operator/= (double val) {*ptTens /= val};};
351 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
352 // si pas possible, ramène 0
353 double GrandeurNumOrdre(int num) const ;
354 int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
355 void Grandeur_brut(ostream & sort,int nbcarr) const ;
356 // ramène le type de grandeur associée de la grandeur de base, car
357 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
358 EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURBH};};
359 // ramène le type de la structure: tableau, liste ext..
360 EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE};};
361 // ramène le type de la grandeur particulière
362 EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return PARTICULIER_TENSEURBH};};
363 // change de repère de la grandeur
364 void Change_repere(const Mat_pleine& beta, const Mat_pleine& gamma);
365 // lecture
366 istream & Lecture_grandeur(istream & ent) { ent >> *this;return ent;};};
367 // ecriture
368 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << *this; return sort;};};
369 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
370 void InitParDefaut() {ptTens->Inita(0.);};};
371 // méthode spécifique d'accès à la grandeur
372 // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
373 // le pb avec le pointeur c'est que l'on peut faire un delete en erreur
374 TenseurBH* ConteneurTenseur() const {return ptTens};};
375 // la référence est un peu plus sur (il est toujours possible de faire un delete sur un
pointeur de ref
376 // mais c'est plus difficile de le faire en se trompant ! enfin j'espère)
377 TenseurBH& RefConteneurTenseur() const {return *ptTens};};
378
379 protected:
380 TenseurBH* ptTens;
381 };
382 /// @} // end of group
383
384 /// @addtogroup Les_grandeurs_particulieres
385 /// @{
386 ///
387
388 //-----
389 /// grandeur TenseurHB: TypeQuelconque::Grandeur::Grandeur_TenseurHB|
390 //-----
391 class Grandeur_TenseurHB : public TypeQuelconque::Grandeur
392 { friend class Tab_Grandeur_TenseurHB;
393 // surcharge de l'opérateur de lecture
394 friend istream & operator » (istream & ent, Grandeur_TenseurHB & a);
395 // surcharge de l'opérateur d'écriture
396 friend ostream & operator « (ostream & sort , const Grandeur_TenseurHB & a);
397 public:
398 // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
399 // ptTens défini, donc -> erreur
400 Grandeur_TenseurHB();
401 // constructeur légal
402 Grandeur_TenseurHB(const TenseurHB& tens): ptTens(NevetzTenseurHB(tens)) {};
403 // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
404 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
405 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
406 Grandeur_TenseurHB(istream & ent);
407 // constructeur de copie
408 Grandeur_TenseurHB(const Grandeur_TenseurHB& a): ptTens(NevetzTenseurHB(*a.ptTens)) {};
409 // destructeur
410 ~Grandeur_TenseurHB();

```

```

411 // ramène une grandeur identique, créée
412 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Grandeur_TenseurHB(*ptTens);};
413 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
414 // le constructeur adoc l'objet correcte et valide
415 void EcriturePourLectureAvecCreation(ostream & sort);
416 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
417 Grandeur & operator = ( const Grandeur_TenseurHB & a); // surcharge d'affectation
418 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
419 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
420 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
421 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
422 // Surcharge de l'operateur +=
423 void operator+= (const Grandeur& c) ;
424 void operator+= (const Grandeur_TenseurHB& aa) {(*ptTens) += *(aa.ptTens)};
425 // Surcharge de l'operateur -=
426 void operator-= (const Grandeur& c) ;
427 void operator-= (const Grandeur_TenseurHB& aa) {(*ptTens) -= *(aa.ptTens)};
428 // Surcharge de l'operateur *=
429 void operator*= (double val) {*ptTens *= val};
430 // Surcharge de l'operateur /= : division par un scalaire
431 void operator/= (double val) {*ptTens /= val};
432 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
433 // si pas possible, ramène 0
434 double GrandeurNumOrdre(int num) const ;
435 int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
436 void Grandeur_brut(ostream & sort,int nbcarr) const ;
437 // ramène le type de grandeur associée de la grandeur de base, car
438 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
439 EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURHB};
440 // ramène le type de la structure: tableau, liste ext..
441 EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE};
442 // ramène le type de la grandeur particulière
443 EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return PARTICULIER_TENSEURHB};
444 // change de repère de la grandeur
445 void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
446 // lecture
447 istream & Lecture_grandeur(istream & ent) { ent » *this;return ent};
448 // ecriture
449 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort « *this; return sort};
450 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
451 void InitParDefaut () {ptTens->Inita(0.);};
452 // méthode spécifique d'accès à la grandeur
453 // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
454 // le pb avec le pointeur c'est que l'on peut faire un delete en erreur
455 TenseurHB* ConteneurTenseur() const {return ptTens};
456 // la référence est un peu plus sur (il est toujours possible de faire un delete sur un
pointeur de ref
457 // mais c'est plus difficile de le faire en se trompant ! enfin j'espère)
458 TenseurHB& RefConteneurTenseur() const {return *ptTens};
459
460 protected:
461 TenseurHB* ptTens;
462 };
463 /// @} // end of group
464
465 /// @addtogroup Les_grandeurs_particulieres
466 /// @{
467 ///
468
469 //-----
470 /// grandeur un tableau de TenseurHH: TypeQuelconque::Grandeur::Tab_Grandeur_TenseurHH|
471 /// tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre
472 //-----
473 class Tab_Grandeur_TenseurHH : public TypeQuelconque::Grandeur
474 { // surcharge de l'operator de lecture
475 friend istream & operator » (istream & ent, Tab_Grandeur_TenseurHH & a);
476 // surcharge de l'operator d'ecriture
477 friend ostream & operator « (ostream & sort , const Tab_Grandeur_TenseurHH & a);
478 public:
479 // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
480 // ptTens défini, donc -> erreur
481 Tab_Grandeur_TenseurHH();
482 // constructeur légal
483 // nb indique le nombre de tenseur
484 Tab_Grandeur_TenseurHH( TenseurHH& tens,int nb);
485 // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
486 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
487 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
488 Tab_Grandeur_TenseurHH(istream & ent);
489 // constructeur de copie
490 Tab_Grandeur_TenseurHH(const Tab_Grandeur_TenseurHH& a) ;
491 // destructeur
492 ~Tab_Grandeur_TenseurHH();
493 // ramène une grandeur identique, créée
494 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_TenseurHH(*this)};};

```

```

495 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
496 // le constructeur adoc l'objet correcte et valide
497 void EcriturePourLectureAvecCreation(ostream & sort);
498 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
499 Grandeur & operator = ( const Tab_Grandeur_TenseurHH & a); // surcharge d'affectation
500 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
501 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
502 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
503 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
504 // Surcharge de l'opérateur +=
505 void operator+=( const Grandeur& c) ;
506 // Surcharge de l'opérateur -=
507 void operator-=( const Grandeur& c) ;
508 // Surcharge de l'opérateur *=
509 void operator*=( double val) ;
510 // Surcharge de l'opérateur /= : division par un scalaire
511 void operator/=( double val) ;
512 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
513 // si pas possible, ramène 0
514 double GrandeurNumOrdre(int num) const ;
515 int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
516 void Grandeur_brut(ostream & sort,int nbcars) const;
517 // ramène le type de grandeur associée de la grandeur de base, car
518 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
519 EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURHH;};
520 // ramène le type de la structure: tableau, liste ext..
521 EnumTypeNiveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
522 // ramène le type de la grandeur particulière
523 EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_TENSEURHH;};
524 // acces au tenseur de numéro i en lecture écriture
525 TenseurHH& operator () ( int i)const {return *(tabTens(i)->ptTens);};
526 // ramène la grandeur_Tenseur associée
527 const Grandeur_TenseurHH& Grandeur_TenseurHH_associee(int i) const {return *(tabTens(i));};
528 // ramène la dimension du tableau de tenseur
529 int Taille() const {return tabTens.Taille();};
530 // changement de taille -> n :
531 // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
532 // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
533 void Change_taille(int n);
534 // change de repère de la grandeur
535 void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
536 // lecture
537 istream & Lecture_grandeur(istream & ent) { ent >> *this;return ent;};
538 // ecriture
539 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << *this; return sort;};
540 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
541 void InitParDefaut();
542
543 protected:
544 // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
545 // utiliser le constructeur par défaut de Grandeur_TenseurHH, d'où la nécessité d'une
construction
546 // en deux temps !!
547 Tableau <Grandeur_TenseurHH* > tabTens;
548 };
549 /// @} // end of group
550
551 /// @addtogroup Les_grandeurs_particulieres
552 /// @{
553 ///
554
555 //-----
556 /// grandeur un tableau de TenseurBH: TypeQuelconque::Grandeur::Tab_Grandeur_TenseurBH|
557 /// tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre
558 //-----
559 class Tab_Grandeur_TenseurBH : public TypeQuelconque::Grandeur
560 { // surcharge de l'opérateur de lecture
561 friend istream & operator » (istream & ent, Tab_Grandeur_TenseurBH & a);
562 // surcharge de l'opérateur d'écriture
563 friend ostream & operator « (ostream & sort , const Tab_Grandeur_TenseurBH & a);
564 public:
565 // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
566 // ptTens défini, donc -> erreur
567 Tab_Grandeur_TenseurBH();
568 // constructeur légal
569 // nb indique le nombre de tenseur
570 Tab_Grandeur_TenseurBH( TenseurBH& tens,int nb);
571 // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
572 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
573 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
574 Tab_Grandeur_TenseurBH(istream & ent);
575 // constructeur de copie
576 Tab_Grandeur_TenseurBH(const Tab_Grandeur_TenseurBH& a) ;
577 // destructeur
578 ~Tab_Grandeur_TenseurBH();

```

```

579         // ramène une grandeur identique, créée
580         TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_TenseurBH(*this);};
581         // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
582         // le constructeur adoc l'objet correcte et valide
583         void EcriturePourLectureAvecCreation(ostream & sort);
584         Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
585         Grandeur & operator = ( const Tab_Grandeur_TenseurBH & a); // surcharge d'affectation
586         // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
587         // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
588         // affectation uniquement des données numériques (pas les pointeurs, pas les string)
589         void Affectation_numerique( const TypeQuelconque::Grandeur & a);
590         // Surcharge de l'opérateur +=
591         void operator+= (const Grandeur& c) ;
592         // Surcharge de l'opérateur -=
593         void operator-= (const Grandeur& c) ;
594         // Surcharge de l'opérateur *=
595         void operator*= (double val) ;
596         // Surcharge de l'opérateur /= : division par un scalaire
597         void operator/= (double val) ;
598         // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
599         // si pas possible, ramène 0
600         double GrandeurNumOrdre(int num) const ;
601         int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
602         void Grandeur_brut(ostream & sort,int nbcarr) const;
603         // ramène le type de grandeur associée de la grandeur de base, car
604         // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
605         EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURBH;};
606         // ramène le type de la structure: tableau, liste ext..
607         EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
608         // ramène le type de la grandeur particulière
609         EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_TENSEURBH;};
610         // acces au tenseur de numéro i en lecture écriture
611         TenseurBH& operator () ( int i)const {return *(tabTens(i)->ptTens);};
612         // ramène la grandeur_Tenseur associée
613         const Grandeur_TenseurBH& Grandeur_TenseurBH_associee(int i) const {return *(tabTens(i));};
614         // ramène la dimension du tableau de tenseur
615         int Taille() const {return tabTens.Taille();};
616         // changement de taille -> n :
617         // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
618         // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
619         void Change_taille(int n);
620         // change de repère de la grandeur
621         void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
622         // lecture
623         istream & Lecture_grandeur(istream & ent) { ent >> *this;return ent;};
624         // ecriture
625         virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << *this; return sort;};
626         // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
627         void InitParDefaut();
628
629         protected:
630         // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
631         // utiliser le constructeur par défaut de Grandeur_TenseurBH, d'où la nécessité d'une
construction
632         // en deux temps !!
633         Tableau <Grandeur_TenseurBH* > tabTens;
634     };
635 /// @} // end of group
636
637 /// @addtogroup Les_grandeurs_particulieres
638 /// @{
639 ///
640
641 //-----
642 /// grandeur un tableau de TenseurBB: TypeQuelconque::Grandeur::Tab_Grandeur_TenseurBB|
643 /// tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre
644 //-----
645     class Tab_Grandeur_TenseurBB : public TypeQuelconque::Grandeur
646     { // surcharge de l'opérateur de lecture
647         friend istream & operator » (istream & ent, Tab_Grandeur_TenseurBB & a);
648         // surcharge de l'opérateur d'écriture
649         friend ostream & operator « (ostream & sort , const Tab_Grandeur_TenseurBB & a);
650     public:
651         // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
652         // ptTens défini, donc -> erreur
653         Tab_Grandeur_TenseurBB();
654         // constructeur légal
655         // nb indique le nombre de tenseur
656         Tab_Grandeur_TenseurBB( TenseurBB& tens,int nb);
657         // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
658         // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
659         // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
660         Tab_Grandeur_TenseurBB(istream & ent);
661         // constructeur de copie

```

```

662     Tab_Grandeur_TenseurBB(const Tab_Grandeur_TenseurBB& a) ;
663     // destructeur
664     ~Tab_Grandeur_TenseurBB();
665     // ramène une grandeur identique, créée
666     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_TenseurBB(*this);};
667     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
668     // le constructeur adoc l'objet correcte et valide
669     void EcriturePourLectureAvecCreation(ostream & sort);
670     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
671     Grandeur & operator = ( const Tab_Grandeur_TenseurBB & a); // surcharge d'affectation
672     // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
673     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
674     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
675     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
676     // Surcharge de l'opérateur +=
677     void operator+= (const Grandeur& c) ;
678     // Surcharge de l'opérateur -=
679     void operator-= (const Grandeur& c) ;
680     // Surcharge de l'opérateur *=
681     void operator*= (double val) ;
682     // Surcharge de l'opérateur /= : division par un scalaire
683     void operator/= (double val) ;
684     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
685     // si pas possible, ramène 0
686     double GrandeurNumOrdre(int num) const ;
687     int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
688     void Grandeur_brut(ostream & sort,int nbcarr) const;
689     // ramène le type de grandeur associée de la grandeur de base, car
690     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
691     EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURBB;};
692     // ramène le type de la structure: tableau, liste ext..
693     EnumTypeNiveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
694     // ramène le type de la grandeur particulière
695     EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_TENSEURBB;};
696     // acces au tenseur de numéro i en lecture écriture
697     TenseurBB& operator () ( int i)const {return *(tabTens(i)->ptTens);};
698     // ramène la grandeur_Tenseur associée
699     const Grandeur_TenseurBB& Grandeur_TenseurBB_associee(int i) const {return *(tabTens(i));};
700     // ramène la dimension du tableau de tenseur
701     int Taille() const {return tabTens.Taille();};
702     // changement de taille -> n :
703     // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
704     // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
705     void Change_taille(int n);
706     // change de repère de la grandeur
707     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
708     // lecture
709     istream & Lecture_grandeur(istream & ent) { ent >> *this;return ent;};
710     // ecriture
711     virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << *this; return sort;};
712     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
713     void InitParDefaut();
714
715     protected:
716     // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
717     // utiliser le constructeur par défaut de Grandeur_TenseurBB, d'où la nécessité d'une
construction
718     // en deux temps !!
719     Tableau <Grandeur_TenseurBB* > tabTens;
720     };
721 /// @} // end of group
722
723 /// @addtogroup Les_grandeurs_particulieres
724 /// @{
725 ///
726
727 //-----
728 /// grandeur un tableau de TenseurHB: TypeQuelconque::Grandeur::Tab_Grandeur_TenseurHB
729 /// tous les tenseurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre
730 //-----
731     class Tab_Grandeur_TenseurHB : public TypeQuelconque::Grandeur
732     { // surcharge de l'opérateur de lecture
733     friend istream & operator » (istream & ent, Tab_Grandeur_TenseurHB & a);
734     // surcharge de l'opérateur d'écriture
735     friend ostream & operator « (ostream & sort , const Tab_Grandeur_TenseurHB & a);
736     public:
737     // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
738     // ptTens défini, donc -> erreur
739     Tab_Grandeur_TenseurHB();
740     // constructeur légal
741     // nb indique le nombre de tenseur
742     Tab_Grandeur_TenseurHB( TenseurHB& tens,int nb);
743     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
744     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction

```

```

745 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
746 Tab_Grandeur_TenseurHB(istream & ent);
747 // constructeur de copie
748 Tab_Grandeur_TenseurHB(const Tab_Grandeur_TenseurHB& a) ;
749 // destructeur
750 ~Tab_Grandeur_TenseurHB();
751 // ramène une grandeur identique, créée
752 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_TenseurHB(*this);};
753 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
754 // le constructeur adoc l'objet correcte et valide
755 void EcriturePourLectureAvecCreation(ostream & sort);
756 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
757 Grandeur & operator = ( const Tab_Grandeur_TenseurHB & a); // surcharge d'affectation
758 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
759 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
760 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
761 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
762 // Surcharge de l'opérateur +=
763 void operator+= (const Grandeur& c) ;
764 // Surcharge de l'opérateur -=
765 void operator-= (const Grandeur& c) ;
766 // Surcharge de l'opérateur *=
767 void operator*= (double val) ;
768 // Surcharge de l'opérateur /= : division par un scalaire
769 void operator/= (double val) ;
770 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
771 // si pas possible, ramène 0
772 double GrandeurNumOrdre(int num) const ;
773 int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
774 void Grandeur_brut(ostream & sort,int nbcarr) const;
775 // ramène le type de grandeur associée de la grandeur de base, car
776 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
777 EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURHB;};
778 // ramène le type de la structure: tableau, liste ext..
779 EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
780 // ramène le type de la grandeur particulière
781 EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_TENSEURHB;};
782 // acces au tenseur de numéro i en lecture écriture
783 TenseurHB& operator () ( int i)const {return *(tabTens(i)->ptTens);};
784 // ramène la grandeur_Tenseur associée
785 const Grandeur_TenseurHB& Grandeur_TenseurHB_associee(int i) const {return *(tabTens(i));};
786 // ramène la dimension du tableau de tenseur
787 int Taille() const {return tabTens.Taille();};
788 // changement de taille -> n :
789 // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
790 // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
791 void Change_taille(int n);
792 // change de repère de la grandeur
793 void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
794 // lecture
795 istream & Lecture_grandeur(istream & ent) { ent » *this;return ent;};
796 // ecriture
797 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort « *this; return sort;};
798 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
799 void InitParDefaut();
800
801 protected:
802 // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
803 // utiliser le constructeur par défaut de Grandeur_TenseurHB, d'où la nécessité d'une
construction
804 // en deux temps !!
805 Tableau <Grandeur_TenseurHB* > tabTens;
806 };
807 /// @} // end of group
808
809 /// @addtogroup Les_grandeurs_particulieres
810 /// @{
811 ///
812
813 //-----
814 /// grandeur un tableau à 2 dim de TenseurHH: TypeQuelconque::Grandeur::Tab2_Grandeur_TenseurHH
815 //-----
816 class Tab2_Grandeur_TenseurHH : public TypeQuelconque::Grandeur
817 { // surcharge de l'opérateur de lecture
818 friend istream & operator » (istream & ent, Tab2_Grandeur_TenseurHH & a);
819 // surcharge de l'opérateur d'écriture
820 friend ostream & operator « (ostream & sort , const Tab2_Grandeur_TenseurHH & a);
821 public:
822 // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
823 // ptTens défini, donc -> erreur
824 Tab2_Grandeur_TenseurHH();
825 // constructeur légal
826 // nb1,nb2 indique la dimension du tableau(nb1,nb2) de tenseur
827 Tab2_Grandeur_TenseurHH( TenseurHH& tens,int nb1, int nb2);

```

```

828 // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
829 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
830 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
831 Tab2_Grandeur_TenseurHH(istream & ent);
832 // constructeur de copie
833 Tab2_Grandeur_TenseurHH(const Tab2_Grandeur_TenseurHH& a) ;
834 // destructeur
835 ~Tab2_Grandeur_TenseurHH();
836 // ramène une grandeur identique, créée
837 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab2_Grandeur_TenseurHH(*this);};
838 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
839 // le constructeur adoc l'objet correcte et valide
840 void EcriturePourLectureAvecCreation(ostream & sort);
841 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
842 Grandeur & operator = ( const Tab2_Grandeur_TenseurHH & a); // surcharge d'affectation
843 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
844 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
845 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
846 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
847 // Surcharge de l'opérateur +=
848 void operator+= (const Grandeur& c);
849 // Surcharge de l'opérateur -=
850 void operator-= (const Grandeur& c) ;
851 // Surcharge de l'opérateur *=
852 void operator*= (double val);
853 // Surcharge de l'opérateur /= : division par un scalaire
854 void operator/= (double val) ;
855 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
856 // si pas possible, ramène 0
857 double GrandeurNumOrdre(int num) const ;
858 int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
859 void Grandeur_brut(ostream & sort,int nbcars) const;
860 // ramène le type de grandeur associée de la grandeur de base, car
861 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
862 EnumTypeGrandeur Type_grandeurAssocie() const {return TENSEURHH};};
863 // ramène le type de la structure: tableau, liste ext..
864 EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU2_T};};
865 // ramène le type de la grandeur particulière
866 EnuTypeQuelParticulier Type_enuGrandeurParticuliere() const{return
PARTICULIER_TABLEAU2_TENSEURHH};};
867 // acces au tenseur de numéro i,j en lecture écriture
868 TenseurHH& operator () ( int i,int j) const {return *(tabTens(i,j)->ptTens)};};
869 // ramène la grandeur_Tenseur associée
870 const Grandeur_TenseurHH& Grandeur_TenseurHH_associee(int i, int j) const {return
*(tabTens(i,j))};};
871 // changement de taille -> n :
872 // si ni < taille_actuelle --> les taille_actuelle - ni sont supprimé
873 // si ni > taille_actuelle --> le dernier élément existant est dupliqué ni-taille_actuelle
fois,
874 void Change_taille(int n1, int n2);
875 // change de repère de la grandeur
876 void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
877 // lecture
878 istream & Lecture_grandeur(istream & ent) { ent » *this;return ent;};
879 // écriture
880 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort « *this; return sort;};
881 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
882 void InitParDefaut();
883
884 protected:
885 // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
886 // utiliser le constructeur par défaut de Grandeur_TenseurHH, d'où la nécessité d'une
construction
887 // en deux temps !!
888 Tableau2 <Grandeur_TenseurHH* > tabTens;
889 };
890 /// @} // end of group
891
892 /// @addtogroup Les_grandeurs_particulieres
893 /// @{
894 ///
895
896 //-----
897 /// grandeur scalaire réel: TypeQuelconque::Grandeur::Grandeur_scalaire_double
898 //-----
899 class Grandeur_scalaire_entier : public TypeQuelconque::Grandeur
900 { friend class Tab_Grandeur_scalaire_entier;
901 // surcharge de l'opérateur de lecture
902 friend istream & operator » (istream & ent, Grandeur_scalaire_entier & a);
903 // surcharge de l'opérateur d'écriture
904 friend ostream & operator « (ostream & sort , const Grandeur_scalaire_entier & a);
905 public:
906 // constructeur par défaut:
907 Grandeur_scalaire_entier();
908 // constructeur légal

```



```

909     Grandeur_scalaire_entier(const double& va): val(va) {};
910     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
911     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
912     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
913     Grandeur_scalaire_entier(istream & ent);
914     // constructeur de copie
915     Grandeur_scalaire_entier(const Grandeur_scalaire_entier& a): val(a.val) {};
916     // destructeur
917     ~Grandeur_scalaire_entier(){};
918     // ramène une grandeur identique, créée
919     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Grandeur_scalaire_entier(val)};};
920     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
921     // le constructeur adoc l'objet correcte et valide
922     void EcriturePourLectureAvecCreation(ostream & sort);
923     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
924     Grandeur & operator = ( const Grandeur_scalaire_entier & a)
925         {val=a.val;return *this}; // surcharge d'affectation
926     Grandeur & operator = ( const double& b) {val=b;return *this}; // surcharge d'affectation
927     // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
928     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
929     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
930     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
931     // Surcharge de l'opérateur +=
932     void operator+= (const Grandeur& c);
933     // Surcharge de l'opérateur -=
934     void operator-= (const Grandeur& c);
935     // Surcharge de l'opérateur *=
936     void operator*=(double x) {val*=x};
937     // Surcharge de l'opérateur /= : division par un scalaire
938     void operator/=(double x) {val/=x};
939     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
940     // si pas possible, ramène 0
941     double GrandeurNumOrdre(int num) const {if(num==1) return val; else return 0.};};
942     int NbMaxiNumeroOrdre() const {return 1}; // récup du nb maxi de numéros d'ordres
943     void Grandeur_brut(ostream & sort,int nbcars) const
944         {sort << " « setw(nbcars) < setprecision(nbcars) << val << " " "};};
945     // ramène le type de grandeur associée de la grandeur de base, car
946     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de base
947     EnumTypeGrandeur Type_grandeurAssocie() const {return SCALAIRE};};
948     // ramène le type de la structure: tableau, liste ext..
949     EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE};};
950     // ramène le type de la grandeur particulière
951     EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_SCALAIRE_ENTIER};};
952     // change de repère de la grandeur
953     void Change_repere(const Mat_pleine& ,const Mat_pleine& ){};
954     // lecture
955     istream & Lecture_grandeur(istream & ent) { ent >> val ;return ent};};
956     // écriture
957     virtual ostream & Ecriture_grandeur(ostream & sort) const { sort << " " << val << " " " ; return
sort};};
958     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
959     void InitParDefaut(){val = 0.};};
960     // méthode spécifique d'accès à la grandeur
961     // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
962     int* ConteneurEntier() {return &val};};
963
964     protected:
965     int val;
966     };
967 /// @} // end of group
968
969 /// @addtogroup Les_grandeurs_particulieres
970 /// @{
971 ///
972
973 //-----
974 /// grandeur tableau de scalaires entière: TypeQuelconque::Grandeur::Tab_Grandeur_scalaire_entiere
975 //-----
976     class Tab_Grandeur_scalaire_entier : public TypeQuelconque::Grandeur
977     { // surcharge de l'opérateur de lecture
978     friend istream & operator >> (istream & ent, Tab_Grandeur_scalaire_entier & a);
979     // surcharge de l'opérateur d'écriture
980     friend ostream & operator << (ostream & sort , const Tab_Grandeur_scalaire_entier & a);
981     public:
982     // constructeur par défaut:
983     Tab_Grandeur_scalaire_entier();
984     // constructeur légal
985     Tab_Grandeur_scalaire_entier(const Tableau <int>& tab_va);
986     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
987     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
988     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
989     Tab_Grandeur_scalaire_entier(istream & ent);
990     // constructeur de copie
991     Tab_Grandeur_scalaire_entier(const Tab_Grandeur_scalaire_entier& a): tab_val(a.tab_val) {};
```



```

992 // destructeur
993 ~Tab_Grandeur_scalaire_entier(){};
994 // ramène une grandeur identique, créée
995 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_scalaire_entier(*this)};};
996 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
997 // le constructeur adoc l'objet correcte et valide
998 void EcriturePourLectureAvecCreation(ostream & sort);
999 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1000 Grandeur & operator = ( const Tableau <int>& b) ; // surcharge d'affectation
1001 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1002 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1003 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1004 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1005 // Surcharge de l'opérateur +=
1006 void operator+= (const Grandeur& c);
1007 // Surcharge de l'opérateur -=
1008 void operator-= (const Grandeur& c);
1009 // Surcharge de l'opérateur *=
1010 void operator*= (double x);
1011 // Surcharge de l'opérateur /= : division par un scalaire
1012 void operator/= (double x);
1013 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1014 // si pas possible, ramène 0
1015 double GrandeurNumOrdre(int num) const;
1016 int NbMaxiNumeroOrdre() const {return tab_val.Taille()}; // récup du nb maxi de numéros
d'ordres
1017 void Grandeur_brut(ostream & sort,int nbcarr) const ;
1018 // ramène le type de grandeur associée de la grandeur de base, car
1019 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1020 EnumTypeGrandeur Type_grandeurAssocie() const {return SCALAIRE};};
1021 // ramène le type de la structure: tableau, liste ext..
1022 EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T};};
1023 // ramène le type de la grandeur particulière
1024 EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_SCALAIRE_DOUBLE};};
1025 // ramène un Grandeur_scalaire_entier associé
1026 Grandeur_scalaire_entier& Grandeur_scalaire_associee(int i) {return tab_val(i)};};
1027 // ramène la dimension du tableau
1028 int Taille() const {return tab_val.Taille()};};
1029 // changement de taille -> n :
1030 // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
1031 // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
1032 void Change_taille(int n);
1033 // change de repère de la grandeur
1034 void Change_repere(const Mat_pleine& ,const Mat_pleine& ){};
1035 // lecture
1036 istream & Lecture_grandeur(istream & ent) { tab_val.Entree(ent);return ent};};
1037 // ecriture
1038 virtual ostream & Ecriture_grandeur(ostream & sort) const { tab_val.Sortir_sansRet(sort);
return sort};};
1039 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1040 void InitParDefaut();
1041 // méthode spécifique d'accès à la grandeur
1042 // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
1043 Tableau <Grandeur_scalaire_entier>& ConteneurTabEntier() {return tab_val};};
1044 // accès au double de numéro i en lecture écriture
1045 int& operator () ( const int i) {return tab_val(i).val};};
1046
1047 protected:
1048 Tableau <Grandeur_scalaire_entier> tab_val;
1049 };
1050 /// @} // end of group
1051
1052 /// @addtogroup Les_grandeurs_particulieres
1053 /// @{
1054 ///
1055
1056 //-----
1057 /// grandeur scalaire réel: TypeQuelconque::Grandeur::Grandeur_scalaire_double
1058 //-----
1059 class Grandeur_scalaire_double : public TypeQuelconque::Grandeur
1060 { friend class Tab_Grandeur_scalaire_double;
1061 // surcharge de l'opérateur de lecture
1062 friend istream & operator » (istream & ent, Grandeur_scalaire_double & a);
1063 // surcharge de l'opérateur d'écriture
1064 friend ostream & operator « (ostream & sort , const Grandeur_scalaire_double & a);
1065 public:
1066 // constructeur par défaut:
1067 Grandeur_scalaire_double();
1068 // constructeur légal
1069 Grandeur_scalaire_double(const double& va): val(va) {};
1070 // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1071 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1072 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale

```

```

1073     Grandeur_scalaire_double(istream & ent);
1074     // constructeur de copie
1075     Grandeur_scalaire_double(const Grandeur_scalaire_double& a): val(a.val) {};
1076     // destructeur
1077     ~Grandeur_scalaire_double(){};
1078     // ramène une grandeur identique, créée
1079     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Grandeur_scalaire_double(val)};};
1080     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1081     // le constructeur adoc l'objet correcte et valide
1082     void EcriturePourLectureAvecCreation(ostream & sort);
1083     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1084     Grandeur & operator = ( const Grandeur_scalaire_double & a)
1085         {val=a.val;return *this}; // surcharge d'affectation
1086     Grandeur & operator = ( const double& b) {val=b;return *this}; // surcharge d'affectation
1087     // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1088     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1089     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1090     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1091     // Surcharge de l'opérateur +=
1092     void operator+= (const Grandeur& c);
1093     // Surcharge de l'opérateur -=
1094     void operator-= (const Grandeur& c);
1095     // Surcharge de l'opérateur *=
1096     void operator*= (double x) {val*=x};
1097     // Surcharge de l'opérateur /= : division par un scalaire
1098     void operator/= (double x) {val/=x};
1099     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1100     // si pas possible, ramène 0
1101     double GrandeurNumOrdre(int num) const {if(num==1) return val; else return 0.};};
1102     int NbMaxiNumeroOrdre() const {return 1}; // récup du nb maxi de numéros d'ordres
1103     void Grandeur_brut(ostream & sort,int nbcar) const
1104         {sort << " " << setw(nbcar) << setprecision(nbcar) << val << " ";};
1105     // ramène le type de grandeur associée de la grandeur de base, car
1106     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1107     EnumTypeGrandeur Type_grandeurAssocie() const {return SCALAIRE};};
1108     // ramène le type de la structure: tableau, liste ext..
1109     EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE};};
1110     // ramène le type de la grandeur particulière
1111     EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_SCALAIRE_DOUBLE};};
1112     // change de repère de la grandeur
1113     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma){};
1114     // lecture
1115     istream & Lecture_grandeur(istream & ent) { ent >> val ;return ent};};
1116     // ecriture
1117     virtual ostream & Ecriture_grandeur(ostream & sort) const { sort << " " << val << " "; return
sort};};
1118     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1119     void InitParDefaut(){val = 0.};};
1120     // méthode spécifique d'accès à la grandeur
1121     // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
1122     double* ConteneurDouble() {return &val};};
1123     // idem en const
1124     double ContDouble() const {return val};};
1125
1126     protected:
1127     double val;
1128     };
1129 /// @} // end of group
1130
1131 /// @addtogroup Les_grandeurs_particulieres
1132 /// @{
1133 ///
1134 ///
1135 //-----
1136 /// grandeur tableau de scalaires réel: TypeQuelconque::Grandeur::Tab_Grandeur_scalaire_double
1137 //-----
1138     class Tab_Grandeur_scalaire_double : public TypeQuelconque::Grandeur
1139     { // surcharge de l'opérateur de lecture
1140     friend istream & operator >> (istream & ent, Tab_Grandeur_scalaire_double & a);
1141     // surcharge de l'opérateur d'écriture
1142     friend ostream & operator << (ostream & sort, const Tab_Grandeur_scalaire_double & a);
1143     public:
1144     // constructeur par défaut:
1145     Tab_Grandeur_scalaire_double();
1146     // constructeur légal
1147     Tab_Grandeur_scalaire_double(const Tableau <double>& tab_va);
1148     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1149     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1150     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1151     Tab_Grandeur_scalaire_double(istream & ent);
1152     // constructeur de copie
1153     Tab_Grandeur_scalaire_double(const Tab_Grandeur_scalaire_double& a): tab_val(a.tab_val) {};
1154     // destructeur

```

```

1155     ~Tab_Grandeur_scalaire_double(){};
1156     // ramène une grandeur identique, créée
1157     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_scalaire_double(*this)};};
1158     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1159     // le constructeur adoc l'objet correcte et valide
1160     void EcriturePourLectureAvecCreation(ostream & sort);
1161     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1162     Grandeur & operator = ( const Tableau <double>& b) ; // surcharge d'affectation
1163     // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1164     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1165     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1166     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1167     // Surcharge de l'opérateur +=
1168     void operator+= (const Grandeur& c);
1169     // Surcharge de l'opérateur -=
1170     void operator-= (const Grandeur& c);
1171     // Surcharge de l'opérateur *=
1172     void operator*= (double x);
1173     // Surcharge de l'opérateur /= : division par un scalaire
1174     void operator/= (double x);
1175     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1176     // si pas possible, ramène 0
1177     double GrandeurNumOrdre(int num) const;
1178     int NbMaxiNumeroOrdre() const {return tab_val.Taille()}; // récup du nb maxi de numéros
d'ordres
1179     void Grandeur_brut(ostream & sort,int nbcar) const ;
1180     // ramène le type de grandeur associée de la grandeur de base, car
1181     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1182     EnumTypeGrandeur Type_grandeurAssocie() const {return SCALAIRE};};
1183     // ramène le type de la structure: tableau, liste ext..
1184     EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T};};
1185     // ramène le type de la grandeur particulière
1186     EnuTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_SCALAIRE_DOUBLE};};
1187     // ramène un Grandeur_scalaire_double associé
1188     Grandeur_scalaire_double& Grandeur_scalaire_associee(int i) {return tab_val(i)};};
1189     // ramène la dimension du tableau
1190     int Taille() const {return tab_val.Taille()};};
1191     // changement de taille -> n :
1192     // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
1193     // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
1194     void Change_taille(int n);
1195     // change de repère de la grandeur
1196     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma){};
1197     // lecture
1198     istream & Lecture_grandeur(istream & ent) { tab_val.Entree(ent);return ent};};
1199     // ecriture
1200     virtual ostream & Ecriture_grandeur(ostream & sort ) const { tab_val.Sortir_sansRet(sort);
return sort};};
1201     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1202     void InitParDefaut();
1203     // méthode spécifique d'accès à la grandeur
1204     // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
1205     Tableau <Grandeur_scalaire_double>& ConteneurTabDouble() {return tab_val};};
1206     // accès au double de numéro i en lecture écriture
1207     double& operator () ( const int i) {return tab_val(i).val};};
1208
1209     protected:
1210     Tableau <Grandeur_scalaire_double> tab_val;
1211     };
1212 /// @} // end of group
1213
1214 /// @addtogroup Les_grandeurs_particulieres
1215 /// @{
1216 ///
1217
1218 //-----
1219 /// grandeur vecteur: TypeQuelconque::Grandeur::Grandeur_vecteur
1220 //-----
1221     class Grandeur_Vecteur : public TypeQuelconque::Grandeur
1222     { friend class Tab_Grandeur_Vecteur;
1223     // surcharge de l'opérateur de lecture
1224     friend istream & operator » (istream & ent, Grandeur_Vecteur & a);
1225     // surcharge de l'opérateur d'écriture
1226     friend ostream & operator « (ostream & sort , const Grandeur_Vecteur & a);
1227     public:
1228     // constructeur par défaut:
1229     Grandeur_Vecteur();
1230     // constructeur légal
1231     Grandeur_Vecteur(const Vecteur& v): ve(v) {}; // à partir d'un vecteur
1232     Grandeur_Vecteur(int n): ve(n) {}; // à partir d'une dimension
1233     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1234     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1235     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale

```

```

1236     Grandeur_Vecteur(istream & ent);
1237     // constructeur de copie
1238     Grandeur_Vecteur(const Grandeur_Vecteur& a): ve(a.ve) {};
1239     // destructeur
1240     ~Grandeur_Vecteur(){};
1241     // ramène une grandeur identique, créée
1242     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Grandeur_Vecteur(ve);};
1243     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1244     // le constructeur adoc l'objet correcte et valide
1245     void EcriturePourLectureAvecCreation(ostream & sort);
1246     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1247     Grandeur & operator = ( const Vecteur& b) {ve=b;return *this;}; // surcharge d'affectation
1248     // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1249     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
    même type
1250     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1251     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1252     // Surcharge de l'opérateur +=
1253     void operator+= (const Grandeur& c);
1254     // Surcharge de l'opérateur -=
1255     void operator-= (const Grandeur& c);
1256     // Surcharge de l'opérateur *=
1257     void operator*= (double val) {ve*=val;};
1258     // Surcharge de l'opérateur /= : division par un scalaire
1259     void operator/= (double val) {ve/=val;};
1260     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1261     // si pas possible, ramène 0
1262     double GrandeurNumOrdre(int num) const;
1263     int NbMaxiNumeroOrdre() const {return ve.Taille();}; // recup du nb maxi de numéros d'ordres
1264     void Grandeur_brut(ostream & sort,int nbcar) const
1265     {sort << " " << setprecision(nbcar) << ve; };
1266     // ramène le type de grandeur associée de la grandeur de base, car
1267     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
    base
1268     EnumTypeGrandeur Type_grandeurAssocie() const {return VECTEUR;};
1269     // ramène le type de la structure: tableau, liste ext..
1270     EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE;};
1271     // ramène le type de la grandeur particulière
1272     EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return PARTICULIER_VECTEUR;};
1273     // change de repère de la grandeur
1274     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
1275     // lecture
1276     istream & Lecture_grandeur(istream & ent) { ent >> ve ;return ent;};
1277     // ecriture
1278     virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << " " << ve << " "; return
    sort;};
1279     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1280     void InitParDefaut() {ve.Zero();};
1281     // ----- méthodes spécifiques d'accès à la grandeur -----
1282     // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
1283     Vecteur* ConteneurVecteur() {return &ve;};
1284     // idem en const
1285     const Vecteur& ConteneurVecteur_const() const {return ve;};
1286
1287     protected:
1288     Vecteur ve;
1289     };
1290 /// @} // end of group
1291
1292 /// @addtogroup Les_grandeurs_particulieres
1293 /// @{
1294 ///
1295 ///-----
1296 /// grandeur un tableau de Vecteur: TypeQuelconque::Grandeur::Tab_Grandeur_Vecteur|
1297 /// tous les Vecteurs doivent avoir la même dimension !! pour la routine GrandeurNumOrdre ?? à voir
1298 ///-----
1299 ///
1300     class Tab_Grandeur_Vecteur : public TypeQuelconque::Grandeur
1301     { // surcharge de l'opérateur de lecture
1302     friend istream & operator » (istream & ent, Tab_Grandeur_Vecteur & a);
1303     // surcharge de l'opérateur d'écriture
1304     friend ostream & operator « (ostream & sort , const Tab_Grandeur_Vecteur & a);
1305     public:
1306     // constructeur par défaut: ne doit pas être utilisé
1307     Tab_Grandeur_Vecteur();
1308     // constructeur légal
1309     // nb indique le nombre de Vecteur
1310     Tab_Grandeur_Vecteur( Vecteur & vect,int nb);
1311     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1312     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1313     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1314     Tab_Grandeur_Vecteur(istream & ent);
1315     // constructeur de copie
1316     Tab_Grandeur_Vecteur(const Tab_Grandeur_Vecteur& a) ;
1317     // destructeur
1318     ~Tab_Grandeur_Vecteur();
1319     // ramène une grandeur identique, créée

```

```

1320     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Tab_Grandeur_Vecteur(*this);};
1321     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1322     // le constructeur adoc l'objet correcte et valide
1323     void EcriturePourLectureAvecCreation(ostream & sort);
1324     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1325     Grandeur & operator = ( const Tab_Grandeur_Vecteur & a); // surcharge d'affectation
1326     // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1327     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
    même type
1328     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1329     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1330     // Surcharge de l'opérateur +=
1331     void operator+= (const Grandeur& c) ;
1332     // Surcharge de l'opérateur -=
1333     void operator-= (const Grandeur& c) ;
1334     // Surcharge de l'opérateur *=
1335     void operator*= (double val) ;
1336     // Surcharge de l'opérateur /= : division par un scalaire
1337     void operator/= (double val) ;
1338     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1339     // si pas possible, ramène 0
1340     double GrandeurNumOrdre(int num) const ;
1341     int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
1342     void Grandeur_brut(ostream & sort,int nbcarr) const;
1343     // ramène le type de grandeur associée de la grandeur de base, car
1344     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
    base
1345     EnumTypeGrandeur Type_grandeurAssocie() const {return VECTEUR;};
1346     // ramène le type de la structure: tableau, liste ext..
1347     EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
1348     // ramène le type de la grandeur particulière
1349     EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_VECTEUR;};
1350     // acces au Vecteur de numéro i en lecture écriture
1351     Vecteur& operator () ( int i)const {return (tabVe(i)->ve);};
1352     // ramène la grandeur_Vecteur associée
1353     const Grandeur_Vecteur& Grandeur_Vecteur_associee(int i) const {return *(tabVe(i));};
1354     // ramène la dimension du tableau de Vecteur
1355     int Taille() const {return tabVe.Taille();};
1356     // changement de taille -> n :
1357     // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
1358     // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
1359     void Change_taille(int n);
1360     // change de repère de la grandeur
1361     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
1362     // lecture
1363     istream & Lecture_grandeur(istream & ent) { ent » *this;return ent;};
1364     // ecriture
1365     virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort « *this; return sort;};
1366     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1367     void InitParDefaut();
1368
1369     protected:
1370     // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
1371     // utiliser le constructeur par défaut de Grandeur_Vecteur, d'où la nécessité d'une
    construction
1372     // en deux temps !!
1373     Tableau <Grandeur_Vecteur* > tabVe;
1374     };
1375 /// @} // end of group
1376
1377 /// @addtogroup Les_grandeurs_particulieres
1378 /// @{
1379 ///
1380
1381 //-----
1382 /// grandeur coordonnée: TypeQuelconque::Grandeur::Grandeur_coorдонnee
1383 //-----
1384     class Grandeur_coorдонnee : public TypeQuelconque::Grandeur
1385     { friend class Tab_Grandeur_Coorдонnee;
1386     // surcharge de l'opérateur de lecture
1387     friend istream & operator » (istream & ent, Grandeur_coorдонnee & a);
1388     // surcharge de l'opérateur d'écriture
1389     friend ostream & operator « (ostream & sort , const Grandeur_coorдонnee & a);
1390     public:
1391     // constructeur par défaut:
1392     Grandeur_coorдонnee();
1393     // constructeur légal
1394     Grandeur_coorдонnee(const Coorдонnee& v): co(v) {};
1395     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1396     // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1397     // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1398     Grandeur_coorдонnee(istream & ent);
1399     // constructeur de copie
1400     Grandeur_coorдонnee(const Grandeur_coorдонnee& a): co(a.co) {};
1401     // destructeur
1402     ~Grandeur_coorдонnee(){};

```

```

1403 // ramène une grandeur identique, créée
1404 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Grandeur_cooronnee(co);};
1405 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1406 // le constructeur adoc l'objet correcte et valide
1407 void EcriturePourLectureAvecCreation(ostream & sort);
1408 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1409 Grandeur & operator = ( const Coordonnee& b) {co=b;return *this;}; // surcharge d'affectation
1410 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1411 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1412 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1413 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1414 // Surcharge de l'opérateur +=
1415 void operator+= (const Grandeur& c);
1416 // Surcharge de l'opérateur -=
1417 void operator-= (const Grandeur& c);
1418 // Surcharge de l'opérateur *=
1419 void operator*= (double val) {co*=val;};
1420 // Surcharge de l'opérateur /= : division par un scalaire
1421 void operator/= (double val) {co/=val;};
1422 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1423 // si pas possible, ramène 0
1424 double GrandeurNumOrdre(int num) const;
1425 int NbMaxiNumeroOrdre() const {return co.Dimension();}; // récup du nb maxi de numéros
d'ordres
1426 void Grandeur_brut(ostream & sort,int nbcар) const
1427 {sort << " " << setprecision(nbcар) ; co.Affiche(sort,nbcар) ;};
1428 // ramène le type de grandeur associée de la grandeur de base, car
1429 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1430 EnumTypeGrandeur Type_grandeurAssocie() const {return COORDONNEE;};
1431 // ramène le type de la structure: tableau, liste ext..
1432 EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE;};
1433 // ramène le type de la grandeur particulière
1434 EnuTypeQuelParticulier Type_enumGrandeurParticuliere() const{return PARTICULIER_COORDONNEE;};
1435 // change de repère de la grandeur
1436 void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
1437 // lecture
1438 istream & Lecture_grandeur(istream & ent) { ent >> co ;return ent;};
1439 // ecriture
1440 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << " " << co << " "; return
sort;};
1441 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1442 void InitParDefaut() {co.Zero();};
1443 // méthode spécifique d'accès à la grandeur
1444 // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
1445 Coordonnee* ConteneurCoordonnee() {return &co;};
1446 // idem en const
1447 const Coordonnee& ConteneurCoordonnee_const() const {return co;};
1448
1449 protected:
1450 Coordonnee co;
1451 };
1452 /// @} // end of group
1453
1454 /// @addtogroup Les_grandeurs_particulieres
1455 /// @{
1456 ///
1457
1458 //-----
1459 /// grandeur un tableau de Coordonnee: TypeQuelconque::Grandeur::Tab_Grandeur_CoordonneeHH
1460 /// tous les coordonnees doivent avoir la même dimension !! pour la routine GrandeurNumOrdre
1461 //-----
1462 class Tab_Grandeur_Coordonnee : public TypeQuelconque::Grandeur
1463 { // surcharge de l'opérateur de lecture
1464 friend istream & operator » (istream & ent, Tab_Grandeur_Coordonnee & a);
1465 // surcharge de l'opérateur d'écriture
1466 friend ostream & operator « (ostream & sort , const Tab_Grandeur_Coordonnee & a);
1467 public:
1468 // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
1469 // co défini, donc -> erreur (en fait ce n'est sûr, mais on fait comme pour les tenseurs)
1470 Tab_Grandeur_Coordonnee();
1471 // constructeur légal
1472 // nb indique le nombre de Coordonnee
1473 Tab_Grandeur_Coordonnee( Coordonnee & coor,int nb);
1474 // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1475 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1476 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1477 Tab_Grandeur_Coordonnee(istream & ent);
1478 // constructeur de copie
1479 Tab_Grandeur_Coordonnee(const Tab_Grandeur_Coordonnee& a) ;
1480 // destructeur
1481 ~Tab_Grandeur_Coordonnee();
1482 // ramène une grandeur identique, créée
1483 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_Coordonnee(*this);};
1484 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via

```

```

1485         // le constructeur adoc l'objet correcte et valide
1486         void EcriturePourLectureAvecCreation(ostream & sort);
1487         Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1488         Grandeur & operator = ( const Tab_Grandeur_Coordonnee & a); // surcharge d'affectation
1489         // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1490         // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1491         // affectation uniquement de deux données numériques (pas les pointeurs, pas les string)
1492         void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1493         // Surcharge de l'opérateur +=
1494         void operator+= (const Grandeur& c) ;
1495         // Surcharge de l'opérateur -=
1496         void operator-= (const Grandeur& c) ;
1497         // Surcharge de l'opérateur *=
1498         void operator*= (double val) ;
1499         // Surcharge de l'opérateur /= : division par un scalaire
1500         void operator/= (double val) ;
1501         // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1502         // si pas possible, ramène 0
1503         double GrandeurNumOrdre(int num) const ;
1504         int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
1505         void Grandeur_brut(ostream & sort,int nbcar) const;
1506         // ramène le type de grandeur associée de la grandeur de base, car
1507         // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1508         EnumTypeGrandeur Type_grandeurAssocie() const {return COORDONNEE;};
1509         // ramène le type de la structure: tableau, liste ext..
1510         EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
1511         // ramène le type de la grandeur particulière
1512         EnuTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_COORDONNEE;};
1513         // acces au Coordonnee de numéro i en lecture écriture
1514         Coordonnee& operator () ( int i)const {return (tabCoor(i)->co);};
1515         // ramène la grandeur_Coordonnee associée
1516         const Grandeur_coordonnee& Grandeur_Coordonnee_associee(int i) const {return *(tabCoor(i));};
1517         // ramène la dimension du tableau de Coordonnee
1518         int Taille() const {return tabCoor.Taille();};
1519         // changement de taille -> n :
1520         // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
1521         // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
1522         void Change_taille(int n);
1523         // change de repère de la grandeur
1524         void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
1525         // lecture
1526         istream & Lecture_grandeur(istream & ent) { ent > *this;return ent;};
1527         // ecriture
1528         virtual ostream & Ecriture_grandeur(ostream & sort) const { sort < *this; return sort;};
1529         // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1530         void InitParDefaut();
1531
1532         protected:
1533         // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
1534         // utiliser le constructeur par défaut de Grandeur_coordonnee, d'où la nécessité d'une
construction
1535         // en deux temps !!
1536         Tableau <Grandeur_coordonnee* > tabCoor;
1537     };
1538     /// @} // end of group
1539
1540     /// @addtogroup Les_grandeurs_particulieres
1541     /// @{
1542     ///
1543
1544     //-----
1545     /// grandeur Ddl_etendu: TypeQuelconque::Grandeur::Grandeur_Ddl_etendu
1546     //-----
1547     class Grandeur_Ddl_etendu : public TypeQuelconque::Grandeur
1548     { friend class Tab_Grandeur_Ddl_etendu;
1549         // surcharge de l'opérateur de lecture
1550         friend istream & operator » (istream & ent, Grandeur_Ddl_etendu & a);
1551         // surcharge de l'opérateur d'écriture
1552         friend ostream & operator « (ostream & sort , const Grandeur_Ddl_etendu & a);
1553     public:
1554         // constructeur par défaut:
1555         Grandeur_Ddl_etendu();
1556         // constructeur légal
1557         Grandeur_Ddl_etendu(const Ddl_etendu& v,const string& nom): co(v) ,nom_ref(nom){};
1558         // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1559         // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1560         // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1561         Grandeur_Ddl_etendu(istream & ent);
1562         // constructeur de copie
1563         Grandeur_Ddl_etendu(const Grandeur_Ddl_etendu& a): co(a.co),nom_ref(a.nom_ref) {};
1564         // destructeur
1565         ~Grandeur_Ddl_etendu(){};
1566         // ramène une grandeur identique, créée
1567         TypeQuelconque::Grandeur* New_idem_grandeur() const {return new

```



```

Grandeur_Ddl_etendu(co,nom_ref);};
1568 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1569 // le constructeur adoc l'objet correcte et valide
1570 void EcriturePourLectureAvecCreation(ostream & sort);
1571 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1572 Grandeur & operator = ( const Grandeur_Ddl_etendu& b)
1573 {co=b.co;nom_ref=b.nom_ref; return *this;}; // surcharge d'affectation
1574 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1575 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1576 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1577 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1578 // Surcharge de l'opérateur +=
1579 void operator+= (const Grandeur& c);
1580 // Surcharge de l'opérateur -=
1581 void operator-= (const Grandeur& c);
1582 // Surcharge de l'opérateur *=
1583 void operator*= (double val) {co*=val;};
1584 // Surcharge de l'opérateur /= : division par un scalaire
1585 void operator/= (double val) {co/=val;};
1586 // récup du nom de référence
1587 virtual const string* Nom_ref() const {return &nom_ref;};
1588 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1589 // si pas possible, ramène 0
1590 double GrandeurNumOrdre(int num) const
1591 {if (num == 1) {return co.ConstValeur();} else {return 0.};};
1592 int NbMaxiNumeroOrdre() const {return 1;}; // récup du nb maxi de numéros d'ordres
1593 void Grandeur_brut(ostream & sort,int nbcarr) const
1594 {sort << " " << setprecision(nbcarr) << co.ConstValeur() << " " ;};
1595 // ramène le type de grandeur associée de la grandeur de base, car
1596 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1597 EnumTypeGrandeur Type_grandeurAssocie() const {return SCALAIRE_DOUBLE};;
1598 // ramène le type de la structure: tableau, liste ext..
1599 EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE};;
1600 // ramène le type de la grandeur particulière
1601 EnuTypeQuelParticulier Type_enumGrandeurParticuliere() const{return PARTICULIER_DDL_ETENDU};;
1602 // change de repère de la grandeur
1603 void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma) // a priori on ne fait rien
sur la grandeur scalaire
1604 {};
1605 // lecture
1606 istream & Lecture_grandeur(istream & ent) { ent >> nom_ref >> co ;return ent;};
1607 // ecriture
1608 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << " " << nom_ref << " " << co << "
"; return sort;};
1609 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1610 void InitParDefaut() {co.Valeur()=0.};;
1611 // méthode spécifique d'accès à la grandeur
1612 // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
1613 Ddl_etendu* ConteneurDdl_etendu() {return &co;};;
1614 void Change_nom_ref(const string& nom) {nom_ref = nom;};
1615
1616 protected:
1617 Ddl_etendu co;
1618 string nom_ref; // un nom de référence qui permet ensuite de discriminer les différents
conteneurs
1619 };
1620 /// @} // end of group
1621
1622 /// @addtogroup Les_grandeurs_particulieres
1623 /// @{
1624 ///
1625
1626 //-----
1627 /// grandeur un tableau de Coordonnee: TypeQuelconque::Grandeur::Tab_Grandeur_CoordonneeHH
1628 /// tous les coordonnees doivent avoir la même dimension !! pour la routine GrandeurNumOrdre
1629 //-----
1630 class Tab_Grandeur_Ddl_etendu : public TypeQuelconque::Grandeur
1631 { // surcharge de l'opérateur de lecture
1632 friend istream & operator >> (istream & ent, Tab_Grandeur_Ddl_etendu & a);
1633 // surcharge de l'opérateur d'écriture
1634 friend ostream & operator << (ostream & sort , const Tab_Grandeur_Ddl_etendu & a);
1635 public:
1636 // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
1637 // co défini, donc -> erreur (en fait ce n'est sûr, mais on fait comme pour les tenseurs)
1638 Tab_Grandeur_Ddl_etendu();
1639 // constructeur légal
1640 // nb indique le nombre de Ddl_etendu
1641 Tab_Grandeur_Ddl_etendu( Grandeur_Ddl_etendu & coor,int nb);
1642 // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1643 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1644 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1645 Tab_Grandeur_Ddl_etendu(istream & ent);
1646 // constructeur de copie
1647 Tab_Grandeur_Ddl_etendu(const Tab_Grandeur_Ddl_etendu& a) ;
1648 // destructeur

```



```

1649     ~Tab_Grandeur_Ddl_etendu();
1650     // ramène une grandeur identique, créée
1651     TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_Ddl_etendu(*this);};
1652     // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1653     // le constructeur adoc l'objet correcte et valide
1654     void EcriturePourLectureAvecCreation(ostream & sort);
1655     Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1656     Grandeur & operator = ( const Tab_Grandeur_Ddl_etendu & a); // surcharge d'affectation
1657     // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1658     // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1659     // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1660     void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1661     // Surcharge de l'opérateur +=
1662     void operator+= (const Grandeur& c) ;
1663     // Surcharge de l'opérateur -=
1664     void operator-= (const Grandeur& c) ;
1665     // Surcharge de l'opérateur *=
1666     void operator*= (double val) ;
1667     // Surcharge de l'opérateur /= : division par un scalaire
1668     void operator/= (double val) ;
1669     // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1670     // si pas possible, ramène 0
1671     double GrandeurNumOrdre(int num) const;
1672     int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
1673     void Grandeur_brut(ostream & sort,int nbcar) const;
1674     // ramène le type de grandeur associée de la grandeur de base, car
1675     // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1676     EnumTypeGrandeur Type_grandeurAssocie() const {return SCALAIRE_DOUBLE;}; // vecteur à 1
élément
1677     // ramène le type de la structure: tableau, liste ext..
1678     EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
1679     // ramène le type de la grandeur particulière
1680     EnuTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_DDL_ETENDU;};
1681     // acces au Coordonnee de numéro i en lecture écriture
1682     Ddl_etendu& operator () ( int i)const {return (tabDdlEte(i)->co);};
1683     // ramène la Grandeur_Ddl_etendu associée
1684     const Grandeur_Ddl_etendu& Grandeur_Ddl_etendu_associee(int i) const {return
*(tabDdlEte(i));};
1685     // ramène la dimension du tableau
1686     int Taille() const {return tabDdlEte.Taille();};
1687     // changement de taille -> n :
1688     // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
1689     // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
1690     void Change_taille(int n);
1691     // change de repère de la grandeur
1692     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma) {}; // ne fait rien ici car
ce sont des scalaires uniques
1693     // lecture
1694     istream & Lecture_grandeur(istream & ent) { ent >> *this;return ent;};
1695     // ecriture
1696     virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << *this; return sort;};
1697     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1698     void InitParDefaut();
1699
1700     protected:
1701     // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
1702     // utiliser le constructeur par défaut , d'où la nécessité d'une construction
1703     // en deux temps !!
1704     Tableau <Grandeur_Ddl_etendu* > tabDdlEte;
1705     };
1706     /// @} // end of group
1707
1708     /// @addtogroup Les_grandeurs_particulieres
1709     /// @{
1710     ///
1711
1712     //-----
1713     /// grandeur vecteur nommé : TypeQuelconque::Grandeur::Grandeur_Vecteur_Nommer
1714     /// contient un nom d'identificateur (utilisé par exemple pour un nom de fonc
1715     //-----
1716     class Grandeur_Vecteur_Nommer : public TypeQuelconque::Grandeur
1717     { // surcharge de l'opérateur de lecture
1718     friend istream & operator >> (istream & ent, Grandeur_Vecteur_Nommer & a);
1719     // surcharge de l'opérateur d'écriture
1720     friend ostream & operator << (ostream & sort , const Grandeur_Vecteur_Nommer & a);
1721     public:
1722     // constructeur par défaut:
1723     Grandeur_Vecteur_Nommer();
1724     // constructeur légal
1725     Grandeur_Vecteur_Nommer(const string & nom_ref_, const int taille,Fonction_nD * fct_ =NULL)
1726     : nom_ref(nom_ref_),fct(fct_)
1727     {v.Change_taille(taille);};
1728     // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée

```

```

1729 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1730 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1731 Grandeur_Vecteur_Nommer(istream & ent);
1732 // constructeur de copie
1733 Grandeur_Vecteur_Nommer(const Grandeur_Vecteur_Nommer& a):
1734     nom_ref(a.nom_ref),v(a.v),fct(a.fct) {};
1735 // destructeur
1736 ~Grandeur_Vecteur_Nommer(){};
1737 // ramène une grandeur identique, créée
1738 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Grandeur_Vecteur_Nommer(*this)};
1739 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1740 // le constructeur adoc l'objet correcte et valide
1741 void EcriturePourLectureAvecCreation(ostream & sort);
1742 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1743 Grandeur & operator = ( const Grandeur_Vecteur_Nommer& b)
1744     {v=b.v;nom_ref = b.nom_ref; fct=b.fct; return *this;}; // surcharge d'affectation
1745 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1746 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1747 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1748 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1749 // Surcharge de l'opérateur +=
1750 void operator+= (const Grandeur& c);
1751 // Surcharge de l'opérateur -=
1752 void operator-= (const Grandeur& c);
1753 // Surcharge de l'opérateur *=
1754 void operator*= (double val) {v *=val;};
1755 // Surcharge de l'opérateur /= : division par un scalaire
1756 void operator/= (double val) {v /=val;};
1757 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1758 // si pas possible, ramène 0
1759 double GrandeurNumOrdre(int num) const
1760     {if ((num <= v.Taille())&&(num>0)) {return v(num);} else {return 0.};};
1761 // récup du nom de référence
1762 virtual const string* Nom_ref() const {return &nom_ref;};
1763 int NbMaxiNumeroOrdre() const {return v.Taille();}; // récup du nb maxi de numéros d'ordres
1764 void Grandeur_brut(ostream & sort,int nbcар const;
1765 // ramène le type de grandeur associée de la grandeur de base, car
1766 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1767 EnumTypeGrandeur Type_grandeurAssocie() const {return VECTEUR;};
1768 // ramène le type de la structure: tableau, liste ext..
1769 EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE;};
1770 // ramène le type de la grandeur particulière
1771 EnuTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_VECTEUR_NOMMER;};
1772 // change de repère de la grandeur
1773 void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma) // a priori on ne fait rien
sur la grandeur vectoriel
1774     {};
1775 // lecture
1776 istream & Lecture_grandeur(istream & ent) { ent >> nom_ref >> v ;return ent;};
1777 // ecriture
1778 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << " " << nom_ref << " " << v ;
return sort;};
1779 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1780 void InitParDefaut() {v.Zero();};
1781 // ----- méthodes spécifiques d'accès à la grandeur -----
1782 // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
1783 Vecteur& ConteneurVecteur() {return v;};
1784 Fonction_nD * Fct() const {return fct;};
1785 // const string& Nom_vec() const {return nom_ref;};
1786 void Change_fonction_nD(Fonction_nD * fct_) {fct = fct_};
1787
1788
1789 protected:
1790 Fonction_nD * fct; // la fonction nD qui permet le calcul du résultat stocké dans v
1791 // n'est pas forcément défini
1792 string nom_ref ; // un nom de référence qui permet de discriminer une même intégrale de
fonction nD, ou une statistique par exemple
1793 // mais sur des volumes ou des ref différents
1794 Vecteur v;
1795 };
1796 /// @} // end of group
1797
1798 /// @addtogroup Les_grandeurs_particulieres
1799 /// @{
1800 ///
1801
1802
1803 /// -----
1804 /// grandeur un tableau de Grandeur_Vecteur_Nommer:
TypeQuelconque::Grandeur::Tab_Grandeur_Grandeur_Vecteur_Nommer
1805
1806 /// -----
class Tab_Grandeur_Vecteur_Nommer : public TypeQuelconque::Grandeur

```

```

1806     { // surcharge de l'operator de lecture
1807         friend istream & operator » (istream & ent, Tab_Grandeur_Vecteur_Nommer & a);
1808         // surcharge de l'operator d'écriture
1809         friend ostream & operator « (ostream & sort, const Tab_Grandeur_Vecteur_Nommer & a);
1810     public:
1811         // constructeur par défaut: ne doit pas être utilisé, car on doit toujours avoir la variable
1812         // co défini, donc -> erreur (en fait ce n'est sûr, mais on fait comme pour les tenseurs)
1813         Tab_Grandeur_Vecteur_Nommer();
1814         // constructeur légal
1815         // nb indique le nombre de Grandeur_Vecteur_Nommer
1816         Tab_Grandeur_Vecteur_Nommer(Grandeur_Vecteur_Nommer & grno, int nb);
1817         // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1818         // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1819         // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1820         Tab_Grandeur_Vecteur_Nommer(istream & ent);
1821         // constructeur de copie
1822         Tab_Grandeur_Vecteur_Nommer(const Tab_Grandeur_Vecteur_Nommer& a);
1823         // destructeur
1824         ~Tab_Grandeur_Vecteur_Nommer();
1825         // ramène une grandeur identique, créée
1826         TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tab_Grandeur_Vecteur_Nommer(*this)};
1827         // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1828         // le constructeur adoc l'objet correcte et valide
1829         void EcriturePourLectureAvecCreation(ostream & sort);
1830         Grandeur & operator = (const Grandeur & a); // surcharge d'affectation
1831         Grandeur & operator = (const Tab_Grandeur_Vecteur_Nommer & a); // surcharge d'affectation
1832         // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1833         // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
1834         // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1835         void Affectation_numerique(const TypeQuelconque::Grandeur & a);
1836         // Surcharge de l'opérateur +=
1837         void operator+= (const Grandeur& c);
1838         // Surcharge de l'opérateur -=
1839         void operator-= (const Grandeur& c);
1840         // Surcharge de l'opérateur *=
1841         void operator*= (double val);
1842         // Surcharge de l'opérateur /= : division par un scalaire
1843         void operator/= (double val);
1844         // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1845         // si pas possible, ramène 0
1846         double GrandeurNumOrdre(int num) const {return 0};
1847         // ici c'est le maxi des vecteurs nommés * le nombre
1848         int NbMaxiNumeroOrdre() const {return 0}; // récup du nb maxi de numéros d'ordres
1849         void Grandeur_brut(ostream & sort, int nbcar) const;
1850         // ramène le type de grandeur associée de la grandeur de base, car
1851         // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
1852         EnumTypeGrandeur Type_grandeurAssocie() const {return VECTEUR}; // vecteur à 1 élément
1853         // ramène le type de la structure: tableau, liste ext..
1854         EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T};
1855         // ramène le type de la grandeur particulière
1856         EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_VECTEUR_NOMMER};
1857         // acces au Grandeur_Vecteur_Nommer de numéro i en lecture écriture
1858         Grandeur_Vecteur_Nommer& operator () (int i) const {return *(tabVN(i))};
1859         // ramène la Grandeur_Vecteur_Nommer associée
1860         const Grandeur_Vecteur_Nommer& Grandeur_Vecteur_Nommer_associee(int i) const {return
*(tabVN(i))};
1861         // ramène la dimension du tableau
1862         int Taille() const {return tabVN.Taille()};
1863         // changement de taille -> n :
1864         // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
1865         // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
1866         void Change_taille(int n);
1867         // change de repère de la grandeur
1868         void Change_repere(const Mat_pleine& beta, const Mat_pleine& gamma) {}; // ne fait rien ici car
ce sont des scalaires uniques
1869         // lecture
1870         istream & Lecture_grandeur(istream & ent) { ent » *this; return ent;};
1871         // écriture
1872         virtual ostream & Ecriture_grandeur(ostream & sort) const { sort « *this; return sort;};
1873         // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1874         void InitParDefaut();
1875     protected:
1876         // on utilise un tableau de pointeur et non de valeur car pour la construction il ne faut pas
1877         // utiliser le constructeur par défaut, d'où la nécessité d'une construction
1878         // en deux temps !!
1879         Tableau <Grandeur_Vecteur_Nommer* > tabVN;
1880     };
1881     /// @} // end of group
1882     /// @}
1883     /// @addtogroup Les_grandeurs_particulieres
1884     /// @{
1885     /// @}
1886     ///

```

```

1887
1888 //-----
1889 /// grandeur BaseH: TypeQuelconque::Grandeur::Grandeur_BaseH
1890 //-----
1891 class Grandeur_BaseH : public TypeQuelconque::Grandeur
1892 { friend class Tab_Grandeur_BaseH;
1893 // surcharge de l'operator de lecture
1894 friend istream & operator » (istream & ent, Grandeur_BaseH & a);
1895 // surcharge de l'operator d'écriture
1896 friend ostream & operator « (ostream & sort, const Grandeur_BaseH & a);
1897 public:
1898 // constructeur par défaut:
1899 Grandeur_BaseH(): baseH() {};
1900 // constructeur fonction du nombre de vecteur et de la dimension des vecteurs de base
1901 Grandeur_BaseH(int dim, int nb_vec ): baseH(dim,nb_vec) {};
1902 // constructeur légal
1903 Grandeur_BaseH(const BaseH& basH): baseH(basH) {};
1904 // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1905 // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1906 // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1907 Grandeur_BaseH(istream & ent);
1908 // constructeur de copie
1909 Grandeur_BaseH(const Grandeur_BaseH& a): baseH(a.baseH) {};
1910 // destructeur
1911 ~Grandeur_BaseH() {};
1912 // ramène une grandeur identique, créée
1913 TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Grandeur_BaseH(baseH);};
1914 // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1915 // le constructeur adoc l'objet correcte et valide
1916 void EcriturePourLectureAvecCreation(ostream & sort);
1917 Grandeur & operator = ( const Grandeur & a); // surcharge d'affectation
1918 Grandeur & operator = ( const Grandeur_BaseH & a) // surcharge d'affectation
1919 {baseH = a.baseH; return *this; }
1920 Grandeur & operator = ( const BaseH & a) // surcharge d'affectation
1921 {baseH = a; return *this; }
1922 // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
1923 // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
1924 même type
1925 // affectation uniquement des données numériques (pas les pointeurs, pas les string)
1926 void Affectation_numerique( const TypeQuelconque::Grandeur & a);
1927 // Surcharge de l'operateur +=
1928 void operator+= (const Grandeur& c) ;
1929 void operator+= (const Grandeur_BaseH& aa);
1930 // Surcharge de l'operateur -=
1931 void operator-= (const Grandeur& c) ;
1932 void operator-= (const Grandeur_BaseH& aa);
1933 // Surcharge de l'operateur *=
1934 void operator*= (double val);
1935 // Surcharge de l'operateur /= : division par un scalaire
1936 void operator/= (double val);
1937 // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
1938 // si pas possible, ramène 0
1939 double GrandeurNumOrdre(int num) const ;
1940 int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
1941 void Grandeur_brut(ostream & sort,int nbcar) const ;
1942 // ramène le type de grandeur associée de la grandeur de base, car
1943 // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
1944 base
1945 EnumTypeGrandeur Type_grandeurAssocie() const {return BASE_H;};
1946 // ramène le type de la structure: tableau, liste ext..
1947 EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE;};
1948 // ramène le type de la grandeur particulière
1949 EnuTypeQuelParticulier Type_enumGrandeurParticuliere() const{return PARTICULIER_BASE_H;};
1950 // change de repère de la grandeur
1951 void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
1952 // lecture
1953 istream & Lecture_grandeur(istream & ent) { ent » *this;return ent;};
1954 // écriture
1955 virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort « *this; return sort;};
1956 // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
1957 void InitParDefaut();
1958 // méthode spécifique d'accès à la grandeur
1959 // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
1960 BaseH* ConteneurBaseH() {return &baseH;};
1961 // la référence est un peu plus sur (il est toujours possible de faire un delete sur un
1962 // pointeur de ref
1963 // mais c'est plus difficile de le faire en se trompant ! enfin j'espère)
1964 const BaseH& RefConteneurBaseH() const {return baseH;};
1965
1966 protected:
1967 BaseH baseH;
1968 };
1969 /// @} // end of group
1970
1971 /// @addtogroup Les_grandeurs_particulieres
1972 /// @{}
1973 ///

```

```

1971
1972 //-----
1973 /// grandeur un tableau de Grandeur_BaseH: TypeQuelconque::Grandeur::Tab_Grandeur_BaseH
1974 /// tous les coordonnees doivent avoir la même dimension !! pour la routine GrandeurNumOrdre
1975 //-----
1976     class Tab_Grandeur_BaseH : public TypeQuelconque::Grandeur
1977     { // surcharge de l'operator de lecture
1978         friend istream & operator » (istream & ent, Tab_Grandeur_BaseH & a);
1979         // surcharge de l'operator d'écriture
1980         friend ostream & operator « (ostream & sort, const Tab_Grandeur_BaseH & a);
1981     public:
1982         // constructeur par défaut:
1983         Tab_Grandeur_BaseH(): tabBaseH() {};
1984         // constructeur légal
1985         // nb indique le nombre de Grandeur_BaseH
1986         Tab_Grandeur_BaseH(Grandeur_BaseH & baseH,int nb): tabBaseH(nb,baseH) {};
1987         // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
1988         // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
1989         // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
1990         Tab_Grandeur_BaseH(istream & ent);
1991         // constructeur de copie
1992         Tab_Grandeur_BaseH(const Tab_Grandeur_BaseH& a) : tabBaseH(a.tabBaseH) {};
1993         // destructeur
1994         ~Tab_Grandeur_BaseH() {};
1995         // ramène une grandeur identique, créée
1996         TypeQuelconque::Grandeur* New_idem_grandeur() const {return new Tab_Grandeur_BaseH(*this);};
1997         // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
1998         // le constructeur adoc l'objet correcte et valide
1999         void EcriturePourLectureAvecCreation(ostream & sort);
2000         Grandeur & operator = (const Grandeur & a); // surcharge d'affectation
2001         Grandeur & operator = (const Tab_Grandeur_BaseH & a) // surcharge d'affectation
2002             {tabBaseH = a.tabBaseH; };
2003         // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
2004         // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
2005         même type
2006         // affectation uniquement des données numériques (pas les pointeurs, pas les string)
2007         void Affectation_numerique(const TypeQuelconque::Grandeur & a);
2008         // Surcharge de l'operateur +=
2009         void operator+= (const Grandeur& c) ;
2010         // Surcharge de l'operateur -=
2011         void operator-= (const Grandeur& c) ;
2012         // Surcharge de l'operateur *=
2013         void operator*= (double val) ;
2014         // Surcharge de l'operateur /= : division par un scalaire
2015         void operator/= (double val) ;
2016         // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
2017         // si pas possible, ramène 0
2018         double GrandeurNumOrdre(int num) const ;
2019         int NbMaxiNumeroOrdre() const; // récup du nb maxi de numéros d'ordres
2020         void Grandeur_brut(ostream & sort,int nbcarr) const;
2021         // ramène le type de grandeur associée de la grandeur de base, car
2022         // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
2023         base
2024         EnumTypeGrandeur Type_grandeurAssocie() const {return BASE__H;};
2025         // ramène le type de la structure: tableau, liste ext..
2026         EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
2027         // ramène le type de la grandeur particulière
2028         EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_BASE_H;};
2029         // acces au Grandeur_BaseH de numéro i en lecture écriture
2030         Grandeur_BaseH& operator () (int i)const {return tabBaseH(i);};
2031         // ramène la grandeur_Grandeur_BaseH associée
2032         const Grandeur_BaseH& Grandeur_Grandeur_BaseH_associee(int i) const {return tabBaseH(i);};
2033         // ramène la dimension du tableau de Grandeur_BaseH
2034         int Taille() const {return tabBaseH.Taille();};
2035         // changement de taille -> n :
2036         // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
2037         // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
2038         void Change_taille(int n);
2039         // change de repère de la grandeur
2040         void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma);
2041         // lecture
2042         istream & Lecture_grandeur(istream & ent) { ent » *this;return ent;};
2043         // écriture
2044         virtual ostream & Ecriture_grandeur(ostream & sort) const { sort « *this; return sort;};
2045         // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
2046         void InitParDefaut();
2047     protected:
2048         // on utilise un tableau
2049         Tableau <Grandeur_BaseH > tabBaseH;
2050     };
2051 /// @} // end of group
2052 /// @addtogroup Les_grandeurs_particulieres
2053 /// @{
2054 ///

```

```

2055
2056
2057 /// grandeur scalaire double nommé + indice: TypeQuelconque::Grandeur::Grandeur_Double_Nommer_indicer
2058 /// contient un nom d'identificateur et un indice qui est sensé resté fixe pendant les opérations
2059 /// la seule grandeur qui varie c'est le double: c'est donc équivalent à un scalaire double
2060 /// l'indice et le nom_ref, servent pour la gestion
2061
2062 -----
2063 class Grandeur_Double_Nommer_indicer : public TypeQuelconque::Grandeur
2064 { // surcharge de l'operator de lecture
2065   friend istream & operator » (istream & ent, Grandeur_Double_Nommer_indicer & a);
2066   // surcharge de l'operator d'écriture
2067   friend ostream & operator « (ostream & sort, const Grandeur_Double_Nommer_indicer & a);
2068   public:
2069   // constructeur par défaut:
2070   Grandeur_Double_Nommer_indicer();
2071   // constructeur légal
2072   Grandeur_Double_Nommer_indicer(const string & nom_ref_,const double& va,int indice_ = 1)
2073     : nom_ref(nom_ref_),val(va),indice(indice_)
2074     {};
2075   // constructeur qui effectue la construction en lisant les informations sur le stream d'entrée
2076   // nécessite d'utiliser la fonction EcriturePourLectureAvecCreation() pour la fonction
2077   // inverse c'est-à-dire l'écriture sur le stream, au lieu de la surcharge d'écriture normale
2078   Grandeur_Double_Nommer_indicer(istream & ent);
2079   // constructeur de copie
2080   Grandeur_Double_Nommer_indicer(const Grandeur_Double_Nommer_indicer& a):
2081     nom_ref(a.nom_ref),val(a.val),indice(a.indice) {};
2082   // destructeur
2083   ~Grandeur_Double_Nommer_indicer(){};
2084   // ramène une grandeur identique, créée
2085   TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Grandeur_Double_Nommer_indicer(*this)};
2086   // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
2087   // le constructeur adoc l'objet correcte et valide
2088   void EcriturePourLectureAvecCreation(ostream & sort);
2089
2090   // la surcharge d'affectation concerne toutes les grandeurs
2091   Grandeur & operator = (const Grandeur & a); // surcharge d'affectation
2092   Grandeur & operator = (const Grandeur_Double_Nommer_indicer& b)
2093     {val=b.val;nom_ref = b.nom_ref; indice=b.indice; return *this;}; // surcharge
d'affectation
2094   // ** opérations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
2095   // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
2096   // affectation uniquement des données numériques (pas les pointeurs, pas les string)
2097
2098   // >>> ici seul le double val est concerné !! <<<<<<<<
2099   void Affectation_numerique(const TypeQuelconque::Grandeur & a);
2100   // Surcharge de l'opérateur +=
2101   void operator+= (const Grandeur& c);
2102   // Surcharge de l'opérateur -=
2103   void operator-= (const Grandeur& c);
2104   // Surcharge de l'opérateur *=
2105   void operator*= (double x) {val *=x;};
2106   // Surcharge de l'opérateur /= : division par un scalaire
2107   void operator/= (double x) {val /=x;};
2108   // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
2109   // si pas possible, ramène 0
2110   double GrandeurNumOrdre(int num) const {if(num==1) return val; else return 0.};;
2111   // récup du nom de référence
2112   virtual const string* Nom_ref() const {return &nom_ref;};
2113   int NbMaxiNumeroOrdre() const {return 1;}; // récup du nb maxi de numéros d'ordres
2114   void Grandeur_brut(ostream & sort,int nbcар const
2115     {sort « " " « setw(nbcар) « setprecision(nbcар) « val « " ";};
2116   // ramène le type de grandeur associée de la grandeur de base, car
2117   // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
2118   EnumTypeGrandeur Type_grandeurAssocie() const {return SCALAIRE;};
2119   // ramène le type de la structure: tableau, liste ext..
2120   EnumType2Niveau Type_structure_grandeurAssocie() const {return TYPE_SIMPLE;};
2121   // ramène le type de la grandeur particulière
2122   EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_SCALAIRE_DOUBLE_NOMMER_INDICER;};
2123   // change de repère de la grandeur
2124   void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma) // a priori on ne fait rien
sur la grandeur scalaire
2125   {};
2126   // lecture
2127   istream & Lecture_grandeur(istream & ent)
2128   { ent » nom_ref » val » indice ;return ent;};
2129   // écriture
2130   virtual ostream & Ecriture_grandeur(ostream & sort) const
2131   { sort « " " « nom_ref « " " « val « " " « indice ;return sort;};
2132   // ----- méthodes spécifiques d'accès à la grandeur -----
2133   // initialise la grandeur à une valeur par défaut: = 0 pour la grandeur double
2134   void InitParDefaut(){val = 0.};;

```

```

2134
2135 // méthode spécifique d'accès à la grandeur
2136 // le fait d'utiliser une méthode permet de savoir qui passe par cette méthode
2137 double* ConteneurDouble() {return &val;};
2138 // idem en const
2139 double ContDouble() const {return val;};
2140 // l'indice
2141 int Indice_const() const {return indice;};
2142 int& Indice() {return indice;};
2143
2144 protected:
2145     string nom_ref ; // un nom de référence
2146     double val; // valeur
2147     int indice;
2148 };
2149 /// @} // end of group
2150
2151 /// @addtogroup Les_grandeurs_particulieres
2152 /// @{
2153 ///
2154
2155 //-----
2156 /// un tableau de grandeur quelconque
2157 //-----
2158 // on arrête le développement car c'est pas sûr que c'est utile en fait !!
2159     class Tableau_Grandeur_quelconque : public TypeQuelconque::Grandeur
2160     { // surcharge de l'operator de lecture
2161         friend istream & operator » (istream & ent, Tableau_Grandeur_quelconque & a);
2162         // surcharge de l'operator d'écriture
2163         friend ostream & operator « (ostream & sort , const Tableau_Grandeur_quelconque & a);
2164     public:
2165         // constructeur par défaut:
2166         Tableau_Grandeur_quelconque();
2167         // constructeur légal: on recopie nfois
2168         // nb indique le nombre de grandeur quelconque que contient le tableau
2169         Tableau_Grandeur_quelconque(const TypeQuelconque::Grandeur& tg,int nb);
2170         // cas d'un tableau déjà connu
2171         // Tableau_Grandeur_quelconque(const Tableau <TypeQuelconque::Grandeur> & tab);
2172         // idem pour des pointeurs
2173         Tableau_Grandeur_quelconque(const Tableau <TypeQuelconque::Grandeur*> & tab);
2174         // idem pour des grandeurs quelconques
2175         Tableau_Grandeur_quelconque(const Tableau <TypeQuelconque> & tab);
2176         // constructeur de copie
2177         Tableau_Grandeur_quelconque(const Tableau_Grandeur_quelconque& a);
2178         // destructeur
2179         ~Tableau_Grandeur_quelconque();
2180         // ramène une grandeur identique, créée
2181         TypeQuelconque::Grandeur* New_idem_grandeur() const {return new
Tableau_Grandeur_quelconque(*this)};
2182         // sauvegarde sur le stream des informations permettant ensuite en lecture de créer via
2183         // le constructeur adoc l'objet correcte et valide
2184         void EcriturePourLectureAvecCreation(ostream & sort);
2185         TypeQuelconque::Grandeur & operator = ( const TypeQuelconque::Grandeur & a); // surcharge
d'affectation
2186         // ** operations simples: si l'opération ne veut rien dire pour le type quelconque -> erreur
2187         // ** les opérations entre deux types quelconques ne doivent concerner que des grandeurs de
même type
2188         // affectation uniquement des données numériques (pas les pointeurs, pas les string)
2189         void Affectation_numerique( const TypeQuelconque::Grandeur & a);
2190         // Surcharge de l'operateur +=
2191         void operator+= (const Grandeur& c);
2192         // Surcharge de l'operateur -=
2193         void operator-= (const Grandeur& c);
2194         // Surcharge de l'operateur *=
2195         void operator*= (double val);
2196         // Surcharge de l'operateur /= : division par un scalaire
2197         void operator/= (double val);
2198         // récupération de la valeur numérique d'une grandeur correspondant à un numéro d'ordre
2199         // si pas possible, ramène 0
2200         double GrandeurNumOrdre(int num) const {return 0;};
2201         int NbMaxiNumeroOrdre() const {return 0;}; // récup du nb maxi de numéros d'ordres
2202         void Grandeur_brut(ostream & sort,int nbcarr) const;
2203         // ramène le type de grandeur associée de la grandeur de base, car
2204         // la grandeur peut être une grandeur complexe comme un tableau ou un liste de grandeur de
base
2205         EnumTypeGrandeur Type_grandeurAssocie() const {return GRANDEUR_QUELCONQUE;};
2206         // ramène le type de la structure: tableau, liste ext..
2207         EnumType2Niveau Type_structure_grandeurAssocie() const {return TABLEAU_T;};
2208         // ramène le type de la grandeur particulière
2209         EnumTypeQuelParticulier Type_enumGrandeurParticuliere() const{return
PARTICULIER_TABLEAU_QUELCONQUE;};
2210         // acces au Grandeur_Vecteur_Nommer de numéro i en lecture écriture
2211         TypeQuelconque::Grandeur& operator () ( int i)const {return *(tab_grandeur(i))};
2212         // ramène la Grandeur_Vecteur_Nommer associée
2213         const TypeQuelconque::Grandeur& ConteneurGrandeurQuelconque(int i) const {return
*(tab_grandeur(i))};
2214         // ramène la dimension du tableau

```



```

2215     int Taille() const {return tab_grandeur.Taille();};
2216     // changement de taille -> n :
2217     // si n < taille_actuelle --> les taille_actuelle - n sont supprimé
2218     // si n > taille_actuelle --> le dernier élément existant est dupliqué n-taille_actuelle fois,
2219     void Change_taille(int n);
2220     // change de repère de la grandeur
2221     void Change_repere(const Mat_pleine& beta,const Mat_pleine& gamma );
2222     // lecture
2223     istream & Lecture_grandeur(istream & ent) { ent >> *this;return ent;};
2224     // ecriture
2225     virtual ostream & Ecriture_grandeur(ostream & sort ) const { sort << *this; return sort;};
2226     // initialise la grandeur à une valeur par défaut: = 0 pour les grandeurs numériques
2227     void InitParDefaut();
2228
2229     protected:
2230     Tableau <TypeQuelconque::Grandeur*> tab_grandeur;
2231     };
2232 /// @} // end of group
2233
2234
2235 #endif

```

## 7.501 Vector\_io.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *      DATE:      23/01/97
31 *
32 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *      PROJET:    Herezh++
35 *
36 *
37 *      BUT: création d'un conteneur vector stl, qui comporte en plus
38 *      une surcharge de lecture écriture.
39 *
40 *      *****
41 *
42 *
43 *****/
44 #ifndef VECTOR_IO_H
45 #define VECTOR_IO_H
46 #include <iostream>
47 #include <stdlib.h>
48 #include <vector>
49 #include "Sortie.h"
50
51 /**
52 *
53 *      BUT: création d'un conteneur vector stl, qui comporte en plus
54 *      une surcharge de lecture écriture.
55 *
56 *
57 * \author   Gérard Rio
58 * \version  1.0
59 * \date    23/01/97
60 * \brief   création de conteneurs type vector stl avec surcharge de lecture écriture
61 *

```



```

62 */
63
64 template <class T>
65 class Vector_io : public vector<T>
66 { // surcharge de l'operator de lecture
67     friend istream & operator « (istream & entree, Vector_io& )
68     { cout « "erreur, cette surcharge ne doit pas être utilisée "
69       « "Vector_io, operator » \n";
70       Sortie(1);
71       return entree;
72     }
73     // surcharge de l'operator d'écriture
74     friend ostream & operator « (ostream & sort, const Vector_io& )
75     { cout « "erreur, cette surcharge ne doit pas être utilisée "
76       « "VectorVector_io, operator « \n";
77       Sortie(1);
78       return sort;
79     }
80
81 public :
82     // CONSTRUCTEURS :
83
84     // DESTRUCTEUR :
85
86     // METHODES PUBLIQUES :
87
88 private :
89     // VARIABLES PROTEGEES :
90
91     // CONSTRUCTEURS :
92
93     // DESTRUCTEUR :
94
95     // METHODES PROTEGEES :
96
97 };
98
99 #endif

```

## 7.502 LaLIST.H

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Grard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE L (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultrieure.
10 //
11 // Copyright (C) 1997-2021 Universit Bretagne Sud (France)
12 // AUTHOR : Grard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /// programme modifi : le nom LaLIST,
31 // pour pouvoir tre utilis avec la bibli
32 // en fait uniquement idem que les listes stl dans un premier temps
33
34 #ifndef LIST_H
35 #define LIST_H
36 #include <list>
37 // typedef list LaLIST;
38 #define LaLIST list
39 #endif
40

```

## 7.503 Algo\_edp.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           12/02/2006                               $   *
31 *                                                         *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)     $   *
33 *                                                         *
34 *   PROJET:        Herezh++                                $   *
35 *                                                         *
36 *****/
37 *   BUT:   Algorithmes de base pour la résolution d'équations
38 *          différentielles.                                $   *
39 *          *
40 *          *
41 *          *
42 *****/
43 #ifndef ALGO_EDP_H
44 #define ALGO_EDP_H
45
46 #include "Vecteur.h"
47 #include "Mat_abstraite.h"
48 #include "Racine.h"
49
50 /** @defgroup Les_classes_algo
51 *
52 *   BUT:   Algorithmes de base pour la résolution d'équations
53 *          différentielles, recherche de zéros, intégration.
54 *
55 *
56 * \author   Gérard Rio
57 * \version  1.0
58 * \date    12/02/2006
59 * \brief   Algorithmes de base pour la résolution d'équations différentielles, recherche de zéros,
60 *          intégration.
61 */
62
63 /// @addtogroup Les_classes_algo
64 /// @{
65 ///
66
67 ///   BUT:   Algorithmes de base pour la résolution d'équations
68 ///   différentielles.
69 class Algo_edp
70 {
71 public :
72     // CONSTRUCTEURS :
73     // constructeur par défaut
74     Algo_edp();
75     // constructeur de copie
76     Algo_edp(const Algo_edp& a);
77     // DESTRUCTEUR :
78     ~Algo_edp();
79
80     // METHODES PUBLIQUES :
81
82
83     /// résolution par runge kutta imbriqué --> estimation d'erreur
84     /// il y a 3 calcul de la fonction dérivée

```

```

85  /// 2 et 3 ième ordre imbriquée
86  /// calcul de la solution à t+dt en fonction de la solution à t,
87  /// en entrée:
88  /// *Ptder_fonc : pointeur de la fonction de calcul de la dérivées
89  ///             en entrée: t et f : temps et la valeur de la fonction a t
90  ///             en sortie: df      : dérivée de la fonction a t
91  ///             erreur : si diff de 0, indique qu'il y a eu une erreur
92  /// *Ptverif_fonc : pointeur de fonction, pour tester val_finale après la prédiction explicite
    finale
93  ///
94  /// val_initiale : valeur des fonctions en t0
95  /// der_initiale : valeur de la dérivée pour t0
96  /// t0           : temps initiale
97  /// deltat      : incrément de temps demandé
98  /// en sortie :
99  /// val_finale  : valeur des fonctions à tdt
100  /// estime_erreur : estimation d'erreur pour chaque fonction
101  /// --> si estime_erreur(i) est >= à ConstMath::tresgrand, la valeur finale = la valeur initiale
    (c-a-d pas de calcul valide)
102  template <class T> void Runge_Kutta_step23(T& instance
103  ,Vecteur& (T::*Ptder_fonc) (const double & t,const Vecteur& f,Vecteur& df,int&
    erreur)
104  ,void (T::*Ptverif_fonc) (const double & t,const Vecteur& f,int& erreur_final)
105  ,const Vecteur& val_initiale,const Vecteur& der_initiale
106  ,const double& t0,const double& deltat
107  ,Vecteur& val_finale,Vecteur& estime_erreur) ;
108
109
110  /// résolution par runge kutta imbriqué --> estimation d'erreur
111  /// il y a 4 calcul de la fonction dérivée
112  /// 3 et 4 ième ordre imbriquée (Fehlberg)
113  /// calcul de la solution à t+dt en fonction de la solution à t,
114  /// en entrée:
115  /// *Ptder_fonc : pointeur de la fonction de calcul de la dérivées
116  ///             en entrée: t et f : temps et la valeur de la fonction a t
117  ///             en sortie: df      : dérivée de la fonction a t
118  ///             erreur : si diff de 0, indique qu'il y a eu une erreur
119  /// *Ptverif_fonc : pointeur de fonction, pour tester val_finale après la prédiction explicite
    finale
120  ///
121  /// val_initiale : valeur des fonctions en t0
122  /// der_initiale : valeur de la dérivée pour t0
123  /// t0           : temps initiale
124  /// deltat      : incrément de temps demandé
125  /// en sortie :
126  /// val_finale  : valeur des fonctions à tdt
127  /// estime_erreur : estimation d'erreur pour chaque fonction
128  /// --> si estime_erreur(i) est >= à ConstMath::tresgrand, la valeur finale = la valeur initiale
    (c-a-d pas de calcul valide)
129  template <class T> void Runge_Kutta_step34(T& instance
130  ,Vecteur& (T::*Ptder_fonc) (const double & t,const Vecteur& f,Vecteur& df,int&
    erreur)
131  ,void (T::*Ptverif_fonc) (const double & t,const Vecteur& f,int& erreur_final)
132  ,const Vecteur& val_initiale,const Vecteur& der_initiale
133  ,const double& t0,const double& deltat
134  ,Vecteur& val_finale,Vecteur& estime_erreur) ;
135
136  /// résolution par runge kutta imbriqué --> estimation d'erreur
137  /// il y a 5 calcul de la fonction dérivée
138  /// l'algorithme s'appuie sur la présentation de numerical recipes
139  /// fifth-order Cash-Karp Runge-Kutta
140  /// calcul de la solution à t+dt en fonction de la solution à t,
141  /// en entrée:
142  /// *Ptder_fonc : pointeur de la fonction de calcul de la dérivées
143  ///             en entrée: t et f : temps et la valeur de la fonction a t
144  ///             en sortie: df      : dérivée de la fonction a t
145  ///             erreur : si diff de 0, indique qu'il y a eu une erreur
146  /// *Ptverif_fonc : pointeur de fonction, pour tester val_finale après la prédiction explicite
    finale
147  ///
148  /// val_initiale : valeur des fonctions en t0
149  /// der_initiale : valeur de la dérivée pour t0
150  /// t0           : temps initiale
151  /// deltat      : incrément de temps demandé
152  /// en sortie :
153  /// val_finale  : valeur des fonctions à tdt
154  /// estime_erreur : estimation d'erreur pour chaque fonction
155  /// --> si estime_erreur(i) est >= à ConstMath::tresgrand, la valeur finale = la valeur initiale
    (c-a-d pas de calcul valide)
156  template <class T> void Runge_Kutta_step45(T& instance
157  ,Vecteur& (T::*Ptder_fonc) (const double & t,const Vecteur& f,Vecteur& df,int&
    erreur)
158  ,void (T::*Ptverif_fonc) (const double & t,const Vecteur& f,int& erreur_final)
159  ,const Vecteur& val_initiale,const Vecteur& der_initiale
160  ,const double& t0,const double& deltat
161  ,Vecteur& val_finale,Vecteur& estime_erreur) ;
162

```

```

163
164 /// un programme de pilotage de l'intégration d'un pas (on s'inspire du programme libre rkf45)
165 /// on distingue une erreur globale et une erreur relative
166 /// l'erreur réelle utilisée sur chaque step est: erreurRelative * |f| + erreurAbsolue
167 /// ---- en entrée: ----
168 /// cas_kutta : donne le type de routine kutta imbriqué a employer:
169 /// =3 pour kutta23, =4 pour kutta34, =5 pour kutta45
170 /// tdebut et tfin : temps de début et de fin du calcul, demandé
171 /// val_initiale : valeur initiale de la fonction (donc à tdebut)
172 /// der_initiale : dérivée initiale de la fonction (donc à tdebut)
173 /// erreurAbsolue : erreur absolue demandée
174 /// erreurRelative : erreur relative (à la fonction) demandée
175 /// Ptder_fonc : pointeur de fonction
176 /// en entrée: t et f : temps et la valeur de la fonction a t
177 /// en sortie: df : dérivée de la fonction a t
178 /// en entrée: t et f : temps et la valeur de la fonction a t
179 /// en sortie: df : dérivée de la fonction a t
180 /// erreur : si diff de 0, indique qu'il y a eu une erreur
181 /// Ptverif_fonc : pointeur de fonction, pour tester l'intégrité du résultat
182 /// cela signifie que val_finale est testée avec cette fonction, systématiquement
183 /// - à la fin de chaque prédiction des kuttas imbriqués
184 /// - au retour du pilotage
185 /// ---- en sortie: ----
186 /// dernierTemps : temps final utilisé = tfin si calcul ok, sinon = le dernier temps calculé
187 /// val_finale : valeur de la fonction à "dernierTemps"
188 /// der_finale : dérivée finale
189 /// dernierdeltat : le dernier deltat utilisé
190 /// nombreAppelF : nombre d'appel de la fonction réellement utilisé dans le programme
191 /// erreurRelative : si retour = 3 ==> erreur relative (à la fonction) minimum possible,
192 /// nb_step : nombre de step utilisé pour le calcul (non compté les steps avec trop
d'erreur)
193 /// erreur_maxi_global : une approximation de l'erreur maxi global cumulant les erreurs sur tous les
steps
194 /// == en retour un entier qui donne le résultat du calcul : ----
195 /// =2 : tout est ok : seul cas où toutes les valeurs de retour sont valides
196 /// =3 : erreur: la précision relative demandée est trop petite, en retour la précision mini
possible
197 /// =4 : erreur: on a dépassé le nombre maxi d'appel fixé, d'où a peu près à (nombreAppelF/6) step
de calcul.
198 /// =6 : erreur: l'intégration pas possible, due aux précisions demandées, on doit augmenter ces
précisions
199 /// souvent du à une solution qui varie très rapidement --> problème potentiel
200 /// =8 : erreur: cas_kutta ne correspond pas à une routine de base existante
201 /// = 0 : erreur inconnue
202
203 template <class T>
204 int Pilotage_kutta(int cas_kutta, T& instance
205 ,Vecteur& (T::*Ptder_fonc) (const double & t,const Vecteur& f,Vecteur& df,int&
erreur)
206 ,void (T::*Ptverif_fonc) (const double & t,const Vecteur& f,int& erreur_final)
207 ,const Vecteur& val_initiale,const Vecteur& der_initiale
208 ,const double& tdebut,const double& tfin
209 ,const double& erreurAbsolue, double& erreurRelative
210 ,Vecteur& val_finale,Vecteur& der_finale
211 ,double& dernierTemps, double& dernierdeltat,int& nombreAppelF,int& nb_step
212 ,double& erreur_maxi_global);
213
214 /// explication erreur
215 void Affiche_Explication_erreur_RG(int type_erreur);
216
217 /// --méthodes pour modifier les différents paramètres
218 /// nombre maxi d'appel de fonction permis
219 /// dans le cas du runge 4-5 -> 6 appels par step (à la louche)
220 void Modif_nbMaxiAppel (int nb) {nbMaxiAppel = nb;};
221 /// initialisation de tous les paramètres à leurs valeurs par défaut
222 void Init_param_val_defaut();
223 /// modifie le niveau d'affichage par exemple pour le débog
224 void Change_niveau_affichage(int niveau) {permet_affichage=niveau;};
225
226 private :
227 // VARIABLES PROTEGEES :
228
229 // ---- controle de la sortie des informations
230 int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les erreurs
et warning
231 int nbMaxiAppel; // nombre maxi d'Appel autorisé de la fonction
232 // ---- variables intermédiaires utilisées par la méthode Runge_Kutta_step23
233 static double AB1,AB2,AB3,A2,aa2,aa3
234 ,bb21,bb31,bb32
235 ,dA1,dA2,dA3;
236 // ---- variables intermédiaires utilisées par la méthode Runge_Kutta_step34
237 static double A_B1,A_B3,A_B4,A_B5,A_1,A_3,A_4
238 ,a_a2,a_a3,a_a4,a_a5
239 ,b_b21,b_b31,b_b32,b_b41,b_b42,b_b43,b_b51,b_b53,b_b54
240 ,d_A1,d_A3,d_A4,d_A5;
241 // --- variables intermédiaires utilisées par la méthode Runge_Kutta_step45
242 Vecteur k2,k3,k4,k5,k6;

```

```

243 //   Vecteur f_inter;
244   static double a2,a3,a4,a5,a6 // les ai de Cash-karp paramètres
245           ,b21,b31,b32,b41,b42,b43 //
246           ,b51,b52,b53,b54,b61,b62,b63,b64,b65 // les bij
247           ,c1,c3,c4,c6; // les ci
248           // c2 =0, c5=0
249   static double ce1,ce3,ce4,ce5,ce6; // ce2=0; : les c-c_étoile_i
250 // --- variables intermédiaires utilisées par la méthode Pilotage_kutta
251   Vecteur val_inter; // valeur de la fonction, intermédiaire
252   Vecteur der_inter; // dérivée de la fonction, intermédiaire
253   Vecteur estime_erreur; // estimation d'erreur calculée
254 // puis des sauvegardes pour pouvoir revenir en arrière
255   Vecteur val_prec_inter; // sauve avant appel: valeur de la fonction, intermédiaire
256   Vecteur der_prec_inter; // sauve avant appel: dérivée de la fonction, intermédiaire
257   Vecteur estime_prec_erreur; // sauve avant appel: estimation d'erreur calculée
258
259   double coef; // coeff de sécurité préconnisé pour les variations de temps
260   static double remin; // pour calculer l'erreur mini acceptable dans le runge
261
262 // METHODES PROTEGEES :
263 // impression du temps
264 void timestamp ();
265 // -- méthode test ---
266 // on définit une fonction a intégrer
267 Vecteur& FoncDeriv(const double & t,const Vecteur& f,Vecteur& df,int& erreur);
268 // on définit une fonction de vérification d'intégrité
269 void FoncVerif_fonc(const double & t,const Vecteur& f,int& erreur);
270 int nb_cas_test; // nb pour choisir entre différents cas test
271 // la méthode qui fait les tests
272 void Test_template();
273
274 };
275
276 // pour faire de l'inline: nécessaire avec les templates
277 // on n'inclut que les méthodes templates
278 #include "Algo_edp_2.cc"
279 #define Algo_edp_deja_inclus
280 /// @} // end of group
281
282 #endif

```

## 7.504 Algo\_Integ1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           14/03/2008
31 *
32 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:         Herezh++
35 *
36 *   BUT:            Algorithme d'intégration 1D
37 *
38 *   *****
39 *
40 *   *****/
41 #ifndef ALGO_INTEG1D_H
42 #define ALGO_INTEG1D_H

```

```

43
44 #include "GeomSeg.h"
45
46 /// @addtogroup Les_classes_algo
47 /// @{
48 ///
49
50
51 /**
52 *
53 *   BUT: Algorithme d'intégration 1D
54 *
55 *
56 * \author   Gérard Rio
57 * \version  1.0
58 * \date     14/03/2008
59 * \brief    Algorithme d'intégration 1D
60 *
61 */
62
63
64 class Algo_Integ1D
65 {
66 public :
67     // CONSTRUCTEURS :
68     // constructeur par défaut
69     // nbptGauss : nombre de point de Gauss, utilisé pour l'intégration
70     Algo_Integ1D(int nbptGauss=2);
71     // constructeur de copie
72     Algo_Integ1D(const Algo_Integ1D& a);
73     // DESTRUCTEUR :
74     ~Algo_Integ1D() {};
75
76     // METHODES PUBLIQUES :
77     // intégration d'une fonction à l'aide de la méthode de Gauss
78     //
79     // en entrée:
80     // *Pt_fonc : pointeur de la fonction,
81     // tfin      : temps finale
82     // deltat   : plage d'intégration
83     // en sortie :
84     // renvoie la valeur de l'intégrale de t0 à t0+deltat
85     template <class T> double IntegGauss(const double& tfin, T& instance
86         ,double (T::*Pt_fonc) (const double & t)
87         ,const double& deltat) ;
88
89     // intégration d'une fonction à l'aide de la méthode de Gauss
90     // mais ici avec un intervalle fixé de -1 à 1
91     // en entrée:
92     // *Pt_fonc : pointeur de la fonction, qui est fonction d'une valeur qui
93     //           doit varier de -1 à 1 (donc il faut faire les interpolations nécessaires)
94     // deltat   : plage d'intégration
95     // en sortie :
96     // renvoie la valeur de l'intégrale de t0 à t0+deltat
97     template <class T> double IntegGauss( T& instance
98         ,double (T::*Pt_fonc) (const double & theta)
99         ,const double& deltat) ;
100
101 private :
102     // VARIABLES PROTEGEES :
103     GeomSeg seg; // l'élément de référence 1D
104
105     // METHODES PROTEGEES :
106
107 };
108 // pour faire de l'inline: nécessaire avec les templates
109 // on n'inclut que les méthodes templates
110 #include "Algo_Integ1D_2.cc"
111 #define Algo_Integ1D_deja_inclus
112 /// @} // end of group
113
114 #endif

```

## 7.505 Algo\_zero.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)

```

```

11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      11/10/2003
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *   *****
37 *   BUT:      Algorithmes de base pour la recherche de zéro d'une
38 *             fonction ou d'un ensemble de fonctions.
39 *
40 *             *****
41 *
42 *   *****/
43 #ifndef ALGO_ZERO_H
44 #define ALGO_ZERO_H
45
46 #include "Vecteur.h"
47 #include "Mat_abstraite.h"
48 #include "Racine.h"
49 #include <math.h>
50
51 /// @addtogroup Les_classes_algo
52 /// @{
53 ///
54
55 /**
56 *
57 *   BUT: pour la gestion d'exception pour non convergence
58 *
59 *
60 *
61 * \author   Gérard Rio
62 * \version  1.0
63 * \date    11/10/2003
64 * \brief   pour la gestion d'exception pour non convergence
65 *
66 */
67
68 class ErrNonConvergence_Newton
69 // =0 cas courant, pas d'information particulière
70 // =1 cas où l'erreur est sévère et ne pourra pas être corrigé en refaisant un calcul avec un
71 // pas de temps plus petit. Il faut refaire le calcul en se positionnant plusieurs pas de temps
72 // auparavant (utilisé par l'hystérésis par exemple)
73 { public :
74     int cas;
75     ErrNonConvergence_Newton () : cas(0) {} ; // par défaut
76     ErrNonConvergence_Newton (int ca) : cas(ca) {} ; // pb
77 };
78 /// @} // end of group
79
80 /// @addtogroup Les_classes_algo
81 /// @{
82 ///
83
84
85 /**
86 *
87 *   BUT:  Algorithmes de base pour la recherche de zéro d'une
88 *         fonction ou d'un ensemble de fonctions.
89 *
90 *
91 * \author   Gérard Rio
92 * \version  1.0
93 * \date    11/10/2003
94 * \brief   Algorithmes de base pour la recherche de zéro d'une fonction ou d'un ensemble de fonctions.
95 *
96 */
97

```

```

98 class Algo_zero
99 {
100 public :
101     // CONSTRUCTEURS :
102     /// constructeur par défaut
103     Algo_zero();
104     /// constructeur de copie
105     Algo_zero(const Algo_zero& a);
106     /// DESTRUCTEUR :
107     ~Algo_zero();
108
109     // METHODES PUBLIQUES :
110
111     /// recherche du zéro d'une équation du second degré dans l'espace des réels: ax^2+bx+c=0.
112     /// cas indique les différents cas:
113     /// = 1 : il y a deux racines , racine2 > racine1
114     /// = 2 : il y a une racine double
115     /// = 0 : pas de racine
116     /// = -1 : pas de racine car a,b,c sont tous inférieur à la précision de traitement
117     /// = -2 : pas de racine car a,b sont inférieur à la précision de traitement tandis que c est non
118     nul
119     /// = -3 : une racine simple car a est inférieur à la précision de traitement, donc considéré
120     /// comme nul
121     /// b et c sont non nul -> equa du premier degré, solution: racine1
122     void SecondDegre(double a, double b, double c, double& racine1, double& racine2, int& cas) const ;
123
124     /// recherche du zéro d'une équation du troisième degré dans l'espace des réels: ax^3+bx^2+cx+d=0.
125     /// cas indique les différents cas:
126     /// = 1 : il y a deux racines , racine2 > racine1
127     /// = 2 : il y a une racine double
128     /// = 3 : il y a une seule racine réelle racine1 (et deux complexes non calculées)
129     /// = 4 : il y a une racine simple: racine1, et une racine double: racine 2
130     /// = 5 : il y a 3 racines réelles: racine1, racine2, racine3
131     /// = 0 : pas de racine
132     /// = -1 : pas de racine car a,b,c sont tous inférieur à la précision de traitement
133     /// = -2 : pas de racine car a,b sont inférieur à la précision de traitement tandis que c est non
134     nul
135     /// = -3 : une racine simple car a est inférieur à la précision de traitement, donc considéré
136     /// comme nul
137     /// b et c sont non nul -> equa du premier degré, solution: racine1
138     void TroisiemeDegre(double a, double b, double c, double d, double& racine1
139     , double& racine2, double& racine3, int& cas) ;
140
141     ///recherche du zéro d'une fonction en utilisant la méthode de Newton-Raphson incrémentale
142     /// ici il s'agit d'une fonction à une variable
143     /// *Ptfonc : le pointeur de la fonction dont il faut chercher le zéro
144     /// *Ptder_fonc : pointeur de la dérivée de la fonction
145     /// pour ces deux fonctions, la variable alpha est un facteur de charge qui varie de
146     /// 0 à 1. Lorsque alpha=0, la racine vaut val_initiale, et ce que l'on
147     /// cherche c'est la racine pour alpha = 1. Lorsque alpha varie progressivement
148     /// de 0 à 1, la racine est sensée varier progressivement de val_initiale à résidu
149     /// l'utilisation d'alpha permet de faire du Newton incrémentale.
150     /// pour ces deux fonctions, l'argument test ramène
151     /// . 1 si le calcul a été ok, -1 s'il y a eu un pb, mais on peut continuer, 0 s'il y a eu un pb
152     /// fatal, qui invalide le calcul de la fonction et/ou de la dérivée
153     /// val_initiale : une valeur initiale de x pour la recherche de zéro
154     /// max_delta_x : si > 0 donne le norme maxi permise sur delta_x au cours d'une itération
155     /// si ||delta_x|| est plus grand on fait delta_x = max_delta_x * delta_x / ||
156     delta_x||
157     /// ramène en sortie:
158     /// un booléen qui indique si la résolution est correcte ou non
159     /// racine : la racine trouvée
160     /// der_at_racine : contient en retour la valeur de la dérivée pour la racine trouvée
161     /// nb_incr_total : le nombre total d'incrément qui a été nécessaire
162     /// nb_iter_total : le nombre total d'itération, qui cumule les iter de tous les incréments
163     template <class T> bool Newton_raphson
164     (T& instance,double (T::*Ptfonc) (double & alpha,double & x,int& test)
165     ,double (T::*Ptder_fonc) (double & alpha,double & x,int& test)
166     ,double val_initiale,double & racine,double & der_at_racine
167     ,int& nb_incr_total, int& nb_iter_total
168     ,double max_delta_x) const ;
169     /// idem précédemment, mais pour une fonction à valeur vectorielle
170     /// les matrices sont telles que: der_at_racine(i,j) = df(i)/d(x(j)), soit: df = der_at_racine * dx
171     template <class T> bool Newton_raphson
172     (T& instance,Vecteur& (T::*Ptfonc) (const double & alpha,const Vecteur & x,int& test)
173     ,Mat_abstraite& (T::*Ptder_fonc) (const double & alpha,const Vecteur & x,int& test)
174     ,const Vecteur& val_initiale,Vecteur & racine,Mat_abstraite & der_at_racine
175     ,int& nb_incr_total, int& nb_iter_total
176     ,double max_delta_x) const;
177     /// idem précédemment, mais pour une fonction à valeur vectorielle et ..
178     /// la méthode externe calcul la valeur de la fonction et de la dérivée en même temps
179     /// mais on utilise également la fonction résidu toute seule
180     /// les matrices sont telles que: der_at_racine(i,j) = df(i)/d(x(j)), soit: df = der_at_racine * dx
181     template <class T> bool Newton_raphson
182     (T& instance,Vecteur& (T::*Ptfonc) (const double & alpha,const Vecteur & x,int& test)
183     ,Mat_abstraite& (T::*Ptder_fonc) (const double & alpha,const Vecteur & x,Vecteur&
184     res,int& test)

```



```

179         ,const Vecteur& val_initiale,Vecteur & racine,Mat_abstraite & der_at_racine
180         ,int& nb_incr_total, int& nb_iter_total
181         ,double max_delta_x) const ;
182
183
184     /// méthodes pour modifier les différents paramètres
185     /// précision sur le résidu à convergence
186     void Modif_prec_res_abs (double eps) {prec_res_abs = eps;}; ///< précision absolue
187     void Modif_prec_res_rel (double eps) {prec_res_rel = eps;}; ///< précision relative
188     /// nombre d'itération maxi pour un incrément
189     void Modif_iter_max (double eps) { iter_max = eps;};
190     /// précision relative sur le mini de l'incrément de x
191     void Modif_coef_mini_delta_x (double eps) {coef_mini_delta_x = eps;};
192     /// minimum de delta x permis
193     void Modif_mini_delta_x (double eps) {mini_delta_x = eps;};
194     /// maximum de delta x permis
195     void Modif_maxi_delta_x (double eps) {maxi_delta_x = eps;};
196     /// nombre maxi d'incrément permis
197     void Modif_nbMaxiIncre (int nb) {nbMaxiIncre = nb;};
198     /// nombre maxi de test -1 permis
199     void Modif_nbMaxi_test_moins1 (int nb) {maxi_test_moins1 = nb;};
200     /// initialisation de tous les paramètres à leurs valeurs par défaut
201     void Init_param_val_defaut();
202     /// le niveau d'affichage
203     void Modif_affichage(int niveau) {permet_affichage = niveau;};
204
205     /// récup en lecture des différents paramètres
206     const double& Prec_res_abs() const {return prec_res_abs;}; ///< précision absolue sur le résidu à
    convergence
207     const double& Prec_res_rel() const {return prec_res_rel;}; ///< précision relative sur le résidu à
    convergence
208     const int& Iter_max() const {return iter_max;}; ///< nombre d'itération maxi pour un incrément
209     const double& Coef_mini_delta_x() const {return coef_mini_delta_x;}; ///< précision relative sur le
    mini de l'incrément de x
210     const double& Mini_delta_x() const {return mini_delta_x;}; ///< minimum de delta x permis
211     const double& Maxi_delta_x() const {return maxi_delta_x;}; ///< maximum de delta x permis
212     const int& NbMaxiIncre() const {return nbMaxiIncre;}; ///< nombre maxi d'incrément permis
213     const int& Maxi_test_moins1() const {return maxi_test_moins1;}; ///< nombre maxi de test -1 permis
214     const int& Permet_affichage() const {return permet_affichage;}; ///< le niveau d'affichage
215
216
217     /// affichage à l'écran des infos
218     void Affiche() const;
219
220     ///----- lecture écriture de restart -----
221     /// cas donne le niveau de la récupération
222     /// = 1 : on récupère tout
223     /// = 2 : on récupère uniquement les données variables (supposées comme telles)
224     void Lecture_base_info(ifstream& ent,const int cas);
225     /// cas donne le niveau de sauvegarde
226     /// = 1 : on sauvegarde tout
227     /// = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
228     void Ecriture_base_info(ofstream& sort,const int cas);
229
230
231 private :
232     // VARIABLES PROTEGEES :
233     double prec_res_abs; // précision absolue sur le résidu à convergence
234     double prec_res_rel; // précision relative sur le résidu à convergence
235     int iter_max; // nombre d'itération maxi pour un incrément
236     double coef_mini_delta_x; // précision relative sur le mini de l'incrément de x
237     double mini_delta_x; // minimum de delta x permis
238     double maxi_delta_x; // maximum de delta x permis
239     int nbMaxiIncre; // nombre maxi d'incrément permis
240     int maxi_test_moins1; // nombre maxi de test = -1 permis
241     // ----- controle de la sortie des informations
242     int permet_affichage; // pour permettre un affichage spécifique dans les méthodes,
243     // pour les erreurs et des warnings
244
245     Quartic algo_quartic; // une suite d'algo pour résoudre une quartic et des cubic
246
247
248     // METHODES PROTEGEES :
249
250 };
251
252 // pour faire de l'inline: nécessaire avec les templates
253 // on n'inclut que les méthodes templates
254 #include "Algo_zero_2.cc"
255 #define Algo_zero_deja_inclus
256 /// @} // end of group
257
258 #endif

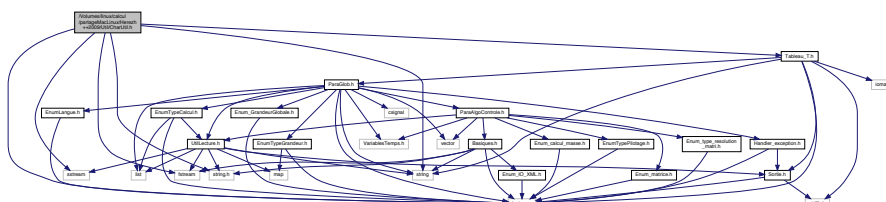
```

## 7.506 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↔ Herezh++2009/Util/CharUtil.h

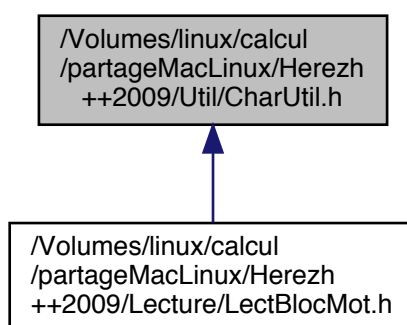
Utilitaires divers concernant les caracteres.

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string.h>
#include <string>
#include "Tableau_T.h"
```

Graphe des dépendances par inclusion de CharUtil.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



### Fonctions

- void **SortBlanc** (int n)  
*sortie de n blancs*
- void **Sortcar** (int n, char c)  
*sortie de n fois le meme caractere*
- int **ChangeEntier** (string st)  
*transformation d'un string ou d'une chaine de caractere en entier*
- int **ChangeEntier** (char \*st)
- double **ChangeReel** (string st)  
*transformation d'un string ou d'une chaine de caractere en relle*
- double **ChangeReel** (char \*st)
- string **ChangeEntierString** (int)  
*transformation d'un entier en chaine de caractere correspondant*
- string **ChangeReelString** (double a)  
*transformation d'un réel en chaine de caractère correspondant*

- char \* **ChangeEntierChar** (int)  
*transformation d'un entier de 0 a 10 en 1 caractere correspondant*
- string **Minuscules** (const string &st)  
*transformation d'un string en minuscules*
- bool **isNumeric** (const char \*pszInput, int nNumberBase)  
*test si un string est numérique ou pas*
- string **lect\_return\_default** (bool avec\_écriture, string val\_default)  
*----- utilitaire pour lire une chaine de caractères:*
- string **lect\_chaine** ()  
*lecture d'une chaine de caractère via getline: pas de validation tant qu'il n'y a rien de lue gestion d'erreur si lecture non correcte*
- string **lect\_o\_n** (bool avec\_écriture)  
*cas d'une valeur o/n , sinon pas acceptable si avec\_écriture = true: on écrit "--> valeur lue : " valeur lue*
- string **lect\_1\_0** (bool avec\_écriture)  
*cas d'une valeur 1/0 , sinon pas acceptable si avec\_écriture = true: on écrit "--> valeur lue : " valeur lue*
- double **lect\_double** ()  
*lecture d'un scalaire réel avec getline gestion d'erreur si lecture non correcte*
- char **Picococar** (ifstream &entr)  
*lecture du prochain caractère non espace et non vide dans le flot sans modifier le flot en remplacement de la méthode peek qui ne marche pas avec code warrior si pb ou aucun caractère, retour du caractère "espacement"*
- char **Picococar** (istream &entr)
- bool **ExisteString** (Tableau< string > &tabMot, string nom)  
*recherche "nom" dans le tableau "tabMot", si nom existe ramene true, sinon false*

## 7.506.1 Description détaillée

Utilitaires divers concernant les caracteres.

## 7.506.2 Documentation des fonctions

### 7.506.2.1 lect\_return\_default()

```
string lect_return_default (
    bool avec_écriture,
    string val_default )
```

----- utilitaire pour lire une chaine de caractères:

cas avec une valeur par défaut accepter si retour chariot, retourne la valeur par défaut passée en paramètre sinon retourne le string lue au clavier si avec\_écriture = true: on écrit "--> valeur par défaut : " val\_default gestion d'erreur si lecture non correcte

## 7.507 CharUtil.h

[Aller à la documentation de ce fichier.](#)

```
1 /*! \file CharUtil.h
2     \brief Utilitaires divers concernant les caracteres.
3 */
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997–2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
```

```

21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32 //
33 /*****
34 *   DATE:      23/01/97
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   *****
41 *   BUT: Utilitaires divers concernant les caracteres.
42 *
43 *   *****
44 *
45 *   *****/
46 #ifndef CHARUTIL_H
47 #define CHARUTIL_H
48
49 #include <iostream>
50 #include <fstream>
51 // #include "Debug.h"
52 #ifndef ENLINUX_STREAM
53 #include <sstream> // pour le flot en memoire centrale
54 #else
55 #include <strstream> // pour le flot en memoire centrale
56 #endif
57 // #ifndef SYSTEM_MAC_OS_X
58 // #include <stringfwd.h> // a priori ce n'est pas portable
59 // #el
60 #if defined SYSTEM_MAC_OS_CARBON
61 #include <stringfwd.h> // a priori ce n'est pas portable
62 #else
63 #include <string.h> // pour le flot en memoire centrale
64 #endif
65 #include <string>
66 // #include <bool.h>
67 #include "Tableau_T.h"
68
69 // // sortie de n blancs
70 inline void SortBlanc(int n)
71 { if (n>0) for (int i=1;i<= n; i++) cout << " ";
72 };
73 // // sortie de n fois le meme caratere
74 inline void Sortcar(int n,char c)
75 { if (n>0) for (int i=1;i<= n; i++) cout << c;
76 };
77 // // transformation d'un string ou d'une chaine de caractere en entier
78 int ChangeEntier(string st);
79 int ChangeEntier(char * st);
80 // // transformation d'un string ou d'une chaine de caractere en relle
81 double ChangeReel(string st);
82 double ChangeReel(char * st);
83 // // transformation d'un entier en chaine de caractere correspondant
84 string ChangeEntierString(int );
85 // // transformation d'un reel en chaine de caractere correspondant
86 string ChangeReelString(double a );
87 // // transformation d'un entier de 0 a 10 en 1 caractere correspondant
88 char* ChangeEntierChar(int );
89
90 // // transformation d'un string en minuscules
91 string Minuscules(const string& st);
92 // // test si un string est numerique ou pas
93 bool isNumeric( const char* pszInput, int nNumberBase );
94 // // ----- utilitaire pour lire une chaine de caracteres:
95
96 // // cas avec une valeur par defaut accepter
97 // // si retour chariot, retourne la valeur par defaut passee en parametre
98 // // sinon retourne le string lue au clavier
99 // // si avec_ecriture = true: on ecrit "--> valeur par defaut : " val_defaut
100 // // gestion d'erreur si lecture non correcte
101 string lect_return_defaut(bool avec_ecriture,string val_defaut);
102
103 // // lecture d'une chaine de caractere via getline: pas de validation
104 // // tant qu'il n'y a rien de lue
105 // // gestion d'erreur si lecture non correcte
106 string lect_chaine();
107

```

```

108 /// cas d'une valeur o/n , sinon pas acceptable
109 /// si avec_ecriture = true: on écrit "--> valeur lue : " valeur lue
110 string lect_o_n(bool avec_ecriture);
111
112 /// cas d'une valeur 1/0 , sinon pas acceptable
113 /// si avec_ecriture = true: on écrit "--> valeur lue : " valeur lue
114 string lect_1_0(bool avec_ecriture);
115
116 /// lecture d'un scalaire réel avec getline
117 /// gestion d'erreur si lecture non correcte
118 double lect_double();
119
120 /// lecture du prochain caractère non espace et non vide dans le flot sans modifier le flot
121 /// en remplacement de la méthode peek qui ne marche pas avec code warrior
122 /// si pb ou aucun caractère, retour du caractère "espacement"
123 char Picococar(istream& entr);
124 #ifndef ENLINUX_STREAM
125 char Picococar(istream& entr);
126 #else
127 char Picococar(istrstream& entr);
128 #endif
129
130 /// recherche "nom" dans le tableau "tabMot", si nom existe ramene true, sinon false
131 bool ExisteString (Tableau<string>& tabMot,string nom);
132
133 #endif

```

## 7.508 Courbe1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe1D.h
30 // classe : Courbe1D
31
32
33 /*****
34 *   DATE:      19/01/2001
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   *****/
41 *   BUT:      Classe virtuelle permettant le calcul d'une fonction 1D
42 *   ainsi qu'éventuellement un certain nombre d'information supplé-
43 *   mentaires telles que dérivées.
44 *   si le nom de la courbe = "_" il s'agit d'une courbe interne
45 *   à un objet, c'est-à-dire gérée seulement par l'entité qui la
46 *   contient, donc pas besoin de nom (elle n'est pas utilisée autre
47 *   part). Si le nom est différent de "_" c'est une courbe qui est
48 *   gérée et référencée dans LesCourbes, donc à partir de son nom,
49 *   on peut la retrouver.
50 *
51 *   *****
52 *
53 *   VERIFICATION:
54 *   ! date !   auteur !           but
55 *   -----

```

```

56 *      !      !      !      !      *
57 *      *      *      *      *      *
58 *      *      *      *      *      *
59 *      *      *      *      *      *
60 *      *      *      *      *      *
61 *      *      *      *      *      *
62 *      *      *      *      *      *
63 *      *      *      *      *      *
64
65 #ifndef COURBE_1_D_H
66 #define COURBE_1_D_H
67
68 #include "UtilLecture.h"
69 #include "EnumCourbe1D.h"
70 #include "Enum_IO_XML.h"
71
72 /** @defgroup Les_courbes_1D Les_courbes_1D
73 *
74 * \author Gérard Rio
75 * \version 1.0
76 * \date 19/01/2001
77 * \brief Def des courbes 1D
78 *
79 */
80
81 /// @addtogroup Les_courbes_1D
82 /// @{
83 ///
84
85 /**
86 *
87 * BUT: Classe virtuelle permettant le calcul d'une fonction 1D
88 * ainsi qu'éventuellement un certain nombre d'information supplé-
89 * mentaires telles que dérivées.
90 * si le nom de la courbe = "_" il s'agit d'une courbe interne
91 * à un objet, c'est-à-dire gérée seulement par l'entité qui la
92 * contient, donc pas besoin de nom (elle n'est pas utilisée autre
93 * part). Si le nom est différent de "_" c'est une courbe qui est
94 * gérée et référencée dans LesCourbes, donc à partir de son nom,
95 * on peut la retrouver.
96 *
97 *
98 * \author Gérard Rio
99 * \version 1.0
100 * \date 19/01/2001
101 * \brief Classe virtuelle permettant le calcul d'une fonction 1D ainsi qu'éventuellement un
102 * certain nombre d'information supplémentaires telles que dérivées.
103 */
104
105 class Courbe1D
106 {
107 public :
108     /// conteneur public pour une valeur et une dérivée
109     class ValDer
110     { public :
111         double valeur;
112         double derivee;
113     };
114     /// conteneur public pour une valeur et un booléen pour le strictement
115     /// inclus
116     class Valbool
117     { public :
118         double valeur;
119         bool dedans;
120     };
121     /// conteneur public pour une valeur et une dérivée et un booléen pour le strictement
122     /// inclus
123     class ValDerbool
124     { public:
125         double valeur;
126         double derivee;
127         bool dedans;
128     };
129     /// conteneur public pour une valeur, une dérivée première et une dérivée seconde
130     class ValDer2
131     { public :
132         double valeur;
133         double derivee;
134         double der_sec;
135     };
136
137     /// CONSTRUCTEURS :
138     /// par défaut
139     Courbe1D( string nom = "", EnumCourbe1D typ = AUCUNE_COURBE1D)
140     : nom_ref(nom), typeCourbe(typ), permet_affichage(0) {};
141     /// de copie

```

```

142 Courbe1D(const Courbe1D& Co) : nom_ref(Co.nom_ref),typeCourbe(Co.typeCourbe)
143 , permet_affichage(Co.permet_affichage) {};
144 // DESTRUCTEUR :
145 virtual ~Courbe1D() {};
146
147 // METHODES PUBLIQUES :
148
149 // ----- virtuelles -----
150
151 // affichage de la courbe
152 virtual void Affiche() const = 0;
153 // ramène le nom de la courbe
154 const string& NomCourbe() const {return nom_ref;};
155 // vérification que tout est ok, pres à l'emploi
156 // ramène true si ok, false sinon
157 virtual bool Complet_courbe()const =0 ;
158
159 // Lecture des donnees de la classe sur fichier
160 // le nom passé en paramètre est le nom de la courbe
161 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
162 // ce nom remplace le nom actuel
163 virtual void LectDonnParticulieres_courbes(const string& nom, UtilLecture * ) = 0;
164
165 // établir le lien entre la courbe et des courbes déjà existantes dont
166 // on connait que le nom
167 // permet ainsi de compléter la courbe
168 // 1) renseigne si la courbe dépend d'autre courbe ou non
169 virtual bool DependAutreCourbes() const {return false;}; ///< par défaut non
170 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
171 virtual list <string>& ListDependanceCourbes(list <string>& lico) const;
172 // 3) établit la connection entre la demande de *this et les courbes passées en paramètres
173 virtual void Lien_entre_courbe (list <Courbe1D *>& ) {};
174
175 // def info fichier de commande
176 virtual void Info_commande_Courbes1D(UtilLecture & entreePrinc) = 0;
177
178 // ramène la valeur
179 virtual double Valeur(double x) = 0;
180
181 // ramène la valeur et la dérivée en paramètre
182 virtual Courbe1D::ValDer Valeur_Et_derivee(double x) = 0;
183
184 // ramène la dérivée
185 virtual double Derivee(double x) = 0;
186
187 // ramène la valeur et les dérivées première et seconde en paramètre
188 virtual Courbe1D::ValDer2 Valeur_Et_der12(double x) = 0;
189
190 // ramène la dérivée seconde
191 virtual double Der_sec(double x) = 0;
192
193 // ramène la valeur si dans le domaine strictement de définition
194 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
195 // si supérieur au x maxi , ramène le valeur maximale possible de y
196 virtual Valbool Valeur_stricte(double x) = 0;
197
198 // ramène la valeur et la dérivée si dans le domaine strictement de définition
199 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
200 // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
201 virtual ValDerbool Valeur_Et_derivee_stricte(double x) = 0;
202
203 //---- lecture écriture de restart ----
204 // cas donne le niveau de la récupération
205 // = 1 : on récupère tout
206 // = 2 : on récupère uniquement les données variables (supposées comme telles)
207 virtual void Lecture_base_info(ifstream& ent,const int cas) = 0;
208 // cas donne le niveau de sauvegarde
209 // = 1 : on sauvegarde tout
210 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
211 virtual void Ecriture_base_info(ofstream& sort,const int cas) = 0;
212 // sortie du schemaXML: en fonction de enu
213 virtual void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) = 0;
214
215 // ----- static -----
216
217 // ramène un pointeur sur la courbe correspondant au type de courbe passé en paramètre
218 // IMPORTANT : il y a création d'une courbe (utilisation d'un new)
219 static Courbe1D* New_Courbe1D(string& nom,EnumCourbe1D typeCourbe);
220 // ramène un pointeur sur une courbe copie de celle passée en paramètre
221 // IMPORTANT : il y a création d'une courbe (utilisation d'un new)
222 static Courbe1D* New_Courbe1D(const Courbe1D& Co);
223
224 // ramène la liste des identificateurs de courbes actuellement disponibles
225 static list <EnumCourbe1D> Liste_courbe_disponible();
226
227 // ----- non virtuelle -----
228

```

```

229     /// ramène le type de la courbe
230     EnumCourbeId Type_courbe() const { return typeCourbe;};
231
232     protected :
233
234     // VARIABLES PROTEGEES :
235     EnumCourbeId typeCourbe; // type de la courbe
236     string nom_ref; // nom de référence de la courbe
237
238     // ---- controle de la sortie des informations: utilisé par les classes dérivées
239     int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les erreurs
    et warning
240
241     // METHODES PROTEGEES :
242     // ramène true si les variables de la classe mère sont complétées
243     bool Complet_var() const;
244 };
245 /// @} // end of group
246
247 #endif

```

## 7.509 Courbe\_cos.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_cos.h
30 // classe : Courbe_cos
31
32
33 /*****
34 *      DATE:      19/01/2001
35 *
36 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *****/
41 *      BUT:      Classe permettant le calcul d'une fonction 1D
42 *              de type : ampli*cos(alpha*x+beta)
43 *              avec un mini et/ou maxi éventuels
44 *              ainsi qu'un certain nombre d'information
45 *              supplémentaires telles que dérivées.
46 *
47 *      *****
48 *
49 *      VERIFICATION:
50 *      ! date ! auteur ! but
51 *      -----
52 *      ! ! !
53 *      $
54 *      *****
55 *      MODIFICATIONS:
56 *      ! date ! auteur ! but
57 *      -----
58 *      $
59 *****/
60
61 #ifndef COURBECOS_1_D_H

```



```

62 #define COURBECOS_1_D_H
63
64 #include "CourbelD.h"
65
66 /// @addtogroup Les_courbes_1D
67 /// @{
68 ///
69
70 /**
71 *
72 *   BUT:   Classe permettant le calcul d'une fonction 1D
73 *         de type : ampli*cos(alpha*x+beta)
74 *         avec un mini et/ou maxi éventuels
75 *         ainsi qu'un certain nombre d'information
76 *         supplémentaires telles que dérivées.
77 *
78 *
79 * \author   Gérard Rio
80 * \version  1.0
81 * \date     19/01/2001
82 * \brief    Classe permettant le calcul d'une fonction 1D de type : ampli*cos(alpha*x+beta)
83 *
84 */
85 class Courbe_cos : public CourbelD
86 {
87 public :
88
89     // CONSTRUCTEURS :
90     Courbe_cos(string nom = "");
91
92     // de copie
93     Courbe_cos(const Courbe_cos& Co);
94     Courbe_cos(const CourbelD& Co);
95
96     // DESTRUCTEUR :
97     ~Courbe_cos();
98
99     // METHODES PUBLIQUES :
100
101     // ----- virtuelles -----
102
103     // affichage de la courbe
104     void Affiche() const ;
105     // ramène true si ok, false sinon
106     bool Complet_courbe()const;
107
108     // Lecture des donnees de la classe sur fichier
109     // le nom passé en paramètre est le nom de la courbe
110     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
111     // ce nom remplace le nom actuel
112     void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
113
114     // def info fichier de commande
115     void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
116
117     // ramène la valeur
118
119     double Valeur(double x) ;
120
121     // ramène la valeur et la dérivée en paramètre
122     CourbelD::ValDer Valeur_Et_derivee(double x) ;
123
124     // ramène la dérivée
125     double Derivee(double x) ;
126
127     // ramène la valeur et les dérivées première et seconde en paramètre
128     CourbelD::ValDer2 Valeur_Et_der12(double x);
129
130     // ramène la dérivée seconde
131     double Der_sec(double x);
132
133     // ramène la valeur si dans le domaine strictement de définition
134     // si c'est inférieur au x mini, ramène la valeur minimale possible de y
135     // si supérieur au x maxi , ramène le valeur maximale possible de y
136     CourbelD::Valbool Valeur_stricte(double x) ;
137
138     // ramène la valeur et la dérivée si dans le domaine strictement de définition
139     // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
140     // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
141     CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
142
143     //----- lecture écriture de restart -----
144     // cas donne le niveau de la récupération
145     // = 1 : on récupère tout
146     // = 2 : on récupère uniquement les données variables (supposées comme telles)
147     void Lecture_base_info(ifstream& ent,const int cas);
148     // cas donne le niveau de sauvegarde

```

```

149 // = 1 : on sauvegarde tout
150 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
151 void Ecriture_base_info(ofstream& sort,const int cas);
152 // sortie du schemaXML: en fonction de enu
153 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
154
155 protected :
156 // VARIABLES PROTEGEES :
157 double ax,bx; // min max
158 bool en_degre; // indique si l'on est en radian ou degré
159 double ampli, alpha, bet; // coefs de la fonction
160 // METHODES PROTEGEES :
161
162 };
163 /// @} // end of group
164
165 #endif

```

## 7.510 Courbe\_expo2\_n.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_expo2_n.h
30 // classe : Courbe_expo2_n
31
32
33 /*****
34 *      DATE:      06/04/2008
35 *
36 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *****/
41 *      BUT:      Classe permettant le calcul d'une fonction 1D 1D
42 *              de type : (gamma+alpha*(x*x)^n)
43 *              ainsi qu'un certain nombre d'information
44 *              supplémentaires telles que dérivées.
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *
50 *      ! date ! auteur ! but
51 *      -----
52 *      ! ! !
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *
58 *****/
59
60 #ifndef COURBEEEXO2_N_1_D_H
61 #define COURBEEEXO2_N_1_D_H
62
63 #include "Courbe1D.h"
64

```

```

65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 *   BUT:   Classe permettant le calcul d'une fonction 1D   1D
72 *         de type : (gamma+alpha*(x*x)^n)
73 *         ainsi qu'un certain nombre d'information
74 *         supplémentaires telles que dérivées.
75 *
76 *
77 * \author   Gérard Rio
78 * \version  1.0
79 * \date     06/04/2008
80 * \brief    Classe permettant le calcul d'une fonction 1D de type : (gamma+alpha*(x*x)^n)
81 *
82 */
83 class Courbe_expo2_n : public CourbelD
84 {
85 public :
86
87     // CONSTRUCTEURS :
88     Courbe_expo2_n(string nom = "");
89
90     // de copie
91     Courbe_expo2_n(const Courbe_expo2_n& Co);
92     Courbe_expo2_n(const CourbelD& Co);
93
94     // DESTRUCTEUR :
95     ~Courbe_expo2_n();
96
97     // METHODES PUBLIQUES :
98
99     // ----- virtuelles -----
100
101     // affichage de la courbe
102     void Affiche() const ;
103     // ramène true si ok, false sinon
104     bool Complet_courbe()const;
105
106     // Lecture des donnees de la classe sur fichier
107     // le nom passé en paramètre est le nom de la courbe
108     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
109     // ce nom remplace le nom actuel
110     void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
111
112     // def info fichier de commande
113     void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
114
115     // ramène la valeur
116
117     double Valeur(double x) ;
118
119     // ramène la valeur et la dérivée en paramètre
120     CourbelD::ValDer Valeur_Et_derivee(double x) ;
121
122     // ramène la dérivée
123     double Derivee(double x) ;
124
125     // ramène la valeur et les dérivées première et seconde en paramètre
126     CourbelD::ValDer2 Valeur_Et_der12(double x);
127
128     // ramène la dérivée seconde
129     double Der_sec(double x);
130
131     // ramène la valeur si dans le domaine strictement de définition
132     // si c'est inférieur au x mini, ramène la valeur minimale possible de y
133     // si supérieur au x maxi , ramène le valeur maximale possible de y
134     CourbelD::Valbool Valeur_stricte(double x) ;
135
136     // ramène la valeur et la dérivée si dans le domaine strictement de définition
137     // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
138     // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
139     CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
140
141     //----- lecture écriture de restart -----
142     // cas donne le niveau de la récupération
143     // = 1 : on récupère tout
144     // = 2 : on récupère uniquement les données variables (supposées comme telles)
145     void Lecture_base_info(ifstream& ent,const int cas);
146     // cas donne le niveau de sauvegarde
147     // = 1 : on sauvegarde tout
148     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
149     void Ecriture_base_info(ofstream& sort,const int cas);
150     // sortie du schemaXML: en fonction de enu
151     void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;

```

```

152
153 protected :
154 // VARIABLES PROTEGEES :
155 double alpha,xn,gamma; // les coefficients de la loi
156
157 // METHODES PROTEGEES :
158
159 };
160 /// @} // end of group
161
162 #endif

```

## 7.511 Courbe\_expo\_n.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_expo_n.h
30 // classe : Courbe_expo_n
31
32
33 /*****
34 * DATE: 19/01/2001 *
35 * *
36 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) $ *
37 * *
38 * PROJET: Herezh++ *
39 * *
40 *****/
41 * BUT: Classe permettant le calcul d'une fonction 1D 1D *
42 * de type : (gamma+alpha*(x)^n) *
43 * ainsi qu'un certain nombre d'information *
44 * supplémentaires telles que dérivées. *
45 * $ *
46 * ***** *
47 * *
48 * VERIFICATION: *
49 * ! date ! auteur ! but ! *
50 * ----- ! *
51 * ! ! ! ! *
52 * $ *
53 * ***** *
54 * MODIFICATIONS: *
55 * ! date ! auteur ! but ! *
56 * ----- ! *
57 * $ *
58 *****/
59
60 #ifndef COURBEEEXO_N_1_D_H
61 #define COURBEEEXO_N_1_D_H
62
63 #include "Courbe1D.h"
64
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *

```

```

71 *     BUT:     Classe permettant le calcul d'une fonction 1D
72 *           de type :
73 *     f(x) = (gamma+alpha*(x)^n)
74 *           ainsi qu'un certain nombre d'information
75 *           supplémentaires telles que dérivées.
76 *
77 *
78 * \author     Gérard Rio
79 * \version    1.0
80 * \date      19/01/2001
81 * \brief     Classe permettant le calcul d'une fonction 1D de type : f(x) = (gamma+alpha*(x)^n)
82 *
83 */
84 class Courbe_expo_n : public CourbelD
85 {
86 public :
87     // CONSTRUCTEURS :
88     Courbe_expo_n(string nom = "");
89
90     // de copie
91     Courbe_expo_n(const Courbe_expo_n& Co);
92     Courbe_expo_n(const CourbelD& Co);
93
94     // DESTRUCTEUR :
95     ~Courbe_expo_n();
96
97     // METHODES PUBLIQUES :
98
99     // ----- virtuelles -----
100
101     // affichage de la courbe
102     void Affiche() const ;
103     // ramène true si ok, false sinon
104     bool Complet_courbe()const;
105
106     // Lecture des donnees de la classe sur fichier
107     // le nom passé en paramètre est le nom de la courbe
108     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
109     // ce nom remplace le nom actuel
110     void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
111
112     // def info fichier de commande
113     void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
114
115     // ramène la valeur
116
117     double Valeur(double x) ;
118
119     // ramène la valeur et la dérivée en paramètre
120     CourbelD::ValDer Valeur_Et_derivee(double x) ;
121
122     // ramène la dérivée
123     double Derivee(double x) ;
124
125     // ramène la valeur et les dérivées première et seconde en paramètre
126     CourbelD::ValDer2 Valeur_Et_der12(double x);
127
128     // ramène la dérivée seconde
129     double Der_sec(double x);
130
131     // ramène la valeur si dans le domaine strictement de définition
132     // si c'est inférieur au x mini, ramène la valeur minimale possible de y
133     // si supérieur au x maxi , ramène la valeur maximale possible de y
134     CourbelD::Valbool Valeur_stricte(double x) ;
135
136     // ramène la valeur et la dérivée si dans le domaine strictement de définition
137     // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
138     // si supérieur au x maxi , ramène la valeur maximale possible de y et Y' correspondant
139     CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
140
141     //----- lecture écriture de restart -----
142     // cas donne le niveau de la récupération
143     // = 1 : on récupère tout
144     // = 2 : on récupère uniquement les données variables (supposées comme telles)
145     void Lecture_base_info(ifstream& ent,const int cas);
146     // cas donne le niveau de sauvegarde
147     // = 1 : on sauvegarde tout
148     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
149     void Ecriture_base_info(ofstream& sort,const int cas);
150     // sortie du schemaXML: en fonction de enu
151     void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
152
153 protected :
154     // VARIABLES PROTEGEES :
155     double alpha,xn,gamma; // les coefficients de la loi
156
157

```

```

158 // METHODES PROTEGEES :
159
160 };
161 /// @} // end of group
162
163 #endif

```

## 7.512 Courbe\_expoaff.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_expoaff.h
30 // classe : Courbe_expoaff
31
32
33 /*****
34 *   DATE:      19/01/2001
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *****/
41 *   BUT:      Classe permettant le calcul d'une fonction 1D 1D
42 *             de type :  $\gamma + \alpha * (|x|^n)$ 
43 *             ainsi qu'un certain nombre d'information
44 *             supplémentaires telles que dérivées.
45 *
46 *             *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !           !           !
53 *
54 *             *****
55 *   MODIFICATIONS:
56 *
57 *   ! date !   auteur !           but
58 *   -----
59 *
60 *****/
61 #ifndef COURBEEEXPOAFF_1_D_H
62 #define COURBEEEXPOAFF_1_D_H
63 #include "Courbe1D.h"
64
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 *   BUT:      Classe permettant le calcul d'une fonction 1D
72 *             de type :
73 *              $f(x) = \gamma + \alpha * (|x|^n)$ 
74 *             ainsi qu'un certain nombre d'information
75 *             supplémentaires telles que dérivées.

```

```

76 *
77 *
78 * \author   Gérard Rio
79 * \version  1.0
80 * \date    19/01/2001
81 * \brief   Classe permettant le calcul d'une fonction 1D de type :  $f(x) = \gamma + \alpha \cdot (|x|^n)$ 
82 *
83 */
84 class Courbe_expoaff : public CourbelD
85 {
86     public :
87
88     // CONSTRUCTEURS :
89     Courbe_expoaff(string nom = "");
90
91     // de copie
92     Courbe_expoaff(const Courbe_expoaff& Co);
93     Courbe_expoaff(const CourbelD& Co);
94
95     // DESTRUCTEUR :
96     ~Courbe_expoaff();
97
98     // METHODES PUBLIQUES :
99
100    // ----- virtuelles -----
101
102    // affichage de la courbe
103    void Affiche() const ;
104    // ramène true si ok, false sinon
105    bool Complet_courbe()const;
106
107    // Lecture des donnees de la classe sur fichier
108    // le nom passé en paramètre est le nom de la courbe
109    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
110    // ce nom remplace le nom actuel
111    void LectDonnParticulieres(const string& nom, UtilLecture * );
112
113    // def info fichier de commande
114    void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
115
116    // ramène la valeur
117
118    double Valeur(double x) ;
119
120    // ramène la valeur et la dérivée en paramètre
121    CourbelD::ValDer Valeur_Et_derivee(double x) ;
122
123    // ramène la dérivée
124    double Derivee(double x) ;
125
126    // ramène la valeur et les dérivées première et seconde en paramètre
127    CourbelD::ValDer2 Valeur_Et_der12(double x);
128
129    // ramène la dérivée seconde
130    double Der_sec(double x);
131
132    // ramène la valeur si dans le domaine strictement de définition
133    // si c'est inférieur au x mini, ramène la valeur minimale possible de y
134    // si supérieur au x maxi , ramène le valeur maximale possible de y
135    CourbelD::Valbool Valeur_stricte(double x) ;
136
137    // ramène la valeur et la dérivée si dans le domaine strictement de définition
138    // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
139    // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
140    CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
141
142    //----- lecture écriture de restart -----
143    // cas donne le niveau de la récupération
144    // = 1 : on récupère tout
145    // = 2 : on récupère uniquement les données variables (supposées comme telles)
146    void Lecture_base_info(ifstream& ent,const int cas);
147    // cas donne le niveau de sauvegarde
148    // = 1 : on sauvegarde tout
149    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
150    void Ecriture_base_info(ofstream& sort,const int cas);
151    // sortie du schemaXML: en fonction de enu
152    void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
153
154
155
156
157     protected :
158     // VARIABLES PROTEGEES :
159     double alpha,xn,gamma; // les coefficients de la loi
160
161     // METHODES PROTEGEES :
162

```

```

163 };
164 /// @} // end of group
165
166 #endif

```

## 7.513 Courbe\_expression\_litterale\_1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_expression_litterale_1D.h
30 // classe : Courbe_expression_litterale_1D
31
32
33 /*****
34 *   DATE:           07/04/2014
35 *
36 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:         Herezh++
39 *
40 *
41 *   BUT:           Classe permettant le calcul d'une fonction 1D
42 *                 de type : une expression littérale, exploitée ensuite
43 *                 par le parser : muparser
44 *                 avec un mini et/ou maxi éventuels
45 *                 ainsi qu'un certain nombre d'information
46 *                 supplémentaires telles que dérivées.
47 *                 IMPORTANT: les dérivées sont calculées de manière
48 *                 numérique.
49 *
50 *   *****
51 *
52 *   VERIFICATION:
53 *
54 *   ! date ! auteur ! but
55 *   -----
56 *   ! ! !
57 *
58 *   *****
59 *   MODIFICATIONS:
60 *   ! date ! auteur ! but
61 *   -----
62 *
63 *   *****/
64
65 #ifndef COURBE_EXPRESSION_LITTERALE_1D_H
66 #define COURBE_EXPRESSION_LITTERALE_1D_H
67
68 #include "Courbe1D.h"
69 #include "muParser.h"
70
71
72 /// @addtogroup Les_courbes_1D
73 /// @{
74 ///
75
76 /**
77 *
78 *   BUT:           Classe permettant le calcul d'une fonction 1D

```



```
79 *           de type : une expression littérale, exploitée ensuite
80 *           par le parser : muparser
81 *           avec un mini et/ou maxi éventuels
82 *           ainsi qu'un certain nombre d'information
83 *           supplémentaires telles que dérivées.
84 *           IMPORTANT: les dérivées sont calculées de manière
85 *           numérique.
86 *
87 *
88 * \author    Gérard Rio
89 * \version  1.0
90 * \date     07/04/2014
91 * \brief    Classe permettant le calcul d'une fonction 1D de type : une expression littérale
92 *
93 */
94
95 class Courbe_expression_litterale_1D : public Courbe1D
96 {
97 public :
98
99     // CONSTRUCTEURS :
100    Courbe_expression_litterale_1D(string nom = "");
101
102    // de copie
103    Courbe_expression_litterale_1D(const Courbe_expression_litterale_1D& Co);
104    Courbe_expression_litterale_1D(const Courbe1D& Co);
105
106    // DESTRUCTEUR :
107    ~Courbe_expression_litterale_1D();
108
109    // METHODES PUBLIQUES :
110
111    // ----- virtuelles -----
112
113    // affichage de la courbe
114    void Affiche() const ;
115    // ramène true si ok, false sinon
116    bool Complet_courbe()const;
117
118    // Lecture des donnees de la classe sur fichier
119    // le nom passé en paramètre est le nom de la courbe
120    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
121    // ce nom remplace le nom actuel
122    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
123
124    // def info fichier de commande
125    void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
126
127    // ramène la valeur
128
129    double Valeur(double x) ;
130
131    // ramène la valeur et la dérivée en paramètre
132    Courbe1D::ValDer Valeur_Et_derivee(double x) ;
133
134    // ramène la dérivée
135    double Derivee(double x) ;
136
137    // ramène la valeur et les dérivées première et seconde en paramètre
138    Courbe1D::ValDer2 Valeur_Et_der12(double x);
139
140    // ramène la dérivée seconde
141    double Der_sec(double x);
142
143    // ramène la valeur si dans le domaine strictement de définition
144    // si c'est inférieur au x mini, ramène la valeur minimale possible de y
145    // si supérieur au x maxi , ramène la valeur maximale possible de y
146    Courbe1D::Valbool Valeur_stricte(double x) ;
147
148    // ramène la valeur et la dérivée si dans le domaine strictement de définition
149    // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
150    // si supérieur au x maxi , ramène la valeur maximale possible de y et Y' correspondant
151    Courbe1D::ValDerbool Valeur_Et_derivee_stricte(double x) ;
152
153    //----- lecture écriture de restart -----
154    // cas donne le niveau de la récupération
155    // = 1 : on récupère tout
156    // = 2 : on récupère uniquement les données variables (supposées comme telles)
157    void Lecture_base_info(ifstream& ent,const int cas);
158    // cas donne le niveau de sauvegarde
159    // = 1 : on sauvegarde tout
160    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
161    void Ecriture_base_info(ofstream& sort,const int cas);
162    // sortie du schemaXML: en fonction de enu
163    void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
164
165 protected :
```

```

166 // VARIABLES PROTEGEES :
167 double ax,bx; // min max
168
169 string expression_fonction; // l'expression littérale de la fonction
170 mu::Parser p; // définition d'une instance de parser
171 double fVal; // la variable interne
172 double delta_xSur_x; // sert pour l'incrément pour le calcul des dérivées
173 // int ordre_troncature; // l'ordre de troncature: 2 par défaut
174
175
176 // METHODES PROTEGEES :
177 // calcul de la dérivée première et seconde par différence finie à l'ordres 4
178 // void Der_1_2(const double & x, double& der1, double& der2);
179 // muParser fourni déjà une dérivée à l'ordre 4 et on pourrait également adjoindre une dérivée
180 // seconde à l'ordre 4 cf. Numerical algorithms with C : page 356
181 // Gisela Engeln-Müllges et Frank Uhlig, Springer, ISBN 3_540-60530-4
182
183
184
185
186
187 };
188 /// @} // end of group
189
190 #endif

```

## 7.514 Courbe\_expression\_litterale\_avec\_derivees\_1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_expression_litterale_avec_derivees_1D.h
30 // classe : Courbe_expression_litterale_avec_derivees_1D
31
32
33 /*****
34 *      DATE:      07/04/2014
35 *
36 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *****/
41 *      BUT:      Classe permettant le calcul d'une fonction 1D
42 *               de type : une expression littérale, exploitée ensuite
43 *               par le parser : muparser
44 *               avec un mini et/ou maxi éventuels
45 *               ainsi qu'un certain nombre d'information
46 *               supplémentaires telles que dérivées.
47 *
48 *      *****
49 *
50 *      VERIFICATION:
51 *
52 *      ! date !   auteur !           but
53 *      -----
54 *      !           !           !
55 *
56 *      *****
57 *      MODIFICATIONS:

```

```

58 *      ! date !   auteur !           but           !      *
59 *      -----
60 *                                     $      *
61 *****/
62
63 #ifndef COURBE_EXPRESSION_LITTERALE_ET_DERIVEES_1_D_H
64 #define COURBE_EXPRESSION_LITTERALE_ET_DERIVEES_1_D_H
65
66 #include "Courbe1D.h"
67 #include "muParser.h"
68
69 /// @addtogroup Les_courbes_1D
70 /// @{
71 ///
72
73 /**
74 *
75 *      BUT:      Classe permettant le calcul d'une fonction 1D
76 *                de type : une expression littérale, exploitée ensuite
77 *                par le parser : muparser
78 *                avec un mini et/ou maxi éventuels
79 *                ainsi qu'un certain nombre d'information
80 *                supplémentaires telles que dérivées.
81 *
82 *
83 * \author      Gérard Rio
84 * \version     1.0
85 * \date       07/04/2014
86 * \brief      Classe permettant le calcul d'une fonction 1D et de sa dérivée de type : une expression
87 *                littérale
88 */
89
90 class Courbe_expression_litterale_avec_derivees_1D : public Courbe1D
91 {
92 public :
93
94     // CONSTRUCTEURS :
95     Courbe_expression_litterale_avec_derivees_1D(string nom = "");
96
97     // de copie
98     Courbe_expression_litterale_avec_derivees_1D(const Courbe_expression_litterale_avec_derivees_1D& Co);
99     Courbe_expression_litterale_avec_derivees_1D(const Courbe1D& Co);
100
101     // DESTRUCTEUR :
102     ~Courbe_expression_litterale_avec_derivees_1D();
103
104     // METHODES PUBLIQUES :
105
106     // ----- virtuelles -----
107
108     // affichage de la courbe
109     void Affiche() const ;
110     // ramène true si ok, false sinon
111     bool Complet_courbe()const;
112
113     // Lecture des donnees de la classe sur fichier
114     // le nom passé en paramètre est le nom de la courbe
115     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
116     // ce nom remplace le nom actuel
117     void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
118
119     // def info fichier de commande
120     void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
121
122     // ramène la valeur
123
124     double Valeur(double x) ;
125
126     // ramène la valeur et la dérivée en paramètre
127     Courbe1D::ValDer Valeur_Et_derivee(double x) ;
128
129     // ramène la dérivée
130     double Derivee(double x) ;
131
132     // ramène la valeur et les dérivées première et seconde en paramètre
133     Courbe1D::ValDer2 Valeur_Et_der12(double x);
134
135     // ramène la dérivée seconde
136     double Der_sec(double x);
137
138     // ramène la valeur si dans le domaine strictement de définition
139     // si c'est inférieur au x mini, ramène la valeur minimale possible de y
140     // si supérieur au x maxi , ramène la valeur maximale possible de y
141     Courbe1D::Valbool Valeur_stricte(double x) ;
142
143     // ramène la valeur et la dérivée si dans le domaine strictement de définition

```

```

144 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
145 // si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant
146 CourbeID::ValDerbool Valeur_Et_derivee_stricte(double x) ;
147
148 //----- lecture écriture de restart -----
149 // cas donne le niveau de la récupération
150 // = 1 : on récupère tout
151 // = 2 : on récupère uniquement les données variables (supposées comme telles)
152 void Lecture_base_info(ifstream& ent,const int cas);
153 // cas donne le niveau de sauvegarde
154 // = 1 : on sauvegarde tout
155 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
156 void Ecriture_base_info(ofstream& sort,const int cas);
157 // sortie du schemaXML: en fonction de enu
158 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
159
160 protected :
161 // VARIABLES PROTEGEES :
162 double ax,bx; // min max
163
164 string expression_fonction; // l'expression littérale de la fonction
165 mu::Parser p; // définition d'une instance de parser
166 double fVal; // la variable interne
167 string expression_der_fonct; // l'expression de la dérivée
168 mu::Parser der_p; // définition de la dérivée
169 double fVal_der; // la variable interne
170 string expression_der2_fonct; // l'expression de la dérivée deuxième
171 mu::Parser der2_p; // définition de la dérivée seconde
172 double fVal_der2; // la variable interne
173
174 // int ordre_troncature; // l'ordre de troncature: 2 par défaut
175
176
177 // METHODES PROTEGEES :
178 // calcul de la dérivée première et seconde par différence finie à l'ordres 4
179 // void Der_1_2(const double & x, double& der1, double& der2);
180 // muParser fourni déjà une dérivée à l'ordre 4 et on pourrait également adjoindre une dérivée
181 // seconde à l'ordre 4 cf. Numerical algorithms with C : page 356
182 // Gisela Engeln-Müllges et Frank Uhlig, Springer, ISBN 3_540-60530-4
183
184
185
186
187
188 };
189 /// @} // end of group
190
191 #endif

```

## 7.515 Courbe\_In\_cosh.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_ln_cosh.h
30 // classe : Courbe_ln_cosh
31 /*****
32 * DATE: 03/10/2014 *
33 * * *
34 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *

```

```

35 *                                     $ *
36 *   PROJET:      Herezh++                                     *
37 *                                     $ *
38 *****
39 *   BUT:      Classe permettant le calcul d'une fonction 1D *
40 *             de type : *
41 *   f(x) = al*(be +ga*x)^n * ln (cosh(de*(he+ee*x)))+ ke*(ge+re*x) *
42 *             avec un mini et/ou maxi éventuels *
43 *             ainsi qu'un certain nombre d'information *
44 *             supplémentaires telles que dérivées. *
45 *                                     $ *
46 *   ////////////////////////////////////////////////// *
47 * *
48 *   VERIFICATION: *
49 * *
50 *   ! date !   auteur !   but *
51 *   ----- *
52 *   !       !       !       ! *
53 *   ////////////////////////////////////////////////// $ *
54 * *
55 *   MODIFICATIONS: *
56 *   ! date !   auteur !   but *
57 *   ----- *
58 *   ////////////////////////////////////////////////// $ *
59 *****/
60
61 #ifndef COURBE_LN_COSH_1_D_H
62 #define COURBE_LN_COSH_1_D_H
63
64 #include "Courbe1D.h"
65
66 /// @addtogroup Les_courbes_1D
67 /// @{
68 ///
69
70 /**
71 *
72 *   BUT:      Classe permettant le calcul d'une fonction 1D
73 *             de type :
74 *   f(x) = al*(be +ga*x)^n * ln (cosh(de*(he+ee*x)))+ ke*(ge+re*x)
75 *             avec un mini et/ou maxi éventuels
76 *             ainsi qu'un certain nombre d'information
77 *             supplémentaires telles que dérivées.
78 *
79 *
80 * \author      Gérard Rio
81 * \version    1.0
82 * \date       19/01/2001
83 * \brief      Classe permettant le calcul d'une fonction 1D de type : f(x) = al*(be +ga*x)^n * ln
84 *             (cosh(de*(he+ee*x)))+ ke*(ge+re*x)
85 */
86 class Courbe_ln_cosh : public Courbe1D
87 {
88 public :
89
90     // CONSTRUCTEURS :
91     Courbe_ln_cosh(string nom = "");
92
93     // de copie
94     Courbe_ln_cosh(const Courbe_ln_cosh& Co);
95     Courbe_ln_cosh(const Courbe1D& Co);
96
97     // DESTRUCTEUR :
98     ~Courbe_ln_cosh();
99
100    // METHODES PUBLIQUES :
101
102    // ----- virtuelles -----
103
104    // affichage de la courbe
105    void Affiche() const ;
106    // ramène true si ok, false sinon
107    bool Complet_courbe()const;
108
109    // Lecture des donnees de la classe sur fichier
110    // le nom passé en paramètre est le nom de la courbe
111    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
112    // ce nom remplace le nom actuel
113    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
114
115    // def info fichier de commande
116    void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
117
118    // ramène la valeur
119
120    double Valeur(double x) ;

```

```

121
122 // ramène la valeur et la dérivée en paramètre
123 CourbelD::ValDer Valeur_Et_derivee(double x) ;
124
125 // ramène la dérivée
126 double Derivee(double x) ;
127
128 // ramène la valeur et les dérivées première et seconde en paramètre
129 CourbelD::ValDer2 Valeur_Et_der12(double x);
130
131 // ramène la dérivée seconde
132 double Der_sec(double x);
133
134 // ramène la valeur si dans le domaine strictement de définition
135 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
136 // si supérieur au x maxi , ramène le valeur maximale possible de y
137 CourbelD::Valbool Valeur_stricte(double x) ;
138
139 // ramène la valeur et la dérivée si dans le domaine strictement de définition
140 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
141 // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
142 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
143
144 //----- lecture écriture de restart -----
145 // cas donne le niveau de la récupération
146 // = 1 : on récupère tout
147 // = 2 : on récupère uniquement les données variables (supposées comme telles)
148 void Lecture_base_info(ifstream& ent,const int cas);
149 // cas donne le niveau de sauvegarde
150 // = 1 : on sauvegarde tout
151 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
152 void Ecriture_base_info(ofstream& sort,const int cas);
153 // sortie du schemaXML: en fonction de enu
154 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
155
156 protected :
157 // VARIABLES PROTEGEES :
158 // f(x) = al*(be +ga*x)^n * ln (cosh(de*(he+ee*x)))+ ke*(ge+re*x)
159 double al,be,ga,de,he,ee,ke,ge, re,se;
160 int n;
161 double ax,bx; // min max
162 // METHODES PROTEGEES :
163
164 };
165 /// @} // end of group
166
167 #endif

```

## 7.516 Courbe\_relax\_expo.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_relax_expo.h
30 // classe : Courbe_relax_expo
31
32
33 /*****
34 * DATE: 19/01/2001 *
35 * * *

```

```

36 *   AUTEUR:      G RIO   (mailto:gerardrio56@free.fr)           *
37 *                                                     $   *
38 *   PROJET:      Herezh++                                     *
39 *                                                     $   *
40 * *****
41 *   BUT:         Classe permettant le calcul d'une fonction 1D *
42 *               de type : (a-b)*exp(-c*x) + b                 *
43 *               ainsi qu'un certain nombre d'information      *
44 *               supplémentaires telles que dérivées.          *
45 *                                                     $   *
46 *   ***** *
47 *   VERIFICATION:                                           *
48 *                                                     *
49 *   ! date !   auteur !           but                       !   *
50 *   ----- *
51 *   !           !           !           !                   !   *
52 *   ***** *
53 *   ***** *
54 *   MODIFICATIONS:                                           *
55 *   ! date !   auteur !           but                       !   *
56 *   ----- *
57 *   ***** *
58 * *****/
59
60 #ifndef COURBEEEXO_RELAX_EXPO_1_D_H
61 #define COURBEEEXO_RELAX_EXPO_1_D_H
62
63 #include "CourbelD.h"
64
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 *   BUT:         Classe permettant le calcul d'une fonction 1D *
72 *               de type :                                     *
73 *   f(x) = (a-b)*exp(-c*x) + b                               *
74 *               avec un mini et/ou maxi éventuels          *
75 *               ainsi qu'un certain nombre d'information    *
76 *               supplémentaires telles que dérivées.        *
77 *
78 *
79 * \author      Gérard Rio
80 * \version    1.0
81 * \date       19/01/2001
82 * \brief      Classe permettant le calcul d'une fonction 1D de type : f(x) = (a-b)*exp(-c*x) + b
83 *
84 */
85 class Courbe_relax_expo : public CourbelD
86 {
87 public :
88
89     // CONSTRUCTEURS :
90     Courbe_relax_expo(string nom = "");
91
92     // de copie
93     Courbe_relax_expo(const Courbe_relax_expo& Co);
94     Courbe_relax_expo(const CourbelD& Co);
95
96     // DESTRUCTEUR :
97     ~Courbe_relax_expo();
98
99     // METHODES PUBLIQUES :
100
101     // ----- virtuelles -----
102
103     // affichage de la courbe
104     void Affiche() const ;
105     // ramène true si ok, false sinon
106     bool Complet_courbe()const;
107
108     // Lecture des donnees de la classe sur fichier
109     // le nom passé en paramètre est le nom de la courbe
110     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
111     // ce nom remplace le nom actuel
112     void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
113
114     // def info fichier de commande
115     void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
116
117     // ramène la valeur
118
119     double Valeur(double x) ;
120
121     // ramène la valeur et la dérivée en paramètre

```

```

122 CourbelD::ValDer Valeur_Et_derivee(double x) ;
123
124 // ramène la dérivée
125 double Derivee(double x) ;
126
127 // ramène la valeur et les dérivées première et seconde en paramètre
128 CourbelD::ValDer2 Valeur_Et_der12(double x);
129
130 // ramène la dérivée seconde
131 double Der_sec(double x);
132
133 // ramène la valeur si dans le domaine strictement de définition
134 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
135 // si supérieur au x maxi, ramène la valeur maximale possible de y
136 CourbelD::Valbool Valeur_stricte(double x) ;
137
138 // ramène la valeur et la dérivée si dans le domaine strictement de définition
139 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
140 // si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant
141 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
142
143 //----- lecture écriture de restart -----
144 // cas donne le niveau de la récupération
145 // = 1 : on récupère tout
146 // = 2 : on récupère uniquement les données variables (supposées comme telles)
147 void Lecture_base_info(ifstream& ent,const int cas);
148 // cas donne le niveau de sauvegarde
149 // = 1 : on sauvegarde tout
150 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
151 void Ecriture_base_info(ofstream& sort,const int cas);
152 // sortie du schemaXML: en fonction de enu
153 void SchemaXML_CourbelD(ofstream& sort,const Enum_IO_XML enu) ;
154
155 protected :
156 // VARIABLES PROTEGEES :
157 double xa,xb,xc; // les coefficients de la loi
158 double xmin, xmax; // éventuellement min max sur x
159
160 // METHODES PROTEGEES :
161
162 };
163 /// @} // end of group
164
165 #endif

```

## 7.517 Courbe\_sin.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_sin.h
30 // classe : Courbe_sin
31
32
33 /*****
34 * DATE: 19/01/2001 *
35 * * $ *
36 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
37 * * $ *
38 * PROJET: Herezh++ *

```



```

39 *                                                                 $ *
40 *****
41 *      BUT:      Classe permettant le calcul d'une fonction 1D
42 *                de type : ampli*sin(alpha*x+beta)
43 *                avec un mini et/ou maxi éventuels
44 *                ainsi qu'un certain nombre d'information
45 *                supplémentaires telles que dérivées.
46 *                                                                 $ *
47 *      //////////// *
48 *
49 *      VERIFICATION:
50 *      ! date !   auteur !           but
51 *      -----
52 *      !           !           !
53 *                                                                 $ *
54 *      //////////// *
55 *      MODIFICATIONS:
56 *      ! date !   auteur !           but
57 *      -----
58 *                                                                 $ *
59 *****/
60
61 #ifndef COURBESIN_1_D_H
62 #define COURBESIN_1_D_H
63
64 #include "Courbe1D.h"
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 *      BUT:      Classe permettant le calcul d'une fonction 1D
72 *                de type :
73 *      f(x) = ampli*sin(alpha*x+beta)
74 *                avec un mini et/ou maxi éventuels
75 *                ainsi qu'un certain nombre d'information
76 *                supplémentaires telles que dérivées.
77 *
78 *
79 * \author      Gérard Rio
80 * \version     1.0
81 * \date       19/01/2001
82 * \brief      Classe permettant le calcul d'une fonction 1D de type : f(x) = ampli*sin(alpha*x+beta)
83 *
84 */
85
86 class Courbe_sin : public Courbe1D
87 {
88     public :
89
90     // CONSTRUCTEURS :
91     Courbe_sin(string nom = "");
92
93     // de copie
94     Courbe_sin(const Courbe_sin& Co);
95     Courbe_sin(const Courbe1D& Co);
96
97     // DESTRUCTEUR :
98     ~Courbe_sin();
99
100    // METHODES PUBLIQUES :
101
102    // ----- virtuelles -----
103
104    // affichage de la courbe
105    void Affiche() const ;
106    // ramène true si ok, false sinon
107    bool Complet_courbe()const;
108
109    // Lecture des donnees de la classe sur fichier
110    // le nom passé en paramètre est le nom de la courbe
111    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
112    // ce nom remplace le nom actuel
113    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
114
115    // def info fichier de commande
116    void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
117
118    // ramène la valeur
119
120    double Valeur(double x) ;
121
122    // ramène la valeur et la dérivée en paramètre
123    Courbe1D::ValDer Valeur_Et_derivee(double x) ;
124

```

```

125 // ramène la dérivée
126 double Derivee(double x) ;
127
128 // ramène la valeur et les dérivées première et seconde en paramètre
129 CourbelD::ValDer2 Valeur_Et_der12(double x);
130
131 // ramène la dérivée seconde
132 double Der_sec(double x);
133
134 // ramène la valeur si dans le domaine strictement de définition
135 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
136 // si supérieur au x maxi , ramène la valeur maximale possible de y
137 CourbelD::Valbool Valeur_stricte(double x) ;
138
139 // ramène la valeur et la dérivée si dans le domaine strictement de définition
140 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
141 // si supérieur au x maxi , ramène la valeur maximale possible de y et Y' correspondant
142 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
143
144 //----- lecture écriture de restart -----
145 // cas donne le niveau de la récupération
146 // = 1 : on récupère tout
147 // = 2 : on récupère uniquement les données variables (supposées comme telles)
148 void Lecture_base_info(ifstream& ent,const int cas);
149 // cas donne le niveau de sauvegarde
150 // = 1 : on sauvegarde tout
151 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
152 void Ecriture_base_info(ofstream& sort,const int cas);
153 // sortie du schemaXML: en fonction de enu
154 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
155
156 protected :
157 // VARIABLES PROTEGEES :
158 double ax,bx; // min max
159 bool en_degre; // indique si l'on est en radian ou degré
160 double ampli, alph, bet; // coefs de la fonction
161 // METHODES PROTEGEES :
162
163 };
164 /// @} // end of group
165 #endif

```

## 7.518 Courbe\_un\_moins\_cos.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Courbe_un_moins_cos.h
30 // classe : Courbe_un_moins_cos
31
32
33 /*****
34 *   DATE:      19/01/2001
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   ****
41 *   BUT:      Classe permettant le calcul d'une fonction 1D

```

```

42 *          de type :  $c*(1-\cos((x-a)\pi/(b-a))$  entre a et b,          *
43 *          en dessous de a,  $\rightarrow 0$  et au dessus de b  $\rightarrow c$           *
44 *          ainsi qu'un certain nombre d'information          *
45 *          supplémentaires telles que dérivées.          *
46 *          $          *
47 *          //*****          *
48 *          VERIFICATION:          *
49 *          *          *
50 *          ! date ! auteur ! but !          *
51 *          -----          *
52 *          ! ! ! !          *
53 *          $          *
54 *          //*****          *
55 *          MODIFICATIONS:          *
56 *          ! date ! auteur ! but !          *
57 *          -----          *
58 *          $          *
59 *          *****/
60
61 #ifndef COURBE_UN_MOINS_COS_1_D_H
62 #define COURBE_UN_MOINS_COS_1_D_H
63
64 #include "CourbelD.h"
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 * BUT: Classe permettant le calcul d'une fonction 1D 1D
72 * de type :  $c*(1-\cos((x-a)\pi/(b-a))$  entre a et b,
73 * en dessous de a,  $\rightarrow 0$  et au dessus de b  $\rightarrow c$ 
74 * ainsi qu'un certain nombre d'information
75 * supplémentaires telles que dérivées.
76 *
77 *
78 * \author Gérard Rio
79 * \version 1.0
80 * \date 19/01/2001
81 * \brief Classe permettant le calcul d'une fonction 1D de type :  $f(x) = c*(1-\cos((x-a)\pi/(b-a))$ 
82 *
83 */
84
85 class Courbe_un_moins_cos : public CourbelD
86 {
87 public :
88
89 // CONSTRUCTEURS :
90 Courbe_un_moins_cos(string nom = "");
91
92 // de copie
93 Courbe_un_moins_cos(const Courbe_un_moins_cos& Co);
94 Courbe_un_moins_cos(const CourbelD& Co);
95
96 // DESTRUCTEUR :
97 ~Courbe_un_moins_cos();
98
99 // METHODES PUBLIQUES :
100
101 // ----- virtuelles -----
102
103 // affichage de la courbe
104 void Affiche() const ;
105 // ramène true si ok, false sinon
106 bool Complet_courbe()const;
107
108 // Lecture des donnees de la classe sur fichier
109 // le nom passé en paramètre est le nom de la courbe
110 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
111 // ce nom remplace le nom actuel
112 void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
113
114 // def info fichier de commande
115 void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
116
117 // ramène la valeur
118
119 double Valeur(double x) ;
120
121 // ramène la valeur et la dérivée en paramètre
122 CourbelD::ValDer Valeur_Et_derivee(double x) ;
123
124 // ramène la dérivée
125 double Derivee(double x) ;
126
127 // ramène la valeur et les dérivées première et seconde en paramètre
128 CourbelD::ValDer2 Valeur_Et_der12(double x);

```

```

129
130 // ramène la dérivée seconde
131 double Der_sec(double x);
132
133 // ramène la valeur si dans le domaine strictement de définition
134 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
135 // si supérieur au x maxi , ramène le valeur maximale possible de y
136 CourbelD::Valbool Valeur_stricte(double x) ;
137
138 // ramène la valeur et la dérivée si dans le domaine strictement de définition
139 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
140 // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
141 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
142
143 //----- lecture écriture de restart -----
144 // cas donne le niveau de la récupération
145 // = 1 : on récupère tout
146 // = 2 : on récupère uniquement les données variables (supposées comme telles)
147 void Lecture_base_info(ifstream& ent,const int cas);
148 // cas donne le niveau de sauvegarde
149 // = 1 : on sauvegarde tout
150 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
151 void Ecriture_base_info(ofstream& sort,const int cas);
152 // sortie du schemaXML: en fonction de enu
153 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
154
155
156
157
158 protected :
159 // VARIABLES PROTEGEES :
160 double ax,bx,cx; // les coefficients de la loi
161
162 // METHODES PROTEGEES :
163
164 };
165 /// @} // end of group
166 #endif

```

## 7.519 CourbePolyHermite1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : CourbePolyHermite1D.h
30 // classe : CourbePolyHermite1D
31
32
33 /*****
34 * DATE: 19/01/2001 *
35 * *
36 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
37 * *
38 * PROJET: Herezh++ *
39 * *
40 *****/
41 * BUT: Classe permettant le calcul d'une fonction 1D poly- *
42 * nomiale 1D par morceau, avec continuité de la dérivée *
43 * via une interpolation de type hermite, *
44 * ainsi qu'un certain nombre d'information *

```

```

45 *      supplémentaires telles que dérivées.      *
46 *      $      *
47 *      ***** *
48 *      *
49 *      VERIFICATION:      *
50 *      ! date ! auteur ! but      !      *
51 *      -----      *
52 *      ! ! !      !      *
53 *      $      *
54 *      ***** *
55 *      MODIFICATIONS:      *
56 *      ! date ! auteur ! but      !      *
57 *      -----      *
58 *      $      *
59 *****/
60
61 #ifndef COURBEPOLYHERMITE_1_D_H
62 #define COURBEPOLYHERMITE_1_D_H
63
64 #include "Courbe1D.h"
65 #include "Tableau_T.h"
66 #include "Coordonnee3.h"
67 /// @addtogroup Les_courbes_1D
68 /// @{
69 ///
70
71 /**
72 *
73 *      BUT:      Classe permettant le calcul d'une fonction 1D poly-
74 *      nomiale 1D par morceau, avec continuité de la dérivée
75 *      via une interpolation de type hermite,
76 *      ainsi qu'un certain nombre d'information
77 *      supplémentaires telles que dérivées.
78 *
79 *
80 * \author      Gérard Rio
81 * \version      1.0
82 * \date      19/01/2001
83 * \brief      Classe permettant le calcul d'une fonction 1D polynomiale 1D par morceau, avec continuité
84 *      de la dérivée via une interpolation de type hermite.
85 */
86
87 class CourbePolyHermite1D : public Courbe1D
88 {
89 public :
90
91     // CONSTRUCTEURS :
92     // par défaut
93     CourbePolyHermite1D(string nom = "");
94     // fonction d'un tableau de points
95     CourbePolyHermite1D(Tableau <Coordonnee3>& pt,string nom = "");
96
97
98     // de copie
99     CourbePolyHermite1D(const CourbePolyHermite1D& Co);
100    CourbePolyHermite1D(const Courbe1D& Co);
101
102    // DESTRUCTEUR :
103    ~CourbePolyHermite1D();
104
105    // METHODES PUBLIQUES :
106
107    // ----- virtuelles -----
108
109    // affichage de la courbe
110    void Affiche() const;
111    // ramène true si ok, false sinon
112    bool Complet_courbe()const;
113
114    // Lecture des donnees de la classe sur fichier
115    // le nom passé en paramètre est le nom de la courbe
116    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
117    // ce nom remplace le nom actuel
118    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
119
120    // def info fichier de commande
121    void Info_commande_Courbes1D(UtilLecture & entreePrinc);
122
123    // ramène la valeur
124
125    double Valeur(double x) ;
126
127    // ramène la valeur et la dérivée en paramètre
128    Courbe1D::ValDer Valeur_Et_derivee(double x) ;
129

```

```

130 // ramène la dérivée
131 double Derivee(double x) ;
132
133 // ramène la valeur et les dérivées première et seconde en paramètre
134 CourbeID::ValDer2 Valeur_Et_der12(double x);
135
136 // ramène la dérivée seconde
137 double Der_sec(double x);
138
139
140 // ramène la valeur si dans le domaine strictement de définition
141 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
142 // si supérieur au x maxi , ramène le valeur maximale possible de y
143 CourbeID::Valbool Valeur_stricte(double x) ;
144
145 // ramène la valeur et la dérivée si dans le domaine strictement de définition
146 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
147 // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
148 CourbeID::ValDerbool Valeur_Et_derivee_stricte(double x) ;
149
150 // méthode pour changer le tableau de points associé
151 void Change_tabPoints(Tableau <Coordonnee3>& pt);
152
153 //----- lecture écriture de restart -----
154 // cas donne le niveau de la récupération
155 // = 1 : on récupère tout
156 // = 2 : on récupère uniquement les données variables (supposées comme telles)
157 void Lecture_base_info(ifstream& ent,const int cas);
158 // cas donne le niveau de sauvegarde
159 // = 1 : on sauvegarde tout
160 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
161 void Ecriture_base_info(ofstream& sort,const int cas);
162 // sortie du schemaXML: en fonction de enu
163 virtual void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
164
165 // ----- méthodes particulières à la fonction poly-linéaire -----
166 // ramène le nombre de point de la polyligne
167 int NombrePoint() {return points.Taille();};
168 // ramène le x et y du point i
169 const Coordonnee3& Pt_nbi(int i) {return points(i);};
170
171 protected :
172 // VARIABLES PROTEGEES :
173 Tableau <Coordonnee3> points; // les points de la courbe: x,y,y' : le z est en fait la dérivée au
point
174 int indice_precedant; // position précédente
175
176 // METHODES PROTEGEES :
177 // fonction d'un type: utilisable par les classes dérivée
178 CourbePolyHermitelD( string nom,EnumCourbeID typ);
179
180 };
181 /// @} // end of group
182 #endif

```

## 7.520 CourbePolyLineaire1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28

```

```

29 // fichier : CourbePolyLineaire1D.h
30 // classe : CourbePolyLineaire1D
31
32
33 /*****
34 *   DATE:      19/01/2001
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   ****
41 *   BUT:      Classe permettant le calcul d'une fonction 1D poly-
42 *             linéaire 1D ainsi qu'un certain nombre d'information
43 *             supplémentaires telles que dérivées.
44 *
45 *   ****
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !           but
50 *   -----
51 *   !           !           !
52 *   ****
53 *   MODIFICATIONS:
54 *
55 *   ! date !   auteur !           but
56 *   -----
57 *   ****
58
59 #ifndef COURBEPOLYLINEAIRE_1_D_H
60 #define COURBEPOLYLINEAIRE_1_D_H
61
62 #include "Courbe1D.h"
63 #include "Tableau_T.h"
64 #include "Coordonnee2.h"
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 *   BUT:      Classe permettant le calcul d'une fonction 1D
72 *             de type :
73 *   f(x) = poly-linéaire c-a-d linéaire par morceaux limités par 2 points
74 *             ainsi qu'un certain nombre d'information
75 *             supplémentaires telles que dérivées.
76 *
77 *
78 * \author    Gérard Rio
79 * \version   1.0
80 * \date      19/01/2001
81 * \brief     Classe permettant le calcul d'une fonction 1D de type : f(x) = poly-linéaire c-a-d
82 *             linéaire par morceaux limités par 2 points
83 */
84
85 class CourbePolyLineaire1D : public Courbe1D
86 {
87 public :
88
89     // CONSTRUCTEURS :
90     // par défaut
91     CourbePolyLineaire1D(string nom = "");
92     // fonction d'un tableau de points
93     CourbePolyLineaire1D(Tableau <Coordonnee2>& pt,string nom = "");
94
95
96     // de copie
97     CourbePolyLineaire1D(const CourbePolyLineaire1D& Co);
98     CourbePolyLineaire1D(const Courbe1D& Co);
99
100    // DESTRUCTEUR :
101    ~CourbePolyLineaire1D();
102
103    // METHODES PUBLIQUES :
104
105    // ----- virtuelles -----
106
107    // affichage de la courbe
108    void Affiche() const;
109    // ramène true si ok, false sinon
110    bool Complet_courbe()const;
111
112    // Lecture des donnees de la classe sur fichier
113    // le nom passé en paramètre est le nom de la courbe

```

```

114 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
115 // ce nom remplace le nom actuel
116 void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
117
118 // def info fichier de commande
119 void Info_commande_Courbes1D(UtilLecture & entreePrinc);
120
121 // ramène la valeur
122
123 double Valeur(double x) ;
124
125 // ramène la valeur et la dérivée en paramètre
126 CourbelD::ValDer Valeur_Et_derivee(double x) ;
127
128 // ramène la dérivée
129 double Derivee(double x) ;
130
131 // ramène la valeur et les dérivées première et seconde en paramètre
132 CourbelD::ValDer2 Valeur_Et_der12(double x);
133
134 // ramène la dérivée seconde
135 double Der_sec(double x);
136
137
138 // ramène la valeur si dans le domaine strictement de définition
139 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
140 // si supérieur au x maxi, ramène la valeur maximale possible de y
141 CourbelD::Valbool Valeur_stricte(double x) ;
142
143 // ramène la valeur et la dérivée si dans le domaine strictement de définition
144 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
145 // si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant
146 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
147
148 // méthode pour changer le tableau de points associé
149 void Change_tabPoints(Tableau <Coordonnee2>& pt);
150
151 //----- lecture écriture de restart -----
152 // cas donne le niveau de la récupération
153 // = 1 : on récupère tout
154 // = 2 : on récupère uniquement les données variables (supposées comme telles)
155 void Lecture_base_info(ifstream& ent,const int cas);
156 // cas donne le niveau de sauvegarde
157 // = 1 : on sauvegarde tout
158 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
159 void Ecriture_base_info(ofstream& sort,const int cas);
160 // sortie du schemaXML: en fonction de enu
161 virtual void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
162
163 // ----- méthodes particulières à la fonction poly-linéaire -----
164 // ramène le nombre de point de la polyligne
165 int NombrePoint() {return points.Taille();};
166 // ramène le x et y du point i
167 const Coordonnee2& Pt_nbi(int i) {return points(i);};
168
169 protected :
170 // VARIABLES PROTEGEES :
171 Tableau <Coordonnee2> points; // les points de la courbe
172 double der_init; // dérivée initiale
173 double der_finale; // dérivée finale
174 int indice_precedant; // position précédente
175
176 // METHODES PROTEGEES :
177 // fonction d'un type: utilisable par les classes dérivée
178 CourbePolyLineaire1D( string nom,EnumCourbelD typ);
179
180 };
181 /// @} // end of group
182 #endif

```

## 7.521 CourbePolyLineaire1D\_simpli.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //

```



```

14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : CourbePolyLineaire1D_simpli.h
30 // classe : CourbePolyLineaire1D_simpli
31
32
33 /*****
34 *   DATE:           03/04/2004
35 *
36 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:         Herezh++
39 *
40 * *****/
41 *   BUT:           Cas d'une définition très simplifié d'une fonction 1D
42 *                 poly-linéaire 1D qui hérite de la fonction poly-linéaire
43 *                 générique. Permet ainsi de minimiser la taille des infos
44 *                 à lire.
45 *
46 *   *****
47 *
48 *   VERIFICATION:
49 *
50 *   ! date ! auteur ! but
51 *   -----
52 *   ! ! !
53 *   *****
54 *
55 *   MODIFICATIONS:
56 *
57 *   ! date ! auteur ! but
58 *   -----
59 *   $
60 *****/
61
62 #ifndef COURBEPOLYLINEAIRE_1_D_SIMPLI_H
63 #define COURBEPOLYLINEAIRE_1_D_SIMPLI_H
64
65 #include "CourbePolyLineaire1D.h"
66 #include "Tableau_T.h"
67 #include "Coordonnee2.h"
68 /// @addtogroup Les_courbes_1D
69 /// @
70 /**
71 *
72 *   BUT:           Cas d'une définition très simplifié d'une fonction 1D
73 *                 poly-linéaire 1D qui hérite de la fonction poly-linéaire
74 *                 générique. Permet ainsi de minimiser la taille des infos
75 *                 à lire.
76 *
77 * \author         Gérard Rio
78 * \version        1.0
79 * \date           03/04/2004
80 * \brief          Classe permettant le calcul d'une fonction 1D de type : poly-linéaire 1D simplifiée, qui
81 *                 hérite de la fonction poly-linéaire
82 */
83
84 class Cpl1D : public CourbePolyLineaire1D
85 {
86 public :
87
88     // CONSTRUCTEURS :
89     // par défaut
90
91     Cpl1D(string nom = "") : CourbePolyLineaire1D(nom,CPL1D) {};
92
93     // de copie
94     Cpl1D(const Cpl1D& Co) : CourbePolyLineaire1D(Co) {};
95     Cpl1D(const Courbe1D& Co);
96
97     // DESTRUCTEUR :
98     ~Cpl1D() {};

```

```

99
100 // METHODES PUBLIQUES :
101
102 // ----- virtuelles -----
103
104 // Lecture des donnees de la classe sur fichier
105 // le nom passé en paramètre est le nom de la courbe
106 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
107 // ce nom remplace le nom actuel
108 void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
109
110 // def info fichier de commande
111 void Info_commande_Courbes1D(UtilLecture & entreePrinc);
112 // sortie du schemaXML: en fonction de enu
113 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
114
115 };
116 /// @} // end of group
117 #endif

```

## 7.522 CourbePolynomiale.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : CourbePolynomiale.h
30 // classe : CourbePolynomiale
31
32
33 /*****
34 *      DATE:      19/01/2001
35 *
36 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *
41 *      BUT:      Classe permettant le calcul d'un polynôme 1D
42 *              de type : a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ...
43 *              ainsi qu'un certain nombre d'information
44 *              supplémentaires telles que dérivées.
45 *
46 *      *****
47 *      VERIFICATION:
48 *
49 *      ! date !   auteur !           but
50 *      -----
51 *      !           !           !
52 *
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date !   auteur !           but
56 *      -----
57 *
58 *      *****/
59
60 #ifndef COURBEPOLYNOMIALE_1_D_H
61 #define COURBEPOLYNOMIALE_1_D_H
62
63 #include "Courbe1D.h"

```

```

64 #include "Tableau_T.h"
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 *   BUT:   Classe permettant le calcul d'un polynôme 1D
72 *         de type :  $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots$ 
73 *         ainsi qu'un certain nombre d'information
74 *         supplémentaires telles que dérivées.
75 *
76 * \author   Gérard Rio
77 * \version  1.0
78 * \date     19/01/2001
79 * \brief    Classe permettant le calcul d'un polynôme 1D de type :  $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots$ 
80 *
81 */
82
83 class CourbePolynomiale : public CourbelD
84 {
85 public :
86
87     // CONSTRUCTEURS :
88     CourbePolynomiale(string nom = "");
89
90     // de copie
91     CourbePolynomiale(const CourbePolynomiale& Co);
92     CourbePolynomiale(const CourbelD& Co);
93
94     // DESTRUCTEUR :
95     ~CourbePolynomiale();
96
97     // METHODES PUBLIQUES :
98
99     // ----- virtuelles -----
100
101     // affichage de la courbe
102     void Affiche() const ;
103     // ramène true si ok, false sinon
104     bool Complet_courbe()const;
105
106     // Lecture des donnees de la classe sur fichier
107     // le nom passé en paramètre est le nom de la courbe
108     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
109     // ce nom remplace le nom actuel
110     void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
111
112     // def info fichier de commande
113     void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
114
115     // ramène la valeur
116
117     double Valeur(double x) ;
118
119     // ramène la valeur et la dérivée en paramètre
120     CourbelD::ValDer Valeur_Et_derivee(double x) ;
121
122     // ramène la dérivée
123     double Derivee(double x) ;
124
125     // ramène la valeur et les dérivées première et seconde en paramètre
126     CourbelD::ValDer2 Valeur_Et_der12(double x);
127
128     // ramène la dérivée seconde
129     double Der_sec(double x);
130
131     // ramène la valeur si dans le domaine strictement de définition
132     // si c'est inférieur au x mini, ramène la valeur minimale possible de y
133     // si supérieur au x maxi , ramène le valeur maximale possible de y
134     CourbelD::Valbool Valeur_stricte(double x) ;
135
136     // ramène la valeur et la dérivée si dans le domaine strictement de définition
137     // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
138     // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
139     CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
140
141     //----- lecture écriture de restart -----
142     // cas donne le niveau de la récupération
143     // = 1 : on récupère tout
144     // = 2 : on récupère uniquement les données variables (supposées comme telles)
145     void Lecture_base_info(ifstream& ent,const int cas);
146     // cas donne le niveau de sauvegarde
147     // = 1 : on sauvegarde tout
148     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
149     void Ecriture_base_info(ofstream& sort,const int cas);

```

```

150 // sortie du schemaXML: en fonction de enu
151 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
152
153
154
155
156 protected :
157 // VARIABLES PROTEGEES :
158 Tableau <double> coef; // coefficients du polynôme
159
160 // METHODES PROTEGEES :
161
162 };
163 /// @} // end of group
164 #endif

```

## 7.523 F1\_plus\_F2.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28 //
29 // fichier : F1_plus_F2.h
30 // classe : F1_plus_F2
31
32
33 /*****
34 * DATE: 14/10/2004 *
35 * * $ *
36 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
37 * * $ *
38 * PROJET: Herezh++ *
39 * * $ *
40 *****/
41 * BUT: Classe permettant le calcul d'une fonction somme *
42 * F(x)= (F1 + F2)(x)=F1(x) + F2(x) *
43 * ainsi qu'un certain nombre d'information *
44 * supplémentaires telles que dérivées. *
45 * * $ *
46 * ***** *
47 * VERIFICATION: *
48 * * *
49 * ! date ! auteur ! but ! *
50 * ----- *
51 * ! ! ! ! *
52 * * $ *
53 * ***** *
54 * MODIFICATIONS: *
55 * ! date ! auteur ! but ! *
56 * ----- *
57 * * $ *
58 *****/
59
60 #ifndef F1_PLUS_F2_1_D_H
61 #define F1_PLUS_F2_1_D_H
62
63 #include "Courbe1D.h"
64 #include "Tableau_T.h"
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///

```

```

68
69 /**
70 *
71 *   BUT:   Classe permettant le calcul d'une fonction 1D somme
72 *         de type :
73 *   F(x)= (F1 + F2)(x)=F1(x) + F2 (x)
74 *         ainsi qu'un certain nombre d'information
75 *         supplémentaires telles que dérivées.
76 *
77 *
78 * \author   Gérard Rio
79 * \version  1.0
80 * \date    19/01/2001
81 * \brief   Classe permettant le calcul d'une fonction 1D de type : F(x)= (F1 + F2)(x)=F1(x) + F2 (x)
82 *
83 */
84
85 class F1_plus_F2 : public CourbelD
86 {
87 public :
88
89     // CONSTRUCTEURS :
90     F1_plus_F2(string nom = "");
91     // constructeur fonction de deux courbes existantes et d'un nom
92     F1_plus_F2(CourbelD* FF1, CourbelD* FF2, string nom = "");
93
94     // de copie
95     F1_plus_F2(const F1_plus_F2& Co);
96     F1_plus_F2(const CourbelD& Co);
97
98     // DESTRUCTEUR :
99     ~F1_plus_F2();
100
101     // METHODES PUBLIQUES :
102
103     // ----- virtuelles -----
104
105     // affichage de la courbe
106     void Affiche() const ;
107     // ramène true si ok, false sinon
108     bool Complet_courbe()const;
109
110     // Lecture des donnees de la classe sur fichier
111     // le nom passé en paramètre est le nom de la courbe
112     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
113     // ce nom remplace le nom actuel
114     void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
115     // dans le cas où les courbes membres sont des courbes externes
116     // fonction pour les définir
117     // les courbes sont défini en interne que si les courbes arguments sont elles même
118     // des courbes locales. c'est-à-dire si FF1->NomCourbe() = "_" alors on recrée une courbe
119     // interne avec new pour F1, sinon F1=FF1 et pas de création; idem pour f2
120     // dans le cas où FF1 ou FF1 est NULL on passe, pas de traitement pour ce pointeur
121     void DefCourbesMembres(CourbelD* FF1, CourbelD* FF2);
122
123     // établir le lien entre la courbe et des courbes déjà existantes dont
124     // on connaît que le nom
125     // permet ainsi de compléter la courbe
126     // 1) renseigne si la courbe dépend d'autre courbe ou non
127     bool DependAutreCourbes() const;
128     // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
129     list <string>& ListDependanceCourbes(list <string>& lico) const;
130     // 3) établit la connection entre la demande de *this et les courbes passées en paramètres
131     void Lien_entre_courbe (list <CourbelD *>& liptco);
132
133
134     // def info fichier de commande
135     void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
136
137     // ramène la valeur
138
139     double Valeur(double x) ;
140
141     // ramène la valeur et la dérivée en paramètre
142     CourbelD::ValDer Valeur_Et_derivee(double x) ;
143
144     // ramène la dérivée
145     double Derivee(double x) ;
146
147     // ramène la valeur et les dérivées première et seconde en paramètre
148     CourbelD::ValDer2 Valeur_Et_der12(double x);
149
150     // ramène la dérivée seconde
151     double Der_sec(double x);
152
153     // ramène la valeur si dans le domaine strictement de définition

```

```

154 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
155 // si supérieur au x maxi, ramène la valeur maximale possible de y
156 CourbelD::Valbool Valeur_stricte(double x) ;
157
158 // ramène la valeur et la dérivée si dans le domaine strictement de définition
159 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
160 // si supérieur au x maxi, ramène la valeur maximale possible de y et Y' correspondant
161 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
162
163 //----- lecture écriture de restart -----
164 // cas donne le niveau de la récupération
165 // = 1 : on récupère tout
166 // = 2 : on récupère uniquement les données variables (supposées comme telles)
167 void Lecture_base_info(ifstream& ent,const int cas);
168 // cas donne le niveau de sauvegarde
169 // = 1 : on sauvegarde tout
170 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
171 void Ecriture_base_info(ofstream& sort,const int cas);
172 // sortie du schemaXML: en fonction de enu
173 void SchemaXML_CourbelD(ofstream& sort,const Enum_IO_XML enu) ;
174
175
176
177
178 protected :
179 // VARIABLES PROTEGEES :
180 CourbelD* F1; // première fonction
181 CourbelD* F2; // seconde fonction
182 double ponder1,ponder2; // pondérations éventuelles des courbes
183 // variable intermédiaires pour la lecture en deux temps
184 // servent également ensuite pour dire si F1 ou F2 sont interne ou pas
185 // s'ils sont interne -> nom_courbei="i_interne_i", sinon ="e_externe_e"
186 string nom_courbel1,nom_courbe2;
187
188 // METHODES PROTEGEES :
189
190 };
191 /// @} // end of group
192 #endif

```

## 7.524 F1\_rond\_F2.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : F1_rond_F2.h
30 // classe : F1_rond_F2
31
32
33 /*****
34 * DATE: 14/10/2004 *
35 * * $ *
36 * AUTEUR: G RIO (mailto:gerardrio56@free.fr) *
37 * * $ *
38 * PROJET: Herezh++ *
39 * * $ *
40 *****/
41 * BUT: Classe permettant le calcul d'une fonction composée *
42 * F(x)= F1(F2(x))=F1 o F2 (x) *
43 * ainsi qu'un certain nombre d'information *

```

```

44 *           supplémentaires telles que dérivées.           *
45 *                                                     $ *
46 *           ////////////////////////////////////////////////// *
47 *           VERIFICATION:                                     *
48 *           ! date !   auteur !           but           ! *
49 *           ----- *
50 *           !           !           !           !           *
51 *           ////////////////////////////////////////////////// *
52 *           ////////////////////////////////////////////////// *
53 *           ////////////////////////////////////////////////// *
54 *           MODIFICATIONS:                                   *
55 *           ! date !   auteur !           but           ! *
56 *           ----- *
57 *                                                     $ *
58 * *****/
59
60 #ifndef F1_ROND_F2_1_D_H
61 #define F1_ROND_F2_1_D_H
62
63 #include "CourbelD.h"
64 #include "Tableau_T.h"
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 *   BUT:   Classe permettant le calcul d'une fonction composé 1D:
72 *           $F(x) = F1(F2(x)) = F1 \circ F2(x)$ 
73 *          ainsi qu'un certain nombre d'information
74 *          supplémentaires telles que dérivées.
75 *
76 *
77 * \author   Gérard Rio
78 * \version  1.0
79 * \date    19/01/2001
80 * \brief   Classe permettant le calcul d'une fonction 1D composé :  $F(x) = F1(F2(x)) = F1 \circ F2(x)$ 
81 *
82 */
83
84 class F1_rond_F2 : public CourbelD
85 {
86 public :
87
88     // CONSTRUCTEURS :
89     F1_rond_F2(string nom = "");
90     // constructeur fonction de deux courbes existantes et d'un nom
91     F1_rond_F2(CourbelD* FF1, CourbelD* FF2, string nom = "");
92
93     // de copie
94     F1_rond_F2(const F1_rond_F2& Co);
95     F1_rond_F2(const CourbelD& Co);
96
97     // DESTRUCTEUR :
98     ~F1_rond_F2();
99
100    // METHODES PUBLIQUES :
101
102    // ----- virtuelles -----
103
104    // affichage de la courbe
105    void Affiche() const ;
106    // ramène true si ok, false sinon
107    bool Complet_courbe()const;
108
109    // Lecture des donnees de la classe sur fichier
110    // le nom passé en paramètre est le nom de la courbe
111    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
112    // ce nom remplace le nom actuel
113    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
114    // dans le cas où les courbes membres sont des courbes externes
115    // fonction pour les définir
116    // les courbes sont défini en interne que si les courbes arguments sont elles même
117    // des courbes locales. c'est-à-dire si FF1->NomCourbe() = "_" alors on recrée une courbe
118    // interne avec new pour F1, sinon F1=FF1 et pas de création; idem pour f2
119    // dans le cas où FF1 ou FF1 est NULL on passe, pas de traitement pour ce pointeur
120    void DefCourbesMembres(CourbelD* FF1, CourbelD* FF2);
121
122    // établir le lien entre la courbe et des courbes déjà existantes dont
123    // on connait que le nom
124    // permet ainsi de compléter la courbe
125    // 1) renseigne si la courbe dépend d'autre courbe ou non
126    bool DependAutreCourbes() const;
127    // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
128    list <string>& ListDependanceCourbes(list <string>& lico) const;
129    // 3) établit la connection entre la demande de *this et les courbes passées en paramètres
130    void Lien_entre_courbe (list <CourbelD *>& liptco);

```

```

131
132
133 // def info fichier de commande
134 void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
135
136 // ramène la valeur
137
138 double Valeur(double x) ;
139
140 // ramène la valeur et la dérivée en paramètre
141 CourbelD::ValDer Valeur_Et_derivee(double x) ;
142
143 // ramène la dérivée
144 double Derivee(double x) ;
145
146 // ramène la valeur et les dérivées première et seconde en paramètre
147 CourbelD::ValDer2 Valeur_Et_der12(double x);
148
149 // ramène la dérivée seconde
150 double Der_sec(double x);
151
152 // ramène la valeur si dans le domaine strictement de définition
153 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
154 // si supérieur au x maxi , ramène le valeur maximale possible de y
155 CourbelD::Valbool Valeur_stricte(double x) ;
156
157 // ramène la valeur et la dérivée si dans le domaine strictement de définition
158 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
159 // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
160 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
161
162 //----- lecture écriture de restart -----
163 // cas donne le niveau de la récupération
164 // = 1 : on récupère tout
165 // = 2 : on récupère uniquement les données variables (supposées comme telles)
166 void Lecture_base_info(ifstream& ent,const int cas);
167 // cas donne le niveau de sauvegarde
168 // = 1 : on sauvegarde tout
169 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
170 void Ecriture_base_info(ofstream& sort,const int cas);
171 // sortie du schemaXML: en fonction de enu
172 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
173
174
175
176
177 protected :
178 // VARIABLES PROTEGEES :
179 CourbelD* F1; // première fonction
180 CourbelD* F2; // seconde fonction
181 // variable intermédiaires pour la lecture en deux temps
182 // servent également ensuite pour dire si F1 ou F2 sont interne ou pas
183 // s'ils sont interne -> nom_courbei="i_interne_i", sinon ="e_externe_e"
184 string nom_courbel,nom_courbe2;
185
186 // METHODES PROTEGEES :
187
188 };
189 /// @} // end of group
190 #endif

```

## 7.525 F\_cycle\_add.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.

```



```

23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : F_cycle_add.h
30 // classe : F_cycle_add
31
32
33 /*****
34 *   DATE:      14/10/2004
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 * *****/
41 *   BUT: Classe permettant le calcul d'une fonction cyclique
42 *        c'est-à-dire qui tous les delta x donnée, retrouve la
43 *        même forme. De plus, on ajoute un facteur d'amplifi-
44 *        cation "ADDITIF" en fonction du nombre de cycles.
45 *        Enfin, à chaque début de cycle, la fonction est translatée
46 *        de la valeur qu'elle avait à la fin du cycle précédent.
47 *
48 *   *****
49 *   VERIFICATION:
50 *
51 *   ! date !   auteur !           but
52 *   -----
53 *   !           !           !
54 *
55 *   *****
56 *   MODIFICATIONS:
57 *   ! date !   auteur !           but
58 *   -----
59 *
60 * *****/
61
62 #ifndef F1_CYCLE_ADD_1_D_H
63 #define F1_CYCLE_ADD_1_D_H
64
65 #include "CourbelD.h"
66 #include "Tableau_T.h"
67 /// @addtogroup Les_courbes_1D
68 /// @{
69 ///
70
71 /**
72 *
73 *   BUT: Classe permettant le calcul d'une fonction cyclique
74 *        c'est-à-dire qui tous les delta x donnée, retrouve la
75 *        même forme. De plus, on ajoute un facteur d'amplifi-
76 *        cation "ADDITIF" en fonction du nombre de cycles.
77 *        Enfin, à chaque début de cycle, la fonction est translatée
78 *        de la valeur qu'elle avait à la fin du cycle précédent.
79 *
80 *
81 * \author   Gérard Rio
82 * \version  1.0
83 * \date    19/01/2001
84 * \brief   Classe permettant le calcul d'une fonction cyclique avec une amplification additive à
85 *          chaque cycle
86 */
87
88 class F_cycle_add : public CourbelD
89 {
90 public :
91
92     // CONSTRUCTEURS :
93     F_cycle_add(string nom = "");
94     // constructeur fonction d'une courbe existante d'un nom
95     F_cycle_add(CourbelD* FF1, string nom = "");
96
97     // de copie
98     F_cycle_add(const F_cycle_add& Co);
99     F_cycle_add(const CourbelD& Co);
100
101     // DESTRUCTEUR :
102     ~F_cycle_add();
103
104     // METHODES PUBLIQUES :
105
106     // ----- virtuelles -----
107
108     // affichage de la courbe

```

```

109 void Affiche() const ;
110 // ramène true si ok, false sinon
111 bool Complet_courbe()const;
112
113 // Lecture des donnees de la classe sur fichier
114 // le nom passé en paramètre est le nom de la courbe
115 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
116 // ce nom remplace le nom actuel
117 void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
118 // dans le cas où la courbe membre est une courbe externe
119 // fonction pour la définir
120 // la courbe est défini en interne que si la courbe argument est elle même
121 // une courbe locale. c'est-à-dire si FF1->NomCourbe() = "_" alors on recrée une courbe
122 // interne avec new pour F1, sinon F=FF1 et pas de création;
123 // dans le cas où FF1 ou FF1 est NULL on passe, pas de traitement pour ce pointeur
124 void DefCourbesMembres(CourbelD* FF1);
125
126 // établir le lien entre la courbe et une courbe déjà existante dont
127 // on connaît que le nom
128 // permet ainsi de compléter la courbe
129 // 1) renseigne si la courbe dépend d'une autre courbe ou non
130 bool DependAutreCourbes() const;
131 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
132 list <string>& ListDependanceCourbes(list <string>& lico) const;
133 // 3) établit la connection entre la demande de *this et les courbes passées en paramètres
134 void Lien_entre_courbe (list <CourbelD *>& liptco);
135
136
137 // def info fichier de commande
138 void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
139
140 // ramène la valeur
141
142 double Valeur(double x) ;
143
144 // ramène la valeur et la dérivée en paramètre
145 CourbelD::ValDer Valeur_Et_derivee(double x) ;
146
147 // ramène la dérivée
148 double Derivee(double x) ;
149
150 // ramène la valeur et les dérivées première et seconde en paramètre
151 CourbelD::ValDer2 Valeur_Et_der12(double x);
152
153 // ramène la dérivée seconde
154 double Der_sec(double x);
155
156 // ramène la valeur si dans le domaine strictement de définition
157 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
158 // si supérieur au x maxi , ramène le valeur maximale possible de y
159 CourbelD::Valbool Valeur_stricte(double x) ;
160
161 // ramène la valeur et la dérivée si dans le domaine strictement de définition
162 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
163 // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
164 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
165
166 //----- lecture écriture de restart -----
167 // cas donne le niveau de la récupération
168 // = 1 : on récupère tout
169 // = 2 : on récupère uniquement les données variables (supposées comme telles)
170 void Lecture_base_info(ifstream& ent,const int cas);
171 // cas donne le niveau de sauvegarde
172 // = 1 : on sauvegarde tout
173 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
174 void Ecriture_base_info(ofstream& sort,const int cas);
175 // sortie du schemaXML: en fonction de enu
176 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
177
178
179
180
181 protected :
182 // VARIABLES PROTEGEES :
183 CourbelD* F1; // fonction de base
184 // variable intermédiaires pour la lecture en deux temps
185 // servent également ensuite pour dire si F1 est interne ou pas
186 // si elle est interne -> nom_courbei="i_interne_i", sinon ="e_externe_e"
187 string nom_courbel;
188
189 double ampli; // facteur d'amplification à chaque cycle
190 double longcycl; // longueur du cycle
191 double decalx,decaly; // decalages initiaux
192
193 // METHODES PROTEGEES :
194
195 };

```

```

196 /// @} // end of group
197 #endif

```

## 7.526 F\_cyclique.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : F_cyclique.h
30 // classe : F_cyclique
31
32
33 /*****
34 *      DATE:      14/10/2004
35 *
36 *      AUTEUR:    G RIO      (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *
41 *
42 *      BUT:      Classe permettant le calcul d'une fonction cyclique
43 *              c'est-à-dire qui tous les delta x donnée, retrouve la
44 *              même forme. De plus, on ajoute un facteur d'amplifi-
45 *              cation "MULTIPLICATIF" en fonction du nombre de cycles.
46 *              Enfin, à chaque début de cycle, la fonction est translatée
47 *              de la valeur qu'elle avait à la fin du cycle précédent.
48 *
49 *              *****
50 *      VERIFICATION:
51 *
52 *      ! date ! auteur ! but
53 *      -----
54 *      ! ! !
55 *
56 *              *****
57 *      MODIFICATIONS:
58 *      ! date ! auteur ! but
59 *      -----
60 *
61 *
62
63 #ifndef F1_CYCLIQUE_1_D_H
64 #define F1_CYCLIQUE_1_D_H
65
66 #include "CourbelD.h"
67 #include "Tableau_T.h"
68 /// @addtogroup Les_courbes_1D
69 /// @{}
70 ///
71
72 /**
73 *
74 *      BUT:      Classe permettant le calcul d'une fonction cyclique
75 *              c'est-à-dire qui tous les delta x donnée, retrouve la
76 *              même forme. De plus, on ajoute un facteur d'amplifi-
77 *              cation "MULTIPLICATIF" en fonction du nombre de cycles.
78 *              Enfin, à chaque début de cycle, la fonction est translatée
79 *              de la valeur qu'elle avait à la fin du cycle précédent.
80 *

```

```

81 *
82 * \author   Gérard Rio
83 * \version  1.0
84 * \date    19/01/2001
85 * \brief    Classe permettant le calcul d'une fonction cyclique avec une amplification multiplicative
             à chaque cycle
86 *
87 */
88
89 class F_cyclique : public CourbelD
90 {
91 public :
92
93     // CONSTRUCTEURS :
94     F_cyclique(string nom = "");
95     // constructeur fonction d'une courbe existante d'un nom
96     F_cyclique(CourbelD* FF1, string nom = "");
97
98     // de copie
99     F_cyclique(const F_cyclique& Co);
100    F_cyclique(const CourbelD& Co);
101
102    // DESTRUCTEUR :
103    ~F_cyclique();
104
105    // METHODES PUBLIQUES :
106
107    // ----- virtuelles -----
108
109    // affichage de la courbe
110    void Affiche() const ;
111    // ramène true si ok, false sinon
112    bool Complet_courbe()const;
113
114    // Lecture des donnees de la classe sur fichier
115    // le nom passé en paramètre est le nom de la courbe
116    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
117    // ce nom remplace le nom actuel
118    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
119    // dans le cas où la courbe membre est une courbe externe
120    // fonction pour la définir
121    // la courbe est défini en interne que si la courbe argument est elle même
122    // une courbe locale. c'est-à-dire si FF1->NomCourbe() ="" alors on recrée une courbe
123    // interne avec new pour Fl, sinon F=FF1 et pas de création;
124    // dans le cas où FF1 ou FF1 est NULL est NULL on passe, pas de traitement pour ce pointeur
125    void DefCourbesMembres(CourbelD* FF1);
126
127    // établir le lien entre la courbe et une courbe déjà existante dont
128    // on connait que le nom
129    // permet ainsi de compléter la courbe
130    // 1) renseigne si la courbe dépend d'une autre courbe ou non
131    bool DependAutreCourbes() const;
132    // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
133    list <string>& ListDependanceCourbes(list <string>& lico) const;
134    // 3) établit la connection entre la demande de *this et les courbes passées en paramètres
135    void Lien_entre_courbe (list <CourbelD *>& liptco);
136
137
138    // def info fichier de commande
139    void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
140
141    // ramène la valeur
142
143    double Valeur(double x) ;
144
145    // ramène la valeur et la dérivée en paramètre
146    CourbelD::ValDer Valeur_Et_derivee(double x) ;
147
148    // ramène la dérivée
149    double Derivee(double x) ;
150
151    // ramène la valeur et les dérivées première et seconde en paramètre
152    CourbelD::ValDer2 Valeur_Et_der12(double x);
153
154    // ramène la dérivée seconde
155    double Der_sec(double x);
156
157    // ramène la valeur si dans le domaine strictement de définition
158    // si c'est inférieur au x mini, ramène la valeur minimale possible de y
159    // si supérieur au x maxi , ramène le valeur maximale possible de y
160    CourbelD::Valbool Valeur_stricte(double x) ;
161
162    // ramène la valeur et la dérivée si dans le domaine strictement de définition
163    // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
164    // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
165    CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
166

```

```

167 //----- lecture écriture de restart -----
168 // cas donne le niveau de la récupération
169 // = 1 : on récupère tout
170 // = 2 : on récupère uniquement les données variables (supposées comme telles)
171 void Lecture_base_info(istream& ent,const int cas);
172 // cas donne le niveau de sauvegarde
173 // = 1 : on sauvegarde tout
174 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
175 void Ecriture_base_info(ofstream& sort,const int cas);
176 // sortie du schemaXML: en fonction de enu
177 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
178
179
180
181
182 protected :
183 // VARIABLES PROTEGEES :
184 Courbe1D* F1; // fonction de base
185 // variable intermédiaires pour la lecture en deux temps
186 // servent également ensuite pour dire si F1 est interne ou pas
187 // si elle est interne -> nom_courbei="i_interne_i", sinon ="e_externe_e"
188 string nom_courbel;
189
190 double ampli; // facteur d'amplification à chaque cycle
191 double longcycl; // longueur du cycle
192 double decalx,decaly; // decalage initiale
193
194 // METHODES PROTEGEES :
195
196 };
197 /// @} // end of group
198 #endif

```

## 7.527 F\_nD\_courbe1D (copy: conflict on 2017-11-21).h

```

1 // fichier : F_nD_courbe1D.h
2 // classe : F_nD_courbe1D
3
4
5 /*****
6 * UNIVERSITE DE BRETAGNE SUD (UBS) --- ENSIBS DE LORIENT *
7 *****/
8 * LIMATB- Equipe DE GENIE MECANIQUE ET MATERIAUX (EG2M) *
9 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
10 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
11 *****/
12 * DATE: 06/03/2023 *
13 * $ *
14 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
15 * phone 0297874573, fax : 0297874588 *
16 * $ *
17 * PROJET: Herezh++ *
18 * $ *
19 *****/
20 * BUT: Classe permettant d'utiliser une fonction courbe *
21 * à l'intérieur d'une fonction nD *
22 * $ *
23 * ***** *
24 * VERIFICATION: *
25 * *
26 * ! date ! auteur ! but ! *
27 * ----- *
28 * ! ! ! ! *
29 * $ *
30 * ***** *
31 * MODIFICATIONS: *
32 * ! date ! auteur ! but ! *
33 * ----- *
34 * $ *
35 *****/
36
37 #ifndef F_ND_COURBE_1D_H
38 #define F_ND_COURBE_1D_H
39
40 #include "Courbe1D.h"
41 #include "Fonction_nD.h"
42 #include "Tableau_T.h"
43
44 class F_nD_courbe1D : public Fonction_nD
45 {
46 public :
47
48 // CONSTRUCTEURS :
49 // constructeur par défaut
50 F_nD_courbe1D(string nom = "");

```

```

51
52 // de copie
53 F_nD_courbelD(const F_nD_courbelD& Co);
54 // de copie à partir d'une instance générale
55 F_nD_courbelD(const Fonction_nD& Co);
56
57 // DESTRUCTEUR :
58 ~F_nD_courbelD();
59
60 // METHODES PUBLIQUES :
61
62 // ----- virtuelles -----
63
64 // Surcharge de l'opérateur = : réalise l'égalité de deux fonctions
65 Fonction_nD& operator=(const Fonction_nD& elt);
66
67 // affichage de la courbe
68 // = 0, 1 ou 2 (le plus précis)
69 void Affiche(int niveau = 0) const;
70 // ramène true si ok, false sinon
71 bool Complet_Fonction()const;
72
73 // Lecture des données de la classe sur fichier
74 // le nom passé en paramètre est le nom de la Fonction
75 // s'il est vide c-a-d = "", la méthode commence par lire le nom sinon
76 // ce nom remplace le nom actuel
77 void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * );
78
79 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
80 // globales en cours de calcul
81 virtual void Mise_a_jour_variables_globales();
82
83 // def info fichier de commande
84 void Info_commande_Fonctions_nD(UtilLecture & entreePrinc) ;
85
86 // // calcul des valeurs de la fonction, retour d'un tableau de scalaires
87 // // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
88 // // NB: il peut y avoir plus d'info que nécessaire
89 // // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
90 // virtual Tableau <double> & Val_FnD_Evoluee(Tableau <double >* xi,Tableau <Coordonnee> * tab_coor
91 // // ,Tableau <TenseurBB* >* tab_tensBB );
92
93 // calcul équivalent, mais pour des paramètres de type ddl enum étendu et/ou type quelconque
94 // pour que l'appel à cette méthode soit correcte, il faut que la dimension de t_enu + celle de tqi
95 // soit identique à celle du tableau Nom_variables() : en fait t_enu et tqi doivent représenter les
    variables
96 virtual Tableau <double> & Valeur_FnD(Tableau <Ddl_etendu> * t_enu,Tableau <TypeQuelconque >* tqi);
97
98 // calcul des valeurs de la fonction, dans le cas où les variables
99 // sont des grandeurs globales: pour l'instant que pour des variables scalaires
100 virtual Tableau <double> & Valeur_pour_variables_globales();
101
102 // ramène le nombre de variable de la fonction
103 int NbVariable()const {return 1;};
104
105 // ramène le nombre de composantes de la fonction
106 int NbComposante() const {return 1;};
107
108 // établir le lien entre la fonction et une courbe ou une fonction déjà existante dont
109 // on connaît que le nom
110 // permet ainsi de compléter la fonction
111 // 1) renseigne si la courbe dépend d'une autre courbe ou non
112 bool DependAutreFoncCourbes() const;
113 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
114 list <string>& ListDependanceCourbes(list <string>& lico) const;
115 // 3) retourne une liste de nom correspondant aux noms de fonctions dont dépend *this
116 // ici cette fonction n'a pas cours, utilisation de la fonction par défaut
117 // list <string>& ListDependanceFonctions(list <string>& lifo) const;
118 // 4) établit la connection entre la demande de *this et les courbes et fonctions passées en
    paramètres
119 void Lien_entre_fonc_courbe (list <Fonction_nD *>& liptfunc,list <CourbelD *>& liptco);
120
121
122 //----- lecture écriture de restart -----
123 // cas donne le niveau de la récupération
124 // = 1 : on récupère tout
125 // = 2 : on récupère uniquement les données variables (supposées comme telles)
126 void Lecture_base_info(ifstream& ent,const int cas);
127 // cas donne le niveau de sauvegarde
128 // = 1 : on sauvegarde tout
129 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
130 void Ecriture_base_info(ofstream& sort,const int cas);
131 // sortie du schemaXML: en fonction de enu
132 void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu) ;
133
134
135

```

```

136
137 protected :
138 // VARIABLES PROTEGEES :
139 Courbe1D* c_Fi; // courbe 1D
140 // variable intermédiaires pour la lecture en deux temps
141 // servent également ensuite pour dire si Fl est interne ou pas
142 // si elle est interne -> nom_courbei="i_interne_i", sinon ="e_externe_e"
143 string nom_courbel;
144 // donc les différentes étapes possibles sont les suivantes:
145 // nom_courbel = ""; : après le constructeur par défaut
146 // nom_courbel = "chaîne" , avec c_Fi = NULL : la courbe interne n'est pas encore
147 // définie, et son nom = chaîne
148 // nom_courbel = "i_interne_i", avec c_Fi non NULL : on a une courbe interne définie en interne
149 // nom_courbel = "e_externe_e", avec c_Fi non NULL : on a une courbe interne
150 // qui pointe sur une courbe définie par ailleurs
151
152 // égal à celui du tableau nom_variables
153 Tableau <double> tab_ret; // le tableau de retour
154
155
156 // METHODES PROTEGEES :
157 // dans le cas où la courbe membre est une courbe externe
158 // méthode pour la définir
159 // la courbe est défini en interne que si la courbe argument est elle même
160 // une courbe locale. c'est-à-dire si c_Fi->NomCourbe() ="_" alors on recrée une courbe
161 // interne avec new pour c_Fi, sinon c_Fi=cc_Fi et pas de création;
162 // dans le cas où cc_Fi est NULL on passe, pas de traitement pour ce pointeur
163 void DefFoncCourbeMembre(Courbe1D* cc_Fi);
164
165 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
166 // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
167 // NB: il peut y avoir plus d'info que nécessaire
168 // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
169 // variables_locales :
170 // vrai : on utilise les tableaux locaux stockés dans la loi pour calculer la fonction
171 // faux : on utilise les tableaux passés en paramètre
172 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* val_ddl_enum,Tableau <Coordonnee >*
173   coord_ddl_enum
174   ,Tableau <TenseurBB >* tens_ddl_enum,bool variables_locales
175 );
176 };
177 #endif

```

## 7.528 F\_nD\_courbe1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : F_nD_courbe1D.h
30 // classe : F_nD_courbe1D
31
32
33 /*****
34 *   DATE:      01/06/2016
35 *
36 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 */

```

```

39 *                                                                                               $ *
40 *****
41 *   BUT: Classe permettant d'utiliser une fonction courbe *
42 *   à l'intérieur d'une fonction nD *
43 *                                                                                               $ *
44 *   //////////////////////////////////// *
45 *   VERIFICATION: *
46 * *
47 *   ! date ! auteur ! but *
48 *   ----- *
49 *   ! ! ! *
50 * *
51 *   //////////////////////////////////// *
52 *   MODIFICATIONS: *
53 *   ! date ! auteur ! but *
54 *   ----- *
55 * *
56 *****/
57
58 #ifndef F_ND_COURBE_1D_H
59 #define F_ND_COURBE_1D_H
60
61 #include "CourbelD.h"
62 #include "Fonction_nD.h"
63 #include "Tableau_T.h"
64
65 /// @addtogroup Les_fonctions_nD
66 /// @{
67 ///
68
69 /**
70 *
71 *
72 * \author Gérard Rio
73 * \version 1.0
74 * \date 01/06/2016
75 * \brief Classe permettant d'utiliser une fonction courbe à l'intérieur d'une fonction nD .
76 *
77 */
78
79 class F_nD_courbelD : public Fonction_nD
80 {
81 public :
82
83 // CONSTRUCTEURS :
84 // constructeur par défaut
85 F_nD_courbelD(string nom = "");
86
87 // de copie
88 F_nD_courbelD(const F_nD_courbelD& Co);
89 // de copie à partir d'une instance générale
90 F_nD_courbelD(const Fonction_nD& Co);
91
92 // DESTRUCTEUR :
93 ~F_nD_courbelD();
94
95 // METHODES PUBLIQUES :
96
97 // ----- virtuelles -----
98
99 // Surcharge de l'opérateur = : réalise l'égalité de deux fonctions
100 Fonction_nD& operator=(const Fonction_nD& elt);
101
102 // affichage de la courbe
103 // = 0, 1 ou 2 (le plus précis)
104 void Affiche(int niveau = 0) const ;
105 // ramène true si ok, false sinon
106 bool Complet_Fonction(bool affichage = true)const;
107
108 // Lecture des données de la classe sur fichier
109 // le nom passé en paramètre est le nom de la Fonction
110 // s'il est vide c-a-d = "", la méthode commence par lire le nom sinon
111 // ce nom remplace le nom actuel
112 void LECTDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * );
113
114 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
115 // globales en cours de calcul
116 virtual void Mise_a_jour_variables_globales();
117
118 // def info fichier de commande
119 void Info_commande_Fonctions_nD(UtilLecture & entreePrinc) ;
120
121 // // calcul des valeurs de la fonction, retour d'un tableau de scalaires
122 // // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
123 // // NB: il peut y avoir plus d'info que nécessaire
124 // // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
125 // virtual Tableau <double> & Val_FnD_Evoluee(Tableau <double> * xi,Tableau <Coordonnee> * tab_coor

```



```

126 //                                     ,Tableau <TenseurBB* >* tab_tensBB );
127
128 // // calcul équivalent, mais pour des paramètres de type ddl enum étendu et/ou type quelconque
129 // // pour que l'appel à cette méthode soit correcte, il faut que la dimension de t_enu + celle de
    tqi
130 // // soit identique à celle du tableau Nom_variables() : en fait t_enu et tqi doivent représenter
    les variables
131 // // virtual Tableau <double> & Valeur_FnD(Tableau <Ddl_etendu> * t_enu,Tableau <TypeQuelconque >*
    tqi);
132
133 // ramène le nombre de composantes de la fonction
134 int NbComposante() const {return 1;};
135
136 // établir le lien entre la fonction et une courbe ou une fonction déjà existante dont
137 // on connaît que le nom
138 // permet ainsi de compléter la fonction
139 // 1) renseigne si la courbe dépend d'une autre courbe ou non
140 bool DependAutreFoncCourbes() const;
141 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
142 list <string>& ListDependanceCourbes(list <string>& lico) const;
143 // 3) retourne une liste de nom correspondant aux noms de fonctions dont dépend *this
144 // ici cette fonction n'a pas cours, utilisation de la fonction par défaut
145 // list <string>& ListDependanceFonctions(list <string>& lifo) const;
146 // 4) établit la connexion entre la demande de *this et les courbes et fonctions passées en
    paramètres
147 void Lien_entre_fonc_courbe (list <Fonction_nD *>& liptfunc,list <Courbe1D *>& liptco);
148
149
150 //----- lecture écriture de restart -----
151 // cas donne le niveau de la récupération
152 // = 1 : on récupère tout
153 // = 2 : on récupère uniquement les données variables (supposées comme telles)
154 void Lecture_base_info(ifstream& ent,const int cas);
155 // cas donne le niveau de sauvegarde
156 // = 1 : on sauvegarde tout
157 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
158 void Ecriture_base_info(ofstream& sort,const int cas);
159 // sortie du schemaXML: en fonction de enu
160 void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu) ;
161
162
163
164
165 protected :
166 // VARIABLES PROTEGEES :
167 Courbe1D* c_Fi; // courbe 1D
168 // variable intermédiaires pour la lecture en deux temps
169 // servent également ensuite pour dire si Fi est interne ou pas
170 // si elle est interne -> nom_courbei="i_interne_i", sinon ="e_externe_e"
171 string nom_courbei;
172 // donc les différentes étapes possibles sont les suivantes:
173 // nom_courbei = ""; : après le constructeur par défaut
174 // nom_courbei = "chaine" , avec c_Fi = NULL : la courbe interne n'est pas encore
175 // définie, et son nom = chaine
176 // nom_courbei = "i_interne_i", avec c_Fi non NULL : on a une courbe interne définie en interne
177 // nom_courbei = "e_externe_e", avec c_Fi non NULL : on a une courbe interne
178 // qui pointe sur une courbe définie par ailleurs
179
180 // égal à celui du tableau nom_variables
181 Tableau <double> tab_ret; // le tableau de retour
182
183
184 // METHODES PROTEGEES :
185 // dans le cas où la courbe membre est une courbe externe
186 // méthode pour la définir
187 // la courbea est défini en interne que si la courbe argument est elle même
188 // une courbe locale. c'est-à-dire si c_Fi->NomCourbe() ="_" alors on recrée une courbe
189 // interne avec new pour c_Fi, sinon c_Fi=cc_Fi et pas de création;
190 // dans le cas où cc_Fi est NULL on passe, pas de traitement pour ce pointeur
191 void DefFoncCourbeMembre(Courbe1D* cc_Fi);
192
193 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
194 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* val_ddl_enum);
195
196 // calcul des valeurs de la fonction, dans le cas où les variables
197 // sont des grandeurs globales
198 virtual Tableau <double> & Valeur_pour_variables_globales_interne();
199
200 };
201 /// @} // end of group
202
203 #endif

```

## 7.529 F\_union\_1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : F_union_1D.h
30 // classe : F_union_1D
31
32
33 /*****
34 *   DATE:      14/10/2004
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *****/
41 *   BUT:  Classe permettant le calcul d'une fonction égale à
42 *         l'union d'une suite de fonctions définies sur des segments
43 *         contigus de manière à couvrir un segment global.
44 *
45 *   *****
46 *   VERIFICATION:
47 *
48 *   ! date !   auteur !   but
49 *   -----
50 *   !       !       !
51 *
52 *   *****
53 *   MODIFICATIONS:
54 *   ! date !   auteur !   but
55 *   -----
56 *
57 *****/
58
59 #ifndef F1_UNION_1D_H
60 #define F1_UNION_1D_H
61
62 #include "Courbe1D.h"
63 #include "Tableau_T.h"
64 /// @addtogroup Les_courbes_1D
65 /// @{
66 ///
67
68 /**
69 *
70 *   BUT:  Classe permettant le calcul d'une fonction égale à
71 *         l'union d'une suite de fonctions définies sur des segments
72 *         contigus de manière à couvrir un segment global.
73 *
74 *
75 * \author   Gérard Rio
76 * \version  1.0
77 * \date    14/10/2004
78 * \brief   Classe permettant le calcul d'une fonction égale à l'union d'une suite de fonctions
79 *          définies sur des segments contigus de manière à couvrir un segment global.
80 */
81
82 class F_union_1D : public Courbe1D
83 {
84 public :

```

```

85
86 // CONSTRUCTEURS :
87 // constructeur par défaut
88 F_union_1D(string nom = "");
89
90 // de copie
91 F_union_1D(const F_union_1D& Co);
92 F_union_1D(const CourbelD& Co);
93
94 // DESTRUCTEUR :
95 ~F_union_1D();
96
97 // METHODES PUBLIQUES :
98
99 // ----- virtuelles -----
100
101 // affichage de la courbe
102 void Affiche() const ;
103 // ramène true si ok, false sinon
104 bool Complet_courbe() const;
105
106 // Lecture des donnees de la classe sur fichier
107 // le nom passé en paramètre est le nom de la courbe
108 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
109 // ce nom remplace le nom actuel
110 void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
111
112 // établir le lien entre la courbe et une courbe déjà existante dont
113 // on connait que le nom
114 // permet ainsi de compléter la courbe
115 // 1) renseigne si la courbe dépend d'une autre courbe ou non
116 bool DependAutreCourbes() const;
117 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
118 list <string>& ListDependanceCourbes(list <string>& lico) const;
119 // 3) établit la connection entre la demande de *this et les courbes passées en paramètres
120 void Lien_entre_courbe (list <CourbelD *>& liptco);
121
122
123 // def info fichier de commande
124 void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
125
126 // ramène la valeur
127
128 double Valeur(double x) ;
129
130 // ramène la valeur et la dérivée en paramètre
131 CourbelD::ValDer Valeur_Et_derivee(double x) ;
132
133 // ramène la dérivée
134 double Derivee(double x) ;
135
136 // ramène la valeur et les dérivées première et seconde en paramètre
137 CourbelD::ValDer2 Valeur_Et_der12(double x);
138
139 // ramène la dérivée seconde
140 double Der_sec(double x);
141
142 // ramène la valeur si dans le domaine strictement de définition
143 // si c'est inférieur au x mini, ramène la valeur minimale possible de y
144 // si supérieur au x maxi , ramène le valeur maximale possible de y
145 CourbelD::Valbool Valeur_stricte(double x) ;
146
147 // ramène la valeur et la dérivée si dans le domaine strictement de définition
148 // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
149 // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
150 CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
151
152 //---- lecture écriture de restart ----
153 // cas donne le niveau de la récupération
154 // = 1 : on récupère tout
155 // = 2 : on récupère uniquement les données variables (supposées comme telles)
156 void Lecture_base_info(ifstream& ent,const int cas);
157 // cas donne le niveau de sauvegarde
158 // = 1 : on sauvegarde tout
159 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
160 void Ecriture_base_info(ofstream& sort,const int cas);
161 // sortie du schemaXML: en fonction de enu
162 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
163
164
165
166
167 protected :
168 // VARIABLES PROTEGEES :
169 Tableau <CourbelD* > Fi; // fonctions de base
170 Tableau <double > Xi; // définit les intervalles de définition des Fi
171 // Fi(i) est défini sur [Xi(i),Xi(i+1)]

```

```

172 // variable intermédiaires pour la lecture en deux temps
173 // servent également ensuite pour dire si Fi est interne ou pas
174 // si elle est interne -> nom_courbei="i_interne_i", sinon ="e_externe_e"
175 Tableau <string > nom_courbei;
176 int indice_precedant; // position précédente
177
178 // METHODES PROTEGEES :
179 // dans le cas où les courbes membres sont des courbes externes
180 // fonction pour les définir
181 // les courbes sont défini en interne que si les courbes argument sont elles même
182 // des courbes locales. c'est-à-dire si FFi(i)->NomCourbe() ="_" alors on recrée une courbe
183 // interne avec new pour Fi(i), sinon Fi(i)=FFi(i) et pas de création;
184 // dans le cas où FFi(i) est NULL on passe, pas de traitement pour ce pointeur
185 void DefCourbesMembres(Tableau <CourbeId* >& FFi);
186
187 };
188 /// @} // end of group
189 #endif

```

## 7.530 Fonc\_scal\_combinees\_nD (copy: conflict on 2017-11-21).h

```

1 // fichier : Fonc_scal_combinees_nD.h
2 // classe : Fonc_scal_combinees_nD
3
4
5 /*****
6 * UNIVERSITE DE BRETAGNE SUD (UBS) --- I.U.P/I.U.T. DE LORIENT *
7 *****/
8 * LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M) *
9 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
10 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
11 *****/
12 * DATE: 19/01/2001 *
13 * $ *
14 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
15 * Tel 0297874571 fax : 02.97.87.45.72 *
16 * $ *
17 * PROJET: Herezh++ *
18 * $ *
19 *****/
20 * BUT: Classe permettant le calcul d'une fonction scalaire nD *
21 * correspondant à une combinaison d'autres fonctions scalaire nD. *
22 * mentaires telles que dérivées. *
23 * La combinaison est définie de manière littérale. *
24 * On utilise pour cela la bibliothèque MuParser *
25 * $ *
26 * ***** *
27 * *
28 * VERIFICATION: *
29 * *
30 * ! date ! auteur ! but ! *
31 * ----- *
32 * ! ! ! ! *
33 * $ *
34 * ***** *
35 * MODIFICATIONS: *
36 * ! date ! auteur ! but ! *
37 * ----- *
38 * $ *
39 *****/
40
41 #ifndef FONCTION_SCAL_COMBINEES_ND_H
42 #define FONCTION_SCAL_COMBINEES_ND_H
43
44 #include "Fonction_nD.h"
45 #include "muParser.h"
46
47 class Fonc_scal_combinees_nD : public Fonction_nD
48 {
49 public :
50
51 // CONSTRUCTEURS :
52 // par défaut
53 Fonc_scal_combinees_nD(string nom = "");
54 // constructeur avec plus d'info
55 Fonc_scal_combinees_nD(const Tableau <string >& var, string nom = "");
56 // de copie
57 Fonc_scal_combinees_nD(const Fonc_scal_combinees_nD& Co);
58 // de copie à partir d'une instance générale
59 Fonc_scal_combinees_nD(const Fonction_nD& Co);
60 // DESTRUCTEUR :
61 virtual ~Fonc_scal_combinees_nD() ;
62
63 // METHODES PUBLIQUES :
64

```

```

65 // ----- virtuelles -----
66
67 // Surcharge de l'operateur = : realise l'egalite de deux fonctions
68 Fonction_nD& operator= (const Fonction_nD& elt);
69
70 // affichage de la Fonction
71 // = 0, 1 ou 2 (le plus précis)
72 void Affiche(int niveau = 0) const ;
73
74 // vérification que tout est ok, pres à l'emploi
75 // ramène true si ok, false sinon
76 bool Completet_Fonction()const;
77
78
79 // ramène le nombre de composantes de la fonction
80 int NbComposante() const {return 1;};
81
82 // Lecture des donnees de la classe sur fichier
83 // le nom passé en paramètre est le nom de la Fonction
84 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
85 // ce nom remplace le nom actuel
86 void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * );
87
88 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
89 // globales en cours de calcul
90 virtual void Mise_a_jour_variables_globales();
91
92 // établir le lien entre la Fonction et des Fonctions déjà existantes dont
93 // on connait que le nom
94 // permet ainsi de compléter la Fonction
95 // 1) renseigne si la Fonction dépend d'autre Fonction ou non
96 bool DependAutreFoncCourbes() const;
97 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
98 list <string>& ListDependanceCourbes(list <string>& lico) const;
99 // 3) retourne une liste de nom correspondant aux noms de Fonction dont dépend *this
100 list <string>& ListDependanceFonctions(list <string>& lico) const;
101 // 4) établit la connection entre la demande de *this et les Fonctions passées en paramètres
102 void Lien_entre_fonc_courbe(list <Fonction_nD *>& liptfonc,list <CourbelD *>& liptco);
103
104 // def info fichier de commande
105 void Info_commande_Fonctions_nD(UtilLecture & entreePrinc);
106
107 // calcul des valeurs de la fonction, dans le cas où les variables
108 // sont des grandeurs globales
109 Tableau <double> & Valeur_pour_variables_globales();
110
111
112 //----- lecture écriture de restart -----
113 // cas donne le niveau de la récupération
114 // = 1 : on récupère tout
115 // = 2 : on récupère uniquement les données variables (supposées comme telles)
116 void Lecture_base_info(ifstream& ent,const int cas);
117 // cas donne le niveau de sauvegarde
118 // = 1 : on sauvegarde tout
119 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
120 void Ecriture_base_info(ofstream& sort,const int cas);
121 // sortie du schemaXML: en fonction de enu
122 void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu);
123
124
125 protected :
126
127 // VARIABLES PROTEGEES :
128 //-----
129 // on considère que l'utilisateur donne une liste d'arguments
130 // qui sont dédiés uniquement à l'expression globale,
131 // car les variables des fonctions internes sont définies
132 // au moment de la définition (lecture) des lois internes
133 // en lecture, il y a détection des variables globales
134 // celles-ci sont rangées dans le tableau enu_variables défini
135 // dans Fonction_nD.h
136 //
137 // par contre au moment de l'appel de la fonction, toutes les variables
138 // nécessaires sont stockés en entrée dans le tableau tab_fVal
139 // NB: ne sont pas incluses les variables globales par ce qu'elles
140 // sont accessibles directement
141 //
142 // l'ordre des variables est le suivant
143 // 1) les variables internes à chaque fonction membre, en suivant
144 // l'ordre d'apparition des fonctions
145 // 2) les variables de la fonction globale
146 //-----
147
148 // 1) la fonction globale
149 string expression_fonction; // l'expression littérale de la fonction
150 mu::Parser p; // définition d'une instance de parser
151 // ici on fait une différence entre :

```

```

152
153 // a) les variables spécifique à "expression_fonction"
154 // ces variables sont dans l'ordre:
155 // . l'identificateur de chaque fonction. Cette
156 // identificateur intervient comme une variable
157 // dans l'appel de l'expression
158 // . les variables patentées de passage
159 // . les variables patentées globales
160 //
161 // du coup:
162 // . les premières valeurs de tab_fVal_int
163 // correspondent aux retours des fonctions
164 // . les dernières aux valeurs des variables patentées
165 Tableau <double> tab_fVal_int; // les variables internes dont
166 // le nombre doit être égal à celui du tableau nom_variables_int
167 Tableau <string > nom_variables_int; // nom fct int + variables de la fonction
168 // de passage et globale
169
170 // b) l'ensemble des variables de la fonction, vu de l'extérieur
171 // qui agglomère dans l'ordre:
172 // . les variables de chaque fonction interne
173 // . + les variables internes à la fonction globale
174 Tableau <double> tab_fVal; // l'ensemble des variables dont le nombre est
175 // égal à celui du tableau nom_variables
176 Tableau <double> tab_ret; // le tableau final de retour
177
178
179 // 2) les fonctions internes
180 Tableau <Fonction_nD* > Fi; // fonctions de base
181
182 // variable intermédiaires pour la lecture en deux temps
183 // servent également ensuite pour dire si Fi est interne ou pas
184 // si elle est interne -> nom_fonctioni="i_interne_i",
185 // sinon ="e_externe_e" tant que le nom de la fonction n'est pas définie
186 // après : LectDonnParticulieres_Fonction_nD :
187 // nom_fonctioni(i) = soit "i_interne_i" sinon le nom de la fonction, qu'elle
188 // soit ou nom définie
189 Tableau <string > nom_fonctioni;
190 // dans le cas où on a affaire à une fonction interne
191 // pour pouvoir l'utiliser dans l'expression littérale
192 // on définit un identificateur associé: ident_interne(i)
193 // s'il s'agit d'une fonction externe : ident_interne(i) = le nom de la fonction;
194 Tableau <string> ident_interne;
195 //--- maintenant les variables de passage
196 Tableau < Tableau <double> > tab_fVal_Fi; // les variables internes pour chaque fonction interne
197 Tableau < Tableau <double> > tab_ret_Fi; // le tableau de retour
198
199
200 int indice_precedant; // position précédente
201
202 // METHODES PROTEGEES :
203 // dans le cas où les fonctions membres sont des fonctions externes
204 // fonction pour les définir
205 // les fonctions sont défini en interne que si les fonctions argument sont elles même
206 // des fonctions locales. c'est-à-dire si FFi(i)->NomFonction() ="_" alors on recrée une fonction
207 // interne avec new pour Fi(i), sinon Fi(i)=FFi(i) et pas de création;
208 // dans le cas où FFi(i) est NULL on passe, pas de traitement pour ce pointeur
209 void DefFonctionsMembres(Tableau <Fonction_nD* >& FFi);
210
211 // initialisation de la fonction analytique
212 void Init_fonction_analytique();
213
214 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
215 // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
216 // NB: il peut y avoir plus d'info que nécessaire
217 // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
218 // vrai : on utilise les tableaux locaux stockés dans la loi pour calculer la fonction
219 // faux : on utilise les tableaux passés en paramètre
220 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* xi,Tableau <Coordonnee> * tab_coor
221 ,Tableau <TenseurBB* >* tab_tensBB,bool variables_locales);
222
223 };
224
225 #endif

```

## 7.531 Fonc\_scal\_combinees\_nD.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.

```

```

9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Fonc_scal_combinees_nD.h
30 // classe : Fonc_scal_combinees_nD
31
32
33 /*****
34 *      DATE:          01/06/2016
35 *
36 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:        Herezh++
39 *
40 *
41 *      *****
42 *      BUT:          Classe permettant le calcul d'une fonction scalaire nD
43 *      correspondant à une combinaison d'autres fonctions scalaire nD.
44 *      La combinaison est définie de manière littérale.
45 *      On utilise pour cela la bibliothèque MuParser
46 *
47 *      *****
48 *      VERIFICATION:
49 *
50 *      ! date !   auteur !           but
51 *      -----
52 *      !           !           !
53 *
54 *      *****
55 *      MODIFICATIONS:
56 *      ! date !   auteur !           but
57 *      -----
58 *
59 *      *****/
60
61 #ifndef FONCTION_SCAL_COMBINEES_ND_H
62 #define FONCTION_SCAL_COMBINEES_ND_H
63
64 #include "Fonction_nD.h"
65 #include "muParser.h"
66
67 /// @addtogroup Les_fonctions_nD
68 /// @{
69 ///
70
71 /**
72 *
73 *      BUT:          Classe permettant le calcul d'une fonction scalaire nD
74 *      correspondant à une combinaison d'autres fonctions scalaire nD.
75 *      La combinaison est définie de manière littérale.
76 *      On utilise pour cela la bibliothèque MuParser
77 *
78 *
79 * \author          Gérard Rio
80 * \version         1.0
81 * \date            01/06/2016
82 * \brief           Classe permettant le calcul d'une fonction scalaire nD
83 *                  correspondant à une combinaison d'autres fonctions scalaire nD.
84 *
85 */
86
87 class Fonc_scal_combinees_nD : public Fonction_nD
88 {
89 public :
90
91     // CONSTRUCTEURS :
92     // par défaut
93     Fonc_scal_combinees_nD(string nom = "");
94     // constructeur avec plus d'info
95     Fonc_scal_combinees_nD(const Tableau <string >& var, string nom = "");

```

```

96 // de copie
97 Fonc_scalcombinees_nD(const Fonc_scalcombinees_nD& Co);
98 // de copie à partir d'une instance générale
99 Fonc_scalcombinees_nD(const Fonction_nD& Co);
100 // DESTRUCTEUR :
101 virtual ~Fonc_scalcombinees_nD() ;
102
103 // METHODES PUBLIQUES :
104
105 // ----- virtuelles -----
106
107 // Surcharge de l'opérateur = : réalise l'égalité de deux fonctions
108 Fonction_nD& operator= (const Fonction_nD& elt);
109
110 // affichage de la Fonction
111 // = 0, 1 ou 2 (le plus précis)
112 void Affiche(int niveau = 0) const ;
113
114 // vérification que tout est ok, pres à l'emploi
115 // ramène true si ok, false sinon
116 bool Complet_Fonction(bool affichage = true) const;
117
118
119 // ramène le nombre de composantes de la fonction
120 int NbComposante() const {return 1;};
121
122 // Lecture des données de la classe sur fichier
123 // le nom passé en paramètre est le nom de la Fonction
124 // s'il est vide c-a-d = "", la méthode commence par lire le nom sinon
125 // ce nom remplace le nom actuel
126 void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * );
127
128 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
129 // globales en cours de calcul
130 virtual void Mise_a_jour_variables_globales();
131
132 // établir le lien entre la Fonction et des Fonctions déjà existantes dont
133 // on connaît que le nom
134 // permet ainsi de compléter la Fonction
135 // 1) renseigne si la Fonction dépend d'autre Fonction ou non
136 bool DependAutreFoncCourbes() const;
137 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
138 list <string>& ListDependanceCourbes(list <string>& lico) const;
139 // 3) retourne une liste de nom correspondant aux noms de Fonction dont dépend *this
140 list <string>& ListDependanceFonctions(list <string>& lico) const;
141 // 4) établit la connexion entre la demande de *this et les Fonctions passées en paramètres
142 void Lien_entre_fonc_courbe(list <Fonction_nD *>& liptfonc, list <CourbeID *>& liptco);
143
144 // def info fichier de commande
145 void Info_commande_Fonctions_nD(UtilLecture & entreePrinc);
146
147 //----- lecture écriture de restart -----
148 // cas donne le niveau de la récupération
149 // = 1 : on récupère tout
150 // = 2 : on récupère uniquement les données variables (supposées comme telles)
151 void Lecture_base_info(ifstream& ent, const int cas);
152 // cas donne le niveau de sauvegarde
153 // = 1 : on sauvegarde tout
154 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
155 void Ecriture_base_info(ofstream& sort, const int cas);
156 // sortie du schemaXML: en fonction de enu
157 void SchemaXML_Fonctions_nD(ofstream& sort, const Enum_IO_XML enu);
158
159
160 protected :
161
162 // VARIABLES PROTEGEES :
163 //-----
164 // on considère que l'utilisateur donne une liste d'arguments
165 // qui sont dédiés uniquement à l'expression globale,
166 // car les variables des fonctions internes sont définies
167 // au moment de la définition (lecture) des lois internes
168 // en lecture, il y a détection des variables globales
169 // celles-ci sont rangées dans le tableau enu_variables défini
170 // dans Fonction_nD.h
171 //
172 // par contre au moment de l'appel de la fonction, toutes les variables
173 // nécessaires sont stockés en entrée dans le tableau tab_fVal
174 // NB: ne sont pas incluses les variables globales par ce qu'elles
175 // sont accessibles directement
176 //
177 // l'ordre des variables est le suivant
178 // 1) les variables internes à chaque fonction membre, en suivant
179 // l'ordre d'apparition des fonctions
180 // 2) les variables de la fonction générale
181 //-----
182

```



```

183 // 1) la fonction globale
184 string expression_fonction; // l'expression littérale de la fonction
185 mu::Parser p; // définition d'une instance de parser
186 // ici on fait une différence entre :
187
188 // a) les variables spécifique à "expression_fonction"
189 // ces variables sont dans l'ordre:
190 // . l'identificateur de chaque fonction. Cette
191 // identificateur intervient comme une variable
192 // dans l'appel de l'expression
193 // . les variables patentées de passage
194 // . les variables patentées globales
195 //
196 // du coup:
197 // . les premières valeurs de tab_fVal_int
198 // correspondent aux retours des fonctions
199 // . les dernières aux valeurs des variables patentées
200 Tableau <double> tab_fVal_int; // les variables internes dont
201 // le nombre doit être égal à celui du tableau nom_variables_int
202 // + les variables globales (enum et nom)
203 Tableau <string > nom_variables_int; // nom fct int + variables de la fonction
204 // "sans" les variables globales !!
205
206 Tableau <Enum_GrandeurGlobale > enu_variables_globale_int; //tableau des énumérés
207 // de variables globales correspondant à la fonction globale
208 // éventuellement vide s'il ne sert pas
209 Tableau <string > nom_variables_globales_int; //tableau des noms en string
210 // de variables globales correspondant à la fonction globale
211 // éventuellement vide s'il ne sert pas
212
213
214 // b) l'ensemble des variables de la fonction, vu de l'extérieur
215 // qui agglomère dans l'ordre:
216 // . les variables de chaque fonction interne
217 // . + les variables internes à la fonction globale
218 Tableau <double> tab_fVal; // l'ensemble des variables dont le nombre est
219 // égal à celui du tableau nom_variables
220 Tableau <double> tab_ret; // le tableau final de retour
221
222
223 // 2) les fonctions internes
224 Tableau <Fonction_nD* > Fi; // fonctions de base
225
226 // variable intermédiaires pour la lecture en deux temps
227 // servent également ensuite pour dire si Fi est interne ou pas
228 // si elle est interne -> nom_fonctioni="i_interne_i",
229 // sinon ="e_externe_e" tant que le nom de la fonction n'est pas définie
230 // après : LectDonnParticulieres_Fonction_nD :
231 // nom_fonctioni(i) = soit "i_interne_i" sinon le nom de la fonction, qu'elle
232 // soit ou nom définie
233 Tableau <string > nom_fonctioni;
234 // dans le cas où on a affaire à une fonction interne
235 // pour pouvoir l'utiliser dans l'expression littérale
236 // on définit un identificateur associé: ident_interne(i)
237 // s'il s'agit d'une fonction externe : ident_interne(i) = le nom de la fonction;
238 Tableau <string> ident_interne;
239 //--- maintenant les variables de passage
240 Tableau < Tableau <double> > tab_fVal_Fi; // les variables internes pour chaque fonction interne
241 Tableau < Tableau <double> > tab_ret_Fi; // le tableau de retour
242
243
244 int indice_precedant; // position précédente
245
246 // METHODES PROTEGEES :
247 // dans le cas où les fonctions membres sont des fonctions externes
248 // fonction pour les définir
249 // les fonctions sont défini en interne que si les fonctions argument sont elles même
250 // des fonctions locales. c'est-à-dire si FFi(i)->NomFonction() ="_" alors on recrée une fonction
251 // interne avec new pour Fi(i), sinon Fi(i)=FFi(i) et pas de création;
252 // dans le cas où FFi(i) est NULL on passe, pas de traitement pour ce pointeur
253 void DefFonctionsMembres(Tableau <Fonction_nD* > & FFi);
254
255 // mise à jour des variables de la classe mère en fonction des fonctions membres
256 void Mise_a_jour_variables_nD();
257
258 // initialisation de la fonction analytique
259 void Init_fonction_analytique();
260
261 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
262 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* xi);
263
264 // calcul des valeurs de la fonction, dans le cas où les variables
265 // sont des grandeurs globales
266 virtual Tableau <double> & Valeur_pour_variables_globales_interne();
267
268
269 };

```

```

270 ///  

271 ///  

272 #endif

```

## 7.532 Fonction\_expression\_litterale\_nD (copy: conflict on 2017-11-21).h

```

1 // fichier : Fonction_expression_litterale_nD.h
2 // classe : Fonction_expression_litterale_nD
3
4
5 /*****
6 * UNIVERSITE DE BRETAGNE SUD (UBS) --- ENSIBS DE LORIENT *
7 *****/
8 * LIMATB- Equipe DE GENIE MECANIQUE ET MATERIAUX (EG2M) *
9 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
10 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
11 *****/
12 * DATE: 06/03/2023 *
13 * $ *
14 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
15 * phone 0297874573, fax : 0297874588 *
16 * $ *
17 * PROJET: Herezh++ *
18 * $ *
19 *****/
20 * BUT: Classe permettant le calcul d'une fonction nD *
21 * de type : une expression littérale, exploitée ensuite *
22 * par le parser : muparser *
23 * $ *
24 * ***** *
25 * *
26 * VERIFICATION: *
27 * *
28 * ! date ! auteur ! but ! *
29 * ----- *
30 * ! ! ! ! *
31 * $ *
32 * ***** *
33 * MODIFICATIONS: *
34 * ! date ! auteur ! but ! *
35 * ----- *
36 * $ *
37 *****/
38
39 #ifndef FONCTION_EXPRESSION_LITTERALE_ND_H
40 #define FONCTION_EXPRESSION_LITTERALE_ND_H
41
42 #include "Fonction_nD.h"
43 #include "muParser.h"
44
45
46 class Fonction_expression_litterale_nD : public Fonction_nD
47 {
48 public :
49
50 // CONSTRUCTEURS :
51 Fonction_expression_litterale_nD(string nom = "");
52
53 // de copie
54 Fonction_expression_litterale_nD(const Fonction_expression_litterale_nD& Co);
55 Fonction_expression_litterale_nD(const Fonction_nD& Co);
56
57 // DESTRUCTEUR :
58 ~Fonction_expression_litterale_nD();
59
60 // METHODES PUBLIQUES :
61
62 // ----- virtuelles -----
63
64 // Surcharge de l'operateur = : realise l'egalite de deux fonctions
65 Fonction_nD& operator= (const Fonction_nD& elt);
66
67 // affichage de la Fonction
68 // = 0, 1 ou 2 (le plus précis)
69 void Affiche(int niveau = 0) const ;
70 // ramène true si ok, false sinon
71 bool Complet_Fonction()const;
72
73 // Lecture des donnees de la classe sur fichier
74 // le nom passé en paramètre est le nom de la Fonction
75 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
76 // ce nom remplace le nom actuel
77 void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * );
78
79 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables

```

```

80 // globales en cours de calcul
81 virtual void Mise_a_jour_variables_globales();
82
83 // def info fichier de commande
84 void Info_commande_Fonctions_nD(UtilLecture & entreePrinc) ;
85
86 // calcul des valeurs de la fonction, dans le cas où les variables
87 // sont toutes des grandeurs globales: pour l'instant que pour des variables scalaires
88 virtual Tableau <double> & Valeur_pour_variables_globales();
89
90 // ramène le nombre de variable de la fonction
91 int NbVariable()const {return tab_fVal.Taille();};
92
93 // ramène le nombre de composantes de la fonction
94 int NbComposante() const {return tab_ret.Taille();};
95
96 //----- lecture écriture de restart -----
97 // cas donne le niveau de la récupération
98 // = 1 : on récupère tout
99 // = 2 : on récupère uniquement les données variables (supposées comme telles)
100 void Lecture_base_info(ifstream& ent,const int cas);
101 // cas donne le niveau de sauvegarde
102 // = 1 : on sauvegarde tout
103 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
104 void Ecriture_base_info(ofstream& sort,const int cas);
105 // sortie du schemaXML: en fonction de enu
106 virtual void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu);
107
108 protected :
109 // VARIABLES PROTEGEES :
110
111 string expression_fonction; // l'expression littérale de la fonction
112
113 mu::Parser p; // définition d'une instance de parser
114 Tableau <double> tab_fVal; // les variables internes dont le nombre doit être
115 // égal à celui du tableau nom_variables
116 Tableau <double> tab_ret; // le tableau de retour
117
118
119
120 // METHODES PROTEGEES :
121
122 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
123 // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
124 // NB: il peut y avoir plus d'info que nécessaire
125 // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
126 // vrai : on utilise les tableaux locaux stockés dans la loi pour calculer la fonction
127 // faux : on utilise les tableaux passés en paramètre
128 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* xi,Tableau <Coordonnee> * tab_coor
129 ,Tableau <TenseurBB* >* tab_tensBB,bool variables_locales);
130
131
132
133
134
135 };
136
137 #endif

```

## 7.533 Fonction\_expression\_litterale\_nD.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License

```

```

25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Fonction_expression_litterale_nD.h
30 // classe : Fonction_expression_litterale_nD
31
32
33 /*****
34 *   DATE:      01/06/2016
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *
41 *   BUT:       Classe permettant le calcul d'une fonction nD
42 *             de type : une expression littérale, exploitée ensuite
43 *             par le parser : muparser
44 *
45 *   *****
46 *
47 *   VERIFICATION:
48 *
49 *   ! date !   auteur !       but
50 *   -----
51 *   !       !       !
52 *
53 *   *****
54 *   MODIFICATIONS:
55 *   ! date !   auteur !       but
56 *   -----
57 *
58 *   *****/
59
60 #ifndef FONCTION_EXPRESSION_LITTERALE_ND_H
61 #define FONCTION_EXPRESSION_LITTERALE_ND_H
62
63 #include "Fonction_nD.h"
64 #include "muParser.h"
65
66
67 /// @addtogroup Les_fonctions_nD
68 /// @{
69 ///
70
71 /**
72 *
73 *   BUT:       Classe permettant le calcul d'une fonction nD
74 *             de type : une expression littérale, exploitée ensuite
75 *             par le parser : muparser
76 *
77 *
78 * \author      Gérard Rio
79 * \version     1.0
80 * \date       01/06/2016
81 * \brief      Classe permettant le calcul d'une fonction nD de type : une expression littérale
82 *
83 */
84
85 class Fonction_expression_litterale_nD : public Fonction_nD
86 {
87 public :
88
89     // CONSTRUCTEURS :
90     // def uniquement du nom, et init par défaut -> après ce constructeur
91     // la fonction doit-être renseignée pour être utilisable
92     Fonction_expression_litterale_nD(string nom = "");
93
94     // def de toutes les grandeurs -> donne une fonction utilisable
95     Fonction_expression_litterale_nD
96     ( string nom_ref // nom de ref de la fonction
97     ,Tableau <string >& nom_variables // les variables non globales
98     ,Tableau <Enum_GrandeurGlobale >& enu_variables_globale // enu globaux
99     ,Tableau <string >& nom_variables_globales // idem sous forme de strings
100     ,string& expression_fonction); // la formule de la fonction
101
102     // de copie
103     Fonction_expression_litterale_nD(const Fonction_expression_litterale_nD& Co);
104     Fonction_expression_litterale_nD(const Fonction_nD& Co);
105
106     // DESTRUCTEUR :
107     ~Fonction_expression_litterale_nD();
108
109     // METHODES PUBLIQUES :
110
111     // ----- virtuelles -----

```

```

112
113 // Surcharge de l'operateur = : realise l'egalite de deux fonctions
114 Fonction_nD& operator= (const Fonction_nD& elt);
115
116 // affichage de la Fonction
117 // = 0, 1 ou 2 (le plus précis)
118 void Affiche(int niveau = 0) const ;
119 // ramène true si ok, false sinon
120 bool Complet_Fonction(bool affichage = true) const;
121
122 // Lecture des donnees de la classe sur fichier
123 // le nom passé en paramètre est le nom de la Fonction
124 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
125 // ce nom remplace le nom actuel
126 void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * );
127
128 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
129 // globales en cours de calcul
130 virtual void Mise_a_jour_variables_globales();
131
132 // def info fichier de commande
133 void Info_commande_Fonctions_nD(UtilLecture & entreePrinc) ;
134
135 // --- erreur il faut utiliser la fonction mère
136 // // ramène le nombre de variable de la fonction
137 // int NbVariable() const {return tab_fVal.Taille();};
138
139 // ramène le nombre de composantes de la fonction
140 int NbComposante() const {return tab_ret.Taille();};
141
142 //----- lecture écriture de restart -----
143 // cas donne le niveau de la récupération
144 // = 1 : on récupère tout
145 // = 2 : on récupère uniquement les données variables (supposées comme telles)
146 void Lecture_base_info(ifstream& ent,const int cas);
147 // cas donne le niveau de sauvegarde
148 // = 1 : on sauvegarde tout
149 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
150 void Ecriture_base_info(ofstream& sort,const int cas);
151 // sortie du schemaXML: en fonction de enu
152 virtual void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu);
153
154 protected :
155 // VARIABLES PROTEGEES :
156
157 string expression_fonction; // l'expression littérale de la fonction
158
159 mu::Parser p; // définition d'une instance de parser
160 Tableau <double> tab_fVal; // les variables internes dont le nombre doit être
161 // égal à celui du tableau nom_variables + les enum globaux + les noms globaux
162 Tableau <double> tab_ret; // le tableau de retour
163
164 // METHODES PROTEGEES :
165
166 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
167 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* xi);
168
169 // calcul des valeurs de la fonction, dans le cas où les variables
170 // sont "tous" des grandeurs globales
171 virtual Tableau <double> & Valeur_pour_variables_globales_interne();
172
173 // initialisation de la fonction analytique
174 void Init_fonction_analytique();
175
176 };
177 /// @} // end of group
178
179 #endif

```

## 7.534 Fonction\_externe\_nD.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify

```

```

15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Fonction_externe_nD.h
30 // classe : Fonction_externe_nD
31
32
33 /*****
34 *   DATE:           01/06/2016
35 *
36 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:        Herezh++
39 *
40 *
41 *   BUT:           Classe permettant le calcul d'une fonction nD
42 *                 externe, définie par l'utilisateur, via 2 pipes nommés.
43 *
44 *   *****
45 *
46 *   VERIFICATION:
47 *
48 *   ! date ! auteur ! but
49 *   -----
50 *   ! ! !
51 *
52 *   *****
53 *   MODIFICATIONS:
54 *   ! date ! auteur ! but
55 *   -----
56 *
57 *   *****/
58
59 #ifndef FONCTION_EXTERNE_ND_H
60 #define FONCTION_EXTERNE_ND_H
61
62 #include "Fonction_nD.h"
63
64 /// @addtogroup Les_fonctions_nD
65 /// @{
66 ///
67
68 /// définition d'une union qui lie les réels, les entiers et les caractères
69 union Tab_car_double_int_1
70 { char tampon[928];
71   double x[116];
72   int n[232];
73 };
74 /// @} // end of group
75
76 /// @addtogroup Les_fonctions_nD
77 /// @{
78 ///
79
80 /**
81 *
82 *
83 * \author Gérard Rio
84 * \version 1.0
85 * \date 01/06/2016
86 * \brief Classe permettant le calcul d'une fonction nD externe, définie par l'utilisateur, via 2
87 *        pipes nommés.
88 */
89
90 class Fonction_externe_nD : public Fonction_nD
91 {
92 public :
93
94 // CONSTRUCTEURS :
95 // def uniquement du nom, et init par défaut -> après ce constructeur
96 // la fonction doit-être renseignée pour être utilisable
97 Fonction_externe_nD(string nom = "");
98
99 // def de toutes les grandeurs -> donne une fonction utilisable
100 Fonction_externe_nD

```

```

101         ( string nom_ref // nom de ref de la fonction
102         ,Tableau <string >& nom_variables // les variables non globales
103         ,Tableau <Enum_GrandeurGlobale >& enu_variables_globale // enu globaux
104         ,Tableau <string >& nom_variables_globales // idem sous forme de strings
105         ,int nb_double_ret); // nombre de grandeur en retour de fonction externe
106
107     // de copie
108     Fonction_externe_nD(const Fonction_externe_nD& Co);
109     Fonction_externe_nD(const Fonction_nD& Co);
110
111     // DESTRUCTEUR :
112     ~Fonction_externe_nD();
113
114     // METHODES PUBLIQUES :
115
116     // ----- virtuelles -----
117
118     // Surcharge de l'operateur = : realise l'egalite de deux fonctions
119     Fonction_nD& operator=(const Fonction_nD& elt);
120
121     // affichage de la Fonction
122     // = 0, 1 ou 2 (le plus précis)
123     void Affiche(int niveau = 0) const;
124     // ramène true si ok, false sinon
125     bool Complet_Fonction(bool affichage = true) const;
126
127     // Lecture des donnees de la classe sur fichier
128     // le nom passé en paramètre est le nom de la Fonction
129     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
130     // ce nom remplace le nom actuel
131     void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture *);
132
133     // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
134     // globales en cours de calcul
135     virtual void Mise_a_jour_variables_globales();
136
137     // def info fichier de commande
138     void Info_commande_Fonctions_nD(UtilLecture & entreePrinc);
139
140     // --- erreur il faut utiliser la fonction mère
141     // // ramène le nombre de variable de la fonction
142     // int NbVariable()const {return tab_fVal.Taille();};
143
144     // ramène le nombre de composantes de la fonction
145     int NbComposante() const {return tab_ret.Taille();};
146
147     //----- lecture écriture de restart -----
148     // cas donne le niveau de la récupération
149     // = 1 : on récupère tout
150     // = 2 : on récupère uniquement les données variables (supposées comme telles)
151     void Lecture_base_info(ifstream& ent,const int cas);
152     // cas donne le niveau de sauvegarde
153     // = 1 : on sauvegarde tout
154     // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
155     void Ecriture_base_info(ofstream& sort,const int cas);
156     // sortie du schemaXML: en fonction de enu
157     virtual void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu);
158
159 protected :
160     // VARIABLES PROTEGEES :
161
162
163     // -- variables pour les tubes nommés -----
164
165     // nom du tube nommé pour l'envoi des données
166     string envoi;
167     // nom du tube nommé pour la reception des données
168     string reception;
169     int passage; // test pour nombre de passage dans l'appel
170     Tab_car_double_int_1 t_car_x_n; // tableau de caractères, réels et entiers
171
172
173
174     // mu::Parser p; // définition d'une instance de parser
175     Tableau <double> tab_fVal; // les variables internes dont le nombre doit être
176     // égal à celui du tableau nom_variables + les enum globaux + les noms globaux
177     Tableau <double> tab_ret; // le tableau de retour
178
179     // METHODES PROTEGEES :
180
181     // calcul des valeurs de la fonction, retour d'un tableau de scalaires
182     virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* xi);
183
184     // calcul des valeurs de la fonction, dans le cas où les variables
185     // sont "tous" des grandeurs globales
186     virtual Tableau <double> & Valeur_pour_variables_globales_interne();
187

```

```

188 // initialisation des pipes
189 void InitialisationPipesNommes();
190
191 // lecture des données sur le pipe
192 void LectureResultatsPipe();
193 // écriture des données sur le pipe
194 void EcritureDonneesPipe();
195
196
197 // changement du nom des pipes
198 void Change_nom_pipe_envoi(const string& env) {envoi=env;};
199 void Change_nom_pipe_reception(const string& recep) {reception=recep;};
200
201
202 };
203 ///< @} // end of group
204
205 #endif

```

## 7.535 Fonction\_nD (conflict on 2019-05-08).h

```

1 // fichier : Fonction_nD.h
2 // classe : Fonction_nD
3
4
5 /*****
6 * UNIVERSITE DE BRETAGNE SUD (UBS) --- I.U.P/I.U.T. DE LORIENT *
7 *****/
8 * LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M) *
9 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
10 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
11 *****/
12 * DATE: 19/01/2001 *
13 * $ *
14 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
15 * Tel 0297874571 fax : 02.97.87.45.72 *
16 * $ *
17 * PROJET: Herezh++ *
18 * $ *
19 *****/
20 * BUT: Classe virtuelle permettant le calcul d'une fonction nD *
21 * ainsi qu'éventuellement un certain nombre d'information supplé- *
22 * mentaires telles que dérivées. *
23 * si le nom de la Fonction = "_" il s'agit d'une Fonction interne *
24 * à un objet, c'est-à-dire gérée seulement par l'entité qui la *
25 * contient, donc pas besoin de nom (elle n'est pas utilisée autre *
26 * part). Si le nom est différent de "_" c'est une Fonction qui est *
27 * gérée et référencée dans LesFonctions, donc à partir de son nom, *
28 * on peut la retrouver. *
29 * $ *
30 * ***** *
31 *
32 * VERIFICATION: *
33 *
34 * ! date ! auteur ! but ! *
35 * ----- *
36 * ! ! ! ! *
37 * $ *
38 * ***** *
39 * MODIFICATIONS: *
40 * ! date ! auteur ! but ! *
41 * ----- *
42 * $ *
43 *****/
44
45 #ifndef FONCTION_N_D_H
46 #define FONCTION_N_D_H
47
48 #include "UtilLecture.h"
49 #include "EnumFonction_nD.h"
50 #include "Enum_IO_XML.h"
51 #include "Tableau_T.h"
52 #include "Enum_GrandeurGlobale.h"
53 #include "Courbe1D.h"
54 #include "Coordonnee.h"
55 #include "Ddl_etendu.h"
56 #include "TypeQuelconque.h"
57 #include "Tenseur.h"
58
59 // gestion d'exception pour des erreurs d'appel
60 class ErrCalculFct_nD
61 { public :
62     ErrCalculFct_nD () {} ; // CONSTRUCTEURS
63     ~ErrCalculFct_nD () {} ;// DESTRUCTEUR :
64 };

```



```

65
66 class Fonction_nD
67 {
68     public :
69
70         // CONSTRUCTEURS :
71         // par défaut
72         Fonction_nD(string nom = "", EnumFonction_nD typ = AUCUNE_FONCTION_nD);
73         // constructeur avec plus d'info
74         // dans le cas ou les variables sont associées à des types quelconques, il
75         // s'agit que de conteneur d'un scalaire simple. Dans le cas contraire il y a erreur !
76         Fonction_nD(const Tableau <string >& var, string nom = "", EnumFonction_nD typ = AUCUNE_FONCTION_nD);
77         // de copie
78         Fonction_nD(const Fonction_nD& Co);
79         // DESTRUCTEUR :
80         virtual ~Fonction_nD();
81
82         // METHODES PUBLIQUES :
83         //
84         // complète les tableaux internes en fonction de la taille des conteneurs
85         // associés aux types quelconques éventuelles
86         // *** doit -être appelée s'il y a des grandeurs quelconque dont les conteneurs
87         // sont associés à plusieurs scalaires
88         // la fonction nD ne doit pas pouvoir fonctionner s'il y a des types complexes
89         // et que cette méthode n'a pas été appelé
90         void Preparation_Grandeur_quelconque(const Tableau <TypeQuelconque >& tqi);
91
92         // ----- virtuelles -----
93
94         // Surcharge de l'operateur = : realise l'egalite de deux fonctions
95         virtual Fonction_nD& operator= (const Fonction_nD& elt) = 0;
96
97         // affichage de la Fonction
98         // = 0, 1 ou 2 (le plus précis)
99         virtual void Affiche(int niveau = 0) const = 0;
100        // ramène le nom de la Fonction
101        const string& NomFonction() const {return nom_ref;};
102        // vérification que tout est ok, pres à l'emploi
103        // ramène true si ok, false sinon
104        virtual bool Complet_Fonction(bool affichage = true) const = 0;
105
106        // ramène le nombre total de variables de la fonction
107        // c-a-dire locale et globales
108        virtual int NbVariable()const
109        {return nom_variables.Taille()+enu_variables_globale.Taille()
110         + nom_variables_globales.Taille();
111         };
112
113        // ramène le nombre de variables locales
114        // c-a-dire à l'exclusion des variables globales
115        virtual int NbVariable_locale()const
116        {return nom_variables.Taille();
117         };
118
119        // ramène le nombre de variables globales
120        // c-a-dire à l'exclusion des variables locales
121        virtual int NbVariable_globale()const
122        {return (enu_variables_globale.Taille()+nom_variables_globales.Taille());
123         };
124
125        // ramène le noms des variables, indépendamment des variables globales
126        const Tableau <string >& Nom_variables() const {return nom_variables;};
127
128        // ramène les énumérés des variables globales éventuelles
129        const Tableau <Enum_GrandeurGlobale >& Enu_variables_globales() const {return
enu_variables_globale;};
130        // ramène les noms string des variables globales éventuelles
131        const Tableau <string >& Nom_variables_globales() const {return nom_variables_globales;};
132
133        // ramène l'équivalent des nom_variables sous forme de Ddl_enum_etendu et
134        // d'EnumTypeQuelconque
135        // permet ensuite d'appeler la méthode Valeur(.. mais à la condition
136        // que l'ensemble des Nom_variables() sont représenté soit par des Ddl_enum_etendu
137        // et ou soit par des EnumTypeQuelconque
138        const Tableau <Ddl_enum_etendu>& Tab_enu_etendu()const {return tab_enu_etendu;};
139        const Tableau <EnumTypeQuelconque>& Tab_enu_quelconque() const {return tab_enu_quelconque;};
140        // un booléen pour noter l'équivalence parfaite ou non
141        bool Equivalence_nom_enu_etendu_et_enu_quelconque() const {return
equivalence_nom_enu_etendu_et_enu_quelconque;};
142
143        // tab_enu_etendu(i) correspond à nom_variables(index_enu_etendu(i))
144        const Tableau <int>& Iindex_enu_etendu() const {return index_enu_etendu;};
145        // tab_enu_quelconque(i) correspond à nom_variables(index_enu_quelconque(i))
146        const Tableau <int>& Iindex_enu_quelconque() const {return index_enu_quelconque;};
147
148        // .. chaque type_des_variables tab_enu_etendu(i) est associé à :
149        // si type_des_variables(i) = 1 -> un scalaire

```

```

150         // si type_des_variables(i) = 2 -> des coordonnées de dim absolue
151         // si type_des_variables(i) = 3 -> un tenseur de dim absolue
152         // si type_des_variables(i) = 4 -> une grandeur quelconque
153     // retour du tableau des indicateurs
154     const Tableau <int>& Type_des_variables_locales() const {return type_des_variables;};
155
156     // retour du tableau posi_ddl_enum, qui est nécessaire pour l'appel de la fonction
157     // Val_FnD_Evoluee(Tableau <double >* xi_,Tableau <Coordonnee> * tab_coor_,Tableau <TenseurBB* >*
tab_tensBB_
158     //         Tableau <TypeQuelconque >* t_quelc_ )
159     // -> Pour le Ddl_enum_etendu (i) dont on récupère la liste via Tab_enu_etendu()
160     // si (i) est un scalaire alors: posi_ddl_enum(i) représente la position que doit
161     // avoir de Ddl_enum_etendu(i) dans xi_
162     // si (i) est de type Coordonnee -> position du ddl_enum_etendu(i) dans tab_coor_
163     // si (i) un tenseur -> position du ddl_enum_etendu(i) dans tab_tensBB_
164     // si (i) une grandeur quelconque -> position du ddl_enum_etendu(i) dans t_quelc_
165     const Tableau <int>& Index_dans_tableau() const {return posi_ddl_enum;};
166
167
168     // retour des tailles qu'ont les différents tableaux associés au ddl_enum
169     // tailles_tab(1) -> nb de scalaires
170     // (2) -> nb de Coordonnee
171     // (3) -> nb de tenseur
172     // (4) -> nb de grandeurs quelconques
173     const Tableau <int>& Tailles_tab() const {return tailles_tab;};
174
175
176     // ramène le nombre de composantes de la fonction
177     virtual int NbComposante() const =0;
178
179     // Lecture des donnees de la classe sur fichier
180     // le nom passé en paramètre est le nom de la Fonction
181     // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
182     // ce nom remplace le nom actuel
183     virtual void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * ) = 0;
184
185     // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
186     // globales en cours de calcul
187     virtual void Mise_a_jour_variables_globales() = 0;
188
189     // établir le lien entre la Fonction et des Fonctions déjà existantes dont
190     // on connaît que le nom
191     // permet ainsi de compléter la Fonction
192     // 1) renseigne si la Fonction dépend d'autre Fonction ou non
193     virtual bool DependAutreFoncCourbes() const {return false;}; // par défaut non
194     // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
195     virtual list <string>& ListDependanceCourbes(list <string>& lico) const;
196     // 3) retourne une liste de nom correspondant aux noms de Fonction dont dépend *this
197     virtual list <string>& ListDependanceFonctions(list <string>& lico) const;
198     // 4) établit la connection entre la demande de *this et les Fonctions passées en paramètres
199     virtual void Lien_entre_fonc_courbe(list <Fonction_nD *>& liptfonc,list <CourbelD *>& liptco) {};
200
201     // def info fichier de commande
202     virtual void Info_commande_Fonctions_nD(UtilLecture & entreePrinc) = 0;
203
204
205     // calcul des valeurs de la fonction, avec comme argument: uniquement des grandeurs
206     // scalaires simples
207     // retour d'un tableau de scalaires
208     Tableau <double> & Valeur_FnD_tab_scalaire(Tableau <double >* val_double)
209     {try
210     {
211         #ifdef MISE_AU_POINT
212         if (permet_affichage > 4)
213         {cout << "\n retour fonction: "«nom_ref» " : ";
214         Tableau <double> & inter = Valeur_FnD_interne(val_double);
215         int nb_val = inter.Taille();
216         for (int i=1;i<=nb_val;i++)
217             cout << "val("«i»)= "«inter(i);
218         if (permet_affichage > 5)
219             {cout << "\n parametres d'appel: ";
220             int nb_var = val_double->Taille();
221             for (int j=1;j<=nb_var;j++)
222                 { cout << " para("«j»)= "« (*val_double) (j);}
223             };
224         return inter;
225         }
226         else
227         #endif
228         return Valeur_FnD_interne(val_double);
229     }
230     catch(...)
231     { cout << "\n ** erreur \n Valeur_FnD_tab_scalaire(..."«endl;
232     this->Affiche();Sortie(1);
233     };
234     // on ne doit jamais arriver ici !
235     return Valeur_FnD_interne(val_double); // pour taire le compil

```

```

236     };
237
238     // calcul des valeurs de la fonction, à l'aide de paramètres = grandeurs évoluées
239     // retour d'un tableau de scalaires
240     // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
241     // NB: il peut y avoir plus d'info que nécessaire
242     // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
243     // il peut y avoir éventuellement des grandeurs quelconques et éventuellement des types non scalaire
244     // dans ce dernier cas il faut renseigner le tableau de numéro d'ordre t_num_ordre
245     Tableau <double> & Val_FnD_Evoluee(Tableau <double >* val_ddl_enum
246                                     ,Tableau <Coordonnee> * coor_ddl_enum
247                                     ,Tableau <TenseurBB* >* tens_ddl_enum
248                                     ,Tableau <TypeQuelconque >* tqi = NULL
249                                     ,Tableau <int> * t_num_ordre = NULL)
250     { if (equivalence_nom_enu_etendu_et_enu_quelconque)
251         {try
252             { Vers_tab_double(t_inter_double, val_ddl_enum, coor_ddl_enum, tens_ddl_enum
253                             , tqi, t_num_ordre);
254                 #ifndef MISE_AU_POINT
255                 if (permet_affichage > 4)
256                     {cout << "\n retour fonction: " << nom_ref << " : ";
257                         Tableau <double> & inter = Valeur_FnD_interne(&t_inter_double);
258                         int nb_val = inter.Taille();
259                         for (int i=1; i<=nb_val; i++)
260                             cout << "val(" << i << ")= " << inter(i);
261                         if (permet_affichage > 5)
262                             {cout << "\n parametres d'appel: ";
263                                 int nb_var = t_inter_double.Taille();
264                                 for (int j=1; j<=nb_var; j++)
265                                     { cout << " para(" << j << ")= " << t_inter_double(j); }
266                             };
267                                 return inter;
268                             }
269                         else
270                             #endif
271                                 return Valeur_FnD_interne(&t_inter_double);
272                     }
273                 catch(...)
274                     { cout << "\n ** erreur \n Valeur_FnD_Evoluee(...) << endl;
275                         this->Affiche(); Sortie(1);
276                     };
277                 }
278                 else {cout << "\n erreur appel Val_FnD_Evoluee(, variables non definies ! ";
279                     this->Affiche(); Sortie(1);
280                     return Valeur_FnD_interne(&t_inter_double); // pour taire le compilateur
281                 };
282                 // on ne doit jamais arriver ici !
283                 return Valeur_FnD_interne(&t_inter_double); // pour taire le compilateur
284             };
285
286             // mise à disposition des conteneurs pour l'appel de Valeur_FnD sans paramètres
287             // il n'est pas autorisé de changer la taille du conteneur, c'est seulement les valeurs qui doivent
288             // être modifiées sinon il y aura des pb
289             // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
290             Tableau <double > & Val_ddl_enum() {return val_ddl_enum;} // valeur associé à l'enu_etendu(i) si
scalaires
291             Tableau <Coordonnee > & Coor_ddl_enum() {return coor_ddl_enum;}; // idem si Coordonnées (en
orthonormée)
292             // retourne le tableau de l'ensemble des grandeurs de type Coordonnee qui sont nécessaire pour
l'appel
293             // chaque type est repéré par premier_famille_Coord(j) ce qui correspond à coor_ddl_enum(j)
294             Tableau <Ddl_enu_etendu> & Premier_famille_Coord() {return premier_famille_Coord;};
295
296             Tableau <TenseurBB* > & Tens_ddl_enum() {return tens_ddl_enum;}; // idem si Tenseur (en orthonormée
!)
297             Tableau <Ddl_enu_etendu> & Premier_famille_tenseur() {return premier_famille_tenseur;};
298
299             // -- retourne une liste de grandeurs quelconques équivalentes aux types évoluées non scalaires
300             // li_equi_Quel_evolue : est l'équivalent des grandeurs évoluées: coor_ddl_enum et tens_ddl_enum
301             // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
évoluées
302             // non scalaires
303             // l'ordre l'apparition dans la liste est telle que l'on a:
304             // 1) tous les Coordonnees 2) puis tous les tenseurs
305             // dans le même ordre que pour les tableaux: coor_ddl_enum puis tens_ddl_enum
306             // ==>==> important: il est permis "que" de changer les valeurs dans les conteneurs
307             List_io <TypeQuelconque > & Li_equi_Quel_evolue() {return li_equi_Quel_evolue;};
308
309             // -- retourne une liste des grandeurs évoluées uniquement scalaire
310             // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
scalaires
311             // est équivalent à val_ddl_enum, l'ordre d'apparition est celui de val_ddl_enum
312             // ==>==> important: il est permis "que" de changer les valeurs dans les conteneurs
313             List_io <Ddl_enu_etendu> & Li_enu_etendu_scalaire() {return li_enu_etendu_scalaire;};
314
315             // calcul équivalent, cf. les paramètres de passage
316             // pour que l'appel à cette méthode soit correcte, il faut que les listes correspondent

```

```

317 // à celles de Li_equi_Quel_evolue(), et Li_enu_etendu_scalaire(),
318 // ==>==> il faut donc utiliser ces listes en les récupérants auparavant !!
319 // Si t_num_ordre est Null cela signifie que tous les tqi sont de grandeurs scalaire
320 // dans le cas contraire t_num_ordre(i) donne le numéro d'ordre du scalaire dans tqi qui correspond
à Nom_variable(i)
321 // ==>==> important: il est permis "que" de changer les valeurs dans les conteneurs
322 Tableau <double> & Valeur_FnD_Evoluee(Tableau <double >* val_ddl_enum
323                                     ,List_io <Ddl_enum_etendu>* li_evolue_scalaire
324                                     ,List_io <TypeQuelconque >* li_evoluee_non_scalaire
325                                     ,Tableau <TypeQuelconque >* tqi
326                                     ,Tableau <int> * t_num_ordre)
327 { if (equivalence_nom_enu_etendu_et_enu_quelconque)
328     { try
329
{Vers_tab_double(t_inter_double, val_ddl_enum, li_evolue_scalaire, li_evoluee_non_scalaire, tqi, t_num_ordre);
330 #ifdef MISE_AU_POINT
331     if (permet_affichage > 4)
332     {cout << "\n retour fonction: " << nom_ref << " : ";
333     Tableau <double> & inter = Valeur_FnD_interne(&t_inter_double);
334     int nb_val = inter.Taille();
335     for (int i=1; i<=nb_val; i++)
336     cout << "val(" << i << ") = " << inter(i);
337     if (permet_affichage > 5)
338     {cout << "\n parametres d'appel: ";
339     int nb_var = t_inter_double.Taille();
340     for (int j=1; j<=nb_var; j++)
341     { cout << " para(" << j << ") = " << t_inter_double(j);
342     };
343     return inter;
344     }
345     else
346     #endif
347     return Valeur_FnD_interne(&t_inter_double);
348     }
349     catch(...)
350     { cout << "\n ** erreur \n Valeur_FnD_Evoluee(...) << endl;
351     this->Affiche(); Sortie(1);
352     };
353     }
354     else {cout << "\n erreur appel Val_FnD_Evoluee(, variables non definies ! ";
355     // on génère une interruption ce qui permettra de dépiler les appels
356     this->Affiche(); Sortie(1);
357     return Valeur_FnD_interne(&t_inter_double); // pour taire le compilateur
358     };
359     // on ne doit jamais arriver ici !
360     return Valeur_FnD_interne(&t_inter_double); // pour taire le compilo
361     };
362
363 // calcul équivalent, mais pour des paramètres de type ddl enum étendu et/ou type quelconque
364 // pour que l'appel à cette méthode soit correcte, il faut que la dimension de t_enu + celle de tqi
365 // soit identique à celle du tableau Nom_variables() : en fait t_enu et tqi doivent représenter les
variables
366 // Si t_num_ordre est Null cela signifie que tous les tqi sont de grandeurs scalaire
367 // dans le cas contraire t_num_ordre(i) donne le numéro d'ordre du scalaire dans tqi qui correspond
à Nom_variable(i)
368 Tableau <double> & Valeur_FnD(Tableau <Ddl_etendu> * t_enu, Tableau <TypeQuelconque >* tqi
369                               , Tableau <int> * t_num_ordre);
370
371 // indicateur permettant de connaître rapidement si
372 // la fonction dépend de la position d'un point M
373 // 0 : la fonction ne dépend pas de la position d'un point M
374 // =i non nul : la fonction dépend de la position d'un point M
375 // et la position de M dans le tableau tab_coor est i (pour un appel de la méthode Valeur(..)
376 // dans ce cas, une au moins des nom_variables a un nom de la même famille que le ddl X1
377 // si = -1 : cela signifie que la fonction dépend "que" de M
378 int Depend_M() const {return depend_M;};
379
380 // calcul des valeurs de la fonction, dans le cas où les variables
381 // sont "tous" des grandeurs globales
382 Tableau <double> & Valeur_pour_variables_globales()
383 { try
384     {
385     #ifdef MISE_AU_POINT
386     if (permet_affichage > 4)
387     {cout << "\n retour fonction: " << nom_ref << " : ";
388     Tableau <double> & inter = Valeur_pour_variables_globales_interne();
389     int nb_val = inter.Taille();
390     for (int i=1; i<=nb_val; i++)
391     cout << "val(" << i << ") = " << inter(i);
392     return inter;
393     }
394     else
395     #endif
396     return Valeur_pour_variables_globales_interne();
397     }
398     catch(...)
399     { cout << "\n ** erreur Valeur_pour_variables_globales(...) << endl;

```

```

400         this->Affiche();Sortie(1);
401     };
402     // on ne doit jamais arriver ici !
403     return Valeur_pour_variables_globales_interne(); // pour taire le compilo
404 };
405
406 // ---- fonctions internes qui ne "doivent pas !!!" être utilisée directement ---
407 // elles sont mises en public, pour pouvoir être appelées par les classes dérivées
408 // sans avoir à déclarer en friend les classes dérivées, sinon cela fait une boucle
409 // sans fin, d'où la solution choisit pour éviter ce pb
410
411 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
412 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* val_ddl_enum) = 0;
413
414 // calcul des valeurs de la fonction, dans le cas où les variables
415 // sont "tous" des grandeurs globales
416 virtual Tableau <double> & Valeur_pour_variables_globales_interne() = 0;
417
418 //----- lecture écriture de restart -----
419 // cas donne le niveau de la récupération
420 // = 1 : on récupère tout
421 // = 2 : on récupère uniquement les données variables (supposées comme telles)
422 virtual void Lecture_base_info(ifstream& ent,const int cas) = 0;
423 // cas donne le niveau de sauvegarde
424 // = 1 : on sauvegarde tout
425 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
426 virtual void Ecriture_base_info(ofstream& sort,const int cas) = 0;
427 // sortie du schemaXML: en fonction de enu
428 virtual void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu) = 0;
429
430 // ----- static -----
431
432 // ramène un pointeur sur la Fonction correspondant au type de Fonction passé en paramètre
433 // IMPORTANT : il y a création d'une Fonction (utilisation d'un new)
434 static Fonction_nD* New_Fonction_nD(string& nom,EnumFonction_nD typeFonction);
435 // ramène un pointeur sur une Fonction copie de celle passée en paramètre
436 // IMPORTANT : il y a création d'une Fonction (utilisation d'un new)
437 static Fonction_nD* New_Fonction_nD(const Fonction_nD& Co);
438
439 // ramène la liste des identificateurs de Fonctions actuellement disponibles
440 static list <EnumFonction_nD> Liste_Fonction_disponible();
441
442 // ----- non virtuelle -----
443
444 // ramène le type de la Fonction
445 EnumFonction_nD Type_Fonction() const { return typeFonction;};
446
447
448 protected :
449
450 // VARIABLES PROTEGEES :
451 EnumFonction_nD typeFonction; // type de la fonction
452 string nom_ref; // nom de référence de la Fonction
453
454 Tableau <string > nom_variables; //variables de la fonction, vu de l'extérieur
455 Tableau <double > t_inter_double; // tableau de même dimension que nom_variable
456 // sert pour passer les infos à la méthode Valeur_FnD_interne(..
457
458 Tableau <Enum_GrandeurGlobale > enu_variables_globale; //tableau des énumérés
459 // de variables globales
460 // éventuellement vide s'il ne sert pas
461 Tableau <string > nom_variables_globales; //tableau des noms en string
462 // de variables globales
463 // éventuellement vide s'il ne sert pas
464
465 // *** important: les tableaux nom_variables, enu_variables_globales et
466 // nom_variables_globales s'ajoutent
467 // ils ne sont pas liés entre eux: le nombre total de variables est
468 // la somme des trois tableaux. En général c'est soit l'un , soit l'autre
469 // ou le dernier mais on peut avoir un mixte
470
471 // ---- tableaux de grandeurs équivalentes aux nom_variables (locales) -----
472 // un tableau tab_enu_etendu et un tableau tab_enu_quelconque:
473 // tab_enu_etendu.Taille()+tab_enu_quelconque.Taille() == nom_variables.Taille() -> grandeurs
locales
474 Tableau <Ddl_enum_etendu> tab_enu_etendu;
475
476 // tab_enu_etendu peut se décomposer de deux manières équivalentes :
477 // 1) tableaux: val_ddl_enum (scalaires) , coord_ddl_enum (Coordonnees) , tens_ddl_enum (tenseur)
478 // permet aux utilisateurs d'obtenir un stockage et un appel avec des grandeurs évoluées,
distinctes
479 // 2) list: li_enu_etendu_scalaire (scalaires) , li_equi_Quel_evolue (Coordonnees "et" tenseurs)
480 // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
scalaires
481 // et permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
évoluées
482 // excluant les scalaire, et appels associés

```

```

483 // li_equi_Quel evolue correspond à des grandeurs quelconques contenant Coordonnees et tenseur,
484 // est indépendant de tab_enu_quelconque, qui lui est complètement quelconque, on ne sait pas a
    priori
485 // ce qu'il y a dedans
486
487 // .. chaque grandeur quelconque doit être associé, lors de l'appel de la fonction, à un numéro
    d'ordre
488 // du coup, pour l'instant on n'utilise qu'un scalaire par grandeur quelconque, mais le conteneur
489 // peut en contenir plusieurs
490 Tableau <EnumTypeQuelconque> tab_enu_quelconque;
491
492 // ----- grandeurs évoluées y compris les grandeurs quelconques -----
493 // .. chaque nom_variables(j) est associé à :
494 Tableau <int> type_des_variables; // si type_des_variables(j) = 1 -> un scalaire
495 // si type_des_variables(j) = 2 -> des coordonnées de dim absolue
496 // si type_des_variables(j) = 3 -> un tenseur de dim absolue
497 // si type_des_variables(j) = 4 -> une grandeur quelconque
498 // si type_des_variables(j) = 0 -> une grandeur qui n'est pas reconnue
    en local
499
500 // .... dans le cas d'un Ddl_enu_etendu : donc c'est relatif au tableau: tab_enu_etendu
501 Tableau <double > val_ddl_enu; // valeur associé au ddl tab_enu_etendu(i) si scalaire
502 // tab_enu_etendu(i) <=> val_ddl_enu(posi_ddl_enu(i))
503 List_io <Ddl_enu_etendu> li_enu_etendu_scalaire; // est équivalent à val_ddl_enu,
504 // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
    scalaires
505
506 Tableau <Ddl_enu_etendu> premier_famille_Coord; // les ddl du premier de famille
507 //coor_ddl_enu(j) est associé avec premier_famille_Coord(j)
508 Tableau <Coordonnee > coor_ddl_enu; // le vecteur Coordonnée associé au ddl tab_enu_etendu(i) si
    Coordonnées
509 Tableau <int > num_dans_coor; // le numéro dans le vecteur associé au ddl tab_enu_etendu(i)
510 // avec j = posi_ddl_enu(i)
511 // tab_enu_etendu(i) <=> coor_ddl_enu(j) (num_dans_coor(i))
512 // bien noter qu'un objet Coordonnee peut contenir plusieurs Ddl_enu_etendu
513
514 Tableau <Ddl_enu_etendu> premier_famille_tenseur; // les ddl du premier de famille
515 // (*tens_ddl_enu)(j) est associé avec premier_famille_tenseur(j)
516 Tableau <TenseurBB* > tens_ddl_enu; // tenseur associé au ddl tab_enu_etendu(i) si Tenseur
517 Tableau <Deuxentiers_enu > ind_tens; // indice dans le tenseur de la grandeur
518 // avec j = posi_ddl_enu(i)
519 // avec k = ind_tens(i).i et l = ind_tens(i).j
520 // tab_enu_etendu(i) <=> (*tens_ddl_enu)(j)(k,l)
521 // bien noter qu'un objet Tenseur peut contenir plusieurs Ddl_enu_etendu
522
523 // li_equi_Quel evolue : est l'équivalent des grandeurs évoluées: coor_ddl_enu et tens_ddl_enu
524 // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
    évoluées
525 List_io <TypeQuelconque > li_equi_Quel evolue; // non scalaires
526
527 // tab_equi_Coor(j) contient un Coordonnee équivalent à coor_ddl_enu(j)
528 Tableau <List_io<TypeQuelconque >::iterator > tab_equi_Coor;
529 //tab_equi_tens(j) contient un tenseur équivalent à tens_ddl_enu(j)
530 Tableau <List_io<TypeQuelconque >::iterator > tab_equi_tens;
531
532
533 // un système d'adressage indirect pour le passage : tab_enu_etendu(i) <-> grandeur évoluée
534 Tableau <int> posi_ddl_enu; // posi_ddl_enu(i) donne la position de tab_enu_etendu(i)
535 // dans le tableau :
536 // si scalaire : dans le tableau val_ddl_enu ->val_ddl_enu(posi_ddl_enu(i))
537 // si Coordonnées : dans le tableau coor_ddl_enu -> coor_ddl_enu(posi_ddl_enu(i))
538 // ici il s'agira de tous les ddl de la même famille,
539 // et c'est num_dans_coor qui permet de trouver la position
540 // si Tenseur : dans le tableau tens_ddl_enu -> tens_ddl_enu(posi_ddl_enu(i))
541 // et c'est ind_tens qui permettra de trouver les indices
542
543 // un système d'adressage indirect pour le passage : nom_variables(j) <-> grandeur évoluée
544 // tab_enu_etendu(i) correspond à nom_variables(index_enu_etendu(i))
545 Tableau <int> index_enu_etendu;
546
547 // .... dans le cas d'une grandeur quelconque
548 // pour les grandeurs quelconques, l'utilisateur doit transmettre le numéro d'ordre correspondant à
    l'équivalence
549 // du ddl_etendu, du coup on ne s'occupe pas de gérer les numéros d'ordre pour les grandeurs
    quelconques
550
551 // un système d'adressage indirect pour le passage : nom_variables(j) <-> grandeur quelconque
552 // tab_enu_quelconque(i) correspond à nom_variables(index_enu_quelconque(i))
553 Tableau <int> index_enu_quelconque;
554
555
556 // stockage des tailles qu'ont les différents tableaux associés
557 Tableau <int> tailles_tab; // tailles_tab(1) -> nb de scalaires
558 // (2) -> nb de Coordonnee
559 // (3) -> nb de tenseur
560 // (4) -> nb de grandeurs quelconques
561 // ----- fin grandeurs évoluées y compris les grandeurs quelconques-----

```

```

562
563 // ces deux tableaux permettent d'appeler la méthode Valeur(..
564 // un booléen pour noter l'équivalence parfaite ou non
565 bool equivalence_nom_enu_etendu_et_enu_quelconque;
566
567 // un tableau intermédiaire qui sert pour Valeur(Tableau <Ddl_etendu> ...
568 Tableau <double > x_x_i;
569
570 int depend_M; // indicateur permettant de connaître rapidement si
571 // la fonction dépend de la position d'un point M
572 // 0 : la fonction ne dépend pas de la position d'un point M
573 // non nul : la fonction dépend de la position d'un point M
574 // dans ce cas, une au moins un des nom_variables
575 // a un nom de la même famille que le ddl X1
576 // et depend_M = position que doit avoir M dans le tableau tab_coor pour un appel
577 // de la méthode Valeur(..,Tableau <Coordonnee> * tab_coor)
578 // si = -1 : cela signifie que la fonction dépend "que" de M
579 bool depend_temps; // indicateur permettant de connaître rapidement si
580 // la fonction dépend du temps ou non
581
582 // ----- controle de la sortie des informations: utilisé par les classes dérivées
583 int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les erreurs
et warning
584
585 //-----
586
587 // METHODES PROTEGEES :
588 // ramène true si les variables de la classe mère sont complètes
589 bool Complet_var() const;
590
591 // définit le paramètre depend_M en fonction des nom_variables
592 void Definition_depend_M();
593
594 // définit le paramètre depend_temps en fonction des nom_variables
595 void Definition_depend_temps();
596
597 // construction à partir des noms de variables, des tableaux tab_enu_etendu et
598 // tab_enu_quelconque
599 // void Construction_enu_etendu_et_quelconque();
600
601 // Construction des index pour les grandeurs évoluées, ainsi que les conteneurs
602 void Construction_index_conteneurs_evoluees();
603
604 // Passage des infos variables évoluées en tableau de réels
605 // I/O : d
606 Tableau <double > & Vers_tab_double(Tableau <double > & di
607 //                                ,const Tableau <double >* val_ddl_enum
608 //                                ,const Tableau <Coordonnee> * coor_ddl_enum
609 //                                ,const Tableau <TenseurBB* >* tens_ddl_enum
610 //                                ,const Tableau <TypeQuelconque >* tqi = NULL
611 //                                ,const Tableau <int> * t_num_ordre = NULL);
612
613 // idem, mais via les listes
614 // val_ddl_enum(k) correspond à li_evolue_scalaire de rang i
615 Tableau <double > & Vers_tab_double(Tableau <double > & di
616 //                                ,const Tableau <double >* val_ddl_enum
617 //                                ,List_io <Ddl_enum_etendu>* li_evolue_scalaire
618 //                                ,List_io <TypeQuelconque >* li_evoluee_non_scalaire
619 //                                ,const Tableau <TypeQuelconque >* tqi = NULL
620 //                                ,const Tableau <int> * t_num_ordre = NULL);
621
622 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
623 // globales en cours de calcul
624 // méthode interne: qui est utilisé par les fonctions dérivées pour la méthode:
625 // Mise_a_jour_variables_globales
626 // retourne false si rien n'a changé
627 bool Mise_a_jour_variables_globales_interne();
628
629 // appel de la fonction sous forme d'une méthode avec un nombre non limité
630 // de paramètres: en fait utilise Valeur
631 // c'est une méthode avec retour uniquement en scalaire, sinon -> erreur
632 // calcul des valeurs de la fonction
633 // ***** fonction a priori dangereuse, à éviter , pour l'instant ne sert pas !! ****
634 double Val_avec_nbArgVariable(double x,...);
635
636 // méthode utilisée par les classes dérivées, pour transférer les infos qui sont
637 // gérées au niveau de Fonction_nD
638 Fonction_nD& Transfert_info(const Fonction_nD& elt);
639
640 // affichage des données internes, utilisée par les fonctions dérivées
641 // niveau donne le degré d'affichage
642 void Affiche_interne(int niveau) const;
643
644 //----- méthodes appelée par les classes dérivées ----
645 // relatives aux données gérées par Fonction_nD
646 // cas donne le niveau de la récupération
647 // = 1 : on récupère tout
648 // = 2 : on récupère uniquement les données variables (supposées comme telles)

```

```

648 void Lect_base_info(ifstream& ent,const int cas);
649 // cas donne le niveau de sauvegarde
650 // = 1 : on sauvegarde tout
651 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
652 void Ecrit_base_info(ofstream& sort,const int cas);
653 // sortie du schemaXML: en fonction de enu
654 void SchemXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu);
655
656 private :
657
658 // un tableau intermédiaire, en général vide, mais qui sert si on utilise
659 // la méthode Val_avec_nbArgVariable
660 Tableau <double >* xinter=NULL;
661
662
663 };
664
665 #endif

```

## 7.536 Fonction\_nD (copy: conflict on 2017-11-21).h

```

1 // fichier : Fonction_nD.h
2 // classe : Fonction_nD
3
4
5 /*****
6 * UNIVERSITE DE BRETAGNE SUD (UBS) --- I.U.P/I.U.T. DE LORIENT *
7 *****/
8 * LABORATOIRE DE GENIE MECANIQUE ET MATERIAUX (LG2M) *
9 * Centre de Recherche Rue de Saint Maudé - 56325 Lorient cedex *
10 * tel. 02.97.87.45.70 fax. 02.97.87.45.72 http://www-lg2m.univ-ubs.fr *
11 *****/
12 * DATE: 19/01/2001 *
13 * $ *
14 * AUTEUR: G RIO (mailto:gerard.rio@univ-ubs.fr) *
15 * Tel 0297874571 fax : 02.97.87.45.72 *
16 * $ *
17 * PROJET: Herezh++ *
18 * $ *
19 *****/
20 * BUT: Classe virtuelle permettant le calcul d'une fonction nD *
21 * ainsi qu'éventuellement un certain nombre d'information supplé- *
22 * mentaires telles que dérivées. *
23 * si le nom de la Fonction = "_" il s'agit d'une Fonction interne *
24 * à un objet, c'est-à-dire gérée seulement par l'entité qui la *
25 * contient, donc pas besoin de nom (elle n'est pas utilisée autre *
26 * part). Si le nom est différent de "_" c'est une Fonction qui est *
27 * gérée et référencée dans LesFonctions, donc à partir de son nom, *
28 * on peut la retrouver. *
29 * $ *
30 * ***** *
31 *
32 * VERIFICATION: *
33 *
34 * ! date ! auteur ! but ! *
35 * ----- *
36 * ! ! ! ! *
37 * $ *
38 * ***** *
39 * MODIFICATIONS: *
40 * ! date ! auteur ! but ! *
41 * ----- *
42 * $ *
43 *****/
44
45 #ifndef FONCTION_N_D_H
46 #define FONCTION_N_D_H
47
48 #include "UtilLecture.h"
49 #include "EnumFonction_nD.h"
50 #include "Enum_IO_XML.h"
51 #include "Tableau_T.h"
52 #include "Enum_GrandeurGlobale.h"
53 #include "Courbe1D.h"
54 #include "Coordonnee.h"
55 #include "Ddl_etendu.h"
56 #include "TypeQuelconque.h"
57 #include "Tenseur.h"
58
59
60 class Fonction_nD
61 {
62 public :
63
64 // CONSTRUCTEURS :

```



```

65 // par défaut
66 Fonction_nD(string nom = "", EnumFonction_nD typ = AUCUNE_FONCTION_nD);
67 // constructeur avec plus d'info
68 // dans le cas ou les variables sont associées à des types quelconques, il
69 // s'agit que de conteneur d'un scalaire simple. Dans le cas contraire il y a erreur !
70 Fonction_nD(const Tableau <string >& var, string nom = "", EnumFonction_nD typ = AUCUNE_FONCTION_nD);
71 // de copie
72 Fonction_nD(const Fonction_nD& Co);
73 // DESTRUCTEUR :
74 virtual ~Fonction_nD();
75
76 // METHODES PUBLIQUES :
77 //
78 // // complète les tableaux internes en fonction de la taille des conteneurs
79 // // associés aux types quelconques éventuelles
80 // // *** doit -être appelée s'il y a des grandeurs quelconque dont les conteneurs
81 // // sont associés à plusieurs scalaires
82 // // la fonction nD ne doit pas pouvoir fonctionner s'il y a des types complexes
83 // // et que cette méthode n'a pas été appelé
84 // void Preparation_Grandeur_quelconque(const Tableau <TypeQuelconque >& tqi);
85
86 // ----- virtuelles -----
87
88 // Surcharge de l'operateur = : realise l'egalite de deux fonctions
89 virtual Fonction_nD& operator= (const Fonction_nD& elt) = 0;
90
91 // affichage de la Fonction
92 // = 0, 1 ou 2 (le plus précis)
93 virtual void Affiche(int niveau = 0) const = 0;
94 // ramène le nom de la Fonction
95 const string& NomFonction() const {return nom_ref;};
96 // vérification que tout est ok, pres à l'emploi
97 // ramène true si ok, false sinon
98 virtual bool Complet_Fonction()const = 0 ;
99
100 // ramène le nombre de variable de la fonction
101 virtual int NbVariable()const
102 {return nom_variables.Taille()+enu_variables_globale.Taille()
103 + nom_variabelles_globales.Taille();
104 };
105
106 // ramène le noms des variables, indépendamment des variables globales
107 const Tableau <string >& Nom_variabelles() const {return nom_variabelles;};
108
109 // ramène les énumérés des variables globales éventuelles
110 const Tableau <Enum_GrandeurGlobale >& Enu_variabelles_globales() const {return
111 enu_variabelles_globale;};
112 // ramène les noms string des variables globales éventuelles
113 const Tableau <string >& Nom_variabelles_globales() const {return nom_variabelles_globales;};
114
115 // ramène l'équivalent des nom_variabelles sous forme de Ddl_enum_etendu et
116 // d'EnumTypeQuelconque
117 // permet ensuite d'appeler la méthode Valeur(.. mais à la condition
118 // que l'ensemble des Nom_variabelles() sont représenté soit par des Ddl_enum_etendu
119 // et ou soit par des EnumTypeQuelconque
120 const Tableau <Ddl_enum_etendu>& Tab_enu_etendu()const {return tab_enu_etendu;};
121 const Tableau <EnumTypeQuelconque>& Tab_enu_quelconque() const {return tab_enu_quelconque;};
122 // un booléen pour noter l'équivalence parfaite ou non
123 bool Equivalence_nom_enu_etendu_et_enu_quelconque() const {return
124 equivalence_nom_enu_etendu_et_enu_quelconque;};
125
126 // tab_enu_etendu(i) correspond à nom_variabelles(index_enu_etendu(i))
127 const Tableau <int>& Iindex_enu_etendu() const {return index_enu_etendu;};
128 // tab_enu_quelconque(i) correspond à nom_variabelles(index_enu_quelconque(i))
129 const Tableau <int>& Iindex_enu_quelconque() const {return index_enu_quelconque;};
130
131 // .. chaque type_des_variabelles tab_enu_etendu(i) est associé à :
132 // si type_des_variabelles(i) = 1 -> un scalaire
133 // si type_des_variabelles(i) = 2 -> des coordonnées de dim absolue
134 // si type_des_variabelles(i) = 3 -> un tenseur de dim absolue
135 // si type_des_variabelles(i) = 4 -> une grandeur quelconque
136 // retour du tableau des indicateurs
137 const Tableau <int>& Type_des_variabelles_locales() const {return type_des_variabelles;};
138
139 // retour du tableau posi_ddl_enum, qui est nécessaire pour l'appel de la fonction
140 // Val_FnD_Evoluee(Tableau <double >* xi_,Tableau <Coordonnee > * tab_coor_,Tableau <TenseurBB* >*
141 // tab_tensBB_
142 //
143 // Tableau <TypeQuelconque >* t_quelc_ )
144 // -> Pour le Ddl_enum_etendu (i) dont on récupère la liste via Tab_enu_etendu()
145 // si (i) est un scalaire alors: posi_ddl_enum(i) représente la position que doit
146 // avoir de Ddl_enum_etendu(i) dans xi_
147 // si (i) est de type Coordonnee -> position du ddl_enum_etendu(i) dans tab_coor_
148 // si (i) un tenseur -> position du ddl_enum_etendu(i) dans tab_tensBB_
149 // si (i) une grandeur quelconque -> position du ddl_enum_etendu(i) dans t_quelc_
150 const Tableau <int>& Index_dans_tableau() const {return posi_ddl_enum;};
151
152
153

```

```

149 // retour des tailles qu'ont les différents tableaux associés au ddl_enum
150 // tailles_tab(1) -> nb de scalaires
151 // (2) -> nb de Coordonnee
152 // (3) -> nb de tenseur
153 // (4) -> nb de grandeurs quelconques
154 const Tableau <int>& Tailles_tab() const {return tailles_tab;};
155
156
157 // ramène le nombre de composantes de la fonction
158 virtual int NbComposante() const =0;
159
160 // Lecture des donnees de la classe sur fichier
161 // le nom passé en paramètre est le nom de la Fonction
162 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
163 // ce nom remplace le nom actuel
164 virtual void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * ) = 0;
165
166 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
167 // globales en cours de calcul
168 virtual void Mise_a_jour_variables_globales() = 0;
169
170 // établir le lien entre la Fonction et des Fonctions déjà existantes dont
171 // on connait que le nom
172 // permet ainsi de compléter la Fonction
173 // 1) renseigne si la Fonction dépend d'autre Fonction ou non
174 virtual bool DependAutreFoncCourbes() const {return false;}; // par défaut non
175 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
176 virtual list <string>& ListDependanceCourbes(list <string>& lico) const;
177 // 3) retourne une liste de nom correspondant aux noms de Fonction dont dépend *this
178 virtual list <string>& ListDependanceFonctions(list <string>& lico) const;
179 // 4) établit la connection entre la demande de *this et les Fonctions passées en paramètres
180 virtual void Lien_entre_fonc_courbe(list <Fonction_nD *>& liptfonc,list <CourbeID *>& liptco) {};
181
182 // def info fichier de commande
183 virtual void Info_commande_Fonctions_nD(UtilLecture & entreePrinc) = 0;
184
185 // calcul des valeurs de la fonction, à l'aide de paramètres = grandeurs évoluées
186 // retour d'un tableau de scalaires
187 // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
188 // NB: il peut y avoir plus d'info que nécessaire
189 // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
190 // il peut y avoir éventuellement des grandeurs quelconques et éventuellement des types non scalaire
191 // dans ce dernier cas il faut renseigner le tableau de numéro d'ordre t_num_ordre
192 virtual Tableau <double> & Val_FnD_Evoluee(Tableau <double >* val_ddl_enum,Tableau <Coordonnee > *
193 // ,Tableau <TenseurBB* >* tens_ddl_enum
194 // ,Tableau <TypeQuelconque >* tqi = NULL // ,Tableau <int> * t_num_ordre = NULL)
195 // { return Valeur_FnD_interne(val_ddl_enum,coor_ddl_enum,tens_ddl_enum, false);};
196
197 // mise à disposition des conteneurs pour l'appel de Valeur_FnD sans paramètres
198 // il n'est pas autorisé de changer la taille du conteneur, c'est seulement les valeurs doivent
199 // être modifiées sinon il y aura des pb
200 // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
201 Tableau <double >& Val_ddl_enum() {return val_ddl_enum;}; // valeur associé à l'enu_etendu(i) si
202 scalaire
203 Tableau <Coordonnee >& Coor_ddl_enum() {return coor_ddl_enum;}; // idem si Coordonnées (en
204 orthonormée)
205 Tableau <TenseurBB* >& Tens_ddl_enum() {return tens_ddl_enum;}; // idem si Tenseur (en orthonormée
206 !)
207
208 // il n'y a pas de mise à disposition des conteneurs de types quelconques, car la fonction ne peut
209 pas gérer tous les
210 // types possibles. En conséquence c'est à l'utilisateur de fournir les infos nécessaires à
211 l'utilisation du conteneur
212 // notamment: le conteneur et numéro d'ordre s'il y a plusieurs données dans le conteneur quelconques
213
214 virtual Tableau <double> & Valeur_FnD()
215 // { return Valeur_FnD_interne(NULL,NULL,NULL, true);};
216
217 // calcul équivalent, mais pour des paramètres de type ddl enum étendu et/ou type quelconque
218 // pour que l'appel à cette méthode soit correcte, il faut que la dimension de t_enu + celle de tqi
219 // soit identique à celle du tableau Nom_variables() : en fait t_enu et tqi doivent représenter les
220 variables
221 // Si t_num_ordre est Null cela signifie que tous les tqi sont de grandeurs scalaire
222 // dans le cas contraire t_num_ordre(i) donne le numéro d'ordre du scalaire dans tqi qui correspond
223 à Nom_variable(i)
224 virtual Tableau <double> & Valeur_FnD(Tableau <Ddl_etendu >* t_enu,Tableau <TypeQuelconque >* tqi
225 // ,Tableau <int> * t_num_ordre);
226
227 // indicateur permettant de connaître rapidement si
228 // la fonction dépend de la position d'un point M
229 // 0 : la fonction ne dépend pas de la position d'un point M
230 // =i non nul : la fonction dépend de la position d'un point M
231 // et la position de M dans le tableau tab_coor est i (pour un appel de la méthode Valeur(..)
232 // dans ce cas, une au moins des nom_variables a un nom de la même famille que le ddl X1
233 int Depend_M() const {return depend_M;};

```

```

227
228 // calcul des valeurs de la fonction, dans le cas où les variables
229 // sont "tous" des grandeurs globales
230 virtual Tableau <double> & Valeur_pour_variables_globales() = 0;
231
232 //----- lecture écriture de restart -----
233 // cas donne le niveau de la récupération
234 // = 1 : on récupère tout
235 // = 2 : on récupère uniquement les données variables (supposées comme telles)
236 virtual void Lecture_base_info(ifstream& ent,const int cas) = 0;
237 // cas donne le niveau de sauvegarde
238 // = 1 : on sauvegarde tout
239 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
240 virtual void Ecriture_base_info(ofstream& sort,const int cas) = 0;
241 // sortie du schemaXML: en fonction de enu
242 virtual void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu) = 0;
243
244 // ----- static -----
245
246 // ramène un pointeur sur la Fonction correspondant au type de Fonction passé en paramètre
247 // IMPORTANT : il y a création d'une Fonction (utilisation d'un new)
248 static Fonction_nD* New_Fonction_nD(string& nom,EnumFonction_nD typeFonction);
249 // ramène un pointeur sur une Fonction copie de celle passée en paramètre
250 // IMPORTANT : il y a création d'une Fonction (utilisation d'un new)
251 static Fonction_nD* New_Fonction_nD(const Fonction_nD& Co);
252
253 // ramène la liste des identificateurs de Fonctions actuellement disponibles
254 static list <EnumFonction_nD> Liste_Fonction_disponible();
255
256 // ----- non virtuelle -----
257
258 // ramène le type de la Fonction
259 EnumFonction_nD Type_Fonction() const { return typeFonction;};
260
261
262 protected :
263
264 // VARIABLES PROTEGEES :
265 EnumFonction_nD typeFonction; // type de la fonction
266 string nom_ref; // nom de référence de la Fonction
267
268 Tableau <string > nom_variables; //variables de la fonction, vu de l'extérieur
269
270 Tableau <Enum_GrandeurGlobale > enu_variables_globale; //tableau des énumérés
271 // de variables globales
272 // éventuellement vide s'il ne sert pas
273 Tableau <string > nom_variables_globales; //tableau des noms en string
274 // de variables globales
275 // éventuellement vide s'il ne sert pas
276
277 // *** important: les tableaux nom_variables, enu_variables_globales et
278 // nom_variables_globales s'ajoutent
279 // ils ne sont pas liés entre eux: le nombre total de variables est
280 // la somme des trois tableaux. En général c'est soit l'un , soit l'autre
281 // ou le dernier mais on peut avoir un mixte
282
283 // ---- tableaux de grandeurs équivalentes aux nom_variables (locales) ----
284 // un tableau tab_enu_etendu et un tableau tab_enu_quelconque:
285 // tab_enu_etendu.Taille()+tab_enu_quelconque.Taille() == nom_variables.Taille() -> grandeurs
locales
286 Tableau <Ddl_enum_etendu> tab_enu_etendu;
287 // .. chaque grandeur quelconque doit être associé, lors de l'appel de la fonction, à un numéro
d'ordre
288 // du coup, pour l'instant on n'utilise qu'un scalaire par grandeur quelconque, mais le conteneur
289 // peut en contenir plusieurs
290 Tableau <EnumTypeQuelconque> tab_enu_quelconque;
291
292
293 // ----- grandeurs évoluées y compris les grandeurs quelconques -----
294 // .. chaque ddl_enum-etendu tab_enu_etendu(i) est associé à :
295 Tableau <int> type_des_variables; // si type_des_variables(i) = 1 -> un scalaire
296 // si type_des_variables(i) = 2 -> des coordonnées de dim absolue
297 // si type_des_variables(i) = 3 -> un tenseur de dim absolue
298 // si type_des_variables(i) = 4 -> une grandeur quelconque
299 Tableau <double > val_ddl_enum; // valeur associé au ddl tab_enu_etendu(i) si scalaire
300 // tab_enu_etendu(i) <=> val_ddl_enum(posi_ddl_enum(i))
301 Tableau <Coordonnee > coor_ddl_enum; // le vecteur Coordonnée associé au ddl tab_enu_etendu(i) si
Coordonnées
302 Tableau <int > num_dans_coor; // le numéro dans le vecteur associé au ddl tab_enu_etendu(i)
303 // avec j = posi_ddl_enum(i)
304 // tab_enu_etendu(i) <=> coor_ddl_enum(j) (num_dans_coor(j))
305
306 Tableau <TenseurBB* > tens_ddl_enum; // tenseur associé au ddl tab_enu_etendu(i) si Tenseur
307 Tableau <Tableau < int > > indice; // indice dans le tenseur de la grandeur
308 // avec j = posi_ddl_enum(i)
309 // avec k = indice(j) (1) et l = indice(j) (2)
310 // tab_enu_etendu(i) <=> (*tens_ddl_enum) (k,l)

```

```

311
312 // pour les grandeurs quelconques, l'utilisateur doit transmettre le numéro d'ordre correspondant à
// l'équivalence
313 // du ddl_etendu, du coup on ne s'occupe pas de gérer les numéros d'ordre pour les grandeurs
// quelconques
314
315 // un système d'adressage indirect
316 Tableau <int> posi_ddl_enum; // posi_ddl_enum(i) donne la position de tab_enu_etendu(i)
317 // dans le tableau :
318 // si scalaire : dans le tableau val_ddl_enum -> val_ddl_enum(posi_ddl_enum(i))
319 // si Coordonnées : dans le tableau coord_ddl_enum -> coord_ddl_enum(posi_ddl_enum(i))
320 // ici il s'agira de tous les ddl de la même famille,
321 // et c'est num_dans_coor qui permet de trouver la position
322 // si Tenseur : dans le tableau tens_ddl_enum -> tens_ddl_enum(posi_ddl_enum(i))
323 // et c'est indice qui permettra de trouver les indices
324
325 // retour des tailles qu'ont les différents tableaux associés
326 Tableau <int> tailles_tab; // tailles_tab(1) -> nb de scalaires
327 // (2) -> nb de Coordonnee
328 // (3) -> nb de tenseur
329 // (4) -> nb de grandeurs quelconques
330 // ----- fin grandeurs évoluées y compris les grandeurs quelconques-----
331
332 // ces deux tableaux permettent d'appeler la méthode Valeur(..
333 // un booléen pour noter l'équivalence parfaite ou non
334 bool equivalence_nom_enu_etendu_et_enu_quelconque;
335
336 // tab_enu_etendu(i) correspond à nom_variables(index_enu_etendu(i))
337 Tableau <int> index_enu_etendu;
338 // tab_enu_quelconque(i) correspond à nom_variables(index_enu_quelconque(i))
339 Tableau <int> index_enu_quelconque;
340 // un tableau intermédiaire qui sert pour Valeur(Tableau <Ddl_etendu> ...
341 Tableau <double > x_x_i;
342
343 int taille_retour; // nombre de grandeurs calculées en retour
344 int depend_M; // indicateur permettant de connaître rapidement si
345 // la fonction dépend de la position d'un point M
346 // 0 : la fonction ne dépend pas de la position d'un point M
347 // non nul : la fonction dépend de la position d'un point M
348 // dans ce cas, une au moins un des nom_variables
349 // a un nom de la même famille que le ddl X1
350 // et depend_M = position que doit avoir M dans le tableau tab_coor pour un appel
351 // de la méthode Valeur(..,Tableau <Coordonnee> * tab_coor)
352 // si = -1 : cela signifie que la fonction dépend "que" de M
353 bool depend_temps; // indicateur permettant de connaître rapidement si
354 // la fonction dépend du temps ou non
355
356
357 // METHODES PROTEGEES :
358 // ramène true si les variables de la classe mère sont complété
359 bool Complet_var() const;
360
361 // définit le paramètre depend_M en fonction des nom_variables
362 void Definition_depend_M();
363
364 // définit le paramètre depend_temps en fonction des nom_variables
365 void Definition_depend_temps();
366
367 // construction à partir des noms de variables, des tableaux tab_enu_etendu et
368 // tab_enu_quelconque
369 void Construction_enu_etendu_et_quelconque();
370
371 // Contruction des index pour les grandeurs évoluées, ainsi que les conteneurs
372 void Construction_index_conteneurs_evoluees();
373
374 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
375 // globales en cours de calcul
376 // méthode interne: qui est utilisé par les fonctions dérivées pour la méthode:
377 // Mise_a_jour_variables_globales
378 // retourne false si rien n'a changé
379 bool Mise_a_jour_variables_globales_interne();
380
381 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
382 // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
383 // NB: il peut y avoir plus d'info que nécessaire
384 // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
385 // variables_locales :
386 // vrai : on utilise les tableaux locaux stockés dans la loi pour calculer la fonction
387 // faux : on utilise les tableaux passés en paramètre
388 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* val_ddl_enum,Tableau <Coordonnee> *
// coor_ddl_enum
389 // ,Tableau <TenseurBB> * tens_ddl_enum,bool variables_locales )
// = 0;
390
391 // appel de la fonction sous forme d'une méthode avec un nombre non limité
392 // de paramètres: en fait utilise Valeur
393 // c'est une méthode avec retour uniquement en scalaire, sinon -> erreur

```

```

394 // calcul des valeurs de la fonction
395 // ***** fonction a priori dangereuse, à éviter , pour l'instant ne sert pas !! ****
396 double Val_avec_nbArgVariable(double x,...);
397
398 // méthode utilisée par les classes dérivées, pour transférer les infos qui sont
399 // gérées au niveau de Fonction_nD
400 Fonction_nD& Transfert_info(const Fonction_nD& elt);
401
402 // affichage des données internes, utilisée par les fonctions dérivées
403 // niveau donne le degré d'affichage
404 void Affiche_interne(int niveau) const;
405
406 private :
407
408 // un tableau intermédiaire, en général vide, mais qui sert si on utilise
409 // la méthode Val_avec_nbArgVariable
410 Tableau <double >* xinter=NULL;
411
412
413 };
414
415 #endif

```

## 7.537 Fonction\_nD.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Fonction_nD.h
30 // classe : Fonction_nD
31
32
33 /*****
34 *      DATE:          01/06/2016
35 *
36 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:        Herezh++
39 *
40 *      *****
41 *      BUT:          Classe virtuelle permettant le calcul d'une fonction nD
42 *      ainsi qu'éventuellement un certain nombre d'information supplé-
43 *      mentaires telles que dérivées.
44 *      si le nom de la Fonction = "-" il s'agit d'une Fonction interne
45 *      à un objet, c'est-à-dire gérée seulement par l'entité qui la
46 *      contient, donc pas besoin de nom (elle n'est pas utilisée autre
47 *      part). Si le nom est différent de "-" c'est une Fonction qui est
48 *      gérée et référencée dans LesFonctions, donc à partir de son nom,
49 *      on peut la retrouver.
50 *
51 *      *****
52 *
53 *      VERIFICATION:
54 *
55 *      ! date !   auteur !           but
56 *      -----
57 *      !           !           !           !
58 *      *****
59 *
60 *      MODIFICATIONS:

```

```

61 *      ! date !      auteur !      but      !      *
62 *      -----
63 *      $      *
64 *****/
65
66 #ifndef FONCTION_N_D_H
67 #define FONCTION_N_D_H
68
69 #include "UtilLecture.h"
70 #include "EnumFonction_nD.h"
71 #include "Enum_IO_XML.h"
72 #include "Tableau_T.h"
73 #include "Enum_GrandeurGlobale.h"
74 #include "CourbeID.h"
75 #include "Coordonnee.h"
76 #include "Ddl_etendu.h"
77 #include "TypeQuelconque.h"
78 #include "Tenseur.h"
79 /** @defgroup Les_fonctions_nD Les_fonctions_nD
80 *
81 * \author      Gérard Rio
82 * \version    1.0
83 * \date      01/06/2016
84 * \brief      Def des fonctions nD
85 *
86 */
87
88
89 /// @addtogroup Les_fonctions_nD
90 /// @{
91 ///
92
93 /// gestion d'exception pour des erreurs d'appel de fonction nD
94 class ErrCalculFct_nD
95 { public :
96     ErrCalculFct_nD () {} ; // CONSTRUCTEURS
97     ~ErrCalculFct_nD () {};// DESTRUCTEUR :
98 };
99 /// @} // end of group
100
101 /// @addtogroup Les_fonctions_nD
102 /// @{
103 ///
104
105 /**
106 *
107 *      BUT:      Classe virtuelle d'interface permettant le calcul d'une fonction nD
108 *      ainsi qu'éventuellement un certain nombre d'informations supplé-
109 *      mentaires telles que dérivées.
110 *      si le nom de la Fonction = "_" il s'agit d'une Fonction interne
111 *      à un objet, c'est-à-dire gérée seulement par l'entité qui la
112 *      contient, donc pas besoin de nom (elle n'est pas utilisée autre
113 *      part). Si le nom est différent de "_" c'est une Fonction qui est
114 *      gérée et référencée dans LesFonctions, donc à partir de son nom,
115 *      on peut la retrouver.
116 *
117 *
118 * \author      Gérard Rio
119 * \version    1.0
120 * \date      01/06/2016
121 * \brief      Classe virtuelle d'interface permettant le calcul d'une fonction nD ainsi
122 *      qu'éventuellement un certain nombre d'informations supplémentaires telles que dérivées.
123 */
124
125 class Fonction_nD
126 {
127     public :
128
129         // CONSTRUCTEURS :
130         // par défaut
131         Fonction_nD(string nom = "", EnumFonction_nD typ = AUCUNE_FONCTION_nD);
132         // constructeur avec plus d'info
133         // dans le cas ou les variables sont associées à des types quelconques, il
134         // s'agit que de conteneur d'un scalaire simple. Dans le cas contraire il y a erreur !
135         Fonction_nD(const Tableau <string >& var, string nom = "", EnumFonction_nD typ =
136             AUCUNE_FONCTION_nD);
137
138         // def de tous les paramètres: utile pour être appelé par les classes dérivées
139         Fonction_nD
140             ( string nom_ref_ // nom de ref de la fonction
141             ,Tableau <string >& nom_variables_non_globales // les variables non globales
142             ,Tableau <Enum_GrandeurGlobale >& enu_variables_globale_ // enu globaux
143             ,Tableau <string >& nom_variables_globales_ // idem sous forme de strings
144             ,EnumFonction_nD typ ); // le type de fonction
145
146         // de copie

```

```

146   Fonction_nD(const Fonction_nD& Co);
147   // DESTRUCTEUR :
148   virtual ~Fonction_nD();
149
150   // METHODES PUBLIQUES :
151   //
152   // // complète les tableaux internes en fonction de la taille des conteneurs
153   // // associés aux types quelconques éventuelles
154   // // *** doit -être appelée s'il y a des grandeurs quelconque dont les conteneurs
155   // // sont associés à plusieurs scalaires
156   // // la fonction nD ne doit pas pouvoir fonctionner s'il y a des types complexes
157   // // et que cette méthode n'a pas été appelé
158   // void Preparation_Grandeur_quelconque(const Tableau <TypeQuelconque >& tqi);
159
160   // ----- virtuelles -----
161
162   // Surcharge de l'opérateur = : realise l'égalite de deux fonctions
163   virtual Fonction_nD& operator= (const Fonction_nD& elt) = 0;
164
165   // affichage de la Fonction
166   // = 0, 1 ou 2 (le plus précis)
167   virtual void Affiche(int niveau = 0) const = 0;
168   // ramène le nom de la Fonction
169   const string& NomFonction() const {return nom_ref;};
170   // vérification que tout est ok, pres à l'emploi
171   // ramène true si ok, false sinon
172   virtual bool Complet_Fonction(bool affichage = true) const = 0;
173
174   // ramène le nombre total de variables de la fonction
175   // c-a-dire locale et globales
176   virtual int NbVariable()const
177   {return nom_variables.Taille()+enu_variables_globale.Taille()
178   + nom_variables_globales.Taille();
179   };
180
181   // ramène le nombre de variables locales
182   // c-a-dire à l'exclusion des variables globales
183   virtual int NbVariable_locale()const
184   {return nom_variables.Taille();
185   };
186
187   // ramène le nombre de variables globales
188   // c-a-dire à l'exclusion des variables locales
189   virtual int NbVariable_globale()const
190   {return (enu_variables_globale.Taille()+nom_variables_globales.Taille());
191   };
192
193   // ramène le noms des variables, indépendamment des variables globales
194   const Tableau <string >& Nom_variables() const {return nom_variables;};
195
196   // ramène les énumérés des variables globales éventuelles
197   const Tableau <Enum_GrandeurGlobale >& Enu_variables_globales() const {return
198   enu_variables_globale;};
199   // ramène les noms string des variables globales éventuelles
200   const Tableau <string >& Nom_variables_globales() const {return nom_variables_globales;};
201
202   // ramène l'équivalent des nom_variables sous forme de Ddl_enum_etendu et
203   // d'EnumTypeQuelconque
204   // permet ensuite d'appeler la méthode Valeur(.. mais à la condition
205   // que l'ensemble des Nom_variables() sont représenté soit par des Ddl_enum_etendu
206   // et ou soit par des EnumTypeQuelconque
207   const Tableau <Ddl_enum_etendu>& Tab_enu_etendu()const {return tab_enu_etendu;};
208   const Tableau <EnumTypeQuelconque>& Tab_enu_quelconque() const {return tab_enu_quelconque;};
209   // un booléen pour noter l'équivalence parfaite ou non
210   bool Equivalence_nom_enu_etendu_et_enu_quelconque() const {return
211   equivalence_nom_enu_etendu_et_enu_quelconque;};
212
213   // tab_enu_etendu(i) correspond à nom_variables(index_enu_etendu(i))
214   const Tableau <int>& Iindex_enu_etendu() const {return index_enu_etendu;};
215   // tab_enu_quelconque(i) correspond à nom_variables(index_enu_quelconque(i))
216   const Tableau <int>& Iindex_enu_quelconque() const {return index_enu_quelconque;};
217
218   // .. chaque type_des_variables tab_enu_etendu(i) est associé à :
219   // si type_des_variables(i) = 1 -> un scalaire
220   // si type_des_variables(i) = 2 -> des coordonnées de dim absolue
221   // si type_des_variables(i) = 3 -> un tenseur de dim absolue
222   // si type_des_variables(i) = 4 -> une grandeur quelconque
223   // retour du tableau des indicateurs
224   const Tableau <int>& Type_des_variables_locales() const {return type_des_variables;};
225
226   // retour du tableau posi_ddl_enum, qui est nécessaire pour l'appel de la fonction
227   // Val_FnD_Evoluee(Tableau <double >* xi_,Tableau <Coordonnee> * tab_coor_,Tableau <TenseurBB> *
228   // tab_tensBB_
229   //
230   // Tableau <TypeQuelconque >* t_quelc_ )
231   // -> Pour le Ddl_enum_etendu (i) dont on récupère la liste via Tab_enu_etendu()
232   // si (i) est un scalaire alors: posi_ddl_enum(i) représente la position que doit
233   // avoir de Ddl_enum_etendu(i) dans xi_

```

```

230 // si (i) est de type Coordonnee -> position du ddl_enum_etendu(i) dans tab_coor_
231 // si (i) un tenseur -> position du ddl_enum_etendu(i) dans tab_tensBB_
232 // si (i) une grandeur quelconque -> position du ddl_enum_etendu(i) dans t_quelc_
233 const Tableau <int>& Index_dans_tableau() const {return posi_ddl_enum;};
234
235
236 // retour des tailles qu'ont les différents tableaux associés au ddl_enum
237 // tailles_tab(1) -> nb de scalaires
238 // (2) -> nb de Coordonnee
239 // (3) -> nb de tenseur
240 // (4) -> nb de grandeurs quelconques
241 const Tableau <int>& Tailles_tab() const {return tailles_tab;};
242
243
244 // ramène le nombre de composantes de la fonction
245 virtual int NbComposante() const =0;
246
247 // Lecture des donnees de la classe sur fichier
248 // le nom passé en paramètre est le nom de la Fonction
249 // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
250 // ce nom remplace le nom actuel
251 virtual void LectDonnParticulieres_Fonction_nD(const string& nom, UtilLecture * ) = 0;
252
253 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
254 // globales en cours de calcul
255 virtual void Mise_a_jour_variables_globales() = 0;
256
257 // établir le lien entre la Fonction et des Fonctions déjà existantes dont
258 // on connait que le nom
259 // permet ainsi de compléter la Fonction
260 // 1) renseigne si la Fonction dépend d'autre Fonction ou non
261 virtual bool DependAutreFoncCourbes() const {return false;}; // par défaut non
262 // 2) retourne une liste de nom correspondant aux noms de courbes dont dépend *this
263 virtual list <string>& ListDependanceCourbes(list <string>& lico) const;
264 // 3) retourne une liste de nom correspondant aux noms de Fonction dont dépend *this
265 virtual list <string>& ListDependanceFonctions(list <string>& lico) const;
266 // 4) établit la connection entre la demande de *this et les Fonctions passées en paramètres
267 virtual void Lien_entre_fonc_courbe(list <Fonction_nD *>& liptfonc,list <CourbeID *>& liptco) {};
268
269 // def info fichier de commande
270 virtual void Info_commande_Fonctions_nD(UtilLecture & entreePrinc) = 0;
271
272
273 // calcul des valeurs de la fonction, avec comme argument: uniquement des grandeurs
274 // scalaires simples
275 // retour d'un tableau de scalaires
276 Tableau <double> & Valeur_FnD_tab_scalaire(Tableau <double >* val_double)
277 {
278     {
279 // #ifdef MISE_AU_POINT
280     if (permet_affichage > 4)
281     {
282         cout << "\n fonction: " << nom_ref << " : ";
283         if (permet_affichage > 5)
284             {cout << "\n parametres d'appel: ";
285              int nb_var = val_double->Taille();
286              for (int j=1;j<=nb_var;j++)
287                  { cout << " para("<j<")= " << (*val_double)(j);}
288              };
289             Tableau <double> & inter = Valeur_FnD_interne(val_double);
290             cout << "\n retour fonction: ";
291             int nb_val = inter.Taille();
292             for (int i=1;i<=nb_val;i++)
293                 cout << " val("<i<")= " << inter(i);
294             return inter;
295         }
296     }
297 // #endif
298     return Valeur_FnD_interne(val_double);
299 }
300 catch (ErrSortieFinale)
301 // cas d'une direction voulue vers la sortie
302 // on relance l'interruption pour le niveau supérieur
303 { ErrSortieFinale toto;
304   throw (toto);
305 }
306 catch(...)
307 { cout << "\n ** erreur \n Valeur_FnD_tab_scalaire(...)"<endl;
308   this->Affiche();
309   ErrCalculFct_nD toto;throw (toto);
310   Sortie(1);
311 };
312 // on ne doit jamais arriver ici !
313 return Valeur_FnD_interne(val_double); // pour taire le compilo
314 };
315
316 // calcul des valeurs de la fonction, à l'aide de paramètres = grandeurs évoluées

```



```

317 // retour d'un tableau de scalaires
318 // les différents tableaux doivent contenir toutes les informations nécessaires pour l'appel
319 // NB: il peut y avoir plus d'info que nécessaire
320 // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
321 // il peut y avoir éventuellement des grandeurs quelconques et éventuellement des types non scalaire
322 // dans ce dernier cas il faut renseigner le tableau de numéro d'ordre t_num_ordre
323 Tableau <double> & Val_FnD_Evoluee(Tableau <double >* val_ddl_enum
324                                     ,Tableau <Coordonnee> * coor_ddl_enum
325                                     ,Tableau <TenseurBB* >* tens_ddl_enum
326                                     ,Tableau <const TypeQuelconque * >* tqi = NULL
327                                     ,Tableau <int> * t_num_ordre = NULL
328                                     )
329 { if (equivalence_nom_enu_etendu_et_enu_quelconque)
330     {try
331         { Vers_tab_double(t_inter_double,val_ddl_enum,coor_ddl_enum,tens_ddl_enum
332                             ,tqi,t_num_ordre);
333 // #ifndef MISE_AU_POINT
334         if (permet_affichage > 4)
335             {cout << "\n fonction: " << nom_ref << " : ";
336                 if (permet_affichage > 5)
337                     {cout << "\n parametres d'appel: ";
338                         int nb_var = t_inter_double.Taille();
339                         for (int j=1;j<=nb_var;j++)
340                             { cout << " para("<j<")= " << t_inter_double(j);}
341                     };
342                     Tableau <double> & inter = Valeur_FnD_interne(&t_inter_double);
343                     cout << "\n retour fonction: ";
344                     int nb_val = inter.Taille();
345                     for (int i=1;i<=nb_val;i++)
346                         cout << " val("<i<")= " << inter(i);
347                     return inter;
348                 }
349             else
350 // #endif
351                 return Valeur_FnD_interne(&t_inter_double);
352         }
353     catch (ErrSortieFinale)
354         // cas d'une direction voulue vers la sortie
355         // on relance l'interruption pour le niveau supérieur
356         { ErrSortieFinale toto;
357             throw (toto);
358         }
359     catch(...)
360         { cout << "\n ** erreur \n Valeur_FnD_Evoluee(...)<<endl;
361             ErrCalculFct_nD toto;throw (toto);
362             this->Affiche();Sortie(1);
363         };
364     }
365     else {cout << "\n erreur appel Val_FnD_Evoluee(, variables non definies ! ";
366             this->Affiche();
367             ErrCalculFct_nD toto;throw (toto);Sortie(1);
368             return Valeur_FnD_interne(&t_inter_double); // pour taire le compilateur
369         };
370     // on ne doit jamais arriver ici !
371     return Valeur_FnD_interne(&t_inter_double); // pour taire le compilo
372 };
373
374 // mise à disposition des conteneurs pour l'appel de Valeur_FnD sans paramètres
375 // il n'est pas autorisé de changer la taille du conteneur, c'est seulement les valeurs qui doivent
376 // être modifiées sinon il y aura des pb
377 // l'ordre des paramètres doit respecter celui donné par les fonctions Index_dans_tableau()
378 Tableau <double >& Val_ddl_enum() {return val_ddl_enum;} // valeur associé à l'enu_etendu(i) si
scalaire
379 Tableau <Coordonnee >& Coor_ddl_enum() {return coor_ddl_enum;}; // idem si Coordonnées (en
orthonormée)
380 // retourne le tableau de l'ensemble des grandeurs de type Coordonnee qui sont nécessaire pour
l'appel
381 // chaque type est repéré par premier_famille_Coord(j) ce qui correspond à coor_ddl_enum(j)
382 Tableau <Ddl_enu_etendu>& Premier_famille_Coord() {return premier_famille_Coord;}
383
384 Tableau <TenseurBB* >& Tens_ddl_enum() {return tens_ddl_enum;}; // idem si Tenseur (en orthonormée
!)
385 int Absolue() const {return absolue;} // indique si les tenseurs sont en absolue ou non
386
387 Tableau <Ddl_enu_etendu>& Premier_famille_tenseur() {return premier_famille_tenseur;}
388
389 // -- retourne une liste de grandeurs quelconques équivalentes aux types évoluées non scalaires
390 // li_equi_Quel_evolue : est l'équivalent des grandeurs évoluées: coor_ddl_enum et tens_ddl_enum
391 // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
évoluées
392 // non scalaires
393 // l'ordre l'apparition dans la liste est telle que l'on a:
394 // 1) tous les Coordonnees 2) puis tous les tenseurs
395 // dans le même ordre que pour les tableaux: coor_ddl_enum puis tens_ddl_enum
396 // ==>==> important: il est permis "que" de changer les valeurs dans les conteneurs
397 List_io <TypeQuelconque >& Li_equi_Quel_evolue() {return li_equi_Quel_evolue;};
398

```

```

399 // -- retourne une liste des grandeurs évoluées uniquement scalaire
400 // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
scalaires
401 // est équivalent à val_ddl_enum, l'ordre d'apparition est celui de val_ddl_enum
402 // =>=> important: il est permis "que" de changer les valeurs dans les conteneurs
403 List_io <Ddl_enum_etendu>& Li_enu_etendu_scalaire() {return li_enu_etendu_scalaire;}
404
405 // calcul équivalent, cf. les paramètres de passage
406 // pour que l'appel à cette méthode soit correcte, il faut que les listes correspondent
407 // à celles de Li_equi_Quel_evoluee(), et Li_enu_etendu_scalaire(),
408 // =>=> il faut donc utiliser ces listes en les récupérant auparavant !!
409 // Si t_num_ordre est Null cela signifie que tous les tqi sont de grandeurs scalaire
410 // dans le cas contraire t_num_ordre(i) donne le numéro d'ordre du scalaire dans tqi qui correspond
à Nom_variable(i)
411 // =>=> important: il est permis "que" de changer les valeurs dans les conteneurs
412 Tableau <double> & Valeur_FnD_Evoluee(Tableau <double >* val_ddl_enum
413 ,List_io <Ddl_enum_etendu>* li_evolue_scalaire
414 ,List_io <TypeQuelconque >* li_evoluee_non_scalaire
415 ,Tableau <const TypeQuelconque * >* tqi
416 ,Tableau <int> * t_num_ordre)
417 { if (equivalence_nom_enu_etendu_et_enu_quelconque)
418 { try
419 {Vers_tab_double(t_inter_double, val_ddl_enum, li_evolue_scalaire, li_evoluee_non_scalaire, tqi, t_num_ordre);
420 // #ifdef MISE_AU_POINT
421 if (permet_affichage > 4)
422 {cout << "\n fonction: " << nom_ref << " : ";
423 if (permet_affichage > 5)
424 {cout << "\n paramètres d'appel: ";
425 int nb_var = t_inter_double.Taille();
426 for (int j=1; j<=nb_var; j++)
427 { cout << " para(" << j << ")= " << t_inter_double(j); }
428 };
429 Tableau <double> & inter = Valeur_FnD_interne(&t_inter_double);
430 cout << "\n retour fonction: ";
431 int nb_val = inter.Taille();
432 for (int i=1; i<=nb_val; i++)
433 cout << " val(" << i << ")= " << inter(i);
434 return inter;
435 }
436 else
437 // #endif
438 return Valeur_FnD_interne(&t_inter_double);
439 }
440 catch (ErrSortieFinale)
441 // cas d'une direction voulue vers la sortie
442 // on relance l'interruption pour le niveau supérieur
443 { ErrSortieFinale toto;
444 throw (toto);
445 }
446 catch(...)
447 { cout << "\n ** erreur \n Valeur_FnD_Evoluee(...)" << endl;
448 this->Affiche();
449 ErrCalculFct_nD toto; throw (toto);
450 Sortie(1);
451 };
452 }
453 else {cout << "\n erreur appel Val_FnD_Evoluee(, variables non definies ! ";
454 // on génère une interruption ce qui permettra de dépiler les appels
455 this->Affiche();
456 ErrCalculFct_nD toto; throw (toto);
457 Sortie(1);
458 return Valeur_FnD_interne(&t_inter_double); // pour taire le compilateur
459 };
460 // on ne doit jamais arriver ici !
461 return Valeur_FnD_interne(&t_inter_double); // pour taire le compilateur
462 };
463
464 // calcul équivalent, mais pour des paramètres de type ddl enum étendu et/ou type quelconque
465 // pour que l'appel à cette méthode soit correcte, il faut que la dimension de t_enu + celle de tqi
466 // soit identique à celle du tableau Nom_variables() : en fait t_enu et tqi doivent représenter les
variables
467 // Si t_num_ordre est Null cela signifie que tous les tqi sont de grandeurs scalaire
468 // dans le cas contraire t_num_ordre(i) donne le numéro d'ordre du scalaire dans tqi qui correspond
à Nom_variable(i)
469 Tableau <double> & Valeur_FnD(Tableau <Ddl_etendu> * t_enu, Tableau <const TypeQuelconque * > * tqi
470 ,Tableau <int> * t_num_ordre);
471
472 // indicateur permettant de connaître rapidement si
473 // la fonction dépend de la position d'un point M: ici au temps tdt
474 // 0 : la fonction ne dépend pas de la position d'un point M
475 // =i non nul : la fonction dépend de la position d'un point M
476 // et = le nombre de composantes demandés (ne sert pas vraiment)
477 // dans ce cas, une au moins des nom_variables a un nom de la même famille que le ddl Xl
478 // si = -1 : cela signifie que la fonction dépend "que" de M
479 int Depend_M() const {return depend_M;}; // cas d'une position courante
480 int Depend_Mt() const {return depend_Mt;}; // cas de la position spécifiquement a t

```

```

481     int Depend_M0() const {return depend_M0;}; // cas de la position initiale
482
483     // calcul des valeurs de la fonction, dans le cas où les variables
484     // sont "tous" des grandeurs globales
485     Tableau <double> & Valeur_pour_variables_globales()
486     { try
487     {
488         // #ifdef MISE_AU_POINT
489         if (permet_affichage > 4)
490         {cout << "\n fonction: "«nom_ref»" : ";
491          Tableau <double> & inter = Valeur_pour_variables_globales_interne();
492          cout << "\n retour fonction: ";
493          int nb_val = inter.Taille();
494          for (int i=1;i<=nb_val;i++)
495              cout << " val("«i»")= "«inter(i)»;
496          return inter;
497        }
498        else
499        // #endif
500        return Valeur_pour_variables_globales_interne();
501    }
502    catch (ErrSortieFinale)
503        // cas d'une direction voulue vers la sortie
504        // on relance l'interruption pour le niveau supérieur
505        { ErrSortieFinale toto;
506          throw (toto);
507        }
508    catch(...)
509        { cout << "\n ** erreur Valeur_pour_variables_globales(...)"«endl;
510          ErrCalculFct_nD toto;throw (toto);
511          this->Affiche();Sortie(1);
512        };
513        // on ne doit jamais arriver ici !
514        return Valeur_pour_variables_globales_interne(); // pour taire le compilé
515    };
516
517 // ---- fonctions internes qui ne "doivent pas !!!" être utilisée directement ---
518 // elles sont mises en public, pour pouvoir être appelées par les classes dérivées
519 // sans avoir à déclarer en friend les classes dérivées, sinon cela fait une boucle
520 // sans fin, d'où la solution choisit pour éviter ce pb
521
522 // calcul des valeurs de la fonction, retour d'un tableau de scalaires
523 virtual Tableau <double> & Valeur_FnD_interne(Tableau <double >* val_ddl_enum) = 0;
524
525 // calcul des valeurs de la fonction, dans le cas où les variables
526 // sont "tous" des grandeurs globales
527 virtual Tableau <double> & Valeur_pour_variables_globales_interne() = 0;
528
529 //----- lecture écriture de restart -----
530 // cas donne le niveau de la récupération
531 // = 1 : on récupère tout
532 // = 2 : on récupère uniquement les données variables (supposées comme telles)
533 virtual void Lecture_base_info(ifstream& ent,const int cas) = 0;
534 // cas donne le niveau de sauvegarde
535 // = 1 : on sauvegarde tout
536 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
537 virtual void Ecriture_base_info(ofstream& sort,const int cas) = 0;
538 // sortie du schemaXML: en fonction de enu
539 virtual void SchemaXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu) = 0;
540
541 // ----- static -----
542
543 // ramène un pointeur sur la Fonction correspondant au type de Fonction passé en paramètre
544 // IMPORTANT : il y a création d'une Fonction (utilisation d'un new)
545 static Fonction_nD* New_Fonction_nD(string& nom,EnumFonction_nD typeFonction);
546 // ramène un pointeur sur une Fonction copie de celle passée en paramètre
547 // IMPORTANT : il y a création d'une Fonction (utilisation d'un new)
548 static Fonction_nD* New_Fonction_nD(const Fonction_nD& Co);
549
550 // ramène la liste des identificateurs de Fonctions actuellement disponibles
551 static list <EnumFonction_nD> Liste_Fonction_disponible();
552
553 // ----- non virtuelle -----
554
555 // ramène le type de la Fonction
556 EnumFonction_nD Type_Fonction() const { return typeFonction;};
557
558
559 protected :
560
561 // VARIABLES PROTEGEES :
562 EnumFonction_nD typeFonction; // type de la fonction
563 string nom_ref; // nom de référence de la Fonction
564
565 Tableau <string > nom_variables; //variables de la fonction, vu de l'extérieur
566 Tableau <double > t_inter_double; // tableau de même dimension que nom_variable
567 // sert pour passer les infos à la méthode Valeur_FnD_interne(..

```

```

568
569 Tableau <Enum_GrandeurGlobale > enu_variables_globale; //tableau des énumérés
570 // de variables globales
571 // éventuellement vide s'il ne sert pas
572 Tableau <string > nom_variables_globales; //tableau des noms en string
573 // de variables globales
574 // éventuellement vide s'il ne sert pas
575
576 // *** important: les tableaux nom_variables, enu_variables_globales et
577 // nom_variables_globales s'ajoutent
578 // ils ne sont pas liés entre eux: le nombre total de variables est
579 // la somme des trois tableaux. En général c'est soit l'un , soit l'autre
580 // ou le dernier mais on peut avoir un mixte
581
582 // ---- tableaux de grandeurs équivalentes aux nom_variables (locales) ----
583 // un tableau tab_enu_etendu et un tableau tab_enu_quelconque:
584 // tab_enu_etendu.Taille()+tab_enu_quelconque.Taille() == nom_variables.Taille() -> grandeurs
locales
585 Tableau <Ddl_enum_etendu> tab_enu_etendu;
586
587 // tab_enu_etendu peut se décomposer de deux manières équivalentes :
588 // 1) tableaux: val_ddl_enum (scalaires) , coor_ddl_enum (Coordonnees) , tens_ddl_enum (tenseur)
589 // permet aux utilisateurs d'obtenir un stockage et un appel avec des grandeurs évoluées,
distinctes
590 // 2) list: li_enu_etendu_scalaire (scalaires) , li_equi_Quel_evolue (Coordonnees "et" tenseurs)
591 // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
scalaires
592 // et permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
évoluées
593 // excluant les scalaire, et appels associés
594 // li_equi_Quel_evolue correspond à des grandeurs quelconques contenant Coordonnees et tenseur,
595 // est indépendant de tab_enu_quelconque, qui lui est complètement quelconque, on ne sait pas a
priori
596 // ce qu'il y a dedans
597
598 // .. chaque grandeur quelconque doit être associé, lors de l'appel de la fonction, à un numéro
d'ordre
599 // du coup, pour l'instant on n'utilise qu'un scalaire par grandeur quelconque, mais le conteneur
600 // peut en contenir plusieurs
601 Tableau <EnumTypeQuelconque> tab_enu_quelconque;
602
603 // ----- grandeurs évoluées y compris les grandeurs quelconques -----
604 // .. chaque nom_variables(j) est associé à :
605 Tableau <int> type_des_variables; // si type_des_variables(j) = 1 -> un scalaire
606 // si type_des_variables(j) = 2 -> des coordonnées de dim absolue
607 // si type_des_variables(j) = 3 -> un tenseur de dim absolue
608 // si type_des_variables(j) = 4 -> une grandeur quelconque
609 // si type_des_variables(j) = 0 -> une grandeur qui n'est pas reconnue
en local
610
611 // .... dans le cas d'un Ddl_enum_etendu : donc c'est relatif au tableau: tab_enu_etendu
612 Tableau <double > val_ddl_enum; // valeur associé au ddl tab_enu_etendu(i) si scalaire
613 // tab_enu_etendu(i) <=> val_ddl_enum(posi_ddl_enum(i))
614 List_io <Ddl_enum_etendu> li_enu_etendu_scalaire; // est équivalent à val_ddl_enum,
615 // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
scalaires
616
617 Tableau <Ddl_enum_etendu> premier_famille_Coord; // les ddl du premier de famille
618 //coor_ddl_enum(j) est associé avec premier_famille_Coord(j)
619 Tableau <Coordonnee > coor_ddl_enum; // le vecteur Coordonnée associé au ddl tab_enu_etendu(i) si
Coordonnees
620 Tableau <int > num_dans_coor; // le numéro dans le vecteur associé au ddl tab_enu_etendu(i)
621 // avec j = posi_ddl_enum(i)
622 // tab_enu_etendu(i) <=> coor_ddl_enum(j) (num_dans_coor(i))
623 // bien noter qu'un objet Coordonnee peut contenir plusieurs Ddl_enum_etendu
624
625 Tableau <Ddl_enum_etendu> premier_famille_tenseur; // les ddl du premier de famille
626 // (*tens_ddl_enum)(j) est associé avec premier_famille_tenseur(j)
627 Tableau <TenseurBB* > tens_ddl_enum; // tenseur associé au ddl tab_enu_etendu(i) si Tenseur
int absolue; // indique si les tenseurs sont en absolue ou non
628 Tableau <Deuxentiers_enu > ind_tens; // indice dans le tenseur de la grandeur
629 // avec j = posi_ddl_enum(i)
630 // avec k = ind_tens(i).i et l = ind_tens(i).j
631 // tab_enu_etendu(i) <=> (*tens_ddl_enum)(j)(K,l)
632 // bien noter qu'un objet Tenseur peut contenir plusieurs Ddl_enum_etendu
633
634 // li_equi_Quel_evolue : est l'équivalent des grandeurs évoluées: coor_ddl_enum et tens_ddl_enum
635 // permet aux utilisateurs d'accéder à un stockage transitoire contenant "que" les grandeurs
évoluées
636
637 List_io <TypeQuelconque > li_equi_Quel_evolue; // non scalaires
638
639 // tab_equi_Coor(j) contient un Coordonnee équivalent à coor_ddl_enum(j)
640 Tableau <List_io<TypeQuelconque >::iterator > tab_equi_Coor;
641 //tab_equi_tens(j) contient un tenseur équivalent à tens_ddl_enum(j)
642 Tableau <List_io<TypeQuelconque >::iterator > tab_equi_tens;
643
644

```

```

645 // un système d'adressage indirect pour le passage : tab_enu_etendu(i) <-> grandeur évoluée
646 Tableau <int> posi_ddl_enum; // posi_ddl_enum(i) donne la position de tab_enu_etendu(i)
647 // dans le tableau :
648 // si scalaire : dans le tableau val_ddl_enum -> val_ddl_enum(posi_ddl_enum(i))
649 // si Coordonnées : dans le tableau coord_ddl_enum -> coord_ddl_enum(posi_ddl_enum(i))
650 // // ici il s'agira de tous les ddl de la même famille,
651 // // et c'est num_dans_coord qui permet de trouver la position
652 // si Tenseur : dans le tableau tens_ddl_enum -> tens_ddl_enum(posi_ddl_enum(i))
653 // // et c'est ind_tens qui permettra de trouver les indices
654
655 // un système d'adressage indirect pour le passage : nom_variables(j) <-> grandeur évoluée
656 // tab_enu_etendu(i) correspond à nom_variables(index_enu_etendu(i))
657 Tableau <int> index_enu_etendu;
658
659 // .... dans le cas d'une grandeur quelconque
660 // pour les grandeurs quelconques, l'utilisateur doit transmettre le numéro d'ordre correspondant à
l'équivalence
661 // du ddl_etendu, du coup on ne s'occupe pas de gérer les numéros d'ordre pour les grandeurs
quelconques
662
663 // un système d'adressage indirect pour le passage : nom_variables(j) <-> grandeur quelconque
664 // tab_enu_quelconque(i) correspond à nom_variables(index_enu_quelconque(i))
665 Tableau <int> index_enu_quelconque;
666
667
668 // stockage des tailles qu'ont les différents tableaux associés
669 Tableau <int> tailles_tab; // tailles_tab(1) -> nb de scalaires
670 // // (2) -> nb de Coordonnee
671 // // (3) -> nb de tenseur
672 // // (4) -> nb de grandeurs quelconques
673 // ----- fin grandeurs évoluées y compris les grandeurs quelconques-----
674
675 // ces deux tableaux permettent d'appeler la méthode Valeur(..
676 // un booléen pour noter l'équivalence parfaite ou non
677 bool equivalence_nom_enu_etendu_et_enu_quelconque;
678
679 // un tableau intermédiaire qui sert pour Valeur(Tableau <Ddl_etendu> ...
680 // c-a-d pour les variables non globales
681 Tableau <double > x_x_i;
682 // un tableau intermédiaire qui sert pour toutes les variables globales
683 Tableau <double > x_glob;
684
685 int depend_M; // indicateur permettant de connaître rapidement si
686 // la fonction dépend de la position courante d'un point M
687 // 0 : la fonction ne dépend pas de la position d'un point M
688 // non nul : la fonction dépend de la position d'un point M
689 // dans ce cas, une au moins un des nom_variables
690 // a un nom de la même famille que le ddl X1
691 // et depend_M = nombre de composantes demandés (ne sert pas vraiment)
692 // si = -1 : cela signifie que la fonction dépend "que" de M
693 int depend_Mt; // idem pour le temps t
694 int depend_M0; // idem pour le temps 0
695
696 bool depend_temps; // indicateur permettant de connaître rapidement si
697 // la fonction dépend du temps ou non
698
699 // ----- controle de la sortie des informations: utilisé par les classes dérivées
700 int permet_affichage; // pour permettre un affichage spécifique dans les méthodes, pour les erreurs
et warning
701
702 //-----
703
704 // METHODES PROTEGEES :
705 // ramène true si les variables de la classe mère sont complété
706 bool Completet_var() const;
707
708 // définit le paramètre depend_M en fonction des nom_variables
709 void Definition_depend_M();
710
711 // définit le paramètre depend_temps en fonction des nom_variables
712 void Definition_depend_temps();
713
714 // construction à partir des noms de variables, des tableaux tab_enu_etendu et
715 // tab_enu_quelconque
716 // void Construction_enu_etendu_et_quelconque();
717
718 // Construction des index pour les grandeurs évoluées, ainsi que les conteneurs
719 void Construction_index_conteneurs_evoluees();
720
721 // Passage des infos variables évoluées en tableau de réels
722 // I/O : d
723 Tableau <double > & Vers_tab_double(Tableau <double > & di
724 // // ,const Tableau <double >* val_ddl_enum
725 // // ,const Tableau <Coordonnee> * coord_ddl_enum
726 // // ,const Tableau <TenseurBB* >* tens_ddl_enum
727 // // ,const Tableau <const TypeQuelconque * >* tqi = NULL
728 // // ,const Tableau <int> * t_num_ordre = NULL);

```

```

729 // idem, mais via les listes
730 // val_ddl_enum(k) correspond à li_evolue_scalaire de rang i
731 Tableau <double > & Vers_tab_double(Tableau <double > & di
732     ,const Tableau <double >* val_ddl_enum
733     ,List_io <Ddl_enum_etendu>* li_evolue_scalaire
734     ,List_io <TypeQuelconque >* li_evoluee_non_scalaire
735     ,const Tableau <const TypeQuelconque * >* tqi = NULL
736     ,const Tableau <int> * t_num_ordre = NULL);
737
738 // mise à jour des variables globales: en fonction de l'apparition de nouvelles variables
739 // globales en cours de calcul
740 // méthode interne: qui est utilisé par les fonctions dérivées pour la méthode:
741 // Mise_a_jour_variables_globales
742 // retourne false si rien n'a changé
743 bool Mise_a_jour_variables_globales_interne();
744
745 // appel de la fonction sous forme d'une méthode avec un nombre non limité
746 // de paramètres: en fait utilise Valeur
747 // c'est une méthode avec retour uniquement en scalaire, sinon -> erreur
748 // calcul des valeurs de la fonction
749 // ***** fonction a priori dangereuse, à éviter , pour l'instant ne sert pas !! ****
750 double Val_avec_nbArgVariable(double x,...);
751
752 // méthode utilisée par les classes dérivées, pour transférer les infos qui sont
753 // gérées au niveau de Fonction_nD
754 Fonction_nD& Transfert_info(const Fonction_nD& elt);
755
756 // affichage des données internes, utilisée par les fonctions dérivées
757 // niveau donne le degré d'affichage
758 void Affiche_interne(int niveau) const;
759
760 //----- méthodes appelée par les classes dérivées -----
761 // relatives aux données gérées par Fonction_nD
762 // cas donne le niveau de la récupération
763 // = 1 : on récupère tout
764 // = 2 : on récupère uniquement les données variables (supposées comme telles)
765 void Lect_base_info(ifstream& ent,const int cas);
766 // cas donne le niveau de sauvegarde
767 // = 1 : on sauvegarde tout
768 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
769 void Ecrit_base_info(ofstream& sort,const int cas);
770 // sortie du schemaXML: en fonction de enu
771 void SchemXML_Fonctions_nD(ofstream& sort,const Enum_IO_XML enu);
772
773 // lecture d'une ou plusieurs variables
774 // peut-être appelée plusieurs fois,
775 // stockage des infos dans Fonction_nD
776 void Lecture_variables(string& nom_lu,UtilLecture * entreePrinc);
777 // méthode pour savoir si le nom_lu est un mot clé relatif à la lecture de variables
778 bool Est_relatif_a_lecture_variable(string& nom_lu)
779     {if ( (nom_lu == "un_argument=")
780         || (nom_lu == "deb_list_var_")
781         )
782         return true;
783         else return false;
784     };
785
786 // affichage des variables de la fonction, dépend du niveau d'impression
787 void Affichage_variables() const;
788
789 // récupération des valeurs des variables globales et stockage dans le tableau
790 // interne x_glob
791 void Recup_Grandeurs_globales();
792
793 private :
794
795 // un tableau intermédiaire, en général vide, mais qui sert si on utilise
796 // la méthode Val_avec_nbArgVariable
797 Tableau <double >* xinter=NULL;
798
799
800 };
801 /// @} // end of group
802
803 #endif

```

## 7.538 LesCourbes1D.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //

```

```

8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      19/01/2001
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *   *****/
37 *   BUT:      gestion des différentes courbes 1D enregistrées.
38 *
39 *   *****
40 *
41 *   VERIFICATION:
42 *
43 *   ! date !   auteur !           but
44 *   -----
45 *   !           !           !           !
46 *   *****
47 *
48 *   MODIFICATIONS:
49 *
50 *   ! date !   auteur !           but
51 *   -----
52 *   !           !           !           !
53 *   *****/
54 #ifndef LESCOURBES1D_H
55 #define LESCOURBES1D_H
56 #include "CourbelD.h"
57 #include <list>
58 #include "UtilLecture.h"
59 #include "MotCle.h"
60 #include <map>
61 #include "Enum_IO_XML.h"
62 /// @addtogroup Les_courbes_1D
63 /// @
64 ///
65
66 /**
67 *
68 *
69 *
70 * \author   Gérard Rio
71 * \version  1.0
72 * \date    19/01/2001
73 * \brief   Classe de gestion des différentes courbes 1D enregistrées.
74 *
75 */
76
77 class LesCourbes1D
78 {
79 public :
80     // un conteneur pour le stockage courant
81     // class Ref_CourbelD
82     // { public:
83     //     string nom_courbe;
84     //     CourbelD* courbe;
85     // };
86     // CONSTRUCTEURS :
87     // constructeur par défaut
88     LesCourbes1D ();
89     // DESTRUCTEUR :
90     ~LesCourbes1D ();
91     // METHODES PUBLIQUES :
92     // lecture des courbes
93     void Lecture(UtilLecture & entreePrinc);

```

```

94
95 // affichage et definition interactive des commandes
96 void Info_commande_lesCourbes1D(UtilLecture & entreePrinc);
97
98 // affichage des courbes
99 void Affiche() const ;
100
101 // test si la courbe de nom st1 existe reellement
102 // retourne false si n'existe pas , true sinon
103 bool Existe(const string & st1) const ;
104
105 // retourne la courbe correspondant a une cle
106 //const
107 Courbe1D * Trouve(const string & st1) const ;
108
109 // vérification que tout est ok, pres à l'emploi
110 // ramène true si ok, false sinon
111 bool Complet();
112
113 // utilitaire pour lire une courbe, soit qui ne sera pas stocké par LesCourbes1D
114 // dans ce cas sont nom est "_", soit son nom correspond à une ref de courbes existante
115 // dans tous les cas on lit un nom et un type de courbe c-a-d un string et un EnumCourbe1D
116 // ----- différents cas -----
117 // ptcourbe: soit == NULL, : on lit un nom, et on lit un type de courbe
118 // 1) si celui-ci est "_", cela signifie que la courbe à lire
119 // est interne à l'utilisateur, on crée une courbe, on lit la courbe
120 // avec Lecture_base_info de la courbe et on ramène un pointeur sur la
121 // nouvelle courbe
122 // 2) si celui-ci est différent de "_", c'est une référence de courbe
123 // on lit la référence et on ramène un pointeur sur la courbe de
124 // LesCourbes1D correspondant
125 //
126 // soit == une courbe existante: on lit un nom, et on regarde le nom actuel de la courbe pointée
127 // par ptcourbe que l'on appellera nom_ref
128 // 1) nom == "_" et nom_ref == "_"
129 // 1-a les deux courbes sont du même type on relie les données avec
130 // Lecture_base_info et on ramène ptcourbe
131 // 1-b les deux courbes sont de type différent, on supprime la courbe pointée par
132 // ptcourbe, on en crée une nouvelle adoc, et on ramène un pointeur dessus
133 // 2) nom != "_" et nom_ref == "_"
134 // la courbes pointé par ptcourbe est supprimé, et on associe le pointeur de retour
135 // a la courbe correspondant à nom de LesCourbes1D
136 // 3) nom == "_" et nom_ref != "_"
137 // on crée une nouvelle courbe adoc, on lie avec Lecture_base_info, et on ramène
138 // un pointeur sur la courbe ainsi créée
139
140 Courbe1D * Lecture_pour_base_info(ifstream& ent,const int cas,Courbe1D * ptcourbe);
141
142 // écriture pour base info
143 // c'est le pendant de Lecture_pour_base_info, de manière à être cohérent
144 static void Ecriture_pour_base_info(ofstream& sort,const int cas,Courbe1D * ptcourbe);
145
146 //----- lecture écriture dans base info -----
147 // cas donne le niveau de la récupération
148 // = 1 : on récupère tout
149 // = 2 : on récupère uniquement les données variables (supposées comme telles)
150 void Lecture_base_info(ifstream& ent,const int cas);
151 // cas donne le niveau de sauvegarde
152 // = 1 : on sauvegarde tout
153 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
154 void Ecriture_base_info(ofstream& sort,const int cas);
155 // sortie du schemaXML: en fonction de enu
156 void SchemaXML_lesCourbes1D(ofstream& sort,const Enum_IO_XML enu) ;
157
158 protected :
159
160 // VARIABLES PROTEGEES :
161 // list <Ref_Courbe1D> listeDeCourbe1D; // liste des courbes 1D
162 // liste de courbes sous forme d'un arbre pour faciliter la recherche
163 // map < string, Ref_Courbe1D , std::less <string> > listeDeCourbe1D;
164 // map < string, Courbe1D * , std::less <string> > listeDeCourbe1D;
165
166
167 static MotCle motCle; // liste des mots clés
168
169 // METHODES PROTEGEES :
170
171 };
172 /// @} // end of group
173
174 #endif

```



## 7.539 LesFonctions\_nD.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:           01/06/2016                               $   *
31 *                                                         $   *
32 *   AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)      $   *
33 *                                                         $   *
34 *   PROJET:        Herezh++                                  *
35 *                                                         *
36 *                                                         $   *
37 *****/
38 *   BUT:gestion des différentes fonctions multivariabes enregistrées.*
39 *                                                         $   *
40 *   ***** *
41 *
42 *   VERIFICATION:
43 *
44 *   ! date !   auteur !           but           !           *
45 *   ----- *
46 *   !           !           !           !           !           *
47 *   $           *
48 *   ***** *
49 *   MODIFICATIONS:
50 *   ! date !   auteur !           but           !           *
51 *   ----- *
52 *   $           *
53 *****/
54 #ifndef LESFONCTIONS_ND_H
55 #define LESFONCTIONS_ND_H
56 #include "Fonction_nD.h"
57
58 #include <list>
59 #include "UtilLecture.h"
60 #include "MotCle.h"
61 #include <map>
62 #include "Enum_IO_XML.h"
63 #include "LesCourbes1D.h"
64
65
66 /// @addtogroup Les_fonctions_nD
67 /// @{
68 ///
69
70 /**
71 *
72 * \author   Gérard Rio
73 * \version  1.0
74 * \date    01/06/2016
75 * \brief   gestion des différentes fonctions multivariabes enregistrées.
76 *
77 */
78
79 class LesFonctions_nD
80 {
81 public :
82     // CONSTRUCTEURS :
83     // constructeur par défaut
84     LesFonctions_nD (const LesCourbes1D* lesC1 = NULL );
85     // DESTRUCTEUR :

```

```

86 ~LesFonctions_nD ();
87 // METHODES PUBLIQUES :
88
89 // changement du pointeur sur LesCourbes1D: prévu une seule fois
90 // sinon cela veut dire que lesCourbes1D n'est pas un singleton !!
91 // donc n'est possible que si lesCourbes1D n'a pas été attribué
92 // ok également si c'est le même pointeur qui est déjà enregistré
93 void Attribut_pointeur_lesCourbes1D(const LesCourbes1D * lesC);
94
95 // lecture des Fonctions
96 void Lecture(UtilLecture & entreePrinc);
97
98 // affichage et definition interactive des commandes
99 void Info_commande_lesFonctions_nD(UtilLecture & entreePrinc);
100
101 // affichage des Fonctions
102 void Affiche() const ;
103
104 // test si la courbe de nom stl existe reellement
105 // retourne false si n'existe pas , true sinon
106 bool Existe(const string & stl) const ;
107
108 // retourne la courbe correspondant a une cle
109 //const
110 Fonction_nD * Trouve(const string & stl) const ;
111
112 // vérification que tout est ok, pres à l'emploi
113 // ramène true si ok, false sinon
114 bool Complet();
115
116 // utilitaire pour lire une courbe, soit qui ne sera pas stocké par LesFonctions_nD
117 // dans ce cas sont nom est "_", soit son nom correspond à une ref de nom de Fonctions existante
118 // dans tous les cas on lit un nom et un type de courbe c-a-d un string et un EnumFonction_nD
119 // ----- différents cas -----
120 // ptcourbe: soit == NULL, : on lit un nom, et on lit un type de courbe
121 // 1) si celui-ci est "_", cela signifie que la courbe à lire
122 // est interne à l'utilisateur, on crée une courbe, on lit la courbe
123 // avec Lecture_base_info de la courbe et on ramène un pointeur sur la
124 // nouvelle courbe
125 // 2) si celui-ci est différent de "_", c'est une référence de courbe
126 // on lit la référence et on ramène un pointeur sur la courbe de
127 // LesFonctions_nD correspondant
128 //
129 // soit == une courbe existante: on lit un nom, et on regarde le nom actuel de la courbe pointée
130 // par ptcourbe que l'on appellera nom_ref
131 // 1) nom = "_" et nom_ref = "_"
132 // 1-a les deux Fonctions sont du même type on relie les données avec
133 // Lecture_base_info
134 // et on ramène ptcourbe
135 // 1-b les deux Fonctions sont de type différent, on supprime la courbe pointée par
136 // ptcourbe, on en crée une nouvelle adoc, et on ramène un pointeur dessus
137 // 2) nom != "_" et nom_ref == "_"
138 // la Fonctions pointé par ptcourbe est supprimé, et on associe le pointeur de
139 // retour
140 // a la courbe correspondant à nom de LesFonctions_nD
141 // 3) nom == "_" et nom_ref != "_"
142 // on crée une nouvelle courbe adoc, on lie avec Lecture_base_info, et on ramène
143 // un pointeur sur la courbe ainsi créée
144
145 Fonction_nD * Lecture_pour_base_info(ifstream& ent,const int cas,Fonction_nD * ptcourbe);
146
147 // écriture pour base info
148 // c'est le pendant de Lecture_pour_base_info, de manière à être cohérent
149 static void Ecriture_pour_base_info(ofstream& sort,const int cas,Fonction_nD * ptcourbe);
150
151 //----- lecture écriture dans base info -----
152 // cas donne le niveau de la récupération
153 // = 1 : on récupère tout
154 // = 2 : on récupère uniquement les données variables (supposées comme telles)
155 void Lecture_base_info(ifstream& ent,const int cas);
156 // cas donne le niveau de sauvegarde
157 // = 1 : on sauvegarde tout
158 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
159 void Ecriture_base_info(ofstream& sort,const int cas);
160 // sortie du schemaXML: en fonction de enu
161 void SchemaXML_lesFonctions_nD(ofstream& sort,const Enum_IO_XML enu) ;
162
163 protected :
164
165 // VARIABLES PROTEGEES :
166 // list <Ref_Fonction_nD> listeDeFonction_nD; // liste des Fonctions 1D
167 // liste de Fonctions sous forme d'un arbre pour faciliter la recherche
168 // map < string, Ref_Fonction_nD , std::less <string> > listeDeFonction_nD;
169 // map < string, Fonction_nD * , std::less <string> > listeDeFonction_nD;
170
171 // un pointeur sur LesCourbes1D ce qui permet de définir des fonctions avec des courbes 1D
172 // ici on considère que LesCourbes1D n'a qu'un membre sinon erreur:

```

```

171 // l'instance est donc un singleton, mais on n'utilise pas une variable static pour éviter les
172 // pb en multistreading
173 const LesCourbes1D * lesCourbes1D;
174
175
176 static MotCle motCle; // liste des mots clés
177
178 // METHODES PROTEGEES :
179
180 };
181 /// @} // end of group
182
183 #endif

```

## 7.540 Poly\_Lagrange.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Poly_Lagrange.h
30 // classe : Poly_Lagrange
31
32
33 /*****
34 *      DATE:      30/03/2008
35 *
36 *      AUTEUR:     G RIO      (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:     Herezh++
39 *
40 *
41 *      BUT:        Classe permettant le calcul d'un polynôme 1D
42 *                 à l'aide d'une interpolation de Lagrange
43 *                 ainsi qu'un certain nombre d'information
44 *                 supplémentaires telles que dérivées.
45 *
46 *      *****
47 *      VERIFICATION:
48 *
49 *      ! date !   auteur !           but
50 *      -----
51 *      !           !           !
52 *
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date !   auteur !           but
56 *      -----
57 *
58 *      *****/
59
60 #ifndef COURBEINTERPOLAGRANGE_1_D_H
61 #define COURBEINTERPOLAGRANGE_1_D_H
62
63 #include "Courbe1D.h"
64 #include "Tableau_T.h"
65 #include "Coordonnee2.h"
66 /// @addtogroup Les_courbes_1D
67 /// @{
68 ///
69

```

```

70 /**
71 *
72 *   BUT:   Classe permettant le calcul d'un polynôme 1D
73 *         à l'aide d'une interpolation de Lagrange
74 *         ainsi qu'un certain nombre d'information
75 *         supplémentaires telles que dérivées.
76 *
77 *
78 * \author   Gérard Rio
79 * \version  1.0
80 * \date    19/01/2001
81 * \brief   Classe permettant le calcul d'un polynôme 1D à l'aide d'une interpolation de Lagrange
82 *
83 */
84
85 class Poly_Lagrange : public CourbelD
86 {
87 public :
88
89     // CONSTRUCTEURS :
90     // par défaut
91     Poly_Lagrange(string nom = "");
92     // fonction d'un tableau de points
93     Poly_Lagrange(Tableau <Coordonnee2>& pt,string nom = "");
94
95     // de copie
96     Poly_Lagrange(const Poly_Lagrange& Co);
97     Poly_Lagrange(const CourbelD& Co);
98
99     // DESTRUCTEUR :
100    ~Poly_Lagrange();
101
102    // METHODES PUBLIQUES :
103
104    // ----- virtuelles -----
105
106    // affichage de la courbe
107    void Affiche() const;
108    // ramène true si ok, false sinon
109    bool Complet_courbe()const;
110
111    // Lecture des donnees de la classe sur fichier
112    // le nom passé en paramètre est le nom de la courbe
113    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
114    // ce nom remplace le nom actuel
115    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
116
117    // def info fichier de commande
118    void Info_commande_Courbes1D(UtilLecture & entreePrinc);
119
120    // ramène la valeur
121
122    double Valeur(double x) ;
123
124    // ramène la valeur et la dérivée en paramètre
125    CourbelD::ValDer Valeur_Et_derivee(double x) ;
126
127    // ramène la dérivée
128    double Derivee(double x) ;
129
130    // ramène la valeur et les dérivées première et seconde en paramètre
131    CourbelD::ValDer2 Valeur_Et_der12(double x);
132
133    // ramène la dérivée seconde
134    double Der_sec(double x);
135
136    // ramène la valeur si dans le domaine strictement de définition
137    // si c'est inférieur au x mini, ramène la valeur minimale possible de y
138    // si supérieur au x maxi , ramène le valeur maximale possible de y
139    CourbelD::Valbool Valeur_stricte(double x) ;
140
141    // ramène la valeur et la dérivée si dans le domaine strictement de définition
142    // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
143    // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
144    CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
145
146    // méthode pour changer le tableau de points associé
147    void Change_tabPoints(Tableau <Coordonnee2>& pt);
148
149    //----- lecture écriture de restart -----
150    // cas donne le niveau de la récupération
151    // = 1 : on récupère tout
152    // = 2 : on récupère uniquement les données variables (supposées comme telles)
153    void Lecture_base_info(ifstream& ent,const int cas);
154    // cas donne le niveau de sauvegarde
155    // = 1 : on sauvegarde tout
156    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)

```

```

157     void Ecriture_base_info(ofstream& sort,const int cas);
158     // sortie du schemaXML: en fonction de enu
159     virtual void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
160
161     // ----- méthodes particulières à la fonction poly-linéaire -----
162     // ramène le nombre de point de la polyligne
163     int NombrePoint() {return points.Taille();};
164     // ramène le x et y du point i
165     const Coordonnee2& Pt_nbi(int i) {return points(i);};
166
167 protected :
168     // VARIABLES PROTEGEES :
169     Tableau <Coordonnee2> points; // les points de la courbe
170     double der_init; // dérivée initiale
171     double der_finale; // dérivée finale
172     int indice_precedant; // position précédente
173
174
175     // METHODES PROTEGEES :
176
177 };
178 /// @} // end of group
179 #endif

```

## 7.541 SixpodeCos3phi.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : SixpodeCos3phi.h
30 // classe : SixpodeCos3phi
31
32
33 /*****
34 *      DATE:          30/03/2008
35 *
36 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:        Herezh++
39 *
40 *      ****
41 *      BUT:          Classe permettant le calcul d'une fonction 1D 1D
42 *                  de type :  $1./(1.+gamma*cos(3*phi)^2)^n$ 
43 *                  ainsi qu'un certain nombre d'information
44 *                  supplémentaires telles que dérivées.
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *      ! date ! auteur ! but
50 *      -----
51 *      ! ! !
52 *
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *
58 *****/

```

```

59
60 #ifndef COURBESIXPODECOS3PHI_1_D_H
61 #define COURBESIXPODECOS3PHI_1_D_H
62
63 #include "CourbelD.h"
64 /// @addtogroup Les_courbes_1D
65 /// @{
66 ///
67
68 /**
69 *
70 *      BUT:      Classe permettant le calcul d'une fonction 1D
71 *              de type :
72 *       $f(x) = 1./(1.+gamma*cos(3*phi)^2)^n$ 
73 *              ainsi qu'un certain nombre d'information
74 *              supplémentaires telles que dérivées.
75 *
76 *
77 * \author      Gérard Rio
78 * \version     1.0
79 * \date        19/01/2001
80 * \brief       Classe permettant le calcul d'une fonction 1D de type :  $f(x) =$ 
81 *               $1./(1.+gamma*cos(3*phi)^2)^n$ 
82 */
83
84 class SixpodeCos3phi : public CourbelD
85 {
86     public :
87
88         // CONSTRUCTEURS :
89         SixpodeCos3phi(string nom = "");
90
91         // de copie
92         SixpodeCos3phi(const SixpodeCos3phi& Co);
93         SixpodeCos3phi(const CourbelD& Co);
94
95         // DESTRUCTEUR :
96         ~SixpodeCos3phi();
97
98         // METHODES PUBLIQUES :
99
100        // ----- virtuelles -----
101
102        // affichage de la courbe
103        void Affiche() const ;
104        // ramène true si ok, false sinon
105        bool Complet_courbe()const;
106
107        // Lecture des donnees de la classe sur fichier
108        // le nom passé en paramètre est le nom de la courbe
109        // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
110        // ce nom remplace le nom actuel
111        void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
112
113        // def info fichier de commande
114        void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
115
116        // ramène la valeur
117
118        double Valeur(double x) ;
119
120        // ramène la valeur et la dérivée en paramètre
121        CourbelD::ValDer Valeur_Et_derivee(double x) ;
122
123        // ramène la dérivée
124        double Derivee(double x) ;
125
126        // ramène la valeur et les dérivées première et seconde en paramètre
127        CourbelD::ValDer2 Valeur_Et_der12(double x);
128
129        // ramène la dérivée seconde
130        double Der_sec(double x);
131
132        // ramène la valeur si dans le domaine strictement de définition
133        // si c'est inférieur au x mini, ramène la valeur minimale possible de y
134        // si supérieur au x maxi , ramène le valeur maximale possible de y
135        CourbelD::Valbool Valeur_stricte(double x) ;
136
137        // ramène la valeur et la dérivée si dans le domaine strictement de définition
138        // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
139        // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
140        CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
141
142        //----- lecture écriture de restart -----
143        // cas donne le niveau de la récupération
144        // = 1 : on récupère tout

```

```

145 // = 2 : on récupère uniquement les données variables (supposées comme telles)
146 void Lecture_base_info(ifstream& ent,const int cas);
147 // cas donne le niveau de sauvegarde
148 // = 1 : on sauvegarde tout
149 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
150 void Ecriture_base_info(ofstream& sort,const int cas);
151 // sortie du schemaXML: en fonction de enu
152 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
153
154
155
156
157 protected :
158 // VARIABLES PROTEGEES :
159 double xn,gamma; // les coefficients de la loi
160
161 // METHODES PROTEGEES :
162
163 };
164 /// @} // end of group
165 #endif

```

## 7.542 TangenteHyperbolique.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : TangenteHyperbolique.h
30 // classe : TangenteHyperbolique
31
32
33 /*****
34 *      DATE:      17/09/2009
35 *
36 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *
41 *      BUT:      Classe permettant le calcul d'une fonction 1D 1D
42 *               de type : a+b*tanh((x-c)/d)
43 *               ainsi qu'un certain nombre d'information
44 *               supplémentaires telles que dérivées.
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *      ! date ! auteur ! but
50 *      -----
51 *      ! ! !
52 *
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *
58 *      *****/
59
60 #ifndef COURBETANGENTEHYPERBOLIQUE_1_D_H

```

```

61 #define COURBETANGENTEHYPERBOLIQUE_1_D_H
62
63 #include "CourbelD.h"
64
65 /// @addtogroup Les_courbes_1D
66 /// @{
67 ///
68
69 /**
70 *
71 *   BUT:   Classe permettant le calcul d'une fonction 1D
72 *         de type :
73 *    $f(x) = a+b*\tanh((x-c)/d)$ 
74 *         ainsi qu'un certain nombre d'information
75 *         supplémentaires telles que dérivées.
76 *
77 *
78 * \author   Gérard Rio
79 * \version  1.0
80 * \date     19/01/2001
81 * \brief    Classe permettant le calcul d'une fonction 1D de type :  $f(x) = a+b*\tanh((x-c)/d)$ 
82 *
83 */
84 class TangenteHyperbolique : public CourbelD
85 {
86     public :
87
88     // CONSTRUCTEURS :
89     TangenteHyperbolique(string nom = "");
90
91     // de copie
92     TangenteHyperbolique(const TangenteHyperbolique& Co);
93     TangenteHyperbolique(const CourbelD& Co);
94
95     // DESTRUCTEUR :
96     ~TangenteHyperbolique();
97
98     // METHODES PUBLIQUES :
99
100    // ----- virtuelles -----
101
102    // affichage de la courbe
103    void Affiche() const ;
104    // ramène true si ok, false sinon
105    bool Complet_courbe()const;
106
107    // Lecture des donnees de la classe sur fichier
108    // le nom passé en paramètre est le nom de la courbe
109    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
110    // ce nom remplace le nom actuel
111    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
112
113    // def info fichier de commande
114    void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
115
116    // ramène la valeur
117
118    double Valeur(double x) ;
119
120    // ramène la valeur et la dérivée en paramètre
121    CourbelD::ValDer Valeur_Et_derivee(double x) ;
122
123    // ramène la dérivée
124    double Derivee(double x) ;
125
126    // ramène la valeur et les dérivées première et seconde en paramètre
127    CourbelD::ValDer2 Valeur_Et_der12(double x);
128
129    // ramène la dérivée seconde
130    double Der_sec(double x);
131
132    // si c'est inférieur au x mini, ramène la valeur minimale possible de y
133    // si supérieur au x maxi , ramène la valeur maximale possible de y
134    CourbelD::Valbool Valeur_stricte(double x) ;
135
136    // ramène la valeur et la dérivée si dans le domaine strictement de définition
137    // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
138    // si supérieur au x maxi , ramène la valeur maximale possible de y et Y' correspondant
139    CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
140
141    //----- lecture écriture de restart -----
142    // cas donne le niveau de la récupération
143    // = 1 : on récupère tout
144    // = 2 : on récupère uniquement les données variables (supposées comme telles)
145    void Lecture_base_info(ifstream& ent,const int cas);
146    // cas donne le niveau de sauvegarde
147    // = 1 : on sauvegarde tout

```



```

148 // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
149 void Ecriture_base_info(ofstream& sort,const int cas);
150 // sortie du schemaXML: en fonction de enu
151 void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;
152
153
154
155
156 protected :
157 // VARIABLES PROTEGEES :
158 double ax,bx,cx,dx; // les coefficients de la fonction
159
160 // METHODES PROTEGEES :
161
162 };
163 /// @} // end of group
164 #endif

```

## 7.543 TripodeCos3phi.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : TripodeCos3phi.h
30 // classe : TripodeCos3phi
31
32
33 /*****
34 *      DATE:          30/03/2008
35 *
36 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:        Herezh++
39 *
40 *
41 *      BUT:           Classe permettant le calcul d'une fonction 1D
42 *                   de type :  $1./(1.+gamma*cos(3*phi))^n$ 
43 *                   ainsi qu'un certain nombre d'information
44 *                   supplémentaires telles que dérivées.
45 *
46 *      *****
47 *
48 *      VERIFICATION:
49 *      ! date ! auteur ! but
50 *      -----
51 *      ! ! !
52 *
53 *      *****
54 *      MODIFICATIONS:
55 *      ! date ! auteur ! but
56 *      -----
57 *
58 *      *****/
59
60 #ifndef COURBETRIPODECOS3PHI_1_D_H
61 #define COURBETRIPODECOS3PHI_1_D_H
62
63 #include "Courbe1D.h"
64 /// @addtogroup Les_courbes_1D

```

```

65 /// @{
66 ///
67
68 /**
69 *
70 *     BUT:     Classe permettant le calcul d'une fonction 1D
71 *             de type :
72 *     f(x) = 1./(1.+gamma*cos(3*phi))^n
73 *             ainsi qu'un certain nombre d'information
74 *             supplémentaires telles que dérivées.
75 *
76 *
77 * \author     Gérard Rio
78 * \version    1.0
79 * \date       30/03/2008
80 * \brief      Classe permettant le calcul d'une fonction 1D de type : f(x) = 1./(1.+gamma*cos(3*phi))^n
81 *
82 */
83
84 class TripodeCos3phi : public CourbelD
85 {
86 public :
87
88     // CONSTRUCTEURS :
89     TripodeCos3phi(string nom = "");
90
91     // de copie
92     TripodeCos3phi(const TripodeCos3phi& Co);
93     TripodeCos3phi(const CourbelD& Co);
94
95     // DESTRUCTEUR :
96     ~TripodeCos3phi();
97
98     // METHODES PUBLIQUES :
99
100    // ----- virtuelles -----
101
102    // affichage de la courbe
103    void Affiche() const ;
104    // ramène true si ok, false sinon
105    bool Complet_courbe()const;
106
107    // Lecture des donnees de la classe sur fichier
108    // le nom passé en paramètre est le nom de la courbe
109    // s'il est vide c-a-d = "", la methode commence par lire le nom sinon
110    // ce nom remplace le nom actuel
111    void LectDonnParticulieres_courbes(const string& nom, UtilLecture * );
112
113    // def info fichier de commande
114    void Info_commande_Courbes1D(UtilLecture & entreePrinc) ;
115
116    // ramène la valeur
117
118    double Valeur(double x) ;
119
120    // ramène la valeur et la dérivée en paramètre
121    CourbelD::ValDer Valeur_Et_derivee(double x) ;
122
123    // ramène la dérivée
124    double Derivee(double x) ;
125
126    // ramène la valeur et les dérivées première et seconde en paramètre
127    CourbelD::ValDer2 Valeur_Et_der12(double x);
128
129    // ramène la dérivée seconde
130    double Der_sec(double x);
131
132    // si c'est inférieur au x mini, ramène la valeur minimale possible de y
133    // si supérieur au x maxi , ramène le valeur maximale possible de y
134    CourbelD::Valbool Valeur_stricte(double x) ;
135
136    // ramène la valeur et la dérivée si dans le domaine strictement de définition
137    // si c'est inférieur au x mini, ramène la valeur minimale possible de y et Y' correspondant
138    // si supérieur au x maxi , ramène le valeur maximale possible de y et Y' correspondant
139    CourbelD::ValDerbool Valeur_Et_derivee_stricte(double x) ;
140
141    //----- lecture écriture de restart -----
142    // cas donne le niveau de la récupération
143    // = 1 : on récupère tout
144    // = 2 : on récupère uniquement les données variables (supposées comme telles)
145    void Lecture_base_info(ifstream& ent,const int cas);
146    // cas donne le niveau de sauvegarde
147    // = 1 : on sauvegarde tout
148    // = 2 : on sauvegarde uniquement les données variables (supposées comme telles)
149    void Ecriture_base_info(ofstream& sort,const int cas);
150    // sortie du schemaXML: en fonction de enu
151    void SchemaXML_Courbes1D(ofstream& sort,const Enum_IO_XML enu) ;

```

```

152
153
154
155
156 protected :
157     // VARIABLES PROTEGEES :
158     double xn,gamma; // les coefficients de la loi
159     bool val_absolu; // indique si on utilise x ou |x|
160
161     // METHODES PROTEGEES :
162
163 };
164 ///< @} // end of group
165 #endif

```

## 7.544 Racine.h

```

1 // méthodes issues de quartic.c
2 // on a laissé les informations des programmes originaux
3 // l'ensemble est organisée sous forme d'une classe pour encapsuler
4
5 //*****
6 //     l'écriture initiale a été modifiée -> C++
7 //
8 //     calcul :
9 //         - racine d'un polynome du second degré
10 //         - racine d'un polynome du 3ieme degré
11 //         - racine d'un polynome du 4ieme degré avec différentes méthodes
12 //             . Descartes' algorithm
13 //             . Ferrari's algorithm
14 //             . Neumark's algorithm
15 //             . Yacoub & Fraidenraich's algorithm
16 //             . Christianson's algorithm
17 //*****
18
19
20
21 #ifndef QUARTIC_H
22 #define QUARTIC_H
23
24 #include <stdio.h>
25
26 class Quartic
27 { public:
28
29
30
31 /*   quartic.c
32     version 54
33
34     a test of relative accuracy of various quartic routines.
35
36     use:
37     quartic [-a] [-c n] [-q n] [-d n]
38     (no parameters) : a loop through 10,000 prechosen quartic coefficients
39     -a               : improve cubic roots (default: no iteration)
40     -c n             : keep reading n cubic roots from standard input
41                     (if n=0 read coefficients), terminate with 'q'.
42     -d n             : set debug value to 'n' (default: 5)
43     -q n             : keep reading n quartic roots from standard input
44                     (if n=0 read coefficients), terminate with 'q'.
45
46     debug <= 1 : print cases used
47
48     method-
49     http://linus.socs.uts.edu.au/~don/pubs/solving.html
50
51     4 Apr 2004 bringing yacfraid notation into line with solving.html
52     4 Apr 2004 fixed bug in final root calculation in yacfraid
53     8 Mar 2004 corrected modified yacfraid algorithm
54     8 Mar 2004 modified yacfraid algorithm
55     31 Jan 2004 printing results when number of roots vary
56     30 Jan 2004 printing error of chosen cubic if debug<1
57     27 Jan 2004 seeking worst coefficients for each algorithm
58     27 Jan 2004 choosing best route for k in chris
59     26 Jan 2004 conforming variables to solving.html
60     26 Jan 2004 fixed bug in yacfraid for multiplicity 3
61     25 Jan 2004 speeding up chris
62     24 Jan 2004 fixed chris error (A <-> B)
63     23 Jan 2004 use debug<1 for diagnostics
64     21 Jan 2004 solve cubic in neumark, yacfraid and chris if d=0
65     20 Jan 2004 3 roots to cubic in chris if e1=0
66     19 Jan 2004 debugging Christianson routine
67     14 Jan 2004 adding Christianson, cut determinant in quadratic call
68     5 Jan 2004 improving cubic roots by Newton-Raphson iteration

```

```

69     23 Dec 2003 seeking snag in yacfraid
70     21 Dec 2003 putting diagnostic printout in yacfraid
71     18 Dec 2003 allowing input of equation coefficients
72     18 Dec 2003 recording cubic root giving least worst error
73     17 Dec 2003 recording cubic root giving the most quartic roots
74     16 Dec 2003 using the cubic root giving the most quartic roots
75     16 Dec 2003 trying consistency of 3 cubic roots where available
76     15 Dec 2003 trying all 3 cubic roots where available
77     13 Dec 2003 trying to fix Neumark
78     13 Dec 2003 cleaning up diagnostic format
79     12 Dec 2003 initialising n,m,po3 in cubic
80     12 Dec 2003 allow cubic to return 3 zero roots if p=q=r=0
81     10 Dec 2003 added setargs
82         2 Dec 2003 finding worst cases
83         2 Dec 2003 negating j if p>0 in cubic
84         1 Dec 2003 changing v in cubic from (sinsqk > doub0) to (sinsqk >= doub0)
85         1 Dec 2003 test changing v in cubic from po3sq+po3sq to doub2*po3sq
86     30 Nov 2003 counting cases in all solving routines
87     29 Nov 2003 testing wsq >= doub0
88     29 Nov 2003 mult by doub2 for v in cubic
89     29 Nov 2003 cut po3cu from cubic
90     29 Nov 2003 better quadratic
91     29 Nov 2003 count agreements
92     17 Nov 2003 count special cases
93     16 Nov 2003 option of loop or read coefficients from input
94     15 Nov 2003 fixed cubic() bug
95     11 Nov 2003 added Brown and Yacoub & Fraidenraich's algorithms
96     21 Jan 1989 quartic selecting Descartes, Ferrari, or Neumark algorithms
97     16 Jul 1981 Don Herbison-Evans
98
99 "Solving Quartics and Cubics for Graphics", D. Herbison-Evans,
100 Graphics Gems V (ed.: A. Paeth) Academic Press (Chesnut Hill), pp. 3-15 (1995).
101
102 "Solving Quartics and Cubics for Graphics", D. Herbison-Evans,
103 Research Report CS-86-56, Department of Computer Science, University of Waterloo (1986)
104
105 "Caterpillars and the Inaccurate Solution of Cubic and Quartic Equations",
106 D. Herbison-Evans, Australian Computer Science Communications,
107 Vol. 5, No. 1, pp. 80-90 (1983)
108
109     subroutines:
110         errors      - find errors in a set of roots
111         acos3       - find arcos(x/3)
112         curoot      - find cube root
113         quadratic   - solve a quadratic
114         cubic        - solve a cubic
115         cubnewton   - improve cubic roots by iteration
116         quartic     - solve a quartic
117         descartes   - use Descartes' algorithm to solve a quartic
118         ferrari     - use Ferrari's algorithm to solve a quartic
119         neumark     - use Neumark's algorithm to solve a quartic
120         yacfraid    - use Yacoub & Fraidenraich's algorithm to solve a quartic
121         chris       - use Christianson's algorithm to solve a quartic
122 */
123
124
125 /*****
126
127 // constructeur permettant d'initialiser les variables
128 Quartic();
129
130 // modification du paramètre de debug et d'itération pour cubic
131 void Change(int debuge, bool iteratee) {debug=debuge;iterate=iteratee;};
132
133 /*****
134 //
135 //   find the errors
136 //
137 //   called by quarticest, docoeff, compare,
138 //   chris, descartes, ferrari, neumark, yacfraid.
139 /*****
140
141 double errors(double a,double b,double c,double d,double rts[4],double rterr[4],int nrts);
142 //double a,b,c,d,rts[4],rterr[4];
143 //int nrts;
144
145 /*****
146 //   find cos(acos(x)/3)
147 //
148 //   16 Jul 1981   Don Herbison-Evans
149 //
150 //   called by cubic .
151 /*****
152
153 double acos3(double x);
154 //   double x ;
155

```

```

156 /*****/
157 //   find cube root of x.
158 //
159 //   30 Jan 1989   Don Herbison-Evans
160 //
161 //   called by cubic .
162 /*****/
163
164 double curoot(double x);
165 //   double x ;
166
167 /*****/
168 //   solve the quadratic equation -
169 //
170 //       x**2 + b*x + c = 0
171 //
172 //   14 Jan 2004   cut determinant in quadratic call
173 //   29 Nov 2003   improved
174 //   16 Jul 1981   Don Herbison-Evans
175 //
176 //   called by   cubic,quartic,chrisc,descartes,ferrari,neumark.
177 /*****/
178
179 int quadratic(double b,double c,double rts[4]);
180 //   double b,c,rts[4];
181
182
183 /*****/
184 //   find the real roots of the cubic -
185 //       x**3 + p*x**2 + q*x + r = 0
186 //
187 //   12 Dec 2003   initialising n,m,po3
188 //   12 Dec 2003   allow return of 3 zero roots if p=q=r=0
189 //   2 Dec 2003   negating j if p>0
190 //   1 Dec 2003   changing v from (sinsq > doub0) to (sinsq >= doub0)
191 //   1 Dec 2003   test changing v from po3sq+po3sq to doub2*po3sq
192 //   16 Jul 1981   Don Herbison-Evans
193 //
194 //   input parameters -
195 //   p,q,r - coeffs of cubic equation.
196 //   iterate : booléen indiquant si l'on veut améliorer les racines via du newton (4 fois)
197 //   debug : si <1 impression d'un paquet de paramètre ??
198 //
199 //   output-
200 //   the number of real roots
201 //   v3 - the roots.
202 //
203 //   global constants -
204 //   rt3 - sqrt(3)
205 //   inv3 - 1/3
206 //   doubmax - square root of largest number held by machine
207 //
208 //   method -
209 //   see D.E. Littlewood, "A University Algebra" pp.173 - 6
210 //
211 //   15 Nov 2003   output 3 real roots: Don Herbison-Evans
212 //   Apr 1981     initial version: Charles Prineas
213 //
214 //   called by   cubictest,quartic,chrisc,yacfraid,neumark,descartes,ferrari.
215 //   calls      quadratic,acos3,curoot,cubnewton.
216 /*****/
217
218 int cubic(double p,double q,double r, double v3[4] );
219
220
221
222 /*****/
223 //   improve roots of cubic by Newton-Raphson iteration
224 //
225 //   5 Jan 2004   Don Herbison-Evans
226 //
227 //   called by cubic.
228 /*****/
229
230 void cubnewton(double p,double q,double r,int n3,double v3[4]);
231 //int n3;
232 //double p,q,r,v3[4];
233 //
234
235 /*****/
236 //   Solve quartic equation using either
237 //   quadratic, Ferrari's or Neumark's algorithm.
238 //
239 //   called by
240 //   calls   descartes, ferrari, neumark, yacfraid.
241 //
242 //   15 Dec 2003   added yacfraid

```

```

243 //      10 Dec 2003  added descartes with neg coeffs
244 //      21 Jan 1989  Don Herbison-Evans
245 //*****
246 int quartic(double a,double b,double c,double d,double rts[4]);
247 //double a,b,c,d,rts[4];
248
249 //*****/
250 //      Solve quartic equation using
251 //      Descartes-Euler-Cardano algorithm
252 //
253 //      called by quartic, compare, quartictest.
254 //
255 //      Strong, T. "Elementary and Higher Algebra"
256 //      Pratt and Oakley, p. 469 (1859)
257 //
258 //      16 Jul 1981  Don Herbison-Evans
259 //*****/
260 int descartes(double a,double b,double c,double d,double rts[4]);
261 //double a,b,c,d,rts[4];
262
263
264 //*****/
265 //      solve the quartic equation -
266 //
267 //       $x^{**4} + a*x^{**3} + b*x^{**2} + c*x + d = 0$ 
268 //
269 //      calls      cubic, quadratic.
270 //
271 //      input -
272 //      a,b,c,e - coeffs of equation.
273 //
274 //      output -
275 //      n4 - number of real roots.
276 //      rts - array of root values.
277 //
278 //      method :  Ferrari - Lagrange
279 //      Theory of Equations, H.W. Turnbull p. 140 (1947)
280 //
281 //      16 Jul 1981 Don Herbison-Evans
282 //
283 //      calls cubic, quadratic
284 //*****/
285 int ferrari(double a,double b,double c,double d,double rts[4]);
286 //      double a,b,c,d,rts[4];
287
288
289
290 //*****/
291 //      solve the quartic equation -
292 //
293 //       $x^{**4} + a*x^{**3} + b*x^{**2} + c*x + d = 0$ 
294 //
295 //      calls      cubic, quadratic.
296 //
297 //      input parameters -
298 //      a,b,c,e - coeffs of equation.
299 //
300 //      output parameters -
301 //      n4 - number of real roots.
302 //      rts - array of root values.
303 //
304 //      method -  S. Neumark
305 //      "Solution of Cubic and Quartic Equations" - Pergamon 1965
306 //
307 //      1 Dec 1985  translated to C with help of Shawn Neely
308 //      16 Jul 1981  Don Herbison-Evans
309 //*****/
310 int neumark(double a,double b,double c,double d,double rts[4]);
311 //      double a,b,c,d,rts[4];
312
313
314
315 //*****/
316 //      solve the quartic equation -
317 //       $x^{**4} + a*x^{**3} + b*x^{**2} + c*x + d = 0$ 
318 //
319 //      calls      cubic, quadratic.
320 //
321 //      input parameters -
322 //      a,b,c,e - coeffs of equation.
323 //
324 //      output parameters -
325 //      n4 - number of real roots.
326 //      rts - array of root values.
327 //
328 //      method -
329 //      K.S. Brown

```

```

330 //      Reducing Quartics to Cubics,
331 //      http://www.seanet.com/~ksbrown/kmath296.htm (1967)
332 //
333 //      Michael Daoud Yacoub & Gustavo Fraidenraich
334 //      "A new simple solution of the general quartic equation"
335 //      Revised 16 Feb 2004
336 //
337 //      14 Nov 2003 Don Herbison-Evans
338 //*****
339 int yacfraid(double a,double b,double c,double d,double rts[4]);
340 // double a,b,c,d,rts[4];
341
342
343 //*****
344 //      Solve quartic equation using
345 //      Christianson's algorithm
346 //      calls errors, quadratic, cubic.
347 //
348 //      Bruce Christianson, Solving Quartics Using Palindromes,
349 //      Mathematical Gazette, Vol. 75, pp. 327-328 (1991)
350 //
351 //      14 Jan 2004 Don Herbison-Evans
352
353 int chris(double a,double b,double c,double d,double rts[4]);
354 //double a,b,c,d,rts[4];
355
356 // données propres
357 protected:
358 #define NCASES 40
359
360 double d3o8,d3o256;      /* double precision constants */
361 double doub0;
362 double doub1,doub2;
363 double doub3,doub4;
364 double doub5,doub6;
365 double doub8,doub9,doub12;
366 double doub16,doub24;
367 double doub27,doub64;
368 double doubmax;          /* approx square root of max double number */
369 double doubmin;          /* smallest double number */
370 double doubtol;          /* tolerance of double numbers */
371 double inv2,inv3,inv4;
372 double inv8,inv16,inv32,inv64,inv128;
373 double qrts[4][3];       /* quartic roots for each cubic root */
374 double rt3;              /* square root of 3 */
375 double rterc[4],rterd[4],rterf[4],rtern[4],rterq[4],rtery[4]; /* errors of roots */
376 double worst3[3];        /* worst error for a given cubic root */
377 int debug;               /* <1 for lots of diagnostics, >5 for none */
378 bool iterate; // indique si l'on veut itérer on pas
379 int j3;
380 int n1,n2,n3,n4[3];
381 int nqud[NCASES];
382 int ncub[NCASES];
383 int nchr[NCASES];
384 int ndes[NCASES];
385 int nfer[NCASES];
386 int nneu[NCASES];
387 int nyac[NCASES];
388 };
389
390
391 #endif

```

## 7.545 Référence du fichier /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Handler\_exception.h

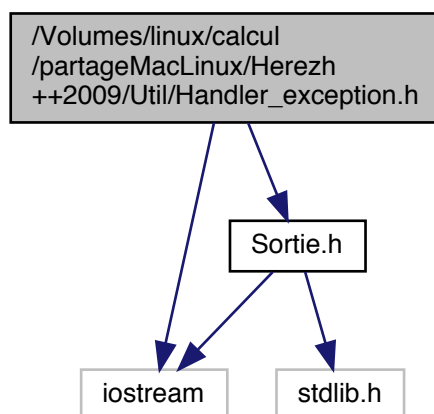
Gestion d'une liste d'Handlers d'exceptions système.

```

#include <iostream>
#include "Sortie.h"

```

Grappe des dépendances par inclusion de Handler\_exception.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

— void **Handler\_signal** (int theSignal)

*méthode utilitaire, utilisable par les classes dérivées, permettant d'agir si un controle-c est émis intervient sur des paramètres stockés dans [ParaAlgoControle](#), qui sont actuellement actif (accessible via [ParaGlob](#)) c'est donc via ces paramètres que les autres programmes peuvent récupérer les infos et agir en conséquence éventuellement*

### 7.545.1 Description détaillée

Gestion d'une liste d'Handlers d'exceptions système.

### 7.546 Handler\_exception.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Handler_exception.h
2   \brief Gestion d'une liste d'Handlers d'exceptions système
3 */
4 // This file is part of the Herezh++ application.
5 //
6 // The finite element software Herezh++ is dedicated to the field
7 // of mechanics for large transformations of solid structures.
8 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
9 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
10 //
11 // Herezh++ is distributed under GPL 3 license ou ultérieure.
12 //
13 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
14 // AUTHOR : Gérard Rio
15 // E-MAIL : gerardrio56@free.fr
16 //
17 // This program is free software: you can redistribute it and/or modify
18 // it under the terms of the GNU General Public License as published by
19 // the Free Software Foundation, either version 3 of the License,
20 // or (at your option) any later version.
  
```



```

21 //
22 // This program is distributed in the hope that it will be useful,
23 // but WITHOUT ANY WARRANTY; without even the implied warranty
24 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
25 // See the GNU General Public License for more details.
26 //
27 // You should have received a copy of the GNU General Public License
28 // along with this program. If not, see <https://www.gnu.org/licenses/>.
29 //
30 // For more information, please consult: <https://herezh.irdl.fr/>.
31
32 /*****
33 *   DATE:      01/03/2007
34 *
35 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
36 *
37 *   PROJET:    Herezh++
38 *
39 *   BUT:       Gestion d'une liste d'Handlers d'exceptions système
40 *
41 *   *****
42 *
43 *   VERIFICATION:
44 *
45 *   ! date !   auteur !           but
46 *   -----
47 *   !       !           !
48 *
49 *   *****
50 *   MODIFICATIONS:
51 *   ! date !   auteur !           but
52 *   -----
53 *
54 *   *****/
55 #ifndef HANDLER_EXCEPTION_SYSTEME_H
56 #define HANDLER_EXCEPTION_SYSTEME_H
57
58 #include <iostream>
59 #include "Sortie.h"
60
61
62     /// méthode utilitaire, utilisable par les classes dérivées, permettant d'agir si un controle-c est
63     /// émis
64     /// intervient sur des paramètres stockés dans ParaAlgoControle, qui sont actuellement actif
65     /// (accessible via ParaGlob)
66     /// c'est donc via ces paramètres que les autres programmes peuvent récupérer les infos et agir
67     /// en conséquence éventuellement
68     void Handler_signal(int theSignal);
69 #endif

```

## 7.547 Référence du fichier /Volumes/linux/calcul/partageMacLinux/↵ Herezh++2009/Util/MathUtil.h

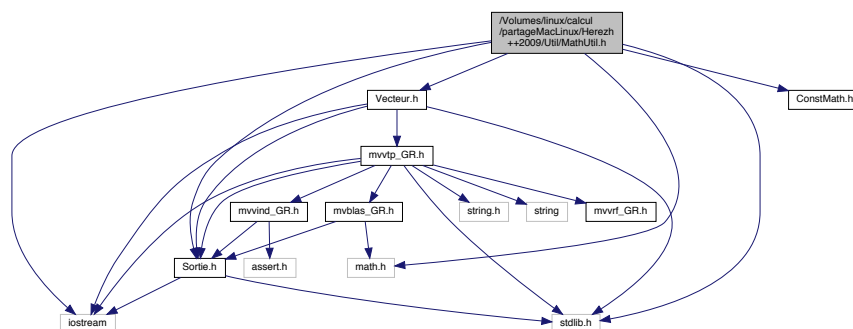
Utilitaires divers de calculs élémentaires.

```

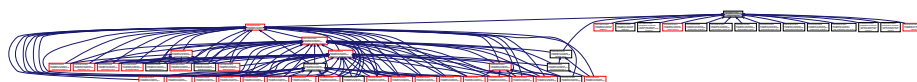
#include <iostream>
#include <math.h>
#include <stdlib.h>
#include "Sortie.h"
#include "Vecteur.h"
#include "ConstMath.h"
#include "MathUtil.cc"

```

Graphe des dépendances par inclusion de MathUtil.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Fonctions

- double **MIN** (double a, double b)  
*minimum de deux doubles*
- double **MaX** (double a, double b)  
*maximum de deux doubles*
- int **MiN** (int a, int b)  
*minimum de deux entiers*
- int **MaX** (int a, int b)  
*maximum de deux entiers*
- double **DabsMiN** (double a, double b)  
*minimum des valeurs absolues de deux doubles*
- double **DabsMaX** (double a, double b)  
*maximum des valeurs absolues de deux doubles*
- double **DabsMaX** (double a, double b, double c)  
*maximum des valeurs absolues de 3 doubles*
- int **DabsMiN** (int a, int b)  
*minimum des valeurs absolues de deux entiers*
- int **DabsMaX** (int a, int b)  
*maximum des valeurs absolues de deux entiers*
- double **DabsMaxiTab** (double \*tab, int n)  
*maximum des valeurs absolu d'un tableau de n réels*
- int **DabsMaxiTab** (int \*tab, int n)  
*maximum des valeurs absolu d'un tableau de n entier relatifs*
- double **Dabs** (double a)  
*valeur absolu d'un reel*
- double **Abs** (double a)
- int **Ab** (int a)
- int **Signe** (double a)  
*ramène 1 ou -1 suivant que a est positif ou non*
- double **DSigne** (double a)  
*idem mais ramène en double 1. ou -1. suivant que a est positif ou non*
- double **Signe** (double a, double b)  
*ramène b ou -b suivant que a est positif ou non*
- bool **diffpetit** (double x, double y)

- calcul la difference entre deux nombres pondéré par la valeur du max des deux nombres + 1 c-a-d ramene vrai si  $|x-y| > \epsilon * (1 + \max(|x|, |y|))$ , faux sinon,  $\epsilon = \text{ConstMath::pasmalpetit}$*
- bool **difftrespetit** (double x, double y)
    - idem mais avec un epsilon plus petit = ConstMath::trespetit*
  - bool **EgaleApeuPres** (double x, double y)
    - un autre nom plus explicite*
  - bool **diffpetit** (double x, double y, double z)
    - calcul la difference entre deux nombres (les deux premiers paramètres) pondéré par la valeur du dernier paramètre c-a-d ramene vrai si  $|x-y| > \epsilon * (|z|)$ , faux sinon,  $\epsilon = \text{ConstMath::pasmalpetit}$*
  - bool **difftrespetit** (double x, double y, double z)
    - idem mais avec un epsilon plus petit = ConstMath::trespetit*
  - bool **diffpourcent** (double x, double y, double z, const double epsilo)
    - idem mais avec un epsilon que l'on choisit en paramètre*
  - template<class T1 >
    - T1 **Sqr** (T1 a)
      - eleve au carre*
  - template<class T1 >
    - T1 **PUISSN** (T1 a, const int n)
      - eleve à la puissance n*

### 7.547.1 Description détaillée

Utilitaires divers de calculs élémentaires.

## 7.548 MathUtil.h

[Aller à la documentation de ce fichier.](#)

```

1  /*! \file MathUtil.h
2  \brief Utilitaires divers de calculs élémentaires
3  */
4
5  // This file is part of the Herezh++ application.
6  //
7  // The finite element software Herezh++ is dedicated to the field
8  // of mechanics for large transformations of solid structures.
9  // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPOUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *   DATE:           23/01/97
35 *
36 *   AUTEUR:         G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:         Herezh++
39 *
40 *****/
41 *   BUT:   Utilitaires divers mathematiques
42 *
43 *   *****
44 *
45 *   VERIFICATION:
46 *   ! date !   auteur !           but           !

```

```

47 * ----- *
48 * ! ! ! ! *
49 * $ *
50 * ***** *
51 * MODIFICATIONS: *
52 * ! date ! auteur ! but ! *
53 * ----- *
54 * $ *
55 * *****/
56 #ifndef MATHUTIL_H
57 #define MATHUTIL_H
58
59 #include <iostream>
60 #include <math.h>
61 #include <stdlib.h>
62 #include "Sortie.h"
63 #include "Vecteur.h"
64 #include "ConstMath.h"
65
66
67 /// minimum de deux doubles
68 double MiN(double a,double b);
69 /// maximum de deux doubles
70 double MaX(double a,double b);
71 /// minimum de deux entiers
72 int MiN(int a,int b);
73 /// maximum de deux entiers
74 int MaX(int a,int b);
75
76 /// minimum des valeurs absolues de deux doubles
77 double DabsMiN(double a,double b);
78 /// maximum des valeurs absolues de deux doubles
79 double DabsMaX(double a,double b);
80 /// maximum des valeurs absolues de 3 doubles
81 double DabsMaX(double a,double b,double c);
82 /// minimum des valeurs absolues de deux entiers
83 int DabsMiN(int a,int b);
84 /// maximum des valeurs absolues de deux entiers
85 int DabsMaX(int a,int b);
86
87 /// maximum des valeurs absolu d'un tableau de n réels
88 double DabsMaxiTab(double * tab, int n) ;
89 /// maximum des valeurs absolu d'un tableau de n entier relatifs
90 int DabsMaxiTab(int * tab, int n) ;
91 /// valeur absolu d'un reel
92 inline double Dabs(double a) { return( a>=0. ? a : -a); };
93 inline double Abs(double a) { return( a>=0. ? a : -a); };
94 inline int Abs(int a) { return( a>=0. ? a : -a); };
95 /// ramène 1 ou -1 suivant que a est positif ou non
96 inline int Signe(double a) { return( a>=0. ? 1 : -1); };
97 /// idem mais ramène en double 1. ou -1. suivant que a est positif ou non
98 inline double DSigne(double a) { return( a>=0. ? 1. : -1.); };
99 /// ramène b ou -b suivant que a est positif ou non
100 inline double Signe(double a,double b) { return( a>=0. ? b : -b); };
101
102
103 /// calcul la difference entre deux nombres pondéré par la valeur du max des deux nombres + 1
104 /// c-a-d ramene vrai si |x-y| > epsilon*(1+max(|x|,|y|)) , faux sinon, epsimon = ConstMath::pasmalpetit
105 inline bool diffpetit(double x,double y)
106 { return((Dabs(x-y) <= ConstMath::petit * (1. + MaX(Dabs(x),Dabs(y))) ) ? false : true); };
107 /// idem mais avec un epsilon plus petit = ConstMath::trespetit
108 inline bool difftrespetit(double x,double y)
109 { return((Dabs(x-y) <= ConstMath::pasmalpetit * (1. + MaX(Dabs(x),Dabs(y))) ) ? false : true); };
110 /// un autre nom plus explicite
111 inline bool EgaleApeuPres(double x,double y)
112 { return((Dabs(x-y) <= ConstMath::pasmalpetit * (1. + MaX(Dabs(x),Dabs(y))) ) ? true : false); };
113 /// calcul la difference entre deux nombres (les deux premiers paramètres) pondéré par la valeur
114 /// du dernier paramètre z
115 /// c-a-d ramene vrai si |x-y| > epsilon*(|z|) , faux sinon, epsimon = ConstMath::pasmalpetit
116 inline bool diffpetit(double x,double y,double z)
117 { return((Dabs(x-y) <= ConstMath::petit * (Dabs(z))) ? false : true); };
118 /// idem mais avec un epsilon plus petit = ConstMath::trespetit
119 inline bool difftrespetit(double x,double y,double z)
120 { return((Dabs(x-y) <= ConstMath::pasmalpetit * ( Dabs(z))) ? false : true); };
121 /// idem mais avec un epsilon que l'on choisit en paramètre
122 inline bool diffpourcent(double x,double y,double z,const double epsilo)
123 { return((Dabs(x-y) <= epsilo * ( Dabs(z))) ? false : true); };
124 /// eleve au carre
125 template <class T1>
126 inline T1 Sqr(T1 a)
127 { return (a * a);};
128 /// eleve à la puissance n
129 template <class T1>
130 inline T1 PUISSN(T1 a, const int n)
131 {
132 #ifdef MISE_AU_POINT
133 if ( n < 1)

```

```

134         { cout << "\n erreur, exposant négatif dans la fonction PUISSN(T1 a, const int n) ";
135           Sortie(1);
136         }
137     #endif
138     int m=n;
139     return ((m > 1) ? (a * PUISSN(a,--m)) : a);
140 };
141
142
143 #ifndef MISE_AU_POINT
144 #include "MathUtil.cc"
145 #define MATHUTIL_H_deja_inclus
146 #endif
147
148
149 #endif

```

## 7.549 MathUtil2.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      23/01/97
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *   *****
37 *   BUT:   Utilitaires divers mathematiques
38 *
39 *   *****
40 *
41 *   VERIFICATION:
42 *
43 *   ! date !   auteur !           but
44 *   !-----!-----!-----!-----!
45 *   !           !           !           !
46 *   *****
47 *   MODIFICATIONS:
48 *
49 *   ! date !   auteur !           but
50 *   !-----!-----!-----!-----!
51 *   *****
52 #ifndef MATHUTIL2_H
53 #define MATHUTIL2_H
54
55 #include "Vecteur.h"
56 #include "Coordonnee.h"
57
58 #include "Mat_pleine.h"
59
60 /** @defgroup Classes_utilitaires_sur_vecteurs_et_matrices Classes_utilitaires_sur_vecteurs_et_matrices
61 *
62 * \author      Gérard Rio
63 * \version    1.0
64 * \date       23/01/97
65 * \brief      Def d'utilitaires divers mathematiques sur vecteurs et matrices

```

```

66 *
67 */
68
69 /// @addtogroup Classes_utilitaires_sur_vecteurs_et_matrices
70 /// @{
71 ///
72
73 /// cas d'une erreur survenue à cause d'une non convergence
74
75 class ErrMathUtil2
76     /// =0 cas courant, pas d'information particulière
77     { public :
78         /// constructeur par défaut (pas d'erreur)
79         ErrMathUtil2 () : cas(0) {} ;
80         /// constructeur à partir d'un indice d'erreur ca donné
81         ErrMathUtil2 (int ca) : cas(ca) {} ;
82
83         /// contient l'erreur
84         int cas;
85     };
86 /// ----- fin gestion des erreurs -----
87 /// @} // end of group
88
89 /// @addtogroup Classes_utilitaires_sur_vecteurs_et_matrices
90 /// @{
91 ///
92
93 class MathUtil2
94     { public :
95         /// vecteurs propre et valeurs propres d'une matrice nxn c-a-d : en particulier 3 3 ou 2 2 ou 1 1
96         /// avec une limitation sur n (< 50)
97         /// *** dans le cas où les matrices sont symétriques !! ***
98         /// la méthode est itérative, --> méthode de Jacobi
99         /// la matrice A est ecrasee, a la sortie chaque colonne represente un vecteur propre
100        /// le vecteur de retour contient les valeurs propres
101        /// s'il y a une erreur dans le calcul : cas = -1
102        static Vecteur VectValPropre(Mat_pleine& A,int& cas);
103
104        ///calcul des vecteurs propres pour une matrice 3x3 réel uniquement, pas forcément symétrique
105        /// dans le cas où on connaît déjà les valeurs propres
106        /// la méthode est directe: --> pas d'itération, donc très rapide
107        /// --> Utilisable que pour des matrices carrées
108        /// en entrée : VP, de dimension 3, Vp contient les valeurs propres
109        ///         les 3 valeurs propres sont considérées classées: VP(1) >= VP(2) >= VP(3)
110        ///         cas : qui indique le type de valeurs propres
111        ///         , cas = 1 si 3 val propres différentes (= 3 composantes du vecteur de retour)
112        ///         , cas = 0 si les 3 sont identiques (= la première composantes du vecteur de
113        retour),
114        ///         , cas = 2 si Vp(1)=Vp(2) ( Vp(1), et Vp(3) dans les 2 premières composantes du
115        retour)
116        ///         , cas = 3 si Vp(2)=Vp(3) ( Vp(1), et Vp(2) dans les 2 premières composantes du
117        retour)
118        /// en sortie : le tableau des vecteurs propres, rangés selon l'ordre d'entrée
119        ///         cas = le cas de l'entrée, sauf s'il y a eu un pb, dans ce cas: cas = -1, et les
120        vecteurs propres sont
121        ///         dans ce cas quelconques
122        static Tableau <Coordonnee> V_Propres3x3(const Mat_pleine& A,const Coordonnee & VP, int & cas );
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

149 //                                     ,Tableau <BaseB>& d_Vi);
150
151 /// calcul d'un changement de base: ceci n'est pas fait dans la classe Coordonnee car il faut y
adjointre
152 /// la classe Mat_pleine qui intègre beaucoup de chose, du coup la classe Coordonnee deviendrait une
usine
153 /// changement de base (cf. théorie) : la matrice beta est telle que:
154 /// gpB(i) = beta(i,j) * gB(j) <==> gp_i = beta_i^j * g_j
155 /// et la matrice gamma telle que:
156 /// gamma(i,j) represente les coordonnees de la nouvelle base duale gpH dans l'ancienne gH
157 /// gpH(i) = gamma(i,j) * gH(j), i indice de ligne, j indice de colonne
158 /// c-a-d= gp^i = gamma^i_j * g^j
159 /// rappel des différentes relations entre beta et gamma
160 /// [beta]^{-1} = [gamma]^T ; [beta]^{-1T} = [gamma]
161 /// [beta] = [gamma]^{-1T} ; [beta]^T = [gamma]^{-1}
162 /// changement de base pour de une fois covariant:
163 /// [Ap_k] = [beta] * [A_i]
164
165 static void ChBase(const CoordonneeB& A_B,const Mat_pleine& beta,CoordonneeB& Ap_B);
166
167 /// changement de base pour de une fois contravariant:
168 /// [Ap^k] = [gamma] * [A^i]
169 static void ChBase(const CoordonneeH& A_H,const Mat_pleine& gamma,CoordonneeH& Ap_H);
170
171 };
172 /// @} // end of group
173
174
175 #ifndef MISE_AU_POINT
176 #include "MathUtil2.cc"
177 #define MATHUTIL2_H_deja_inclus
178 #endif
179
180
181 #endif

```

## 7.550 MvtSolide.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      19/01/2007
31 *
32 *   AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 ******
37 *   BUT:      gérer les mouvements solides.
38 *            + homothétie qui est une extension à l'objectif initial !
39 *
40 *            *****
41 *
42 ******/
43
44 #ifndef MOUVEMENT_SOLIDE_H
45 #define MOUVEMENT_SOLIDE_H
46
47 #include "UtilLecture.h"

```

```

48 #include "Coordonnee.h"
49 #include "Tableau_T.h"
50 #include "Basiques.h"
51
52 ///
53 /// \author Gérard Rio
54 /// \version 1.0
55 /// \date 19/01/2007
56
57 class MvtSolide
58 { // surcharge de l'operator de lecture
59 // les informations sont typées
60 friend istream & operator » (istream &, MvtSolide &);
61 // surcharge de l'operator d'écriture typée
62 friend ostream & operator « (ostream &, const MvtSolide &);
63
64 public :
65 // CONSTRUCTEURS :
66
67 // par défaut
68 MvtSolide();
69 // de copie
70 MvtSolide (const MvtSolide& nd);
71 // DESTRUCTEUR :
72 ~MvtSolide ();
73
74
75 // METHODES PUBLIQUES :
76
77 // Affiche les donnees
78 void Affiche () const ;
79
80 // Realise l'egalite
81 MvtSolide& operator= (const MvtSolide& d);
82 //Surcharge d'opérateur logique
83 bool operator == ( const MvtSolide& a) const ;
84 bool operator != ( const MvtSolide& a) const { return !(*this == a);};
85
86 // lecture des mouvements solides
87 void Lecture_mouvements_solides(UtilLecture * entreePrinc);
88 // mot clé d'existence de mouvement solide en lecture
89 static string MotCleMvtSolide() {return string("mouvement_solide_");};
90
91 // 0) indique si oui ou non il y a un mouvement solide
92 bool ExisteMvtSolide() const {return (tab_tcr.Taille() != 0); };
93 // retour de la première rotation: en fait l'axe de rotation
94 const Coordonnee& Premiere_rotation()const;
95 // indique s'il existe seulement une seule rotation
96 bool ExisteUneSeuleRotation()const;
97 // récupération des centres de rotation noeud, éventuellement
98 // 1) donne en retour la liste des identificateurs de centre de noeud
99 const list <String_et_entier>& Liste_ident_centreNoeud()const {return lis_centre_noeud;};
100 // 2) renseigne les centres en cours pour la prochaine applications des mouvements solides
101 void RenseigneCentreNoeud(const list <Coordonnee> & lis_coor) {list_coor_centre_noeud = lis_coor;};
102 // 3) application des mouvements solides aux points transmis et retour du point
103 // transformé
104 Coordonnee & AppliqueMvtSolide (Coordonnee & M) const ;
105 // 3) idem que précédemment, mais avec un coefficient d'intensité
106 // dans le cas d'une translation, celle-ci est multiplié par le coef
107 // dans le cas d'un nouveau centre, sa position n'est pas affectée par le coef
108 // dans le cas d'une rotation, celle-ci est multiplié par le coef
109 // dans le cas d'une homothétie: celle-ci est multiplié par le coef
110 Coordonnee & AppliqueMvtSolide (Coordonnee & M, const double& coef) const ;
111 // affichage et definition interactive des commandes
112 void Info_commande_MvtSolide(ostream & sort);
113
114 // effacement du mouvement solide actuel
115 void EffaceMvtSolide();
116
117 private :
118 // VARIABLES PROTEGEES :
119 Tableau <Coordonnee> tab_tcr; // tableau des translations, centres et rotations
120 Tableau <int> tab_indic; // indique les types de transformations effectuées
121 // =1 cas d'une translation normale
122 // =2 cas de la définition du centre de rotation
123 // =3 cas d'une rotation
124 // =4 cas d'un centre de rotation donné par les coordonnées d'un noeud
125 // =5 cas d'une homothétie
126 // cas où l'on utilise des positions de noeuds pour définir les centres de rotation
127 list <String_et_entier> lis_centre_noeud; // doit avoir la dimension du nombre de fois où il y a
128 // apparition de tab_indic = 4, le string contient le nom du maillage, et l'entier
129 // le numéro du noeud
130 list <Coordonnee> list_coor_centre_noeud; // list de travail qui sert à récupérer la position des
131 // noeuds
132
133
134 // METHODES PROTEGEES :

```



```
135
136 };
137
138 #endif
```

## 7.551 Référence du fichier

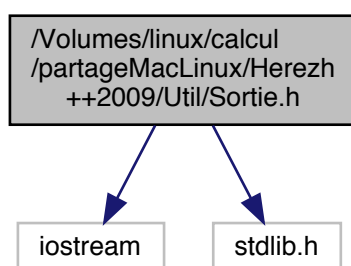
### /Volumes/linux/calcul/partageMacLinux/Herezh++2009/Util/Sortie.h

classes de gestion de l'arrêt d'Herezh

```
#include <iostream>
```

```
#include <stdlib.h>
```

Grappe des dépendances par inclusion de Sortie.h:



Ce graphe montre quels fichiers incluent directement ou indirectement ce fichier :



## Classes

- class [ErrSortie](#)  
*gestion d'exception pour Sortie*
- class [ErrSortieFinale](#)  
*gestion d'exception pour Sortie finale*

## Fonctions

- void [Sortie](#) (int n)  
*fonction pour capter un appel à la sortie*

### 7.551.1 Description détaillée

classes de gestion de l'arrêt d'Herezh

## 7.552 Sortie.h

[Aller à la documentation de ce fichier.](#)

```

1 /*! \file Sortie.h
2  \brief classes de gestion de l'arrêt d'Herezh
3 */
4
5 // This file is part of the Herezh++ application.
6 //
7 // The finite element software Herezh++ is dedicated to the field
8 // of mechanics for large transformations of solid structures.
9 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
10 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
11 //
12 // Herezh++ is distributed under GPL 3 license ou ultérieure.
13 //
14 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
15 // AUTHOR : Gérard Rio
16 // E-MAIL : gerardrio56@free.fr
17 //
18 // This program is free software: you can redistribute it and/or modify
19 // it under the terms of the GNU General Public License as published by
20 // the Free Software Foundation, either version 3 of the License,
21 // or (at your option) any later version.
22 //
23 // This program is distributed in the hope that it will be useful,
24 // but WITHOUT ANY WARRANTY; without even the implied warranty
25 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
26 // See the GNU General Public License for more details.
27 //
28 // You should have received a copy of the GNU General Public License
29 // along with this program. If not, see <https://www.gnu.org/licenses/>.
30 //
31 // For more information, please consult: <https://herezh.irdl.fr/>.
32
33 /*****
34 *      DATE:      23/01/97                      $      *
35 *
36 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)      $      *
37 *
38 *      PROJET:    Herezh++                                $      *
39 *
40 *****/
41 *      BUT:      Gestion de la fin du programme.          $      *
42 *
43 *
44 *      ***** *
45 *
46 *****/
47 #ifndef SORTIE_H
48 #define SORTIE_H
49
50 #include <iostream>
51 using namespace std;
52
53 #include <stdlib.h>
54
55 /** @defgroup Classes_gestions_arret_Herezh Classes_gestions_arret_Herezh
56 *
57 * \author Gérard Rio
58 * \version 1.0
59 * \date 23/01/97
60 * \brief Def gestion de l'arrêt d'Herezh
61 *
62 */
63
64 /// @addtogroup Classes_gestions_arret_Herezh
65 /// @{
66 ///
67
68 /**
69 *
70 * \brief gestion d'exception pour Sortie
71 *
72 */
73 class ErrSortie
74 { public :
75     ErrSortie () {} ; // CONSTRUCTEURS
76     ~ErrSortie () {} ;// DESTRUCTEUR :
77 };
78 /// @} // end of group
79
80
81 /// @addtogroup Classes_gestions_arret_Herezh
82 /// @{
83
84 /**
85 *
86 * \brief gestion d'exception pour Sortie finale
87 *

```

```

88 */
89
90 class ErrSortieFinale
91 { public :
92     ErrSortieFinale () {} ; // CONSTRUCTEURS
93     ~ErrSortieFinale () {} ; // DESTRUCTEUR :
94 };
95     /// @} // end of group
96
97
98 /// @addtogroup Classes_gestions_arret_Herezh
99 /// @{
100 ///
101
102 /// fonction pour capter un appel à la sortie
103 void Sortie(int n);
104 /// @} // end of group
105
106 #endif

```

## 7.553 LesSuitesReel.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 /*****
30 *   DATE:      14/11/2007
31 *
32 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
33 *
34 *   PROJET:    Herezh++
35 *
36 *
37 *   BUT:      gestion des différentes Suites de réels enregistrées.
38 *
39 *
40 *   *****
41 *
42 *   *****/
43
44 #ifndef LES_SUITE_REEL_H
45 #define LES_SUITE_REEL_H
46 #include "SuiteReel.h"
47
48 #include <list>
49 #include "SuiteReel.h"
50
51
52
53 /// @addtogroup def_classes_suites_reel
54 /// @{
55 ///
56
57 /// gestion des différentes Suites de réels enregistrées.
58 class LesSuiteReel
59 {
60     public :
61         // CONSTRUCTEURS :
62         // constructeur par défaut
63         LesSuiteReel () {} ;

```

```

64 // DESTRUCTEUR :
65 ~LesSuiteReel ();
66 // METHODES PUBLIQUES :
67
68 // lecture d'une nouvelle suite, création et retour d'un pointeur sur la suite
69 // si amplitude != 0 alors il faut n non nulle et:
70 // on a U_n=amplitude, ce qui permet pour les suites de réduire le nombre de paramètre à lire
71 static SuiteReel * Lecture_uneSuiteReel(double amplitude=0., int n=0) ;
72
73 protected :
74
75 // VARIABLES PROTEGEES :
76 static list < SuiteReel* > listeDeSuites; // la liste des Suites
77
78 // METHODES PROTEGEES :
79
80 };
81 /// @} // end of group
82
83 #endif

```

## 7.554 Suite\_arithmetique.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Suite_arithmetique.h
30 // classe : Suite_arithmetique
31
32
33 /*****
34 *      DATE:      14/11/2007
35 *
36 *      AUTEUR:    G RIO (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *
41 *      BUT:      Classe permettant des calculs relatifs à des suites
42 *               arithmétique:  $u_n=q U_{(n-1)}$ 
43 *
44 *      *****
45 *
46 * *****/
47
48 #ifndef SUITE_ARITHMETIQUE_M_H
49 #define SUITE_ARITHMETIQUE_M_H
50
51 #include "SuiteReel.h"
52
53
54 /// @addtogroup def_classes_suites_reel
55 /// @{
56 ///
57
58 /// Classe permettant des calculs relatifs à des suites arithmétique:  $u_n=q U_{(n-1)}$ 
59 class Suite_arithmetique : public SuiteReel
60 {
61 public :
62

```

```

63 // CONSTRUCTEURS :
64 // par défaut
65 Suite_arithmetique()
66 : SuiteReel(SUITE_ARITHMETIQUE), U_0(0.), p(0.) {};
67 // de copie
68 Suite_arithmetique(const Suite_arithmetique& Co)
69 : SuiteReel(Co), U_0(Co.U_0), p(Co.p) {};
70 // DESTRUCTEUR :
71 ~Suite_arithmetique(){};
72
73 // METHODES PUBLIQUES :
74
75 // affichage de la courbe
76 void Affiche() const ;
77 // ramène la valeur d'un élément n de la suite
78 double U_n(int n) ;
79 // ramène la somme de la suite de m à n
80 // de manière arbitraire pour n=-1 ==> 0
81 // pour n < -1 --> erreur
82 double Somme_Suite(int n);
83
84 // interactif écran-clavier pour saisir les paramètres d'une suite
85 // si amplitude != 0 alors il faut n non nulle et:
86 // on a Somme_suite(n) = amplitude, ce qui permet pour les suites de réduire le nombre de paramètre à lire
87 void Def_suite(double amplitude, int n) ;
88
89 protected :
90
91 // VARIABLES PROTEGEES :
92 double U_0 ; // la valeur du terme initiale de la suite
93 double p ; // raison arithmétique
94
95 };
96 /// @} // end of group
97
98 #endif

```

## 7.555 Suite\_equidistante.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Suite_equidistante.h
30 // classe : Suite_equidistante
31
32
33 /*****
34 *      DATE:      14/11/2007
35 *
36 *      AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *      PROJET:    Herezh++
39 *
40 *****/
41 *      BUT:      Classe permettant des calculs relatifs à des suites
42 *               equidistante: u_n=U_(n-1)
43 *
44 *      *****
45 *

```

```

46  *****/
47
48 #ifndef SUITE_EQUIDISTANTE_M_H
49 #define SUITE_EQUIDISTANTE_M_H
50
51 #include "SuiteReel.h"
52
53 /// @addtogroup def_classes_suites_reel
54 /// @{
55 ///
56
57 /// Suite_equidistante: Classe permettant des calculs relatifs à des suites equidistante: u_n=U_(n-1)
58 class Suite_equidistante : public SuiteReel
59 {
60 public :
61
62     // CONSTRUCTEURS :
63     // par défaut
64     Suite_equidistante( )
65     : SuiteReel(SUITE_EQUIDISTANTE), U_0(0.) {};
66     // de copie
67     Suite_equidistante(const Suite_equidistante& Co)
68     : SuiteReel(Co), U_0(Co.U_0) {};
69     // DESTRUCTEUR :
70     ~Suite_equidistante(){};
71
72     // METHODES PUBLIQUES :
73
74     // affichage de la courbe
75     void Affiche() const ;
76     // ramène la valeur d'un élément n de la suite
77     double U_n(int ) { return U_0; };
78     // ramène la somme de la suite de 0 à n
79     // de manière arbitraire pour n=-1 ==> 0
80     // pour n < -1 --> erreur
81     double Somme_Suite(int n) ;
82
83     // si amplitude != 0 alors il faut n non nulle et:
84     // on a U_n=amplitude, ce qui permet pour les suites de réduire le nombre de paramètre à lire
85     void Def_suite(double amplitude, int n) ;
86
87 protected :
88
89     // VARIABLES PROTEGEES :
90     double U_0 ; // la valeur du terme courant de la suite
91
92 };
93 /// @} // end of group
94
95 #endif

```

## 7.556 Suite\_geometrique.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.
5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : Suite_geometrique.h
30 // classe : Suite_geometrique
31
32

```

```

33 /*****
34 *   DATE:      14/11/2007
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   ****
41 *   BUT:      Classe permettant des calculs relatifs à des suites
42 *
43 *   *****
44 *
45 *****/
46
47 #ifndef SUITE_GEOMETRIQUE_M_H
48 #define SUITE_GEOMETRIQUE_M_H
49
50 #include "SuiteReel.h"
51 #include "Sortie.h"
52
53 /// @addtogroup def_classes_suites_reel
54 /// @{
55 ///
56
57 /// Suite_geometrique: cas d'une suite géométrique
58 class Suite_geometrique : public SuiteReel
59 {
60 public :
61
62     // CONSTRUCTEURS :
63     // par défaut
64     Suite_geometrique()
65         : SuiteReel(SUITE_GEOMETRIQUE), U_0(0.), p(0.) {};
66     // de copie
67     Suite_geometrique(const Suite_geometrique& Co)
68         : SuiteReel(Co), U_0(Co.U_0), p(Co.p) {};
69     // DESTRUCTEUR :
70     ~Suite_geometrique(){};
71
72     // METHODES PUBLIQUES :
73
74     // affichage de la courbe
75     void Affiche() const ;
76     // ramène la valeur d'un élément n de la suite
77     double U_n(int n) ;
78     // ramène la somme de la suite de 0 à n
79     // de manière arbitraire pour n=-1 ==> 0
80     // pour n < -1 --> erreur
81     double Somme_Suite(int n);
82
83     // interactif écran-clavier pour saisir les paramètres d'une suite
84     // si amplitude != 0 alors il faut n non nulle et:
85     // on a Somme_suite(n) = amplitude, ce qui permet pour les suites de réduire le nombre de paramètre à lire
86     void Def_suite(double amplitude, int n) ;
87
88 protected :
89
90     // VARIABLES PROTEGEES :
91     double U_0 ; // la valeur du terme initiale de la suite
92     double p ; // raison géométrique
93
94     double PUISSn(double a, const int n)
95     {
96         #ifdef MISE_AU_POINT
97             if ( n < 0)
98                 { cout << "\n erreur, exposant négatif dans la fonction PUISSN(Tl a, const int n) ";
99                   Sortie(1);
100                 }
101             #endif
102             int m=n;
103             return ((m > 0) ? (a * PUISSn(a,--m) : 1.);
104         };
105
106 };
107 /// @} // end of group
108
109 #endif

```

## 7.557 SuiteReel.h

```

1 // This file is part of the Herezh++ application.
2 //
3 // The finite element software Herezh++ is dedicated to the field
4 // of mechanics for large transformations of solid structures.

```

```

5 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
6 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
7 //
8 // Herezh++ is distributed under GPL 3 license ou ultérieure.
9 //
10 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
11 // AUTHOR : Gérard Rio
12 // E-MAIL : gerardrio56@free.fr
13 //
14 // This program is free software: you can redistribute it and/or modify
15 // it under the terms of the GNU General Public License as published by
16 // the Free Software Foundation, either version 3 of the License,
17 // or (at your option) any later version.
18 //
19 // This program is distributed in the hope that it will be useful,
20 // but WITHOUT ANY WARRANTY; without even the implied warranty
21 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
22 // See the GNU General Public License for more details.
23 //
24 // You should have received a copy of the GNU General Public License
25 // along with this program. If not, see <https://www.gnu.org/licenses/>.
26 //
27 // For more information, please consult: <https://herezh.irdl.fr/>.
28
29 // fichier : SuiteReel.h
30 // classe : SuiteReel
31
32
33 /*****
34 *   DATE:      14/11/2007
35 *
36 *   AUTEUR:    G RIO   (mailto:gerardrio56@free.fr)
37 *
38 *   PROJET:    Herezh++
39 *
40 *   *****
41 *   BUT:      Classe permettant des calculs relatifs à des suites reel *
42 *
43 *   *****
44 *
45 *   *****/
46
47
48 #ifndef SUITE_M_H
49 #define SUITE_M_H
50
51 #include "Enum_Suite.h"
52
53 /** @defgroup def_classes_suites_reel def_classes_suites_reel
54 *
55 * \author Gérard Rio
56 * \version 1.0
57 * \date 14/11/2007
58 * \brief Classe permettant des calculs relatifs à des suites reel
59 *
60 */
61
62 /// @addtogroup def_classes_suites_reel
63 /// @{
64 ///
65
66 /// classe d'interface (virtuelle pure) pour la création et l'utilisation de suite
67 class SuiteReel
68 {
69 public :
70
71 // CONSTRUCTEURS :
72 // par défaut
73 SuiteReel( Enum_Suite typ = SUITE_NON_DEFINIE)
74 : typeSuite(typ) {};
75 // de copie
76 SuiteReel(const SuiteReel& Co) : typeSuite(Co.typeSuite) {};
77 // DESTRUCTEUR :
78 virtual ~SuiteReel(){};
79
80 // METHODES PUBLIQUES :
81
82 // affichage de la courbe
83 virtual void Affiche() const = 0 ;
84
85 // ramène la valeur d'un élément n de la suite
86 virtual double U_n(int n) =0 ;
87
88 // ramène la somme de la suite de 0 à n en considérant la suite:
89 // 0. U_0 U_1 U_2 etc
90 // ici = sum_0^n U_i
91 // de manière arbitraire pour n=-1 ==> 0

```



```

92 // pour n < -1 --> erreur
93 virtual double Somme_Suite(int n) = 0;
94
95 // ramène le type de la suite
96 Enum_Suite Type_Suite() const { return typeSuite;};
97
98 // interactif écran-clavier pour saisir les paramètres d'une suite
99 // si amplitude != 0 alors il faut n non nulle et:
100 // on a Somme_suite(n) = amplitude, ce qui permet pour les suites de réduire le nombre de paramètre
à lire
101 virtual void Def_suite(double amplitude, int n) =0;
102
103 protected :
104
105 // VARIABLES PROTEGEES :
106 Enum_Suite typeSuite; // type de la suite
107
108
109 };
110 /// @} // end of group
111
112 #endif

```

## 7.558 Util.h

```

1
2 // This file is part of the Herezh++ application.
3 //
4 // The finite element software Herezh++ is dedicated to the field
5 // of mechanics for large transformations of solid structures.
6 // It is developed by Gérard Rio (APP: IDDN.FR.010.0106078.000.R.P.2006.035.20600)
7 // INSTITUT DE RECHERCHE DUPUY DE LÔME (IRDL) <https://www.irdl.fr/>.
8 //
9 // Herezh++ is distributed under GPL 3 license ou ultérieure.
10 //
11 // Copyright (C) 1997-2021 Université Bretagne Sud (France)
12 // AUTHOR : Gérard Rio
13 // E-MAIL : gerardrio56@free.fr
14 //
15 // This program is free software: you can redistribute it and/or modify
16 // it under the terms of the GNU General Public License as published by
17 // the Free Software Foundation, either version 3 of the License,
18 // or (at your option) any later version.
19 //
20 // This program is distributed in the hope that it will be useful,
21 // but WITHOUT ANY WARRANTY; without even the implied warranty
22 // of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
23 // See the GNU General Public License for more details.
24 //
25 // You should have received a copy of the GNU General Public License
26 // along with this program. If not, see <https://www.gnu.org/licenses/>.
27 //
28 // For more information, please consult: <https://herezh.irdl.fr/>.
29
30 /*****
31
32 *      DATE:          23/01/97
33 *
34 *      AUTEUR:        G RIO   (mailto:gerardrio56@free.fr)
35 *
36 *      PROJET:        Herezh++
37 *
38 *
39 *      BUT:           Fonctions utilitaires.
40 *
41 *      *****
42 *
43 *****/
44 #ifndef UTIL_H
45 #define UTIL_H
46
47 #include "Vecteur.h"
48 #include "Tableau_T.h"
49 #include "Enum_ddl.h"
50 #include "Ddl.h"
51 #include "Base.h"
52 #include "PtTabRel.h"
53
54
55 /// @addtogroup Classes_utilitaires_sur_vecteurs_et_matrices
56 /// @{
57 ///
58
59 /// Util: divers utilitaires sur vecteurs, coordonnées, matrices ...
60 class Util

```

```

61
62 { public :
63     /// PRODUIT VECTORIEL DE DEUX VECTEURS EN COORDONNEES ASBOLUS
64     /// on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les
        coordonnées
65     /// static Vecteur ProdVec( const Vecteur & v1, const Vecteur & v2);
66     /// idem en coordonnées absolues avec le type Coordonnee
67     static Coordonnee ProdVec_coor( const Coordonnee & v1, const Coordonnee & v2);
68     /// idem en coordonnées contravariantes avec le type CoordonneeH
69     static CoordonneeH ProdVec_coorH( const CoordonneeH & v1, const CoordonneeH & v2);
70     /// idem en coordonnées covariantes avec le type CoordonneeB
71     static CoordonneeB ProdVec_coorB( const CoordonneeB & v1, const CoordonneeB & v2);
72     /// idem en coordonnées covariantes avec le type CoordonneeB, et en retour un coordonnee normal
73     static Coordonnee ProdVec_coorBN( const CoordonneeB & v1, const CoordonneeB & v2);
74     /// CALCUL DU DETERMINANT DE TROIS VECTEURS
75     /// on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les
        coordonnées
76     /// static double Determinant( const Vecteur & v1, const Vecteur & v2, const Vecteur & v3);
77     /// idem en coordonnées covariantes
78     static double DeterminantB( const CoordonneeB & v1, const CoordonneeB & v2, const CoordonneeB & v3);
79     /// idem en coordonnées absolues avec le type Coordonnee
80     static double Determinant( const Coordonnee & v1, const Coordonnee & v2, const Coordonnee & v3);
81     /// CALCUL DU DETERMINANT DE DEUX VECTEURS
82     /// on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les
        coordonnées
83     /// static double Determinant( const Vecteur & v1, const Vecteur & v2);
84     /// idem en coordonnées covariantes
85     static double DeterminantB( const CoordonneeB & v1, const CoordonneeB & v2);
86     /// idem en coordonnées absolues avec le type Coordonnee
87     static double Determinant( const Coordonnee & v1, const Coordonnee & v2);
88     /// CALCUL DU DETERMINANT DE UN VECTEUR
89     /// on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les
        coordonnées
90     /// static double Determinant( const Vecteur & v1);
91     /// idem en coordonnées covariantes
92     static double DeterminantB( const CoordonneeB & v1);
93     /// idem en coordonnées absolues avec le type Coordonnee
94     static double Determinant( const Coordonnee & v1);
95     /// calcul de la variation d'un vecteur unitaire connaissant la variation du
96     /// vecteur non norme
97     /// v : le vecteur non norme, Dv : la variation de v, nor : la norme de v
98     /// en retour : la variation de vecteur : le vecteur peut-être de dimension > 3
99     ///!!!!!!!!!!!!!!!! très important: il doit s'agir de vecteur exprimé dans un repère orthonormé
100    /// ceci est vrai quelque soit la variance affichée: car ici on ne prend pas en compte la variation
101    /// d'une métrique associée à un repère non orthonormé
102    /// ex: les variations des gi sont ok car en fait les gi représentent les coordonnées dans un repère
        absolu
103    static Vecteur VarUnVect( const Vecteur & v, const Vecteur & Dv, double nor);
104    /// idem avec des coordonnées, donc dim <= 3
105    static Coordonnee VarUnVect_coor( const Coordonnee & v, const Coordonnee & Dv, double nor);
106    /// idem avec des coordonnéesB, donc dim <= 3
107    static CoordonneeB VarUnVect_coorB( const CoordonneeB & v, const CoordonneeB & Dv, double nor);
108    /// idem avec des coordonnéesH, donc dim <= 3
109    static CoordonneeH VarUnVect_coorH( const CoordonneeH & v, const CoordonneeH & Dv, double nor);
110    /// calcul du tableau de variation d'un vecteur unitaire connaissant le tableau de variation du
111    /// vecteur non norme
112    /// v : le vecteur non norme, Dv : la variation de v, nor : la norme de v
113    /// en retour : le tableau de variation
114    static Tableau <Vecteur> VarUnVect( const Vecteur & v, const Tableau <Vecteur >& Dv, double nor);
115    static Tableau <Coordonnee> VarUnVect_coor( const Coordonnee & v, const Tableau <Coordonnee >& Dv,
        double nor);
116    static Tableau <CoordonneeB> VarUnVect_coorB( const CoordonneeB & v, const Tableau <CoordonneeB >&
        Dv, double nor);
117    /// idem et le tableau de retour passé en paramètre
118    static Tableau <Vecteur>& VarUnVect( const Vecteur & v, const Tableau <Vecteur >& Dv, double nor,
        Tableau <Vecteur>& retour);
119    static Tableau <Coordonnee>& VarUnVect_coor( const Coordonnee & v, const Tableau <Coordonnee >& Dv,
        double nor, Tableau <Coordonnee>& retour );
120    static Tableau <CoordonneeB>& VarUnVect_coorB( const CoordonneeB & v, const Tableau <CoordonneeB >&
        Dv, double nor,Tableau <CoordonneeB>& retour);
121    /// la variation du vecteur est supposé se trouver dans le premier vecteur de la base
122    static Tableau <Coordonnee>& VarUnVect_coorBN( const CoordonneeB & v, const Tableau <BaseB>& Dv,
        double nor,Tableau <Coordonnee>& retour);
123
124    /// calcul de la variation d'un produit vectoriel
125    /// vi et Dvi les vecteurs du produit vectoriel et leurs variations
126    /// on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les
        coordonnées
127    /// static Tableau <Vecteur> VarProdVect( const Vecteur & v1, const Vecteur & v2,
128    /// const Tableau <Vecteur >& Dv1, const Tableau <Vecteur >& Dv2);
129    static Tableau <Coordonnee> VarProdVect_coor( const Coordonnee & v1, const Coordonnee & v2,
130    const Tableau <Coordonnee >& Dv1, const Tableau <Coordonnee >& Dv2);
131    /// on supprime la fonction relative aux vecteurs, car elle "doit" être remplacée par celle sur les
        coordonnées
132    /// // idem que le précédent module mais avec Dv qui est la référence des vecteurs Dv1 et Dv2
133    /// static Tableau <Vecteur> VarProdVect( const Vecteur & v1, const Vecteur & v2,
134    /// const Tableau <BaseB >& Dv);

```

```

135 // idem que le précédent module mais avec un retour en coordonnée, et un tableau de BaseB
136 static Tableau <Coordonnee> VarProdVect_coor(const Coordonnee & v1, const Coordonnee & v2,
137     const Tableau <BaseB >& Dv);
138 // idem que le précédent module mais avec in-out en coordonnéeB, et un tableau de BaseB
139 static Tableau <CoordonneeB> VarProdVect_coorB(const CoordonneeB & v1, const CoordonneeB & v2,
140     const Tableau <BaseB >& Dv);
141
142 // idem avec le tableau de retour passé en paramètre
143 static Tableau <Coordonnee>& VarProdVect_coor(const Coordonnee & v1, const Coordonnee & v2,
144     const Tableau <Coordonnee >& Dv1, const Tableau <Coordonnee >& Dv2
145     ,Tableau <Coordonnee>& retour);
146 static Tableau <Coordonnee>& VarProdVect_coor(const Coordonnee & v1, const Coordonnee & v2,
147     const Tableau <BaseB >& Dv,Tableau <Coordonnee>& retour);
148 static Tableau <CoordonneeB>& VarProdVect_coorB(const CoordonneeB & v1, const CoordonneeB & v2,
149     const Tableau <BaseB >& Dv,Tableau <CoordonneeB>& retour);
150 static Tableau <Coordonnee>& VarProdVect_coorBN(const CoordonneeB & v1, const CoordonneeB & v2,
151     const Tableau <BaseB >& Dv,Tableau <Coordonnee>& retour);
152
153 // calcul de la variation de la norme d'un vecteur, connaissant la variation du vecteur,
154 // le vecteur, et sa norme, si la norme est trop petite on met à 0 la variation
155 static Tableau <double> VarNorme(const Tableau <Coordonnee >& Dv,const Coordonnee& V,const double&
    norme);
156 // ici le vecteur peut-être de dimension quelconque > 3 par exemple
157 static Tableau <double> VarNorme(const Tableau <Vecteur >& Dv,const Vecteur& V,const double& norme);

158 // retourne le numero du ddl recherche identifie par en, dans le tableau passé en paramètre
159 // s'il existe sinon 0
160 static int Existe(const Tableau<Ddl >& tab_ddl,Enum_ddl en);
161 // calcul du produit mixte des vecteurs d'une base en coordonnées covariantes
162 static double ProduitMixte(const BaseB & tab_v );
163 // calcul de l'inverse d'une matrice 3x3 donnée par ces coordonnées, en retour la matrice
164 // d'entrée est remplacée par la matrice inverse
165 // 1) cas d'une matrice non symétrique (quelconque), rangement des valeurs:
166 // (1,1) ; (1,2) ; (1,3) ; (2,1) ; (2,2) ; (2,3) ; (3,1) ; (3,2) ; (3,3)
167 static void Inverse_mat3x3(listdouble9Iter & i9Iter);
168 // 2) cas d'une matrice symétrique , rangement des valeurs:
169 // (1,1) ; (2,2) ; (3,3) ; (2,1) = (1,2) ; (3,2) = (2,3) ; (3,1) = (1,3) ;
170 static void Inverse_mat3x3(listdouble6Iter & i6Iter);
171 // calcul de l'inverse d'une matrice 2x2 donnée par ces coordonnées, en retour la matrice
172 // d'entrée est remplacée par la matrice inverse
173 // 1) cas d'une matrice non symétrique (quelconque), rangement des valeurs:
174 // (1,1) ; (2,2) ; (2,1) ; (1,2) ;
175 static void Inverse_mat2x2(listdouble4Iter & i4Iter);
176 // 2) cas d'une matrice symétrique , rangement des valeurs:
177 // (1,1) ; (2,2) ; (2,1) = (1,2) ;
178 static void Inverse_mat2x2(listdouble3Iter & i3Iter);
179
180
181 };
182 // @} // end of group
183
184 #ifndef MISE_AU_POINT
185 #include "Util.cc"
186 #define Util_H_deja_inclus
187 #endif
188
189 #endif

```

